°

REDES PRIMAVERA 2014

Benemérita Universidad Autónoma De Puebla

FACULTAD DE CIENCIAS DE LA COMPUTACION

Redes De Computadoras

"PROYECTO 2"

Dr. Olmos Pineda

Integrantes:
Mara Patricia Vivanco Rendón
Gabriela González Méndez
Elizabeth Chimalt Reyes

Heroica Puebla de Zaragoza a, **[Seleccione la fecha]**

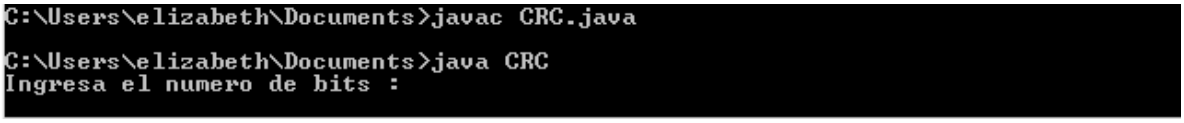
“IMPLEMENTACION CRC”

En este proyecto el objetivo es implementar CRC.

CRC es la comprobación de redundancia cíclica que es un código de detección de errores usado frecuentemente en redes digitales y en dispositivos de almacenamiento para detectar cambios accidentales en los datos.

Nuestro proyecto no cumple con los requisitos ya que el proyecto original debe leer un archivo y convertir ese archivo en bits lo cual no logramos hacer y lo que hicimos fue una sencilla implementación de CRC.

Lo que hace nuestra aplicación es leer los datos, es decir, le pedimos al usuario que ingrese el número de bits:

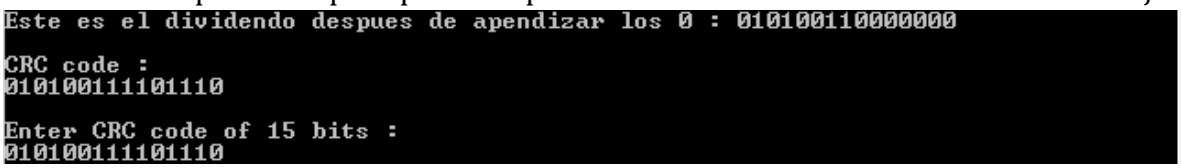


Y le pedimos que ingrese manualmente el arreglo de bits

Después pedimos el número de bits de divisor

De igual manera pedimos que ingrese el arreglo de bits del divisor

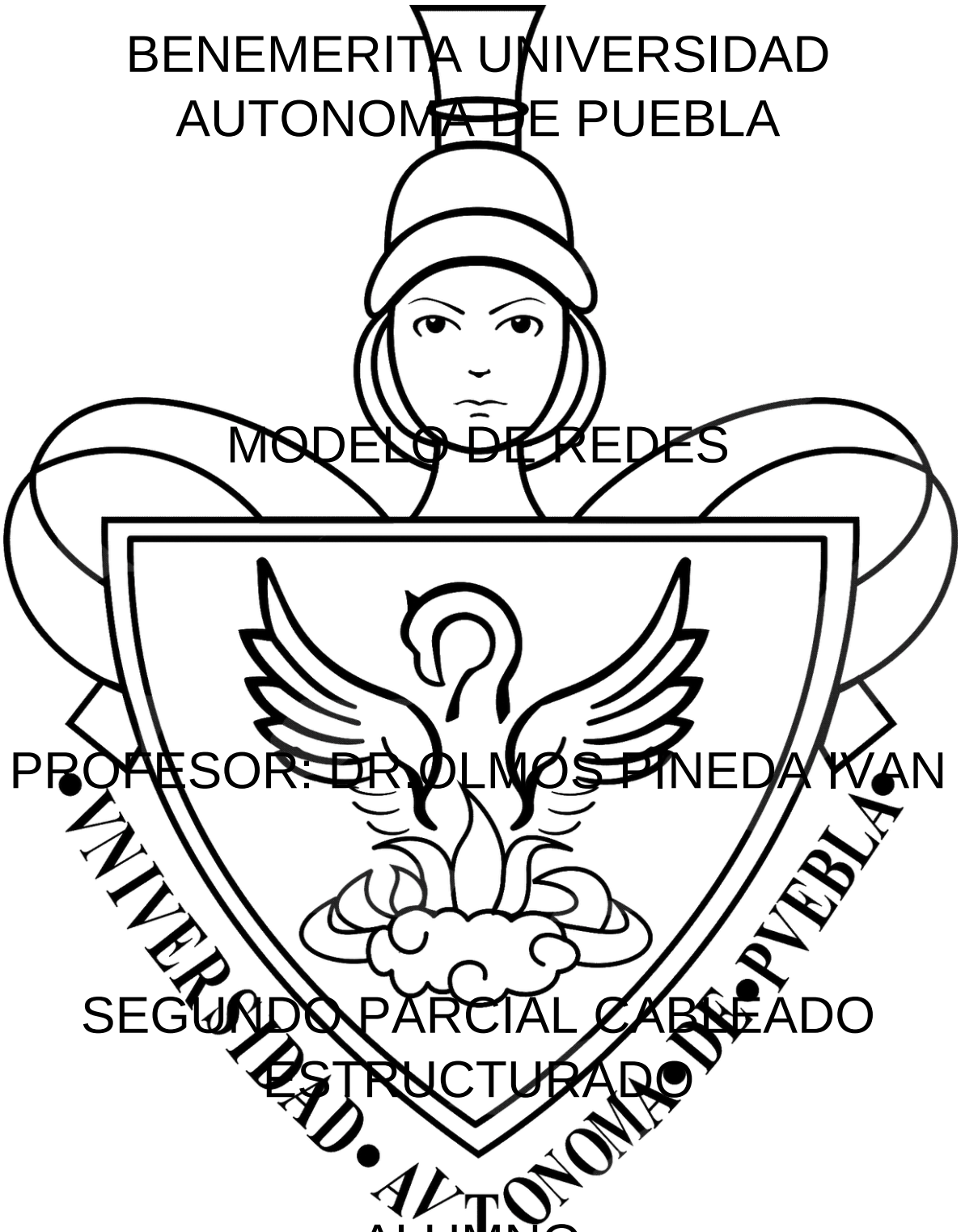
Después hace operaciones y apendiza el dividendo con los 0 y nos da el código CRC el cual debemos poner después para comprobar si hubo errores o no al enviar el mensaje.



Y final mente hace la comprobacion y nos muestra un letrero donde nos dice si el mensaje tuvo errores o no.

En conclusión el CRC se utiliza como una detección de errores de código, el cual tiene una serie de aplicaciones usadas cuando se implementa mediante normas, convirtiéndolo así en un sistema práctico.

BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA



MODELO DE REDES

PROFESOR: DR. OLMOS PINEDA IVAN

SEGUNDO PARCIAL CABLEADO ESTRUCTURADO

ALUMNO:

JOSE LUIS HERRERA ESCAMILLA

OBED Z. GONZALEZ

OSCAR SANDOVAL MARTINEZ

CARLOS ISAI GONZALEZ AGUILAR

CABLEADO ESTRUCTURADO

Introducción

¿Qué es un cableado estructurado?

Se conoce como cableado estructurado al sistema de cables, conectores, canalizaciones y dispositivos que permiten establecer una infraestructura de telecomunicaciones en un edificio, que integran los servicios de voz, datos y vídeo en conjunto con sistemas de administración, dentro de una edificación tales como los sistemas de alarmas, seguridad de acceso y sistemas de energía, etc. Resumiendo, es un cableado para todos los servicios que implican información y control en una edificación.

Desarrollo

De esta manera, el apego del cableado estructurado a un estándar permite que este tipo de sistemas ofrezca flexibilidad de instalación e independencia de proveedores y protocolos, además de brindar una amplia capacidad de crecimiento y de resultar fáciles de administrar. El tendido de cables del cableado estructurado suele desarrollarse con cable de par trenzado de cobre (para redes de tipo IEEE 802.3), aunque también puede utilizarse cable de fibra óptica o cable coaxial. El tendido supone cierta complejidad, cuando se trata de cubrir áreas extensas tales como un edificio de varias plantas. En este sentido hay que tener en cuenta las limitaciones de diseño que impone la tecnología de red de área local que se desea implantar:

- la segmentación del tráfico de red.
- la longitud máxima de cada segmento de red.
- la presencia de interferencias electromagnéticas.
- la necesidad de redes locales virtuales.
- etc.

El sistema de cableado horizontal, es la porción del sistema de cableado de telecomunicaciones que se extiende del área de trabajo al cuarto de telecomunicaciones o viceversa. El cableado horizontal consiste de dos elementos básicos:

1. Cable Horizontal y Hardware de Conexión (también llamado cable horizontal), proporcionan los medios básicos para transportar señales de telecomunicaciones entre el área de trabajo y el cuarto de telecomunicaciones. Estos componentes son los contenidos de las rutas y espacios horizontales.
2. Rutas y Espacios Horizontales (también llamado sistemas de distribución horizontal). Las rutas y espacios horizontales son utilizados para distribuir y soportar cable horizontal y conectar hardware entre la salida del área de trabajo y el cuarto de telecomunicaciones. Estas rutas y espacios son los contenedores del cableado horizontal.

Algunas recomendaciones para las técnicas de cableado:

- si existiera cable suspendido, se recomienda la utilización de canaletas para transportar los cables horizontales.
- Una tubería de 3/4 de pulgada, por cada dos cables de UTP.
- una tubería de 1 pulgada, por cada cable de dos fibra ópticas.
- los radios mínimos de curvatura deben ser bien implementados.

El cableado horizontal incluye:

- las salidas (cajas/placas/conectores) de telecomunicaciones en el área de trabajo (Work Area Outlets WAO).
- Cables y conectores de transición instalados entre las salidas del área de trabajo y el cuarto de telecomunicaciones.
- Paneles de parcheo y cables de empalme, utilizados para configurar las conexiones de cableado horizontal en el cuarto de telecomunicaciones.
- Se deben hacer ciertas consideraciones a la hora de seleccionar el cableado horizontal: contiene la mayor cantidad de cables individuales en el edificio.

Topología.

La norma EIA/TIA 568A hace las siguientes recomendaciones, en cuanto la topología del cableado horizontal. El cableado horizontal debe seguir una topología tipo estrella. Cada toma/conector de telecomunicaciones del área de trabajo debe conectarse a una interconexión en el cuarto de telecomunicaciones.

Distancias.

Sin importar el medio físico, la distancia horizontal máxima no debe exceder 90m. La distancia se mide desde la terminación mecánica del medio en la interconexión horizontal en el cuarto de telecomunicaciones hasta la toma/conector de telecomunicaciones en el área de trabajo. Además, se recomiendan las siguientes distancias, se separan 10m para los cables del área de trabajo y los

cables del cuarto de telecomunicaciones (cordones de parcheo, jumpers y cables de equipo). Medios reconocidos, se reconocen tres tipos de cables para el sistema del cableado horizontal:

- cables de par trenzado sin blindar (UTP) de 100ohm y cuatro pares
- cables de par trenzado blindados (STP) de 150 ohm y cuatro pares
- cables de fibra óptica multímodo de 62.5/125 um y dos fibras

En la siguiente tabla, se muestran las zonas de trabajo para cada edificio de la facultad de ciencias de la computación:

EDIFIFICOS	ZONAS DE TRABJO	USUARIOS (X ÁREA DE TRABAJO)	METROS USADOS
104A (cubículos)	1	46	819
104B			
104C (laboratorios)	4	35	1500
104D	3	255	4600

Edificio 104A **Edificio 104B**

Edificio 104C **Edificio 104D**

Cableado Backbone

El propósito del cableado del backbone, es proporcionar interconexiones entre cuartos de entrada de servicios de edifico, cuartos de equipo y cuartos de telecomunicaciones. El cableado del backbone incluye la conexión vertical entre pisos en edificios de varios pisos. El cableado del backbone incluye medios de transmisión, puntos principales e intermedios de conexión cruzada y terminaciones mecánicas. El cableado vertical realiza la conexión entre los diferentes gabinetes de telecomunicaciones y entre estos y la sala de equipamiento. En este componente del sistema de cableado ya no resulta económico mantener la estructura general utilizada en el cableado horizontal, sino que es conveniente realizar instalaciones independientes. El backbone de datos, se puede implementar con cables UTP o con fibra óptica. En el caso de utilizar UTP, el mismo será de categoría 6a y se dispondrá un número de cables desde cada gabinete al gabinete seleccionando como centro de estrella. Actualmente, la diferencia de costo provocada por la utilización de fibra óptica se ve compensada por la mayor flexibilidad y posibilidad de crecimiento que brinda esta tecnología. Se construye el backbone, llevando un cable de fibra desde cada gabinete al gabinete centro de la estrella. Si bien, para una configuración mínima Ethernet basta con utilizar cable de 2

fibras, resulta conveniente utilizar cable con mayor cantidad de fibra (6 a 12), ya que la diferencia de costo no es importante y se posibilita por una parte disponer de conductores de reserva para el caso de que alguno falle, y por otra parte la utilización en el futuro de otras topologías que requieran más conductores, como FDDI o sistemas resistentes a fallas. La norma EIA/TIA 568, prevé la ubicación de la transmisión de cableado vertical a horizontal, y la ubicación de los dispositivos necesarios para lógralo, en habitaciones independientes con puerta destinada a tal fin, ubicadas por lo menos una por piso, denominados closet de comunicaciones. Se utilizan habitualmente gabinetes estándar de 19 pulgadas de ancho, con puertas de aproximadamente 50cm de profundidad y de una altura entre 1.5 a 2m, en dichos gabinetes se disponen generalmente de las siguientes secciones:

- **acometida de los puestos de trabajo:** 2 cables UTP llegan desde cada puesto de trabajo.
- **acometida de datos:** cables de fibra óptica, que se llevan a una bandeja de conexión adecuada.
- **electrónica de la red de datos:** Hubs, Switches, Bridges y otros dispositivos necesarios.
- **alimentación eléctrica.**
- **iluminación interna** para facilitar cualquier trabajo dentro del gabinete.
- **Ventilación.**

EIDIFICIO	DISTANCIAS	MATERIAL
104A- 104B	42m	Fibra óptica
104B - 104C	42m	Fibra óptica
104C - 104D	87m	Fibra óptica
Distancia Extra	15m	Fibra óptica

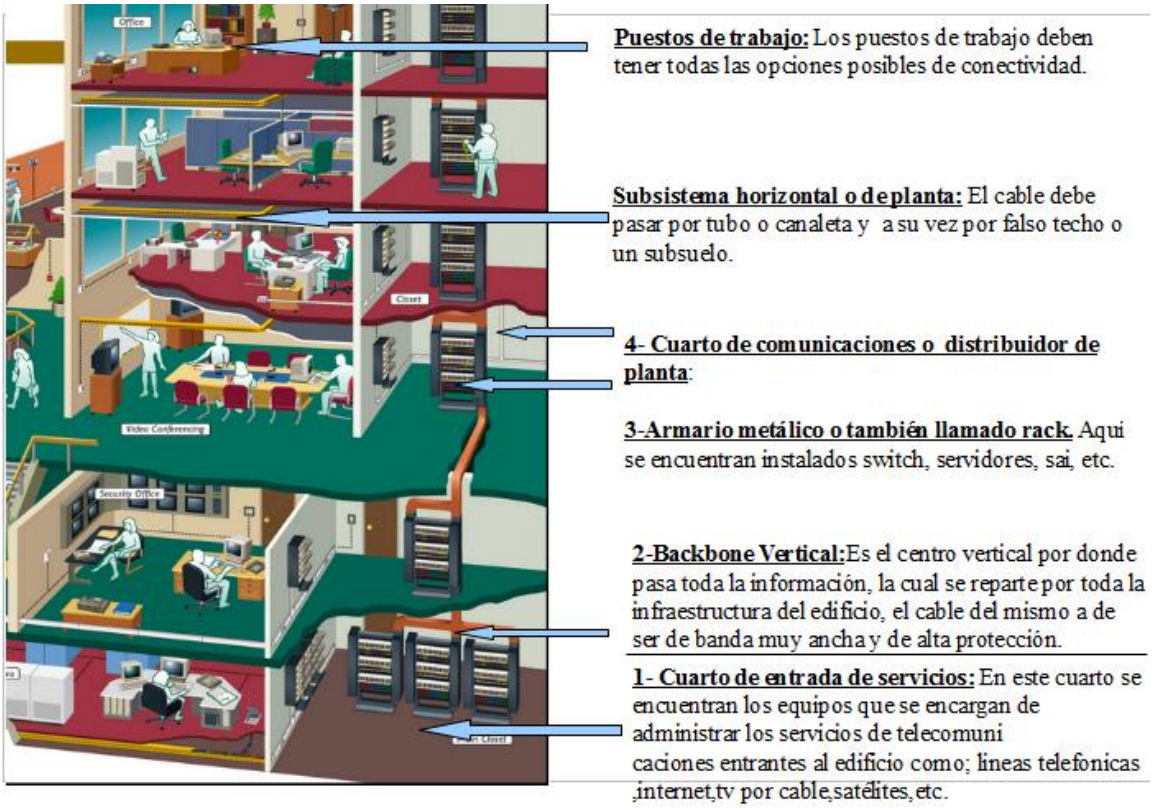
Velocidad según la categoría de la red

- **categoría 1**: se utiliza para comunicaciones telefónicas y no es adecuado para la transmisión de datos ya que sus velocidades no alcanzan los 512kbps
- **categoría 2:** puede transmitir datos a velocidades de hasta 4Mbps
- **categoría 3:** se utiliza en redes 10BaseT y puede transmitir datos a velocidades de hasta 10Mbps
- **categoría 4:** se utiliza en redes Token Ring y puede transmitir datos a velocidades de hasta 16Mbps
- **categoría 5:** puede transmitir datos a velocidades de hasta 100Mbps
- **categoría 6**: redes de alta velocidad hasta 1Gbps
- **categoría 6a:** redes de alta velocidad de hasta 10Gbps

Impedancia y Distorsión por Retardo

Las líneas de transmisión tendrán en alguna porción ruido de fondo, generando por fuentes externas, el transmisor o las líneas adyacentes. Este ruido se combina con la señal transmitida. La distorsión resultante puede ser menor, pero la atenuación puede provocar que la señal transmitida. La distorsión resultante puede ser menor, pero la atenuación puede provocar que la señal digital desciende al nivel de la señal de ruido. El nivel de la señal digital es mayor que el nivel de la señal de ruido, pero se acerca al nivel de la señal de ruido a medida que se acerca al receptor. Una señal formada por varías frecuencias es propensa a la distorsión por retardo causada por la impedancia, la cual es la resistencia al cambio de las diferentes frecuencias. Esta puede provocar que los diferentes componentes de frecuencia que contiene las señales lleguen fuera de tiempo al receptor. Si la frecuencia se incrementa, el efecto empeora y el receptor estará imposibilitado de interpretar las señales correctamente. Este problema puede resolverse disminuyendo el largo del cable. El cable debe tener una impedancia de 100ohm en la frecuencia usada para transmitir datos. Es importante mantener un nivel de señal sobre el nivel de ruido. La mayor fuente de ruido en un cable par trenzado con varios alambres es la interferencia. La interferencia, es una ruptura de los cables adyacentes y no es un problema típico de cables. El ruido ambiental en los circuitos digitales, es provocado por las lámparas fluorescentes, motores, hornos de microondas y equipos de oficina (computadoras, teléfonos, copiadoras). Para medir la interferencia se inyecta una señal de valor conocido en un extremo y se mide la interferencia en los cables vecinos.

Implementación de cableado estructurado



En este caso usaremos un BACKBONE horizontal.

Herramientas que usaremos en el cableado estructurado

Pinzas ponchadoras

Estas pinzas las usamos para colocar el conector RJ-45 a un cable UTP, en la imagen podemos apreciar que en la parte de abajo tiene una navaja para cortar el cable y poder pelarlo. Y en la parte del centro es donde se mete el conector RJ-45 ya armado con los cables y se poncha.

Herramienta de presión

Esta se usa para colocar el cable UTP en el Jack o hembra RJ-45.

Tester

Esta herramienta nos permite verificar la continuidad de un cable UTP que hayamos armado, así como también detectar cruzamientos, es decir, si al armar el cable intercambiamos la posición de algún alambre.

Flexometro

Indispensable para medir el cable UTP y no desperdiciar.

Pinzas de corte

Herramienta manual que se utiliza para cortar cable.

Cortador de canaletas

Esta herramienta como su nombre lo indica corta canaletas, además de hacer en solo segundos hace un corte recto y sin rebabas.

Rollo de velcro

Esta no es una herramienta pero es un material necesario para poder agrupar los cables UTP.

Herramientas en general

En esta parte entran los que son desatornilladores, taladro, etc. Que son herramientas no exclusivas para instalación de cableado estructurado.

Material para un cableado estructurado

Cable UTP cat 6ª

El nombre correcto es cable de par trenzado, esto es debido a que se trata de una funda plástica externa blindada ó no blindada, que contiene un conjunto de 8 cables que se encuentran trenzados entre sí de dos en dos, básicamente de la forma blanco/verde - verde, blanco/naranja - naranja, blanco/café - café y blanco/azul -azul, lo anterior no indica que al momento de su uso sea del mismo modo, sino que se combinan según las necesidades. Este cable permite ser utilizado para la transmisión de datos en las redes informáticas, así como de señales telefónicas.

Rack

Un rack es un soporte metálico destinado a alojar equipamiento electrónico, informático y de comunicaciones. Las medidas para la anchura están normalizadas para que sean compatibles con equipamiento de distintos fabricantes. También son llamados bastidores, cabinas, gabinetes o armarios.

Patch panel

Front

Back

Los llamados Patch Panel son utilizados en algún punto de una red informática donde todos los cables de red terminan. Se puede definir como paneles donde se ubican los puertos de una red, normalmente localizados en un bastidor o rack de telecomunicaciones. Todas las líneas de entrada y salida de los equipos (ordenadores, servidores, impresoras... etc.) tendrán su conexión a uno de estos paneles.

Placas de pared RJ-45

Estas son montadas en las cajas de pared y en ellas son montados los Conectores hembra RJ-45.

Caja para pared

Esta es montada en la pared y por dentro van los cables UTP.

Canaleta

Se empotra a la pared y por dentro van los cables UTP.

Conector hembra RJ-45

Se utiliza para colocar en placas o paneles de parcheo.

Conector RJ-45

El conector RJ-45 (RJ significa *Registered Jack*) es uno de los conectores principales utilizados con tarjetas de red Ethernet, que transmite información a través de cables de par trenzado. Por este motivo, a veces se le denomina puerto Ethernet.

Patchcords

Cable de conexión (*patch cord*) se usa en redes de computadoras o sistemas informáticos o electrónicos para conectar un dispositivo electrónico con otro.

Fibra de cuatro canales

Equipo electrónico que se usa en una red de cableado estructurado

Router

Un “Router” es como su propio nombre indica, y fácilmente se puede traducir, un enrutador o encaminador que nos sirve para interconectar redes de ordenadores y que actualmente implementan puertas de acceso a internet como son los router para ADSL, los de Cable o 3G.

Switch

Un switch es un dispositivo que sirve para conectar varias elementos dentro de una red. Estos pueden ser un PC de escritorio, una impresora, la misma televisión, tu PS3 o cualquier aparato que posea una tarjeta Ethernet.

En cualquier oficina o lugar de trabajo es muy común tener al menos un switch por planta que permite la interconexión de los distintos equipos.

Regulador de voltaje

El propósito de un regulador de voltaje es mantener el voltaje en un circuito relativamente cerca de un valor deseado. Son uno de los componentes electrónicos más comunes, ya que las fuentes de alimentación con frecuencia producen corrientes que, sin el regulador, dañarían alguno de los componentes en el circuito. Estos dispositivos tienen una variedad de funciones específicas, dependiendo de su aplicación particular.

COTIZACION DE LA INSTALACION DE RED ESTRUCTURADA

Rack de 42 UR PESADO	6	$9,708.00
Organizador vertical 2U rack	12	$5,136.00
Organizador Horizontal 2U rack	12	$4,276.00
Cable UTP cat 6a	6919	$228,327.00
Fibra Optica de 4 Hilos	200	$10,600
Jack RJ45 cat 6a	376	$1,880.00
Conector SC	20	$1,280.00
Placas de 2 ventanas	188	$4,888.00
Patch panel 24 puertos cat 6a	5	$25,830.00
Patch panel 48 puertos cat 6a	1	$9,228.00
Patch cord 1 metro cat 6a	70	$25,620.00
Rollo de velcro	6	$2,322.00

Cajas de montaje	188	$23,688.00
Switch 16 puertos	28	$201,068.00
Switch 48 Puertos	1	$41,370.00
Switch 24 puertos	5	$112,700.00
Router alambricos	6	$21,180.00
Routers AP inalambricos	9	$37,674.00
Regulador voltaje	6	$16,800.00
Rejillas porta cables	200	$45,000.00
TOTAL		$828,575.00

Conclusión

El Cableado Estructurado es una técnica o un sistema de cableado de redes que sigue una serie de normativas de manera modular con el fin de proporcionar una obra física apropiada para el usuario desde el punto de vista de las necesidades de telecomunicaciones.
El diseño de una red hoy en día, debe ser cuidadosamente analizado. Entre los factores que influyen para lograr un buen diseño se deben citar: la flexibilidad con respecto a los servicios soportados, la vida útil requerida, el tamaño del sitio y la cantidad de usuarios que estarán conectados y los costos, entre otros. Teniendo en cuenta estos factores no se debe dudar en utilizar el mecanismo que provea las facilidades de estandarización, orden, rendimiento, durabilidad, integridad y facilidad de expansión como el cableado estructurado provee.

[ESCRIBA EL NOMBRE DE LA COMPAÑÍA]

FACULTAD DE CIENCIAS DE LA COMPUTACION

en 1

[Escriba el subtítulo del documento]

Dr. Olmos Pineda

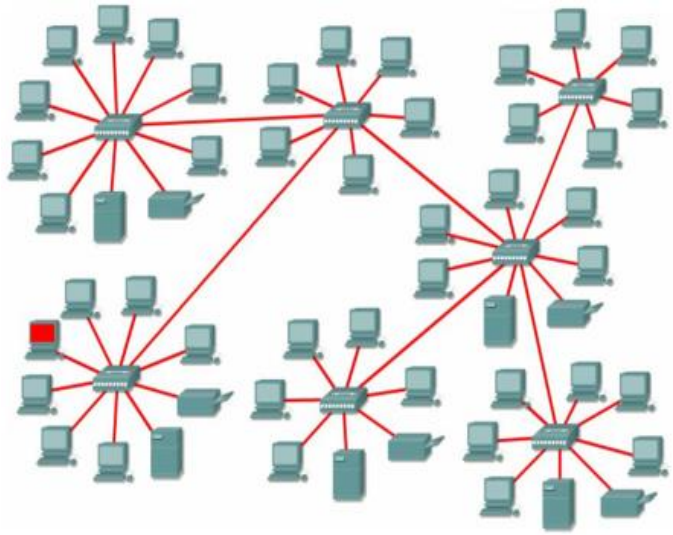
user

Heroica Puebla de Zaragoza a, **[Seleccione la fecha]**

Introducción

LATENCIA

La latencia de una línea de transmisión expresa el tiempo que tardan los datos en entrar por un extremo del enlace hasta que aparecen por el otro.



Depende de tres factores:

- Del retardo de propagación de las señales a través del medio de transmisión. Este tiempo depende de la velocidad de propagación de las señales a través del medio (3,0 x 10^8 metros/segundo en el vació). El retardo de propagación es insalvable y se conoce como latencia mínima.

- El mensaje a transferir normalmente se divide en bloques o paquetes. Por tanto el tiempo que tarda un BIT depende de la tasa de transferencia de la red.

- También depende de la congestión de la red.

Por lo tanto la latencia es igual a:

Latencia = Retardo de propagación + Tiempo de emisión + Tiempo de cola + Retardo de procesamiento

Donde:

- ✓ Retardo de propagación: es el tiempo que tarda la información en llegar desde su punto de partida al destino, depende de la trayectoria física y del medio físico de transmisión.

- ✓ Tiempo de emisión: es el tiempo requerido para enviar todos los bits en un paquete al medio de transmisión utilizado.

- ✓ Tiempo de cola: es el tiempo que un paquete es puesto en una cola hasta que es transmitido, el número de paquetes en cola dependerá en la cantidad y el tipo de tráfico de la red.

- ✓ Retardo de procesamiento: es el tiempo requerido para analizar el encabezado de un paquete y decidir a donde enviarlo.

Algoritmo de Dijkstra

También llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista. Su nombre se refiere a Edsger Dijkstra, quien lo describió por primera vez en 1959.

La idea subyacente en este algoritmo consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen, al resto de vértices que componen el grafo, el algoritmo se detiene.

Algoritmo en pseudocódigo

```
1   método Dijkstra(Grafo,origen):
2       creamos una cola de prioridad Q
3       agregamos origen a la cola de prioridad Q
4       mientras Q no este vacío:
5           sacamos un elemento de la cola Q llamado u
6           si u ya fue visitado continuo sacando elementos de Q
7           marcamos como visitado u
8           para cada vértice v adyacente a u en el Grafo:
9               sea w el peso entre vértices ( u , v )
10              si v no ah sido visitado:
11                  Relajacion( u , v , w )

1   método Relajacion( actual , adyacente , peso ):
2       si distancia[ actual ] + peso < distancia[ adyacente ]
3          distancia[ adyacente ] = distancia[ actual ] + peso
4          agregamos adyacente a la cola de prioridad Q
```

Desarrollo

Problema

Implementar un programa que dada la velocidad de transmisión, distancia de segmento, y los tiempos de cola pueda encontrar el camino más rápido entre un nodo inicial y otro final designados por el usuario, así como calcular el tiempo de transmisión de un archivo.

Solución

El método de entrada de datos fue a través de un archivo de texto plano. Dicho archivo debe contener el número de nodos y la cantidad de enlaces. En la siguiente línea debe tener el nodo origen y el nodo destino. La tercera línea especificar el nodo origen, nodo destino, longitud, velocidad de transferencia, datos de control, datos de usuario. Y por ultimo tiempo de cola.

Para poder resolver el problema de encontrar el camino mas corto de la red, trate de implementar el algoritmo de Dijkstra en lenguaje C, el cual tiene como objetivo obtener el camino más corto de un grafo.

Entonces el algoritmo de dijkstra trabaja así:

Primero marcamos todos los vértices como no utilizados. El algoritmo parte de un vértice origen que será ingresado, a partir de ese vértices evaluaremos sus adyacentes.

Se realiza la búsqueda en todos los vértices adyacentes, y el que esté más cerca de nuestro punto origen, lo tomamos como punto intermedio y vemos si podemos llegar más rápido a través de este vértice a los demás. Después escogemos al siguiente más cercano y repetimos el proceso. Esto lo hacemos hasta que el vértice no utilizado más cercano sea nuestro destino. Al proceso de actualizar las distancias tomando como punto intermedio al nuevo vértice se le conoce como relajación (relajación).

Para el recorrido usaremos una cola de prioridad, debe tener la propiedad cada vez que extraiga un elemento me debe devolver el de menor valor, en este caso dicho valor será el peso acumulado en los nodos.

Después de calcular el posible camino se necesita calcular el tiempo de transmisión y el tiempo de propagación empleando las formulas dadas en clase. A continuación de esto obtendremos la latencia.

Pero al final no logre que el programa leyera los datos desde el archivo y ocurre una interrupción de la ejecución.

BIBLIOGRAFIA

http://www.ecured.cu/index.php/Latencia_inform%C3%A1tica

https://sites.google.com/site/comdatosgrupo4/contenidos/cap5_arendredes#TOC-Latencia

http://jariasf.wordpress.com/2012/03/19/camino-mas-corto-algoritmo-de-dijkstra/

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Materia:
Redes de Computadoras

Profesor:
Iván Olmos Pineda

Trabajo:
Reporte del Primer Examen Parcial

Alumno:
Alejandro Blanca Calderón

Matricula:
201041063

Fecha de Entrega:
25/02/2014

Benemérita Universidad Autónoma De Puebla

FACULTAD DE CIENCIAS DE LA COMPUTACION

Redes De Computadoras

Primer Examen Parcial
“Latencia”

Dr. Olmos Pineda

Elizabeth Chimalt Reyes

Heroica Puebla de Zaragoza a, **[Seleccione la fecha]**

Introducción

En redes informáticas de datos se denomina latencia a la suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red.

Otros factores que influyen en la latencia de una red son:

- Tiempo de propagación del bit por el medio, que depende del tiempo de propagación de la corriente o luz por el medio, además de la distancia recorrida

- Máxima cantidad de datos que pueden ser transmitidos por la red sin segmentarse. Al tiempo de propagación del bit en un paquete se le llama tiempo de transmisión

- Tiempos de espera para difundirse un paquete a través de un conmutador, además del tráfico de la red. A este tiempo se le denomina tiempo de cola

Para calcular la latencia tenemos la siguiente formula:

Latencia = Tiempo de propagación + Tiempo de transmisión + Tiempo de

Dónde:

- Tiempo de propagación = velocidad de la luz/distancia a recorrer
- Tiempo de transmisión = tasa de transferencia teórica/tamaño del paquete.

"Desarrollo del tema"

Para resolver este problema lo programe en C++ utilizando arreglos y matrices donde calculo la distancia, la velocidad y la latencia,

- primero limpio los arreglos para evitar cualquier confusión.

```
void ceros(){
  int i;
  for(i=0;i<20;i++){AD[i][0]=0; V[i]=0;}
  cam=0;
}
```

- Después calculamos la Latencia= TP+TT+TC

```
float latencia_xy(int x, int y){
    float latency, TP, TT, TC = 0;
    TP=((float)dis[x][y]*1000)/300000;
    TT=(size_pack[x][y]*8)/speed[x][y];
    if (t_cola[x] != 0)
        TC = t_cola[x];
    latency=TP+TT+TC;
    return latency;
}
```

- Calculamos todos los caminos posibles

```
int do_it(int X){
int i;
 V[X]=1;
 print[indice]=X;
 indice++;
 if(X==fin){for(i=0;i<indice;i++)camino[cam][i]=print[i]; indice--; cam++;  V[X]=0; return;}
 for(i=1;i<=AD[X][0];i++)
   if(!V[AD[X][i]])do_it(AD[X][i]);
   V[X]=0;
   indice--;
   return;
}
```

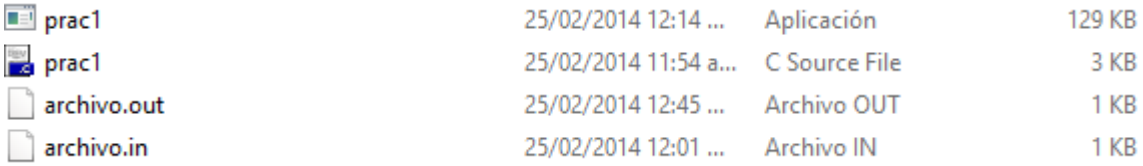
“Conclusión”

El programa lo ejecutamos de la siguiente manera:

1.- abrimos símbolo del sistema

```
C:\Users\elizabeth>cd C:\Users\elizabeth\Documents\ParcialRed

C:\Users\elizabeth\Documents\ParcialRed>prac1.exe

C:\Users\elizabeth\Documents\ParcialRed>
```

2.- nos genera un archivo .out



3.- abrimos el archivo que es el que contiene nuestro resultado

```
 1 - 2 - 5 -
LATENCIA: 424.1303
 1 - 2 - 3 - 4 - 5 -
LATENCIA: 480.5493
 1 - 3 - 4 - 5 -
LATENCIA: 30.6140
 1 - 3 - 2 - 5 -
LATENCIA: 62.1616
Numero total de caminos4
El camino minimo es que tiene la 30.613998 de latencia
```

http://www.cs.buap.mx/~iolmos/redes/3_Rendimiento.pdf

http://www.ecured.cu/index.php/Latencia_inform%C3%A1tica

BENEMERITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

ALUMNO: *DAVID GIL MEJÍA*

ASIGNATURA: REDES DE COMPUTADORAS

ÁREA: REDES Y HARDWARE

CATEDRÁTICO(A): Dr. Iván Olmos Pineda

REPORTE TÉCNICO DE PROYECTO “LATENCIA MENOR”

Puebla de Zaragoza, Puebla. a lunes, 24 de febrero de 2014

1. INTRODUCCIÓN

En este curso lectivo de Redes de Computadoras se estudia diversas partes de lo que abarca la interfaces de redes, capas, etc. Por lo que como evaluación de primer parcial se propuso un Proyecto-Examen que es realizar un programa que imprima camino de menor latencia incluyendo los demás recorridos posibles con su respectiva latencia. Para empezar, se desglosará en este reporte técnico lo que se hizo en este proyecto mediante una de las plataformas de programación ya sea en C/C++, Java de acuerdo a nuestra experiencia.

En lo primero ¿Qué es la Red? Es un conjunto de computadoras autónomas interconectadas entre si por algún medio, ya sea cable físico, enrutador, o cualquier fuente de conectividad o no guiado como red inalámbrica, y tiene como tarea compartir recursos de una máquina a otra máquina. Sus objetivos de esta red son lograr que los recursos(datos) estén disponibles para un usuario de la red, en cualquier parte, alta confiabilidad que atienden peticiones a pesar de que haya fallas en ciertos recursos, bajos costos, y proporcionar un medio de comunicación eficaz y confiable. Los sectores que se aplican la red son en varios sitios como milicia, bancos, tráfico aéreo, seguridad de reactores nucleares, y demás que se enfoquen a la eficiencia corporativa. Incluso tiene para la comunicación entre 2 persona o mas, juegos en línea, y datos remotos. Por lo tanto la red es la fuente en donde establece el compartimiento de recursos e interacción de máquinas remotas.

El sistema de comunicación forma parte de la interacción y sus elementos constan lo siguiente:

- **Mensaje**: Datos que se quiera enviar.
- **Emisor**: El que envía mensaje o datos, se prepara para ser enviados por el medio, tanto en lo cardinal como en calidad. (Origen).
- **Medio**: Elemento a través del cual se envía el mensaje.
- **Receptor**: El que recibe datos (Destino).

Acerca de esto, se verán diversas partes lo que forma una latencia. ¿Qué es Latencia? Es el tiempo que mide en segundos que toma un bit que viaja desde un

inicio hasta el fin y sus factores consisten en que el tiempo de propagación depende del tiempo de propagación de la luz por el medio, la máxima cantidad de datos suelen ser transmitidos sin segmentación, y tiempos de espera para la difusión de paquetes a través de un conmutador. Incluso hay latencia de uso musical, como los transformadores de audio digital a vinilos analógicos. El traspaso de información de un punto a otro sufrirá este retardo mediante la división de paquetes en bits, que normalmente está estimado en milisegundos en algunos casos pequeños, en otro más notorio. La latencia es obtenida por obtención de los 3 tiempos que son:

- Tiempo de Propagación
- Tiempo de Transmisión
- Tiempo de Cola

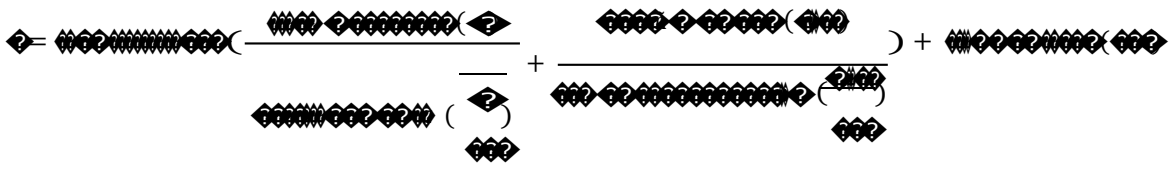
Un punto muy central es que siempre va a haber cierta latencia, aun cuando se trate de latencia cero, la cuestión es que esta es imperceptible (3 ms aproximadamente). Por tanto, se define como tiempo que dura llegar desde su punto de inicio hasta su punto final, es decir cuando la tarea se consuma. Se denominan latencias en una memoria RAM a los diferentes retardos producidos en el acceso a los distintos componentes de esta última. Estos retardos influyen en el tiempo de acceso al procesador, el cual mide en nanosegundos.

Como se explicara a continuación es el desarrollo y los resultados obtenidos con respecto a nuestro proyecto.

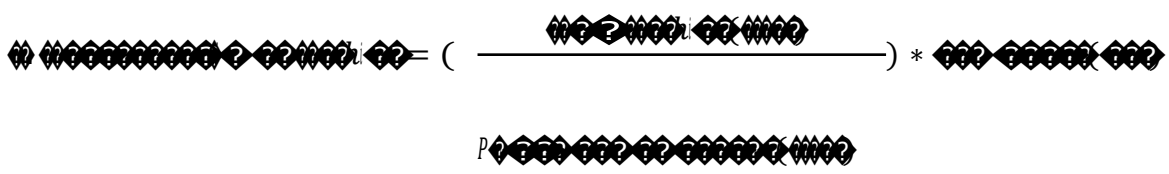
2. ANÁLISIS

Obtener la información con sus respectivos datos de nodos y adyacencias basando en tamaño de paquetes, tiempo de transferencia teórica, tiempo de cola. Imprimir todas las posibilidades, comenzando con la mejor latencia. El tamaño de un archivo será analizado conforme el tiempo de transferencia a través de los

enrutadores desde su dispositivo inicial hasta el final. Su respectiva fórmula de latencia para obtener en cada camino es:



Para la mejor latencia se calcula el tiempo de transferencia que por su ecuación es:



3. DETALLES

Considerar el tamaño de un archivo en bits sin importar la magnitud de memoria física ya sea (byte, Kilobyte, Megabyte, Gigabyte).

Obtener recorridos posibles con sus respectivas latencias tomando en cuenta primero el recorrido de menor latencia encontrada de todos. El algoritmo que se utiliza para los recorridos es Fuerza Bruta por Permutación.

3.1. Fuerza Bruta por Permutación

Este tipo de algoritmo propuesto consiste en que los nodos intermedios permuten de acuerdo desde 1 hasta (n-2) nodos intermedios que son routers, y de cada uno se almacena en la lista los recorridos que cumpla con los requisitos de

adyacencia de un nodo tras otro nodo. Lo que hace primero es tomar los nodos sin considerar el nodo inicial y el nodo final que son máquinas en un arreglo de nodos intermedios, y luego almacenar los recorridos en una lista de recorridos posibles. Al final se utiliza para el cálculo total de latencia de cada camino.

4. DESARROLLO

Este programa se inicializa desde el símbolo de sistema con el fin de que el usuario observe los datos capturados desde una información de topología, por lo

que es un archivo de texto (.txt). Después, se realiza la importación mediante selector de archivo de su clase ***JFileChooser***, done nos permite importar un archivo y almacenar en una objeto de la clase File, por lo que en este caso consideramos cualquier ubicación de un archivo. La ventana que se muestra, es la interfaz que se usa únicamente para la importación de archivo y la implementación de la operación.

Interfaz gráfica al inicio de programa

El botón “Upload Network topology info” sirve para subir un archivo .txt con su respectivo formato de captura de datos de enrutadores y enlaces. El “Run Latency Operation” implementa el cálculo de latencia, este botón esta deshabilitado al ejecución de la aplicación, se habilita una vez que importe un archivo, y vuelve a deshabilitar cuando se presiona el botón “Reset”, lo que hace este botón es liberar el archivo y poder subir otro como parte de requisito del programa. En dado caso de que el usuario quiera subir otro archivo sin liberar el anterior, el mensaje de error aparece así,

Mensaje de error

Esta dialogo de error quiere decir que primero debe liberar el archivo anterior o sea presionar “Reset”. Se aplica esta característica para evitar algún conflicto con los valores y estructuras de datos.

4.1 Implementación

Yendo a lo que es la implementación de la interfaz, lo que se hace es abrir el símbolo de sistema o consola después dirigir a una ruta o carpeta donde está ubicado el archivo jar que lo nombramos “Latencia.jar” una vez extraído posteriormente. Este archivo se generó usando 5 clases que son estructura de datos y la clase principal como se nombraron lo siguiente:

- ***adyacence.class (de enlaces)***
- ***Nodo_arrcomb.class (lista de recorridos)***
- ***Nodo_router.class (tiempos de cola de enrutadores)***
- ***NodoSimple.class (para división de paquetes)***
- ***Nodo_RecorridosFinales.class(para imprimir caminos con su respectivo tiempo de latencia de cada uno)***
- ***Proyecto_LatenciaMenor.class (Clase Principal)***

La forma que se generó la aplicación está definida con esta sintaxis:

```
> jar cvfe Latencia.jar Proyecto_LatenciaMenor *.class
```

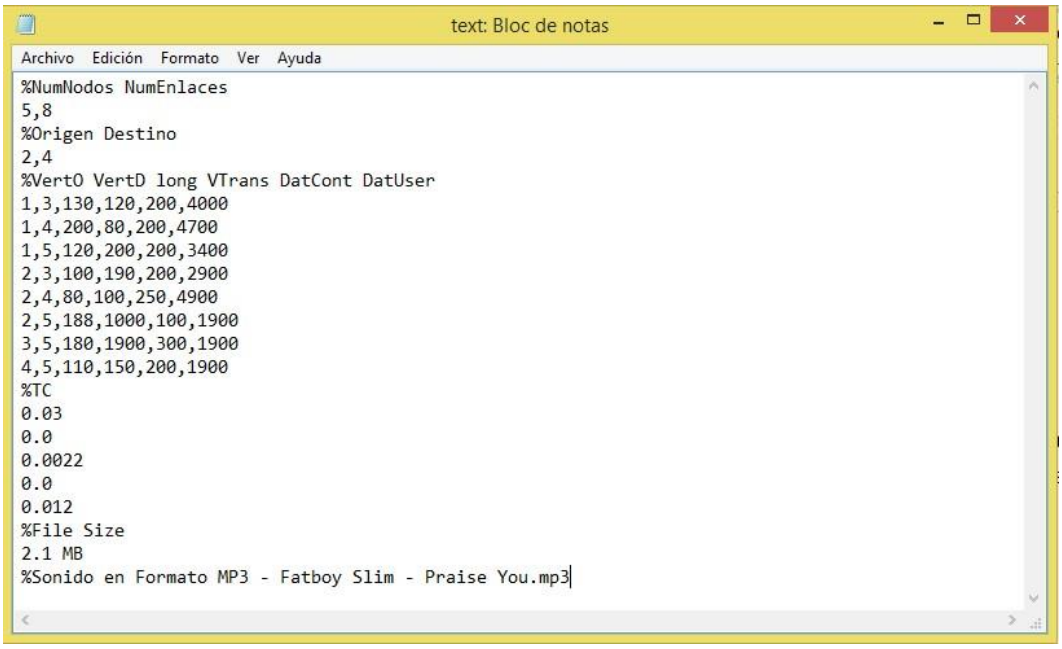
En la parte de “*.class” hace que obtenga todos los archivos contengan este sufijo de sus nombres, o sea, saca un conjunto de archivos generados de código java.

La sintaxis para ejecutar la aplicación es:

```
> java -jar Latencia.jar
```

Con eso nos muestra la ventana principal usado con la clase JFrame como se mostró al inició del desarrollo. Y realizar sus respectivas operaciones definidas en el código lógico.

El formato de cada archivo .txt se elaboró de esta manera:



De todos los datos, la parte de tamaño de archivo se puede introducir la cantidad en decimal y la medida de magnitud de memoria física. Además, este tamaño proporcionado depende mucho al momento de dividir paquetes, porque cuanto más tamaño se le calcule más tardado será la implementación de esta función, por el uso de estructura dinámica de acuerdo con el análisis de algoritmos se ocupa más el tiempo y a la misma vez la memoria de cabecera.

En este programa se utiliza matriz de adyacencias para los enlaces de grafos, luego las listas ligadas para el uso de recorridos de caminos y paquetes; sin embargo, se usó más la memoria dinámica, porque a diferencia de la estática se establece un solo tamaño tomado de la memoria, y la dinámica depende de la memoria de datos. En fin, lo que se hace es cuando se le un archivo va almacenando los datos de un nodo a otro nodo en la matriz de adyacencia mediante el objeto[n][n] de la clase “adyacence”, luego almacenar tiempos de cola en una lista ligada de clase “Nodo_router”, la división de paquetes se utiliza con listas ligadas de dos ligas cada una(***next y down***) de la clase “NodoSimple”, como funcionalidad es insertar división de paquetes conforme cada una de los paquetes de cada división

antecedente. Y por último las listas ligadas de recorridos con su respectivo tiempo de latencia están basadas de la clase “Nodo_RecorridoFinal”.

Aquí se presentan las 4 estructuras de datos en código fuente de los nodos que se usan para las operaciones de este proyecto.

```
class adyacence { // Adyacencia de un nodo a otro nodo
  long dist;
  long speed_th;
  long control_data;
  long user_data;

    public adyacence(long dist, long speed_th, long control_data,
long user_data) {
        this.dist=dist;
        this.speed_th=speed_th;
        this.control_data=control_data;
        this.user_data=user_data;
    }
}

class NodoSimple { // Paquetes con su respectivo tamaño
  long num;
  NodoSimple next;
  NodoSimple down;

    public NodoSimple(long num) {
        this.num=num;
        next=null;
        down=null;
    }
}
```

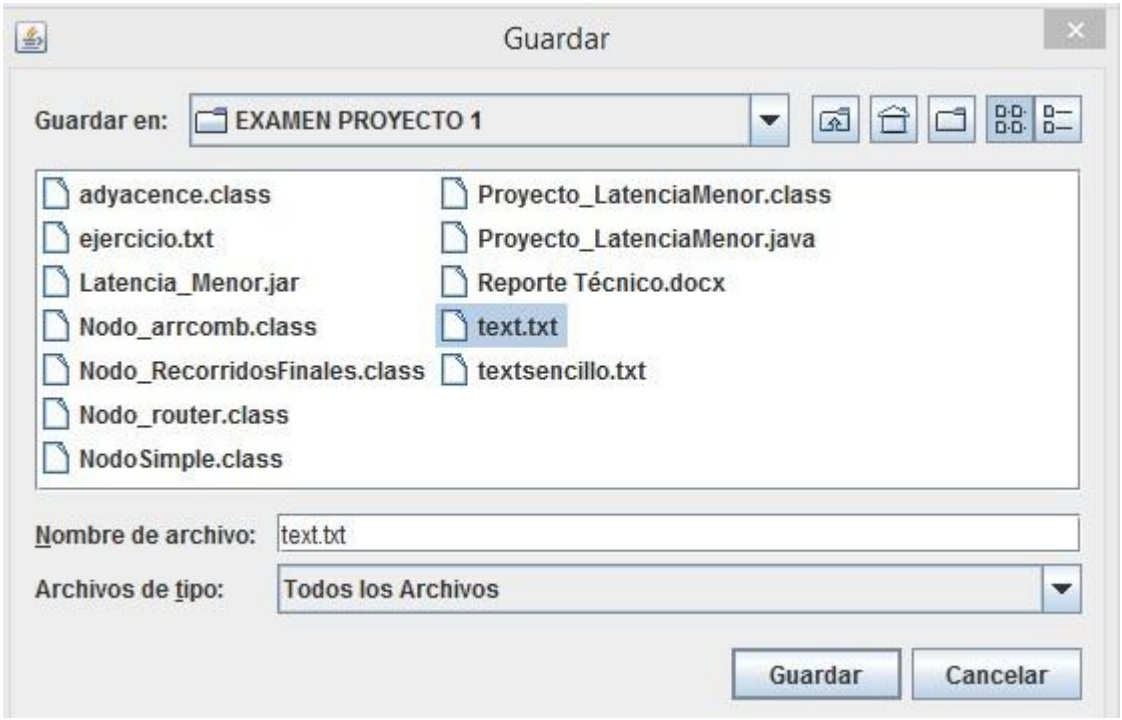
```
class Nodo_router { // Router con su respectivo tiempo de cola
  int id;

  double tiempo_cola;

    Nodo_router next;


    public Nodo_router(int id, double tiempo_cola) {

        this.id=id;
        this.tiempo_cola=tiempo_cola;
        next=null;

    }

}
```

```
   class Nodo_arrcomb{ // Recorridos posibles por Fuerza Bruta por
Permutacion int[] rec;
     Nodo_arrcomb nex;


       public Nodo_arrcomb(int[] rec) {
           this.rec=rec;
           this.nex=null;
       }
   }
```

4.2 Resultados

Estos resultados muestran cómo se implementaron con distintas topologías de red, lo primero se muestra lo que se hizo para hacer funcionar este programa.

Cuando presionemos el botón “Upload Network topology info”, nos mostrara el cuadro de explorador de archivos, lo que elegimos es el archivo con la información de grafo y dar en el botón “Guardar”, y posteriormente registra el nombre de archivo de cualquier ruta especifica no necesariamente en donde está el código fuente y el compilado.


Guardar
Guardar en:
EXAMEN PROYECTO 1
adyacence.class
ejercicio.txt
Latencia_Menor.jar
Nodo_arrcomb.class
Nodo_RecorridosFinales.class
Nodo_router.class
NodoSimple.class
Proyecto_LatenciaMenor.class
Proyecto_LatenciaMenor.java
Reporte Técnico.docx
text.txt
textsencillo.txt
Nombre de archivo:
text.txt
Archivos de tipo:
Todos los Archivos
Guardar
Cancelar

Así es, como nos muestra en el campo de texto de la clase **JTextField** y en esto deshabilitamos la edición con el método setEnable(false), lo que hace es evitar a que el usuario no realice escrituras sobre ella accidentalmente. Por tanto se usa como letrero.

En el campo de texto se muestra el nombre de archivo elegido

Luego en la consola se muestra datos de la topografía:

```
F:\PRIMAVERA 2014\Redes de Computadoras\EXAMEN PROYECTO 1>java -jar Latencia_Menor.jar
        Nodos: 5        Enlaces: 8
        Nodo Origen: 2  Nodo Destino: 4
(1,3)  |Length:130 m.| |TTT:120 Mbps| |Datos Control:500| |Datos Usuario:24000|
(1,4)  |Length:200 m.| |TTT:80 Mbps| |Datos Control:500| |Datos Usuario:34700|
(1,5)  |Length:120 m.| |TTT:200 Mbps| |Datos Control:500| |Datos Usuario:43400|
(2,3)  |Length:100 m.| |TTT:190 Mbps| |Datos Control:500| |Datos Usuario:52900|
(2,4)  |Length:80 m.| |TTT:100 Mbps| |Datos Control:550| |Datos Usuario:64900|
(2,5)  |Length:188 m.| |TTT:1000 Mbps| |Datos Control:500| |Datos Usuario:61900|
(3,5)  |Length:180 m.| |TTT:1900 Mbps| |Datos Control:500| |Datos Usuario:61900|
(4,5)  |Length:110 m.| |TTT:150 Mbps| |Datos Control:500| |Datos Usuario:69000|
Tiempo de cola del Nodo 1 >> 0.34
Tiempo de cola del Nodo 2 >> 0.0
Tiempo de cola del Nodo 3 >> 2.02
Tiempo de cola del Nodo 4 >> 0.0
Tiempo de cola del Nodo 5 >> 0.9
File Size: 20971520 bits
```

Lo que esta mostrado son la cantidad de nodos, enlaces, el nodo origen, el nodo destino, las adyacencia y sus respectivo datos(Longitud, Tiempo de Transferencia Teórica, Datos de Control y Datos de Usuario), los tiempo de colas de cada nodo y el tamaño de archivo, que en este caso consideramos en MB y KB convertidos a bits.

En seguida se hace corrida de la operación de latencia utilizando las respectivas estructuras de datos que definimos, para comenzar se presiona “Run Operation Latency”, que permite a que el programa realice el cálculo de latencia a cada uno

de los recorridos y al final lo imprima con los paquetes totales divididos. Y se almacena en la lista de recorridos de la clase Nodo_RecorridosFinales.

```
Ejecutando calculo de latencia. Esto puede llevar varios minutos ...

Recorridos Finales:

RECORRIDO MENOR => 2 - 5 - 4 - --> Lat:0.9seg. ¦ Tiempo Transf.:304.2seg. ¦ Total paq divi
didos: 678
2 - 3 - 1 - 4 - --> Lat:2.36seg. ¦ Total paq divididos: 2775
2 - 3 - 5 - 4 - --> Lat:2.92seg. ¦ Total paq divididos: 1191
2 - 5 - 1 - 4 - --> Lat:1.24seg. ¦ Total paq divididos: 2034
2 - 3 - 1 - 5 - 4 - --> Lat:3.26seg. ¦ Total paq divididos: 3964
2 - 3 - 5 - 1 - 4 - --> Lat:3.26seg. ¦ Total paq divididos: 2776
2 - 5 - 3 - 1 - 4 - --> Lat:3.26seg. ¦ Total paq divididos: 2712
```

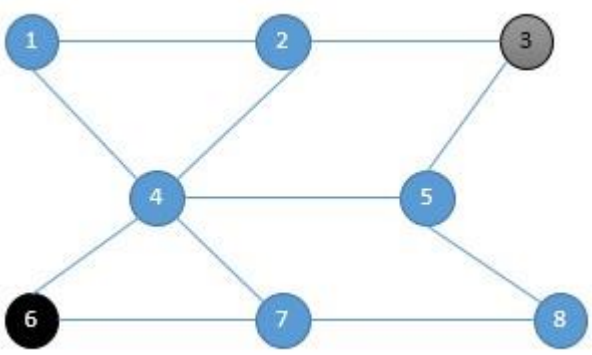
Cálculo de Latencia y Paquetes divididos. Únicamente en la mejor latencia se obtiene el tiempo de transferencia

El tiempo de transferencia se imprime únicamente en el recorrido de menor latencia con el puntero *auxmenor* de esta misma clase que se almacena los recorridos con sus respectivos datos. Además, si se desea calcular los recorridos de otros grafos, se libera el archivo presionando el botón “Reset”.

Este mensaje quiere decir que el usuario no ha liberado el archivo guardado en un objeto de tipo File, y es aplicado igual cuando solo se sube el archivo sin realizar calculo alguno. En caso de modificar alguna información en el archivo de texto, deberá hacer el mismo paso de liberación.

Estos son los siguientes resultados que se muestran junto con el diagrama mostrando solo los nodos y líneas de adyacencia.

Resultado 2:



Sea el nodo inicial 6 y el nodo final 12. Tamaño de entrada de archivo: 1.2 MB

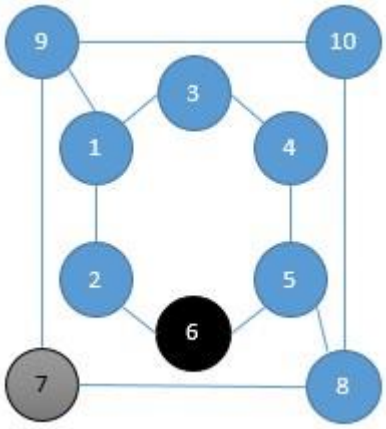
Los datos de este respectivo grafo

```
       Nodos: 8        Enlaces: 11
       Nodo Origen: 3  Nodo Destino: 6
(1,2)  |Length:300 m.| |TTT:1200 Mbps| |Datos Control:200| |Datos Usuario:2700|
(1,4)  |Length:300 m.| |TTT:1200 Mbps| |Datos Control:200| |Datos Usuario:300|
(2,4)  |Length:500 m.| |TTT:1200 Mbps| |Datos Control:200| |Datos Usuario:1200|
(2,3)  |Length:600 m.| |TTT:1200 Mbps| |Datos Control:200| |Datos Usuario:900|
(3,5)  |Length:300 m.| |TTT:1200 Mbps| |Datos Control:200| |Datos Usuario:2100|
(4,5)  |Length:700 m.| |TTT:1200 Mbps| |Datos Control:200| |Datos Usuario:1000|
(4,6)  |Length:800 m.| |TTT:1200 Mbps| |Datos Control:200| |Datos Usuario:1000|
(4,7)  |Length:700 m.| |TTT:1200 Mbps| |Datos Control:200| |Datos Usuario:1200|
(7,8)  |Length:700 m.| |TTT:1200 Mbps| |Datos Control:200| |Datos Usuario:340|
(6,7)  |Length:1200 m.| |TTT:1200 Mbps| |Datos Control:200| |Datos Usuario:700|
(5,8)  |Length:230 m.| |TTT:1200 Mbps| |Datos Control:200| |Datos Usuario:1100|
Tiempo de cola del Nodo 1 >> 0.45
Tiempo de cola del Nodo 2 >> 0.10
Tiempo de cola del Nodo 3 >> 0.0
Tiempo de cola del Nodo 4 >> 1.02
Tiempo de cola del Nodo 5 >> 0.67
Tiempo de cola del Nodo 6 >> 0.0
Tiempo de cola del Nodo 7 >> 0.5
Tiempo de cola del Nodo 8 >> 0.93
File Size: 10066329 bits
```

Las corridas:

```
RECORRIDO MENOR => 3 - 2 - 4 - 6 - --> Lat:1.12seg. ¦ Tiempo Transf.:12526.080000000002seg. ¦ Total paq
divididos: 33555
3 - 5 - 4 - 6 - --> Lat:1.69seg. ¦ Total paq divididos: 33556
3 - 2 - 1 - 4 - 6 - --> Lat:1.57seg. ¦ Total paq divididos: 89480
3 - 2 - 4 - 7 - 6 - --> Lat:1.62seg. ¦ Total paq divididos: 55925
3 - 5 - 4 - 7 - 6 - --> Lat:2.19seg. ¦ Total paq divididos: 57524
3 - 5 - 8 - 7 - 6 - --> Lat:2.1seg. ¦ Total paq divididos: 81491
3 - 2 - 1 - 4 - 7 - 6 - --> Lat:2.0700000000000003seg. ¦ Total paq divididos: 123035
3 - 5 - 8 - 7 - 4 - 6 - --> Lat:3.12seg. ¦ Total paq divididos: 115046
3 - 2 - 4 - 5 - 8 - 7 - 6 - --> Lat:3.22seg. ¦ Total paq divididos: 111850
3 - 2 - 1 - 4 - 5 - 8 - 7 - 6 - --> Lat:3.6700000000000004seg. ¦ Total paq divididos: 190145
```

Resultado 3:



Sea el nodo inicial 7 y el nodo final 6. Tamaño de entrada de archivo: 1.88 MB

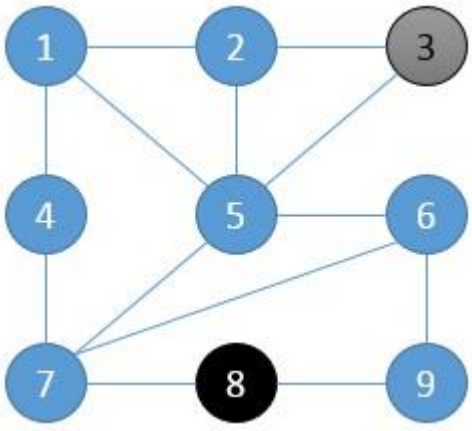
Los datos del respectivo grafo:

```
        Nodos: 10       Enlaces: 12
        Nodo Origen: 7  Nodo Destino: 6
(1,2)   |Length:100 m.| |TTT:200 Mbps| |Datos Control:200| |Datos Usuario:5000|
(1,9)   |Length:100 m.| |TTT:1200 Mbps| |Datos Control:200| |Datos Usuario:5000|
(1,3)   |Length:100 m.| |TTT:330 Mbps| |Datos Control:200| |Datos Usuario:2300|
(2,6)   |Length:100 m.| |TTT:120 Mbps| |Datos Control:200| |Datos Usuario:2300|
(5,6)   |Length:100 m.| |TTT:300 Mbps| |Datos Control:200| |Datos Usuario:2300|
(5,8)   |Length:100 m.| |TTT:500 Mbps| |Datos Control:200| |Datos Usuario:2300|
(4,5)   |Length:100 m.| |TTT:450 Mbps| |Datos Control:200| |Datos Usuario:1000|
(3,4)   |Length:100 m.| |TTT:780 Mbps| |Datos Control:200| |Datos Usuario:5000|
(9,10)   |Length:1000 m.| |TTT:200 Mbps| |Datos Control:200| |Datos Usuario:8700|
(8,10)   |Length:1000 m.| |TTT:200 Mbps| |Datos Control:200| |Datos Usuario:8700|
(7,8)   |Length:1000 m.| |TTT:500 Mbps| |Datos Control:200| |Datos Usuario:7700|
(7,9)   |Length:1000 m.| |TTT:210 Mbps| |Datos Control:200| |Datos Usuario:5000|
Tiempo de cola del Nodo 1 >> 0.05
Tiempo de cola del Nodo 2 >> 0.10
Tiempo de cola del Nodo 3 >> 0.054
Tiempo de cola del Nodo 4 >> 1.02
Tiempo de cola del Nodo 5 >> 0.67
Tiempo de cola del Nodo 6 >> 0.0
Tiempo de cola del Nodo 7 >> 0.0
Tiempo de cola del Nodo 8 >> 0.93
Tiempo de cola del Nodo 9 >> 0.0002
Tiempo de cola del Nodo 10 >> 0.003
File Size: 15770583 bits
```

Corrida:

```
RECORRIDO MENOR => 7 - 9 - 1 - 2 - 6 - --> Lat:0.1502seg. ¦ Tiempo Transf.:473.7308seg. ¦ Total paq divi
didos: 18928
7 - 8 - 5 - 6 - --> Lat:1.6seg. ¦ Total paq divididos: 18435
7 - 9 - 10 - 8 - 5 - 6 - --> Lat:1.6032000000000002seg. ¦ Total paq divididos: 28391
7 - 9 - 1 - 3 - 4 - 5 - 6 - --> Lat:1.7942seg. ¦ Total paq divididos: 69394
7 - 8 - 5 - 4 - 3 - 1 - 2 - 6 - --> Lat:2.824seg. ¦ Total paq divididos: 112647
```

Resultado 4:



Sea el nodo inicial 3 y el nodo final 8. Tamaño de entrada de archivo: 955 KB

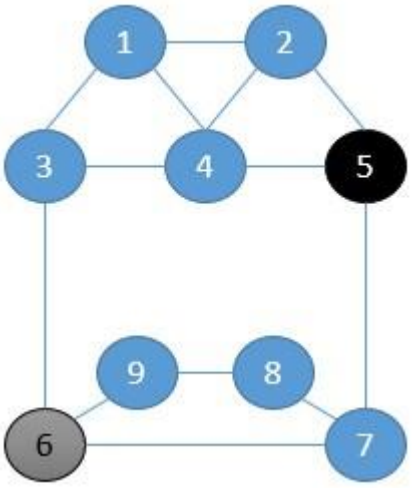
Los datos del grafo:

```
       Nodos: 9        Enlaces: 13
       Nodo Origen: 3  Nodo Destino: 8
(1,2)  |Length:150 m.| |TTT:1250 Mbps| |Datos Control:200| |Datos Usuario:12000|
(1,4)  |Length:150 m.| |TTT:2000 Mbps| |Datos Control:200| |Datos Usuario:3000|
(1,5)  |Length:150 m.| |TTT:410 Mbps| |Datos Control:200| |Datos Usuario:6000|
(2,3)  |Length:150 m.| |TTT:700 Mbps| |Datos Control:200| |Datos Usuario:4000|
(2,5)  |Length:150 m.| |TTT:1200 Mbps| |Datos Control:200| |Datos Usuario:12000|
(3,5)  |Length:150 m.| |TTT:1380 Mbps| |Datos Control:200| |Datos Usuario:14000|
(4,7)  |Length:150 m.| |TTT:1221 Mbps| |Datos Control:200| |Datos Usuario:25000|
(5,6)  |Length:150 m.| |TTT:2100 Mbps| |Datos Control:200| |Datos Usuario:2000|
(5,7)  |Length:150 m.| |TTT:200 Mbps| |Datos Control:600| |Datos Usuario:15000|
(6,7)  |Length:150 m.| |TTT:1000 Mbps| |Datos Control:700| |Datos Usuario:2000|
(6,9)  |Length:150 m.| |TTT:1800 Mbps| |Datos Control:200| |Datos Usuario:1000|
(8,7)  |Length:150 m.| |TTT:1100 Mbps| |Datos Control:200| |Datos Usuario:900|
(8,9)  |Length:150 m.| |TTT:900 Mbps| |Datos Control:600| |Datos Usuario:12000|
Tiempo de cola del Nodo 1 >> 0.5
Tiempo de cola del Nodo 2 >> 0.2
Tiempo de cola del Nodo 3 >> 0.0
Tiempo de cola del Nodo 4 >> 0.02
Tiempo de cola del Nodo 5 >> 0.07
Tiempo de cola del Nodo 6 >> 0.34
Tiempo de cola del Nodo 7 >> 1.03
Tiempo de cola del Nodo 8 >> 0.0
Tiempo de cola del Nodo 9 >> 0.02
File Size: 7823360 bits
```

Corrida:

```
RECORRIDO MENOR => 3 - 5 - 6 - 9 - 8 - --> Lat:0.430000000000000005seg. | Tiempo Transf.:239.940000000000
03seg. | Total paq divididos: 20119
3 - 5 - 7 - 8 - --> Lat:1.1seg. | Total paq divididos: 10059
3 - 2 - 5 - 7 - 8 - --> Lat:1.3seg. | Total paq divididos: 15647
3 - 5 - 6 - 7 - 8 - --> Lat:1.44seg. | Total paq divididos: 20118
3 - 2 - 1 - 4 - 7 - 8 - --> Lat:1.75seg. | Total paq divididos: 23471
3 - 2 - 1 - 5 - 7 - 8 - --> Lat:1.8seg. | Total paq divididos: 17603
3 - 2 - 5 - 6 - 7 - 8 - --> Lat:1.6400000000000001seg. | Total paq divididos: 23471
3 - 2 - 5 - 6 - 9 - 8 - --> Lat:0.6300000000000001seg. | Total paq divididos: 23472
3 - 5 - 1 - 4 - 7 - 8 - --> Lat:1.62seg. | Total paq divididos: 18440
3 - 5 - 7 - 6 - 9 - 8 - --> Lat:1.4600000000000002seg. | Total paq divididos: 20678
3 - 2 - 1 - 5 - 6 - 7 - 8 - --> Lat:2.14seg. | Total paq divididos: 25427
3 - 2 - 1 - 5 - 6 - 9 - 8 - --> Lat:1.1300000000000001seg. | Total paq divididos: 25428
3 - 2 - 5 - 1 - 4 - 7 - 8 - --> Lat:1.82seg. | Total paq divididos: 25427
3 - 2 - 5 - 7 - 6 - 9 - 8 - --> Lat:1.6600000000000001seg. | Total paq divididos: 25428
3 - 5 - 2 - 1 - 4 - 7 - 8 - --> Lat:1.82seg. | Total paq divididos: 18998
3 - 2 - 1 - 4 - 7 - 6 - 9 - 8 - --> Lat:2.11seg. | Total paq divididos: 33252
3 - 2 - 1 - 5 - 7 - 6 - 9 - 8 - --> Lat:2.16seg. | Total paq divididos: 27384
3 - 5 - 1 - 4 - 7 - 6 - 9 - 8 - --> Lat:1.9800000000000002seg. | Total paq divididos: 28501
3 - 2 - 1 - 4 - 7 - 5 - 6 - 9 - 8 - --> Lat:2.18seg. | Total paq divididos: 37164
3 - 2 - 5 - 1 - 4 - 7 - 6 - 9 - 8 - --> Lat:2.18seg. | Total paq divididos: 35208
3 - 5 - 2 - 1 - 4 - 7 - 6 - 9 - 8 - --> Lat:2.18seg. | Total paq divididos: 29059
```

Resultado 5:



Sea el nodo origen: 6 y el nodo destino 5. Tamaño de entrada de archivo:
Los datos de este grafo:

```
        Nodos: 9        Enlaces: 13
        Nodo Origen: 6  Nodo Destino: 5
(1,2)  |Length:140 m.| |TTT:700 Mbps| |Datos Control:200| |Datos Usuario:2300|
(1,3)  |Length:140 m.| |TTT:400 Mbps| |Datos Control:200| |Datos Usuario:4500|
(1,4)  |Length:140 m.| |TTT:750 Mbps| |Datos Control:200| |Datos Usuario:1000|
(2,4)  |Length:140 m.| |TTT:990 Mbps| |Datos Control:200| |Datos Usuario:2000|
(2,5)  |Length:140 m.| |TTT:1000 Mbps| |Datos Control:200| |Datos Usuario:1100|
(3,4)  |Length:140 m.| |TTT:3200 Mbps| |Datos Control:200| |Datos Usuario:4100|
(3,6)  |Length:300 m.| |TTT:300 Mbps| |Datos Control:200| |Datos Usuario:3000|
(4,5)  |Length:140 m.| |TTT:1000 Mbps| |Datos Control:200| |Datos Usuario:3300|
(5,7)  |Length:300 m.| |TTT:2100 Mbps| |Datos Control:200| |Datos Usuario:1300|
(6,9)  |Length:80 m.| |TTT:1000 Mbps| |Datos Control:200| |Datos Usuario:9000|
(6,7)  |Length:300 m.| |TTT:1400 Mbps| |Datos Control:200| |Datos Usuario:9300|
(8,9)  |Length:80 m.| |TTT:2400 Mbps| |Datos Control:200| |Datos Usuario:10000|
(8,7)  |Length:80 m.| |TTT:900 Mbps| |Datos Control:200| |Datos Usuario:12300|
Tiempo de cola del Nodo 1 >> 0.12
Tiempo de cola del Nodo 2 >> 0.02
Tiempo de cola del Nodo 3 >> 0.102
Tiempo de cola del Nodo 4 >> 0.02
Tiempo de cola del Nodo 5 >> 0.0
Tiempo de cola del Nodo 6 >> 0.0
Tiempo de cola del Nodo 7 >> 0.001
Tiempo de cola del Nodo 8 >> 0.02
Tiempo de cola del Nodo 9 >> 1.21
File Size: 29360128 bits
```

Corrida:

```
RECORRIDO MENOR => 6 - 7 - 5 - --> Lat:0.001seg. | Tiempo Transf.:3.157seg. | Total paq divididos: 28415

6 - 3 - 4 - 5 - --> Lat:0.122seg. | Total paq divididos: 29361
6 - 3 - 1 - 2 - 5 - --> Lat:0.24199999999999997seg. | Total paq divididos: 78293
6 - 3 - 1 - 4 - 5 - --> Lat:0.24199999999999997seg. | Total paq divididos: 78296
6 - 3 - 4 - 2 - 5 - --> Lat:0.142seg. | Total paq divididos: 68509
6 - 9 - 8 - 7 - 5 - --> Lat:1.2309999999999999seg. | Total paq divididos: 32625
6 - 3 - 1 - 2 - 4 - 5 - --> Lat:0.26199999999999996seg. | Total paq divididos: 97867
6 - 3 - 1 - 4 - 2 - 5 - --> Lat:0.26199999999999996seg. | Total paq divididos: 107657
6 - 3 - 4 - 1 - 2 - 5 - --> Lat:0.262seg. | Total paq divididos: 107657
```

5. CONCLUSIÓN

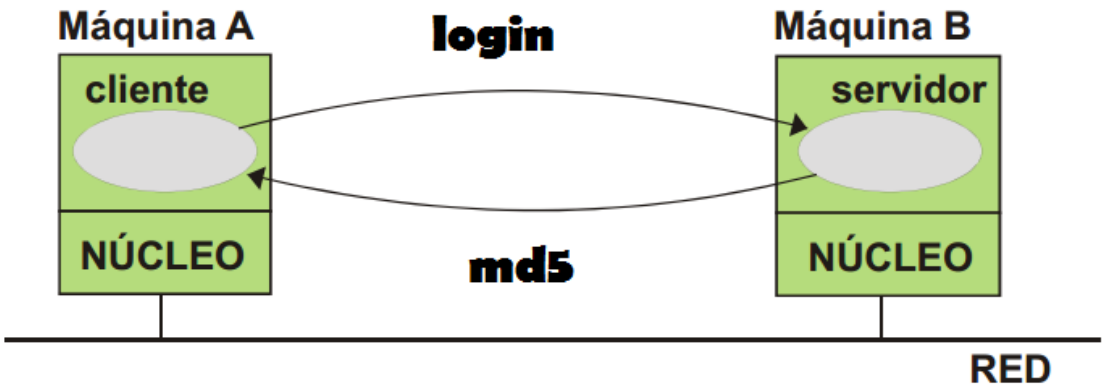
Así pues, podemos ver que el entorno del programa nos permita tener una herramienta más que realice cálculos de ciertas interfaces y sabiendo que este proyecto ha llevado a que sea útil tanto en la vida cotidiana como en la laboral, por eso el programa que calcula la latencia y obtiene el menor para que a la vez sepamos cuanto tiempo se tardó en transferir un archivo, eso es una interfaz amigable que se ha considerado hasta ahora. Siento que este proyecto ha cumplido con los requisitos; sin embargo, aún queda más por ver.

Reporte técnico "Cliente/Servidor con encriptación MD5"

Introducción

En nuestro proyecto, se realiza la implementación de un cliente/servidor con encriptación con el algoritmo MD5.

- Usuario envía un ID (login) de su usuario al servidor
- El servidor genera y envía un mensaje aleatorio al usuario
- El servidor y el sistema del usuario hacen un cálculo con MD5 del mensaje aleatorio y del password del usuario
- El sistema de usuario envía el mensaje con la encriptación en MD5, y el servidor verifica en base de datos que este coincida con el MD5 que ha obtenido.
- Si es el caso, se válida su contraseña



Algoritmo MD5

El algoritmo MD5 recibe un mensaje m de cualquier tamaño, y de éste siempre obtiene una serie de 128 bits, que es única para cualquier mensaje. El algoritmo consta de 5 pasos, los cuales se explican a continuación:

1. El mensaje m es rellenado con n bits, de tal manera que el nuevo mensaje m', tenga una longitud menor a 64 bits comparado con un múltiplo de 512 bits. Para ello, el primer bit de relleno es un "1" y los restantes - 1 bits son "0".
2. La nueva longitud tras añadir los bits de relleno es almacenada en una representación de 64 bits y añadida al final del mensaje en forma de dos palabras de 32 bits, yendo en primer lugar la que contiene los bits menos significativos. Si la longitud del mensaje fuera mayor que 2

REDES DE COMPUTADORAS

Javier Torres Timal

REPORTE DE PRIMER EXAMEN PARCIAL

INTRODUCCION

Una red de computadoras, es una colección interconectada de computadoras autónomas. Dos computadoras se consideran interconectadas cuando son capaces de intercambiar información

Estos dispositivos conectados se denominan hosts o sistemas terminales.

Una red en general es un sistema de transmisión de datos que permite el intercambio de información entre dispositivos electrónicos (computadores) que

toman el nombre de HOST. El HOST es todo dispositivo electrónico (computador)

conectado a una red.

En definición más específica, una red es un conjunto de computadoras que van a compartir archivos (carpetas, datos, imágenes, audio, video, etc.) o recursos (disco duro, lectora, disketera, monitor, impresora, fotocopiadora, web cam, etc.), éstas computadoras pueden estar interconectadas por un medio físico o inalámbrico.

La transmisión de datos se produce a través de un medio de transmisión o combinación de distintos medios: cables de par trenzado, cables coaxiales, cables de fibra óptica, tecnología inalámbrica, enlace bluetooth, enlace infrarrojo, enlace vía satélite.

Cuando se intercambian datos los paquetes se inician en un host (el origen), atraviesan una serie de routers y terminan su viaje en otro host (el destino). Cuando un paquete viaja de un nodo (host o router) al siguiente nodo (host o router) a lo largo de una ruta, el paquete sufre varios tipos de retardo en cada uno de los nodos de dicha ruta. A esto tipo retardos en conjunto se le denomina Latencia la cual depende de los siguientes factores:

- Tiempo de Propagacion
- Tiempo de Transmision
- Tiempo de Cola

Dónde:

Tiempo de propagación = distancia a recorrer/velocidad de la luz

Tiempo de transmisión = tamaño del paquete / tasa de transferencia teórica

Para calcular la latencia de un punto A a un punto B se aplica la siguiente Formula

$$L = TP + TT + TC$$

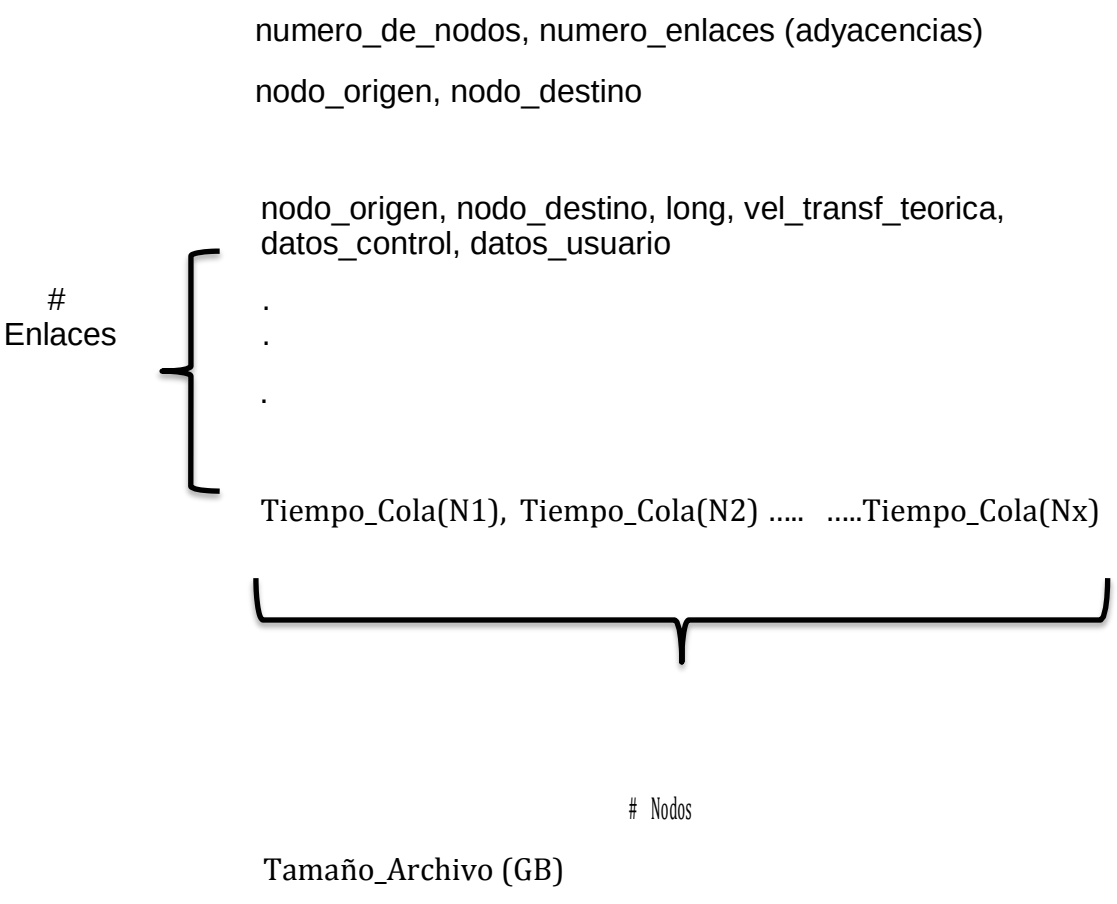
DESARROLLO

El objetivo de este proyecto consiste en realizar un programa el cual, dado una

Red de Computadoras, proporcione todos los caminos posibles para transferir un archivo de un punto A a un punto B y de todos estos caminos seleccione el que menos tiempo se tarde en transferir. Esta red únicamente tendrá 2 dispositivos (computadoras) y el resto de nodos se tomaran como Routers con protocolos diferentes.

La Red de Computadoras se proporcionara a través de un archivo .txt para que asi sea mucho mas rápido tomar los datos de entrada necesarios.

El formato del archivo .txt será el siguiente



Todos los datos son en formato Numerico, el dato *long* se proporcionara en metros (m), el dato *vel_transf_teorica* se dará en Megabits por Segundo (Mbps), los datos: *datos_control y datos_usuario* se proporcionaran en Bytes (bytes), los *Tiempo_Cola* correspondientes se darán en segundos (S) y por ultimo, en el ultimo renglón del archivo .txt se indicara el tamaño del archivo a transferir en Gigabytes (GB). A excepción del *nodo_origen* y *nodo_destino*, los demás nodos serán considerados Routers.

El contenido del archivo .txt serán los únicos datos de entrada y a partir de esto el programa deberá trabajar para que al final muestre el mejor camino, la lantencia de ese camino y el tiempo que se tardara en transferir el archivo del nodo inicial al nodo destino.

¿Qué lenguaje se utilizara para la implementación?

El programa se realizará en lenguaje JAVA ya que al ser Orientado a Objetos facilitará el trabajo considerablemente.

¿Qué estructuras de datos se ocupara?

- Grafo.- Se ira construyendo la Red de Computadoras como un grafo.
- Matriz de adyacencia.- Para saber como están enlazados los routers y dispositivos
- Lista.- Para manipular datos así como también para guardar los caminos posibles

¿Qué metodología se seguirá para realiza el programa?

Se seguirá la filosofía de divide y vencerás para hacer el programa. Se dividirá el problema completo en pequeños problemas y a su vez se ira resolviendo cada uno de estos subproblemas para después unirlo y resolver el problema completo.
Se dividirá en 5 partes las cuales son:

1. Se extraen los datos del archivo .txt para ir formando el Grafo así como también la matriz de adyacencia.
2. Teniendo el grafo construido se obtendrán todos los caminos posibles para transferir el archivo del nodo inicial al nodo destino.
3. Se obtendrá la cantidad de paquetes que se estarán enviando de punto a punto a través de cada camino posible
4. Se ira tomando cada camino y se ira calculando la latencia hasta encontrar el camino con la menor latencia.
5. Por ultimo, ya teniendo el mejor camino se estimara el tiempo que se tardaría en transferir un archivo de un tamaño ya especificado a través de dicho camino seleccionado.

PRIMERA PARTE

Se asume que la primera parte que se trata de extraer los datos del archivo .txt no tiene mayor complejidad, puesto que aparte de ser sencillo, si no se entendiera existen infinidad de textos en el cual se explica el tema de extraer datos de un archivo por lo cual se omitirá la explicación de esta parte del programa.

Para formar el grafo y la matriz de adyacencia se contara con dos tipos de Nodos:

1. NodoCab.- En el cual se tendrá únicamente el valor del nodo, tiempo de cola y un apuntador a una lista de adyacencia
2. Nodo.- Que en conjunto formaran la lista de adyacencia de cada nodo cabecera y cada nodo almacenara el resto de los datos(valor del nodo, distancia, velocidad, etc).

Con lo mencionado anteriormente y sabiendo el formato del archivo .txt se estará formando el grafo y la matriz de adyacencia utilizando un par de ciclos de

iteración.

SEGUNDA PARTE

Ya que se tiene el grafo armado se recorrerá usando la idea de Busqueda primero en profundidad (BPP) asi como también algunos aspecto del algoritmo de Dijkstra para obtener todos los caminos posibles. La idea es que se empezara visitando el nodo inicial y si aun no esta visitado se insertara en un lista que ira almacenando todos los caminos posibles aquí es donde se toma la idea de Dijkstra, después de ser insertado se visitara cada nodo de su lista de adyacencia y se seguirá el

mismo procedimiento hasta que se llegue al nodo destino, en esta parte se toma la idea de búsqueda primero en profundidad (BPP) . Una vez que se haya llegado al nodo destino significa que ya se tiene un camino completo, se guarda y se borrar el ultimo nodo para que continúe con el resto de nodos adyacentes y así vaya sacando de igual manera los demás caminos, en esta ultima parte de borrar el ultimo dato, la pertenece al algoritmo de Búsqueda en Amplitud, sin embargo seria la única idea que se tomo de dicho algoritmo . Por la naturalidad del problema se implementará de manera recursiva, a continuación se muestra el fragmento de código que realizará la tarea de obtener todos los caminos:

```
aux=r.inigrafo; // SE APUNTA AL INICIO DEL GRAFO

            while(aux.vertice!=nodo_ini)

            {

                aux=aux.sig;        //  RECORREMOS EL NODO HASTA LLEGAR AL QUE ES EL NODO

INICIAL

            }

            aux2=aux.listady;       //  APUNTAMOS A SU LISTA DE ADYACENCIA
            auxiliar.inserta(aux.vertice);// INSERTA CABEZA EN LA LISTA
            busca_caminos(aux.vertice,nodo_fin);  // FUNCION PARA BUSCAR TODOS LOS CAMINOS


public static void busca_caminos(int vertice, int nodofinal)

    {

        // RECIBE VERTICE ACTUAL Y NODO FINAL
        NodoCab aux;

        aux=r.inigrafo; Nodo
        auxiliar_ady;
        while(aux.vertice!=vertice)

        {

            aux=aux.sig;// SE RECORRE NODO HASTA LLEGAR A VERTICE

        }

        auxiliar_ady=aux.listady;//APUNTAMOS A LISTA DE ADYACENCIA

        while(auxiliar_ady!=null)

        {

            if(auxiliar.verifica(auxiliar_ady.vertice)==false) // SE VERIFICA SI EL NODO
ESTA O NO EN LA LISTA

            {

                auxiliar.inserta(auxiliar_ady.vertice); // SI NO ESTA VISITADO SE INSERTA

EN LA LISTA

RECURSIVIDAD    if (auxiliar_ady.vertice==nodofinal) // CONDICION DE PARO DE LA


                {
GUARDA LA LISTA
                    dijkstra.agrega_lista(auxiliar); // CUANDO YA LLEGO AL NODO FINAL SE


                    dijkstra.inserta(666); // SE INSERTA UNA BANDERA PARA SABER QUE A

PARTIR DE ESTE PUNTO EMPIEZA UNA NUEVA LISTA
```

```
                    auxiliar.borrafin(); // SE BORRA EL ULTIMO NODO PARA SEGUIR CON LAS
ADYACENCIAS FALTANTES

                }

                else

                {

                    busca_caminos(auxiliar_ady.vertice,nodofinal); // SI AUN NO SE LLEGA AL
NODO FINAL SE LLAMA RECURSIVAMENTE LA FUNCION AHORA CON LA ADYACENCIA COMO NODO INICIAL

                }

            }

            auxiliar_ady=auxiliar_ady.liga;

        }

        auxiliar.borrafin(); // SE BORRA EL ULTIMO NODO PARA SEGUIR CON LAS ADYACENCIAS
FALTANTES



    }
```

TERCERA PARTE

Para la división de paquetes se ira tomando todos los caminos posibles uno por uno y se ira calculando cuantos paquetes iran viajando de punto a punto o de nodo a nodo. Como por cada camino habrá una lista en especial, será mucho mas fácil manipular la misma. La idea es que se recorra toda la lista a través de dos nodos de referencia, uno que apunte al lugar actual y el otro siempre apuntando a una posición anterior al actual, teniendo esto, únicamente se dividirán los paquetes de la siguiente manera:

- Al principio siempre será un paquete el que viajara, después de eso, se iran dividiendo según sea el caso.
- Se manejara un acarreo que es la cantidad menor de datos en bytes que iran circulando por la red (cantidad de datos menor sin contar el residuo de la división)
- Si el tamaño de datos del nodo anterior es menor o igual al del nodo actual simplemente siguen circulando la misma cantidad de paquetes que el anterior, no hay necesidad de hacer divisiones de paquetes.
- Si el tamaño de datos del nodo anterior es mayor al del nodo actual entonces se divide los datos del nodo anterior entre los datos del nodo actual y el resultado entero de esa división, multiplicado por la cantidad de paquetes que iban circulando en el anterior será la nueva cantidad de paquetes que ahora iran circulando, el residuo de la división significa que va otra paquete mas pequeño y de igual manera se contabiliza en la cantidad de paquetes total.
- Se ira realizando el mismo procedimiento hasta llegar al final de la lista y hasta agotar todas las listas(caminos).
- El resultado de la división de paquetes se ira almacenando en una nueva lista, la cual tendrá el vértice actual, el vértice siguiente, y la cantidad de paquetes que van circulando de punto a punto. Con esta será mas fácil las operaciones siguientes.

A continuación se muestra el fragmento de código que realizara la tarea de división de paquetes:

```
                nodito_aux=lista_paquetes.iniLista; //APUNTAMOS AL INICIO DE LA LISTA DE
TODOS LOS CAMINOS

                noditoant_aux = nodito_aux; //APUNTAMOS AL NODO ANTERIOR DEL NODO ACTUAL

                nodito_aux.num_paquetes = 1; //COMO ES EL INICIO EL NUM DE PAQUETES QUE
CIRCULARA ES 1

                nodito_aux.num_paquetes_total = 1;

                nodito_aux.extra=0; //EXTRA QUIERE DECIR EL RESIDUO DE LA DIVISON DE DU DEL
NODO ACTUAL ENTRE EL NODO ANTERIOR
```

```
                acarreo = nodito_aux.DU; // ACARREO ES EL TAMAÑO DE DU QUE VA CIRCULANDO
POR LA RED
                nodito_aux=nodito_aux.liga; // APUNTAMOS AL SIG NODO
                while(nodito_aux!=null) //MIENTRAS NO SE LLEGUE AL FINAL DE LA LISTA DE
CAMINOS PSOSIBLES

                {
                    if (noditoant_aux.DU <= nodito_aux.DU)
                    {
                        nodito_aux.num_paquetes= 1 * noditoant_aux.num_paquetes;

                        nodito_aux.num_paquetes_total= 1 *
```

```
noditoant_aux.num_paquetes_total;
                        if ((acarreo<nodito_aux.DU) &&(noditoant_aux.DU <= nodito_aux.DU))
                        {
                            acarreo = acarreo;
                        }
                    }
                    else
                    {
                        nodito_aux.num_paquetes = noditoant_aux.DU/nodito_aux.DU;
                        if((noditoant_aux.DU%nodito_aux.DU)!=0)
                        {
                            nodito_aux.extra = 1;
                        }
                        else
                            nodito_aux.extra = 0;
                        if(noditoant_aux.extra == 1){
                        nodito_aux.num_paquetes_total = (nodito_aux.num_paquetes +
nodito_aux.extra)* noditoant_aux.num_paquetes + 1;
                        }
                        else
                            nodito_aux.num_paquetes_total = (nodito_aux.num_paquetes +
nodito_aux.extra)* noditoant_aux.num_paquetes;
                        if ((acarreo<nodito_aux.DU))
                        {
                            acarreo = acarreo;
                            nodito_aux.extra = 0;
                        }
                        else
                            acarreo = nodito_aux.DU;
                    } noditoant_aux=nodito_aux;
                    nodito_aux=nodito_aux.liga;
                }
```

CUARTA PARTE

Como ya se tiene la división de paquetes en una lista aparte, únicamente analizamos cada camino y aplicamos la formula para calcular la latencia. A continuación se

muestra el fragmento de código que realizara la tarea de calcular la latencia de cada camino:

```
latencia_temp = lista_paquetes.latencia(r);

                if(latencia_temp<= latencia_final)

                {

                    lista_final = new ListaD(); latencia_final
                    = latencia_temp;
                    lista_final.agrega_lista2(lista_paquetes);

                }
```

```
float latencia(Grafo grafo)

    {

        float TP;
        float TT;
        float TC1;

        float latencia1=0;

        Nodo2 auxiliar= iniLista;
        NodoCab aux = grafo.inigrafo;
        while(auxiliar!=null)

        {

            aux = grafo.inigrafo;

            while(aux.vertice!=auxiliar.valor_sig)

            {

                aux=aux.sig;

            }
```

```
            TC1=aux.TC;
            TP=(((float)auxiliar.distancia)/1000)/300000
            ; TT =(((float)auxiliar.DC +
(float)auxiliar.DU)*8)/(((float)auxiliar.velocidad)*1000000);
            latencia1= latencia1 + ((float)auxiliar.num_paquetes_total * (TP + TT + TC1));
            auxiliar=auxiliar.liga;
        }
        return latencia1;
    }
```

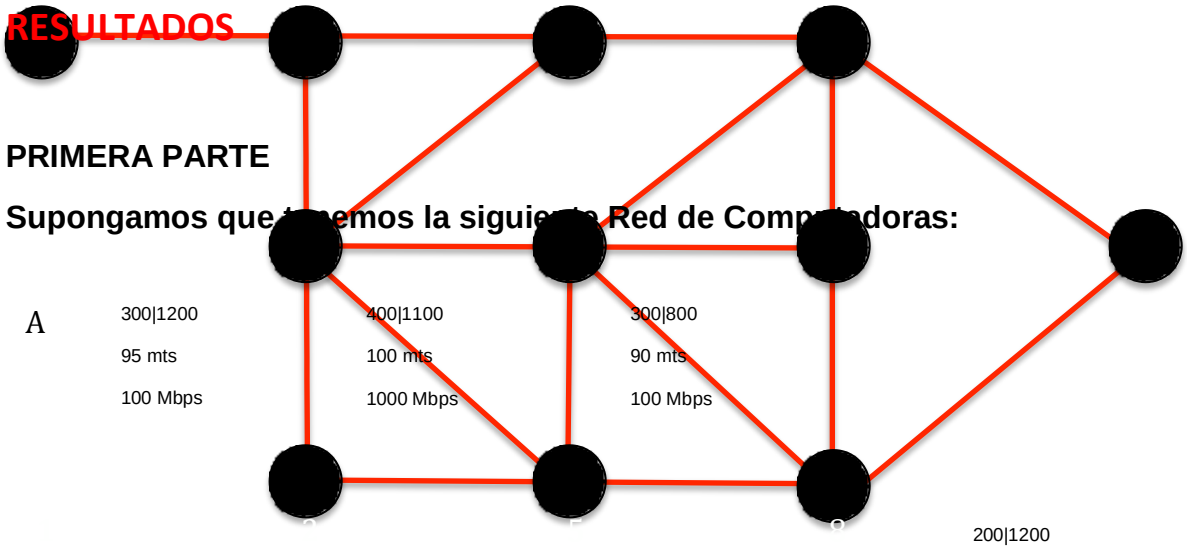
QUINTA PARTE

Ya que se tiene el camino con la latencia menor únicamente tomamos el tamaño del archivo a transferir, lo dividimos entre los datos de usuario inicial y el resultado lo multiplicamos por la latencia menor y ya tenemos el resultado esperado.

RESULTADOS

PRIMERA PARTE

Supongamos que tenemos la siguiente Red de Computadoras:



A

300|1200
95 mts
100 Mbps

400|1100
100 mts
1000 Mbps

300|800
90 mts
100 Mbps

200|1200
150 mts
100 Mbps

200|1300
90 mts
10 Mbps

300|1200
150 mts
150 Mbps

400|700
95 mts
500 Mbps

500|1000
130 mts
1000 Mbps

00|1800
5 mts
00 Mbps

80 mts
200 Mbps

B

400|1100
100 mts
100 Mbps

500|800

500|1000

200|1300

300|1100

90 mts

1000 Mbps

160 mts

10 Mbps

50 mts

1500 Mbps

100 mts

10 Mbps

200|1000

120 mts

1000 Mbps

300|1100

100 mts

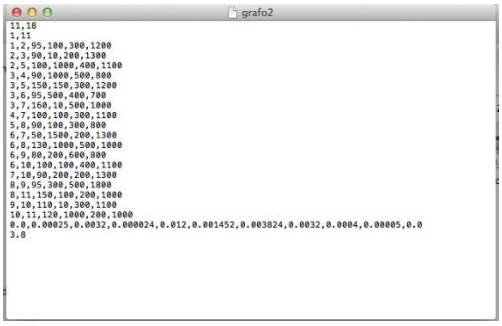
100 Mbps

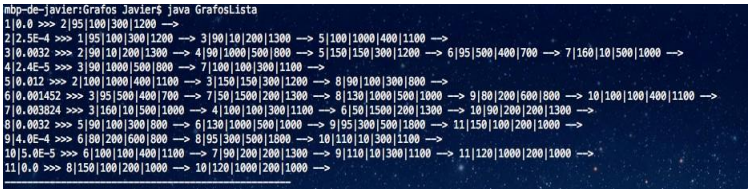
200|1300

90 mts

200 Mbps

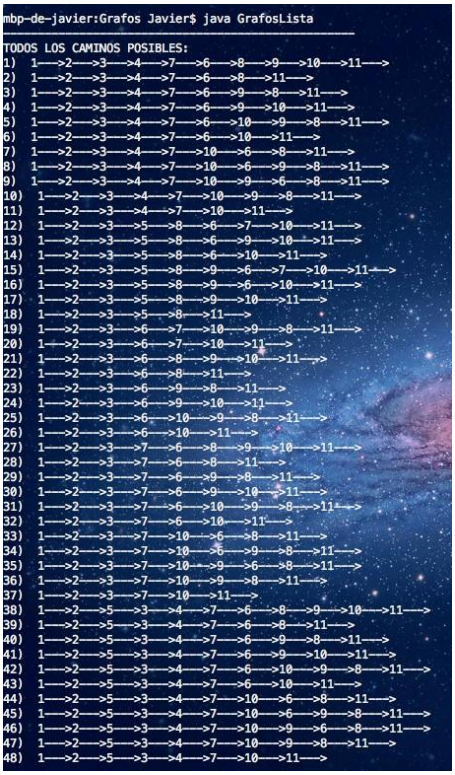
Como resultado esperaríamos el grafo armado asi como también la matriz de adyacencia:

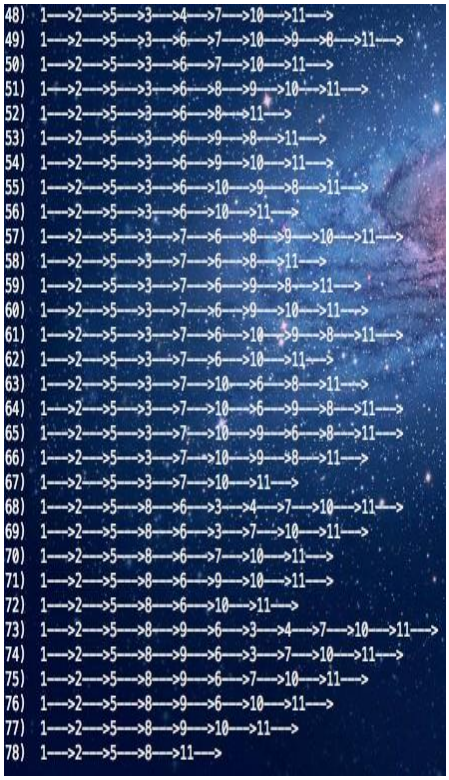




SEGUNDA PARTE

En esta parte se esperaría que el programa mostrara todos los caminos posibles para llegar de A a B. Haciendo una rigurosa prueba de escritorio se encontró que en total existen 78 caminos distintos para llegar de A a B por que lo el programa debería encontrar la misma cantidad de caminos:





TERCERA PARTE

En esta parte se debería de formar una lista diferente por cada camino posible que existe, en la cual, almacenaría la cantidad de paquetes que iran circulando de punto a punto:

CUARTA PARTE

Teniendo la lista de paquetes aparte, únicamente aplicamos la formula para calcular la latencia de punto a punto obviamente tomando en cuenta la división de paquetes. De igual manera se realizo una rigurosa prueba de escritorio, y tomando como un ejemplo el primer camino posible, se calculo que la latencia de dicho camino fue:

Camino: 1--->2--->3--->4--->7--->6--->8--->9--->10--->11

1 a 2 = 1 paquete

2 a 3 = 1 paquete

3 a 4 = 2 paquetes

4 a 7 = 2 paquetes

7 a 6 = 2 paquetes

6 a 8 = 2 paquetes

8 a 9 = 2 paquetes

9 a 10 = 2 paquetes

10 a 11 = 2 paquetes

Latencia = 0.02225 segundos, en el programa debería salir el mismo resultado para el mismo camino:

QUINTA PARTE

Cuando ya se tiene el mejor camino únicamente verificamos que tiempo tardaría en transferir un archivo de un determinado tamaño. De igual manera se realizo un prueba de escritorio, y con varias pruebas se identifico que el mejor camino posible es:

1--->2--->3--->6--->10--->11

Con una latencia de:

Latencia = 0.008071117 seg(s)

Y si consideramos un archivo de 3.8 GB tardaría un total de:

3.8 GB = 4080218931.2 bytes

(4080218931.2 bytes) / 1(200 bytes) = 3400182.44266666666667 paquetes

(3400182.44266666666667) * (0.008071117 segs) = 27443.45052577792 segs

Por lo cual, transferir un archivo de 3.8 GB por el mejor camino tardaría en total:

27443.45052577792 segs

o 457.39084209629867 mins o
7.62318070160498 hrs

Se esperaría que en el programa apareciera el mismo resultado:

CONCLUSIÓN

La latencia es un factor importante al momento de transferir archivos de un punto a otro. En esta practica se logro implementar un programa que calculara el mejor camino con la menor latencia, funcionaria con cualquier conjunto de datos de entrada de tal manera que podría utilizarse para problemas aplicados de la vida real lo cual deja una

agradable satisfacción. Se aprendió como es que funcionan las redes de computadoras, el internet y otras cosas que usamos del diario así como también se entendió que existen diversos protocolos para intercambiar datos o información.

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Modelos de Redes

Prof. Dr. Iván Olmos Pineda

Saúl Pérez Vásquez

Trabajo:

Proyecto

Introducción

A lo largo de la historia del manejo de datos, ha sido muy relevante la capacidad de poder esconder información, ya sea por ser de modo personal o sea por ser información exclusiva y clasificada. Durante este tiempo han surgido diversos mecanismos para lograr pasar desapercibidos estos datos mientras se envían, de tal forma que no puedan ser interceptados o decifrados con facilidad.

Una buena encriptación ha de proporcionar seguridad para que la información que deseemos proteger no esté suceptible a ser descubierta. Recordemos que esto tuvo sus inicios en la guerra y con uno de los padres de la computación: Alan Turing. En nuestros días estamos cada vez más suceptibles a los ataques cibernéticos por lo cual es más importante proteger nuestra información personal. Una técnica para encriptación muy eficiente es la MD5 la cual explicaremos más adelante.

Desarrollo

Para este proyecto implementaremos un servicio de chat, para el cuál haya que proteger los datos del usuario, en específico su contraseña. Se utilizará una función de encriptación mediante MD5, garantizando así una protección de la contraseña del usuario.

¿En qué consiste MD5?

Este algoritmo fue desarrollado por Ronald Rivest en 1995 y está basado en dos algoritmos anteriores MD2 y MD4. Todos estos protocolos producen un número de 128 bits a partir de un texto de cualquier longitud.

MD4 fue desarrollado para mejorar el rendimiento de MD2 , sin embargo, varios problemas fueron detectados y en 1996 fueron publicados elementos que hacen hoy en día inservible el algoritmo. MD5 sustituyó a MD4 y aunque no tiene el rendimiento de su antecesor, hasta

el momento no han sido publicados elementos que comprometan su integridad y funcionamiento.

MD5 comienza rellenando el mensaje a una longitud congruente en módulo 448 mod 512. Es decir la longitud del mensaje es 64 bits menos que un entero

múltiplo de 512. El relleno consiste en un bit en 1 seguido por cuentos bits en 0 sean necesarios. La longitud original del mensaje es almacenada en los últimos 64 bits del relleno.

ExplicacióndetalladadelalgoritmoMD5

•Terminologías y notaciones

```
   El símbolo "+" significa suma de
palabras.
   X <<< s se interpreta por una rotación
de bits a la izquierda sobre 'X', 's'
posiciones
   not(x) se entiende como el complemento
de x
```

Empezamos suponiendo que tenemos un mensaje de 'b' bits de entrada, y que nos gustaría encontrar su resumen. Aquí 'b' es un valor arbitrario entero no

negativo, pero puede ser cero, no tiene por qué ser múltiplo de ocho, y puede ser muy largo. Imaginemos los bits del mensaje escritos así:

```
m0 m1 ... m{b-1}
```

Los siguientes cinco pasos son efectuados para calcular el resumen del mensaje.

Paso 1. Adición de bits

El mensaje será extendido hasta que su longitud en bits sea congruente con 448, módulo 512. Esto es, si se le resta 448 a la longitud del mensaje tras este paso, se obtiene un múltiplo de 512. Esta extensión se realiza siempre, incluso si la longitud del mensaje es ya congruente con 448, módulo 512.

La extensión se realiza como sigue: un solo bit "1" se añade al mensaje, y después se añaden bits "0" hasta que la longitud en bits del mensaje extendido se haga congruente con 448, módulo 512. En todos los mensajes se añade al menos un bit y como máximo 512.

Paso 2. Longitud del mensaje

Un entero de 64 bits que represente la longitud 'b' del mensaje (longitud antes de añadir los bits) se concatena al resultado del paso anterior. En el supuesto no deseado de que 'b' sea mayor que 2^64, entonces sólo los 64 bits de menor peso de 'b' se usarán.

En este punto el mensaje resultante (después de rellenar con los bits y con 'b') se tiene una longitud que

es un múltiplo exacto de 512 bits. A su vez, la longitud del mensaje es múltiplo de 16 palabras (32 bits por palabra). Con M[0 ... N-1] denotaremos las palabras del mensaje resultante, donde N es múltiplo de 16.

Paso 3. Inicializar el búfer MD

Un búfer de cuatro palabras (A, B, C, D) se usa para calcular el resumen del mensaje. Aquí cada una de las letras A, B, C, D representa un registro de 32 bits. Estos registros se inicializan con los siguientes valores hexadecimales, los bytes de menor peso primero:

```
palabra A: 01 23 45 67
palabra B: 89 ab cd ef
palabra C: fe dc ba 98
palabra D: 76 54 32 10
```

Paso 4. Procesado del mensaje en bloques de 16 palabras

Primero definimos cuatro funciones auxiliares que toman como entrada tres palabras de 32 bits y su salida es una palabra de 32 bits.

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

Los operadores $\oplus, \wedge, \vee, \neg$ son las funciones XOR, AND, OR y NOT respectivamente.

En cada posición de cada bit X actúa como un condicional: si X, entonces Z sino Y. La función Z podría haber sido definida usando + en lugar de v ya que XY y not(x) Z nunca tendrán unos ('1') en la misma posición de bit. Es interesante resaltar que si los bits de X, Y y Z son independientes y no sesgados, cada uno de los bits de F(X,Y,Z) será independiente y no sesgado.

Las funciones G, H e I son similares a la función F, ya que actúan "bit a bit en paralelo" para producir sus salidas de los bits de X, Y y Z, en la medida que si cada bit correspondiente de X, Y y Z son independientes y no sesgados, entonces cada bit de G(X,Y,Z), H(X,Y,Z) e I(X,Y,Z) serán independientes y no sesgados. Nótese que la función H es la comparación bit a bit "xor" o función "paridad" de sus entradas.

Este paso usa una tabla de 64 elementos T[1 ... 64] construida con la función Seno. Denotaremos por T[i] el elemento i-ésimo de esta tabla, que será igual a la parte

entera del valor absoluto del seno de 'i' 4294967296 veces, donde.....

Código del MD5:

```
 /* Procesar cada bloque de 16 palabras. */
para i = 0 hasta N/16-1 hacer


   /* Copiar el bloque 'i' en X. */
   para j = 0 hasta 15 hacer
     hacer X[j] de M[i*16+j].
   fin para /* del bucle 'j' */


 /* Guardar A como AA, B como BB, C como CC,
y D como DD. */


 /* Ronda 1. */
 /* [abcd k s i] denotarán la operación
      a = b + ((a + F(b, c, d) + X[k] + T[i])
<<< s). */
 /* Hacer las siguientes 16 operaciones. */
```

```
 [ABCD  0  7  1] [DABC  1 12  2] [CDAB  2 17
3] [BCDA  3 22  4]
 [ABCD  4  7  5] [DABC  5 12  6] [CDAB  6 17
7] [BCDA  7 22  8]
```

```
 [ABCD  8  7  9] [DABC  9 12 10] [CDAB 10 17
11] [BCDA 11 22 12]
 [ABCD 12  7 13] [DABC 13 12 14] [CDAB 14 17
15] [BCDA 15 22 16]


 /* [abcd k s i] denotarán la operación
     a = b + ((a + G(b, c, d) + X[k] + T[i])
<<< s). */
```

```
 /* Hacer las siguientes 16 operaciones. */
 [ABCD  1  5 17] [DABC  6  9 18] [CDAB 11 14
19] [BCDA  0 20 20]
 [ABCD  5  5 21] [DABC 10  9 22] [CDAB 15 14
23] [BCDA  4 20 24]
 [ABCD  9  5 25] [DABC 14  9 26] [CDAB  3 14
27] [BCDA  8 20 28]
 [ABCD 13  5 29] [DABC  2  9 30] [CDAB  7 14
31] [BCDA 12 20 32]


 /* Ronda 3. */
 /* [abcd k s t] denotarán la operación
     a = b + ((a + H(b, c, d) + X[k] + T[i])
<<< s). */
 /* Hacer las siguientes 16 operaciones. */
 [ABCD  5  4 33] [DABC  8 11 34] [CDAB 11 16
35] [BCDA 14 23 36]
 [ABCD  1  4 37] [DABC  4 11 38] [CDAB  7 16
39] [BCDA 10 23 40]
 [ABCD 13  4 41] [DABC  0 11 42] [CDAB  3 16
43] [BCDA  6 23 44]
 [ABCD  9  4 45] [DABC 12 11 46] [CDAB 15 16
47] [BCDA  2 23 48]
```

```
 /* Ronda 4. */
 /* [abcd k s t] denotarán la operación
     a = b + ((a + I(b, c, d) + X[k] + T[i])
<<< s). */
 /* Hacer las siguientes 16 operaciones. */
 [ABCD  0  6 49] [DABC  7 10 50] [CDAB 14 15
51] [BCDA  5 21 52]
 [ABCD 12  6 53] [DABC  3 10 54] [CDAB 10 15
55] [BCDA  1 21 56]
```

```
 [ABCD  8  6 57] [DABC 15 10 58] [CDAB  6 15
59] [BCDA 13 21 60]
 [ABCD  4  6 61] [DABC 11 10 62] [CDAB  2 15
63] [BCDA  9 21 64]

 /* Ahora realizar las siguientes sumas.
(Este es el incremento de cada
    uno de los cuatro registros por el valor
que tenían antes de que
    este bloque fuera inicializado.) */


    A = A + AA
    B = B + BB
    C = C + CC
    D = D + DD

 fin para /* del bucle en 'i' */
```

Paso 5. Salida

El resumen del mensaje es la salida producida por A, B, C y D. Esto es, se comienza el byte de menor peso de A y se acaba con el byte de mayor peso de D.

Explicación de la parte del chat:

Para realizar el servicio de chat utilicé una arquitectura orientada objetos llamada CORBA *(*Common Object Request Broker Architecture) , la cual presenta muchas características de otros sistemas orientados a objetos, incluyendo la herencia de interfaces y el polimorfismo. Lo que hace a CORBA más interesante es que proporciona estas capacidades, incluso cuando es utilizado en lenguajes no orientados a objeto como C o COBOL, aunque CORBA trabaja particularmente bien con los lenguajes orientados a objeto como C++ y Java.

Permite la implementación de programas tipo cliente-servidor. Por lo cual se decidió utilizar esta arquitectura.

Acontinuación detallamos cada paso para la creación de un programa en corba:

Interfaz IDL

Interfaz IDL:

El primer paso al crear una aplicación CORBA es especificar todos tus objetos y tus interfaces usando el
IDL tiene sintaxis similar a C++ y puede ser usada al definir modulos, interfaces, estructuras de datos y m
El siguiente código esta escrito en IDL, y descibr un objeto CORBA que dice hola (sayHello()), esta oper
y un método que apaga el ORB (shutdown()).

Hello.idl

```
module HelloApp
```

```
{
  interface Hello
  {
   string sayHello();
   oneway void shutdown();
  };
};
```

NOTA: Cuando escribimos la interfaz no usamos el mismo nombre del modulo. Esto puede llevar a obte
cuando compilamos con herramientas de diferentes proveedores.

Programa servidor:

EL ejemplo del servidor consiste de dos clases, el sirviente (servant) y el servidor (server). El sirviente, H
implementación
de la interfaz IDL "hello"; cada instancia Hello e implementada por una instancia HelloImpl. El sirviente
HelloPOA, que es
generado por el compilador idlj del ejemplo IDL. El sirviente contiene un método para cada operación ID
métodos sayHello()
y shutdown(). Los métodos del sirviente son como métodos ordinarios de java.

La clase HelloServer, en su método main():

- Crea e inicializa una instancia ORB
- Obtiene una referencia a la raíz POA y activa el POAManager
- Crea una instancia del sirviente y llama al ORB a través de él
- Obtiene un objeto CORBA de referencia para llamado contexto en que registra el nuevo objeto C
- Obtiene la raíz del llamado contexo
- Registra el nuevo objeto en el llamado contexto bajo el nombre "Hello"

- Espera para la invoación del nuevo objeto de el cliente.

Este ejmplo provee de un ejemplo de un objeto servidor temporal. Para un ejemplo de un programa "Hola
servidor persistente

ver Ejemplo 2: Hola Mundo con un estado persistente.

HelloServer.java

```
import HelloApp.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;

import java.util.Properties;
```

```
class HelloImpl extends HelloPOA {
  private ORB orb;

  public void setORB(ORB orb_val) {
    orb = orb_val;
  }

  // implement sayHello() method
  public String sayHello() {
            return "\nHello world !!\n";
  }

  // implement shutdown() method
  public void shutdown() {
    orb.shutdown(false);
  }
}

public class HelloServer {

  public static void main(String args[]) {
    try{
      // create and initialize the ORB
        ORB orb = ORB.init(args, null);

      // get reference to rootpoa & activate the POAManager
      POA rootpoa = POAHelper.narrow(orb.resolve_initial_ref
      rootpoa.the_POAManager().activate();

      // create servant and register it with the ORB
      HelloImpl helloImpl = new HelloImpl();
      helloImpl.setORB(orb);

      // get object reference from the servant
      org.omg.CORBA.Object ref = rootpoa.servant_to_referenc
      Hello href = HelloHelper.narrow(ref);

      // get the root naming context
```

```
  // NameService invokes the name service
  org.omg.CORBA.Object objRef =

              orb.resolve_initial_references("NameServic

  // Use NamingContextExt which is part of the Interoper

  // Naming Service (INS) specification.

  NamingContextExt ncRef = NamingContextExtHelper.narrow


  // bind the Object Reference in Naming

  String name = "Hello";

  NameComponent path[] = ncRef.to_name( name );

          ncRef.rebind(path, href);


          System.out.println("HelloServer ready and wait


  // wait for invocations from clients
  orb.run();

}


          catch (Exception e) {
```

```
            System.err.println("ERROR: " + e);
            e.printStackTrace(System.out);
          }

          System.out.println("HelloServer Exiting ...");

        }
      }
```

Programa cliente:

El ejemplo de la aplicacion del cliente:

- Crea e inicializa un ORB
- Obtiene una referencia a la raíz del llamado contexto
- Busca "Hello" en el llamdo contexto y recive una referencia a ese objeto CORBA
- Invoca las operaciones sayHello() y shutdown() del objeto e imprime el resultado

HelloClient.java

```
import HelloApp.*;
        import org.omg.CosNaming.*;
       import org.omg.CosNaming.NamingContextPackage.*;
            import org.omg.CORBA.*;

            public class HelloClient
            {
              static Hello helloImpl;

              public static void main(String args[])
```

```
{
  try{
    // create and initialize the ORB
    ORB orb = ORB.init(args, null);

    // get the root naming context
    org.omg.CORBA.Object objRef =
        orb.resolve_initial_references("NameService");
    // Use NamingContextExt instead of NamingContext. Th
    // part of the Interoperable naming Service.
    NamingContextExt ncRef = NamingContextExtHelper.narr

    // resolve the Object Reference in Naming
    String name = "Hello";
    helloImpl = HelloHelper.narrow(ncRef.resolve_str(nam

    System.out.println("Obtained a handle on server obje
    System.out.println(helloImpl.sayHello());
    helloImpl.shutdown();
```

```
            } catch (Exception e) {
              System.out.println("ERROR : " + e) ;
              e.printStackTrace(System.out);

              }

        }


    }
```

Para nuestro programa utilizamos solamente la misma receta implementada con servicios típicos de un chat, login, alta, baja, y envio de mensajes.

Servidor:

```
import org.omg.PortableServer.*;
import org.omg.CosNaming.*;
import java.security.MessageDigest;//md5

public class ServidorChat
{
    public static void main (String[] args)
    {
```

```
        Ejm09_EncriptacionMD5 obj = new
Ejm09_EncriptacionMD5();


      try {
         JOptionPane.showMessageDialog(null,
obj.getMD5("palabrasecreta"));
      } catch (Exception ex) {
         System.out.println(ex.getMessage());
      }
```

```
            try
            {

                    // 1. Inicializar ORB
            org.omg.CORBA.ORB orb =
org.omg.CORBA.ORB.init(args, null);


            // 2.1 Obtener POA raiz
            POA raizPOA =
POAHelper.narrow(orb.resolve_initial_references("RootPOA"));


            // 2.2 Activar el POA manager
            raizPOA.the_POAManager().activate();


            ServicioChatImpl ServicioChat_servant = new
ServicioChatImpl();


            // 3.2 Registrar en el POA y obtener referencia al objeto
(IOR)
            org.omg.CORBA.Object ServicioChatCorba =
raizPOA.servant_to_reference(ServicioChat_servant);
```

```
        // 4.1 Obtener el initial naming context
        org.omg.CORBA.Object ncCorba =
orb.resolve_initial_references("NameService");
        NamingContextExt nc =
NamingContextExtHelper.narrow(ncCorba);


        // 4.2 Asociar un nombre (en el primer nivel)
        nc.rebind(nc.to_name("ServicioChat"), ServicioChatCorba);
```

```
    // 5 Quedar a la espera de peticiones
    System.out.println("Proceso servidor en espera ... ");
    orb.run();


       }catch(Exception e) {
    System.out.println("Error:" + e.getMessage());
    System.exit(1);


  }
  }

public String getMD5(String cadena) throws Exception {

  MessageDigest md = MessageDigest.getInstance("MD5");
  byte[] b = md.digest(cadena.getBytes());


  int size = b.length;
  StringBuilder h = new StringBuilder(size);
  for (int i = 0; i < size; i++) {


    int u = b[i] & 255;


    if (u < 16)
    {
       h.append("0").append(Integer.toHexString(u));
```

```
    }
    else
    {
        h.append(Integer.toHexString(u));
    }
}
return h.toString();
```

```
    }
}
```

Conclusiones

MD5 se utiliza extensamente en el mundo del software para proporcionar la seguridad de que un archivo descargado de Internet no se ha alterado. Comparando una suma MD5 publicada con la suma de comprobación del archivo descargado, un usuario puede tener la confianza suficiente de que el archivo es igual que el publicado por los desarrolladores. Esto protege al usuario contra los

'Caballos de Troya' o 'Troyanos' y virus que algún otro usuario malicioso pudiera incluir en el software. La comprobación de un archivo descargado contra su suma MD5 no detecta solamente los archivos alterados de una manera maliciosa, también reconoce una descarga corrupta o incompleta.

Para comprobar la integridad de un archivo descargado de Internet se puede utilizar una herramienta MD5 para comparar la suma MD5 de dicho archivo con un archivo MD5SUM con el resumen MD5 del primer archivo. En los sistemas UNIX, el comando de*md5sum* es un ejemplo de tal

herramienta. Además, también está implementado en el lenguaje de*scripting*PHPcomo MD5("") entre otros.

En sistemas UNIX y GNU/Linux se utiliza el algoritmo MD5 para calcular el *hash* de las claves de los usuarios. En el disco se guarda el resultado del MD5 de la clave que se introduce al dar de alta un usuario, y cuando éste quiere entrar en el sistema se compara el *hash* MD5 de la clave introducida con el *hash* que hay guardado en el disco duro. Si coinciden, es la misma clave y el usuario será autenticado. Los sistemas actuales GNU/Linux utilizan funciones de hash más seguras, como pueden ser SHA-2 o SHA-3.

en 1

[Escriba el subtítulo del documento]

user

[Seleccione la fecha]

Introducción:

En redes informáticas de datos se denomina **latencia** a la suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red.

Otros factores que influyen en la latencia de una red son:

- El tamaño de los paquetes transmitidos.
- El tamaño de los buffers dentro de los equipos de conectividad. Ellos pueden producir un Retardo Medio de Encolado

Este proyecto tiene como finalidad similar una red y calcular el camino con latencia menor, para ello serán necesarios conocimientos básicos de estructuras de datos como lo son Grafos, así como de técnicas de programación avanzada, el programa deberá importar un archivo de formato especifico lo que implica una programación que permita generar distintos tipos de grafos y recorridos a todos ellos.

Desarrollo

Comencé el programa primero importando los datos del archivo con el formato especificado, para ello construí una función recursiva que permite saltarse todas las líneas que incluyan el carácter “%”, el cual marcara a una línea de comentarios, en caso de no existir, tomara la siguiente línea valida.

```
public String leeArchivo()throws IOException{
    String str;
    str = leector.readLine();
    if( str.charAt(0)== '%' ){
        str=leeArchivo();
    }
    return str;
}
```

Una ves logrado esto, estableci el tipo de estructura de datos que utilizaría para el almacenamiento de valores importantes, para esto utilice la clase

```
import java.util.*;
```

la cual me permite el manejo de arreglos dinámicos, (ArrayList) los cuales utilize para guardar las vértices y aristas del grafo:

```
List<Enlace> adyacencia = new ArrayList<Enlace>();
List<Nodo> iteraciones = new ArrayList<Nodo>();;
List<Nodo> inicioFinal = new ArrayList<Nodo>();;
List<Nodo> router = new ArrayList<Nodo>();
```

La variable adyacencia es quien tendrá los vértices, iteraciones tendrá el número de nodos y numero de Vertices, inicioFinal aloja el nombre del nodo inicial y el nodo final y router, todos los datos (tiempos de cola) de cada nodo del grafo

Ya que en el formato se indica que iteraciones e inicio final vienen de la misma manera, construi esta función con la que retiro 2 datos de una línea del archivo, lo asigno a la variable array que llega por parámetros y los regreso para asignarlos a el contenedor final

```
public void intData(String s,List<Nodo> array){
    String docValue[] = new String[2];
    int v=0;
    for(int i=0; i < s.length() ;i++ ){
        char caracter = s.charAt(i);
        if(caracter == ','){
        }
        else{
            docValue[v] = ""+caracter;
            array.add(new Nodo(docValue[v]));
            v++;
        }
    }
}
```

Estas son las funciones utilizadas por mi programa, muchas de ellas muy complejas y largas por lo que no incluyo fotos de ellas ya que van en los archivos adjuntos, solo

```
public void getFileSize()throws IOException..
public void getTC()throws IOException..
public void getLinks() throws IOException..
public void allPaths()..
public String leeArchivo()throws IOException..
public void calculaPaths(Nodo currentNode, List<Enlace> l, Nodo endNode, List<Enlace>Path, List<Nodo> visitedNodes)..
public void imprimeCamino(List<Enlace> l)..

public void updateLinks()..
```

explicare la función del programa

Como vimos antes, ya con los valores de iretacion utilizo las funciones getLinks y getTC para obtener enlaces y nodos respectivamente, updateLinks, actualiza los tiempos de cola de los enlaces ya que se guardaron en estructuras de datos diferentes.

La función principal de mi Clase Archivo, la encargada de la lectura y almacenamiento de datos es la siguiente:

```
public void construyeGrafo()throws IOException, NumberFormatException, ArrayIndexOutOfBoundsException{
    String dato;
    dato = leeArchivo();
    intData(dato,iteraciones);
    dato = leeArchivo();
    intData(dato,inicioFinal);
    System.out.println("Nodos Totales: "+ iteraciones.get(0).name()+ " Vertices Totales: "+ iteraciones.get(1).name());
    System.out.println("Equipo A(Origen) :"+inicioFinal.get(0).name());
    System.out.println("Equipo B(Destino):"+inicioFinal.get(1).name() );
    getLinks();
    getTC();
    updateLinks();
    getFileSize();
    allPaths();

}
```

En ella se alcanza a apreciar la secuencia en la que trabaja el software, primero retiro los datos de iteración en las primeras 5 líneas de código, continuo imprimiendo los datos en pantalla para luego obtener Enlaces, Obtener Tiempos de Cola, actualizar los enlaces, obtener el tamaño del archivo y calcular todos los caminos con la función allPaths().

allPaths se encarga de comenzar a trazar los caminos posibles con la ayuda de la función calculaPaths la cual recibe por parámetros los nodos destino, el arreglo de adyacencia, el nodo final, una lista que cargara el camino trazado y una lista con los arreglos visitados

```
public void allPaths(){
    System.out.println("Todos los recorridos de A -> B Posibles son:");
    List<Enlace> c = new ArrayList<Enlace>();
    List<Nodo> pathRecord = new ArrayList<Nodo>();
    for(Enlace i:adyacencia){
         if(i.origen.name().equals(inicioFinal.get(0).name()) && pathRecord.contains(i.destino)==false){
            c.clear();
            pathRecord.clear();
            c.add(i);
            pathRecord.add(i.origen);
            calculaPaths(i.destino,adyacencia,inicioFinal.get(1),c,pathRecord);
        }
    }
}
```

calculaPaths hace uso de la recursión para el cálculo de los caminos:

```
public void calculaPaths(Nodo currentNode, List<Enlace> l, Nodo endNode, List<Enlace>Path, List<Nodo> visitedNodes){
    if(currentNode.name().equals(endNode.name())){
        Latencia latency = new Latencia();
        System.out.println(" Path :"+Path.size());
        imprimeCamino(Path);
        double tempLatency = latency.latencyOf(Path);
        System.out.print(" Latencia "+String.format("%f", tempLatency));
    }
    else{
        for(Enlace i:l){
            if(currentNode.name().equals(i.origen.name())){
                if(visitedNodes.contains(i.destino)==false){  //problemas...
                    visitedNodes.add(i.origen);
                    Path.add(i);
                    calculaPaths(i.destino,l,endNode,Path,visitedNodes);
                    visitedNodes.remove(i.origen);
                    Path.remove(i);
                }
            }
        }
    }
```

Como podemos ver en la condición de paro mando a llamar a la función latencyOf() de la clase Latencia, la cual recibe una lista con el camino recorrido y calcula su latencia.

```
public double latencyOf(List<Enlace> l){
    int firstPaq = l.get(0).userData();
    List<Pakage> pak = new ArrayList<Pakage>();
    double flat=0,ltemp=0;
    int rr=1;
    Pakage frst = new Pakage(firstPaq);
    Pakage sqnd = new Pakage(1300);
    pak.add(frst);
    int smallerSize = firstPaq;
        for(Enlace i:l){
            if(i.datosUsuario<smallerSize){
                smallerSize=i.datosUsuario;
            }
            pak=getP2(pak,smallerSize);
            //System.out.println(i.longitud+i.velocidad);
            ltemp=(i.makeOperations()*pak.size());
            //System.out.println("Enlace "+rr+" * "+pak.size()+" = "+String.format("%f", ltemp));
            flat=flat+ltemp;
            rr++;
        }
return  flat;
```

Y con la función makeOperations, hago las operaciones pertinentes para obtener el TT, TP y TC, multiplicado por el numero de paquetes devuelto por la función pak2().

Esta función manda cada a cortar los paquetes con la función pak2, solo en caso de ser

```
public List<Pakage> getP2 (List<Pakage> r,int s){
    List<Pakage> temp = new ArrayList<Pakage>();
    int sobra;
    for(Pakage i:r){
        if(i.getSize()>s){
            System.out.println("Dividiendo");
            sobra = i.getSize()-s;
            temp.add(new Pakage(s));
            while(sobra>s){
                sobra = sobra-s;
                temp.add(new Pakage(s));
            }
            temp.add(new Pakage(sobra));
        }
        else temp.add(new Pakage(i.getSize()));
    }
    return temp;
```

necesario}

Funcion que calcula TT,TP y TC:

```
public double makeOperations(){
    double result = getTP() + getTT() + destino.getTC();
    return result;
}
```

Conclusiones

Para la realización de este programa, me encontré con el obstáculo más grande en el manejo de estructuras de datos más que en el cálculo de la latencia en si ya que se reducía a una operación matemática, fueron necesarias horas de investigación y desarrollo para pensar la manera más adecuada de almacenamiento que congeniara con una eficiente organización a la hora de realizar los recorridos de los grafos, que fue en mi consideración, la parte mas difícil.

Resultados:

Estos son los outputs del programa:

Output	Archivo de Prueba
Archivo Elegido doc.txt Nodos Totales: 5 Vertices Totales: 7 Equipo A(Origen) :1 Equipo B(Destino):5 Enlaces Posibles: 1 -> 2 2 -> 5 3 -> 2 2 -> 3 1 -> 3 3 -> 5 1 -> 4 4 -> 5 1 Todos los recorridos de A -> B Posibles son: Path :2 1 -> 2 -> B Latencia 0.000117 Path :3 1 -> 2 -> 3 -> B Latencia 0.000176 Path :3 1 -> 3 -> 2 -> B Latencia 0.000176 Path :2 1 -> 3 -> B Latencia 0.000117 Path :2 1 -> 4 -> B Latencia 0.000117 Process completed.	%prueba 1 5,7 %prueba 3 1,5 %prueba 4 1,2,100,300,1200,1000 2,5,100,300,1200,1000 2,3,100,300,1200,1000 1,3,100,300,1200,1000 3,5,100,300,1200,1000 1,4,100,300,1200,1000 4,5,100,300,1200,1000 %prueba5 0.0 0.111 0.111 0.111 0.0 %pueba6 2.5,Gb %F85AC21DE4

Y esta la latencia resultado en una prueba con los valores de ejemplo de la clase:

Output	Datos de Ejemplo

Latencia 0.005514
 	
Process completed.
	Nodo n1 = new Nodo("1",tc);
	Nodo n2 = new Nodo("2",0.00025);
	System.out.println();
	Nodo n3 = new Nodo("3",0.001);
	Nodo n4 = new Nodo("4",tc);
	Enlace e1 = new Enlace(n1,n2,95,100,300,1200);
	Enlace e2 = new Enlace(n2,n3,80,1000,500,1000);
	Enlace e3 = new Enlace(n3,n4,111,10,500,800);
	grafo.add(e1);
	grafo.add(e2);
	grafo.add(e3);
	tc = l.latencyOf(grafo);
	System.out.println("Latencia "+String.format("%f", t

Benemérita Universidad Autónoma De Puebla

Facultad De Ciencias De La Computación

Primer Examen

Materia: Redes De Computadoras.

Catedrático: Dr. Iván Olmos Pineda.

Heroica Puebla de Zaragoza, a 30 de Enero de 2014.

Alumna:

Vivanco Rendón Mara Patricia

INTRODUCCION

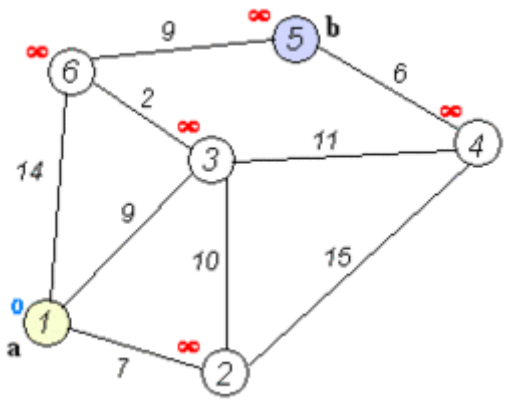
La latencia, o retardo, es el tiempo que una trama o paquete tarda en hacer el recorrido desde la estación origen hasta su destino final. Es importante determinar con exactitud la cantidad de latencia que existe en la ruta entre el origen y el destino para las LAN y las WAN. En el caso específico de una LAN Ethernet, un buen entendimiento de la latencia y de su efecto en la temporización de la red es de importancia fundamental para determinar si CSMA/CD podrá funcionar correctamente.
La latencia no depende únicamente de la distancia y de la cantidad de dispositivos. Por ejemplo, si dos estaciones de trabajo están separadas por tres switches correctamente configurados, las estaciones de trabajo pueden experimentar una latencia menor de la que se produciría si estuvieran separadas por dos routers correctamente configurados. Esto se debe a que los routers ejecutan funciones más complejas y que llevan más tiempo. Un router debe analizar los datos de la Capa 3.
Es el tiempo que le toma a un sólo paquete en viajar desde el nodo fuente hasta el nodo destino.

Hay situaciones en las que es más interesante conocer el tiempo en que tarda un paquete en ir y regresar. A ese tiempo se le conoce como Round Trip Time (RTT).

Hay situaciones en las que es más interesante conocer el tiempo en que tarda un paquete en ir y regresar. A ese tiempo se le conoce como Round Trip Time (RTT).

2) Tiempo en Cola: A los disposisitivos de comunicación les toma cierto tiempo en retransmitir un paquete .

3) Transmisión: la cantidad de tiempo que toma transmitir una unidad de datos a través de un enlace.



El algoritmo parte de un vértice origen que será ingresado, a partir de ese vértices evaluaremos sus adyacentes, como dijkstra usa una técnica greedy - La técnica greedy utiliza el principio de que para que un camino sea óptimo, todos los caminos

que contiene también deben ser óptimos- entre todos los vértices adyacentes, buscamos el que esté más cerca de nuestro punto origen, lo tomamos como punto intermedio y vemos si podemos llegar más rápido a través de este vértice a los demás. Después escogemos al siguiente más cercano (con las distancias ya actualizadas) y repetimos el proceso. Esto lo hacemos hasta que el vértice no utilizado más cercano sea nuestro destino. Al proceso de actualizar las distancias tomando como punto intermedio al nuevo vértice se le conoce como relajación (relaxation).

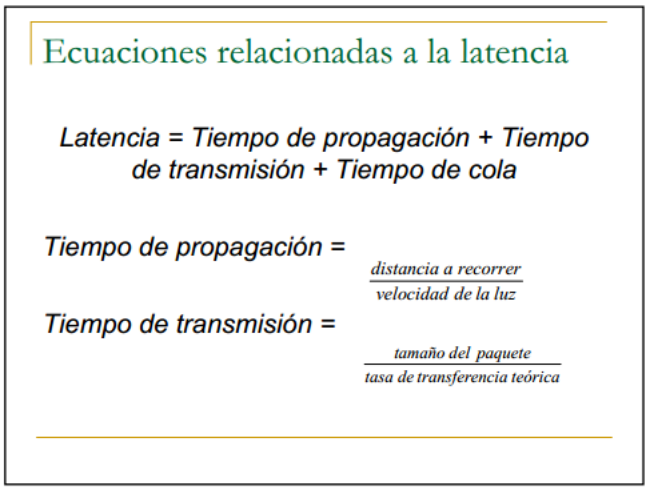
Dijkstra es muy similar a BFS, si recordamos BFS usaba una Cola para el recorrido para el caso de Dijkstra usaremos una Cola de Prioridad o Heap, este Heap debe tener la propiedad de Min-Heap es decir cada vez que extraiga un elemento del Heap me debe devolver el de menor valor, en nuestro caso dicho valor será el peso acumulado en los nodos.

Tanto java como C++ cuentan con una cola de prioridad ambos implementan un Binary Heap aunque con un Fibonacci Heap la complejidad de dijkstra se reduce haciéndolo mas eficiente, pero en un concurso mas vale usar la librería que intentar programar una nueva estructura como un Fibonacci Heap, claro que particularmente uno puede investigar y programarlo para saber como funciona internamente.

PROBLEMA

Implementar un programa que dada una topología de red (Velocidad de transmisión, Distancia de segmento, Tiempo de cola) se pueda encontrar el camino más rápido entre 2 nodos del sistema , así como calcule el Tiempo de Transmisión dado un tamaño de archivo predefinido.

Solución



Para poder solucionarlo tuvimos que utilizar el algoritmo de dijkstra para poder encontrar el camino más corto para poder calcular la latencia

```
/*Aqui empieza Dijkstra*/
struct cmp {
    bool ope complex<float> &operator e &x , const Node &y ) {
        return x.second > y.second;
    }
};
vector< Node > ady[ MAX ];
float distancia[ MAX ];
bool visitado[ MAX ];
priority_queue< Node , vector<Node> , cmp > Q;
int V;  //numero de vertices
int previo[ MAX ];  //imprime el camino
```

Como ya sabemos debemos poner calcular la latencia se tiene que calcular lo que se tiene en la parte inferior , pero ahora lo que tenemos que hacer es poder progrmarlo para que nos arroje un reseltuda de cada uno de ellas.

Debemos enviar los paquetes con los datos de usuario y los datos de control para asi ver si llega en un solo paquete o se debe de dividir en en varios para que pueda llegar al destino . una vez que divide los paquetes y calcula los caminos es asi como podremos saber cuanto tarda en transferirse con la latencia.

CONCLUSION

En el programa no se puede leer desde un txt y todo se debe poner desde consola.

Bliografia

http://www.cs.buap.mx/~iolmos/redes/3_Rendimiento.pdf

http://jariasf.wordpress.com/2012/03/19/camino-mas-corto-algoritmo-de-dijkstra/

Blog de WordPress.com. *El tema Coraline*.


BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

EXAMEN-PROYECTO
"MODELO DE REDES"

PROFESOR
IVAN OLMOS PINEDA

ALUMNO
RUIZ ESCAPITA CHRISTOPHER JESUS 201136965

PERIODO
PRIMAVERA 2014

INTRODUCCIÓN

Para el análisis de las redes muchas veces es necesario realizar operaciones con grafos, es una práctica común en el uso de las redes, se pueden hacer los cálculos que uno necesite y dependiendo de lo que se necesite es de la manera en la que se va a realizar el grafo, en nuestro caso calcularemos las latencias de los caminos que puede recorrer un paquete y así mostrar el camino con la latencia más corta.

ALGORITMOS

Para poder resolver este problema usamos un algoritmo en backtracking recursivamente, lo que hace este algoritmo es que recibe el nodo inicial y el nodo final, posteriormente va marcando en un arreglo de visitados que ya paso por ese nodo, y en la matriz de nodos adyacentes busca los adyacentes del nodo que no estén visitados y manda a llamar en recursión a la función y el nodo inicial ahora es el adyacente del nodo anterior, después de agregar el nodo va limpiando el arreglo de visitados para que posteriormente pueda buscar otros caminos. La condición de paro del método es que cuando el nodo inicial sea igual al final imprima el camino

```
194
195     public void paths(int ni, int nf){
196         int temp=0;
197         boolean first;
198
199         visitados[ni]=1;
200        if(ni==nf){
201             System.out.println();
202             first=true;
203             for(i=0;i<=NN;i++){
204                 if(visitados[i]==1)
205                     if(nf ==i)temp=i;
206                 else
207                    System.out.print(i+" ");
208                 first=false;
209             }
210             System.out.print(temp+" ");
211
212        }
213        //System.out.println(ni+" "+nf);
214         for(int v=1;v<=NN;v++){
215             if(M[ni][v]==1 && visitados[v]==0){
216                // lat=tp(ni,v)+tt(ni,v);
217                 paths(v,nf);
218
219             }
220         }
221         visitados[ni]=0;
222     }
```

LOGICA

La lógica más importante para la resolución de este problema es la del backtracking, por esa razón mostraremos el pseudocódigo para mostrar en que me base para resolver el problema.

Algoritmo en pseudocódigo:

```
método DFS( origen,final):
    marcamos origen como visitado
    recuperar el path si se llego a final
        para cada vertice v adyacente a origen en el Grafo:
            si v no ah sido visitado:
               marcamos como visitado v
               llamamos recursivamente DFS( v )
    marcamos origen como no visitado
```

FORMA DE TRABAJO

Para el tratado del grafo es que leemos los datos de un archivo de texto, el número de vértices y aristas se guardan en un, las adyacencias que tienen cada vértices se guardan en una matriz de Adyacencias, y los datos de distancia, velocidad, tamaño del paquete y datos de usuario se almacenan en una matriz para su manipulación posteriormente.

Los tiempos de cola se guardan en un arreglo llamado “tiempos[]”, el nodo inicial y el nodo final se guardan en variables que hacen referencia a su nombre y esos se mandan a llamar en la función paths(); y por último se alamcena el tamaño de paquete en una variable llamada “tampaq”

Posteriormente se calculan los caminos por los que puede mandarse el paquete. Y después de eso se calculan las latencias de los caminos y se tiene que imprimir el camino con la menor latencia.

NOTA: En mi caso me falto imprimir calcular las latencias, y así mismo imprimir el camino con menor latencia

CONCLUSION

Se encontraron los caminos, pero no encontré una lógica, digamos adecuada o fácil para calcular las latencias, por lo que a consideración propia se tiene aproximadamente un 70%.

en 1

[Escriba el subtítulo del documento]

[Seleccione la fecha]

user

Detección de Errores

La **detección y corrección de errores** es una importante práctica para el mantenimiento e integridad de los datos a través de diferentes procedimientos y dispositivos como medios de almacenamiento confiables.

Paridad Simple

Consiste en añadir un bit de más a la cadena que queremos enviar, y que nos indicará si el número de unos (bits puestos a 1) es par o es impar. Si es par incluiremos este bit con el valor = 0, y si no es así, lo incluiremos con valor = 1.

Suma de Comprobación

Es un método sencillo pero eficiente sólo con cadenas de palabras de una longitud pequeña, es por esto que se suele utilizar en cabeceras de tramas importantes u otras cadenas importantes y en combinación con otros métodos. Consiste en agrupar el mensaje a transmitir en cadenas de una longitud determinada L no muy grande, de por ejemplo 16 bits. Considerando a cada cadena como un número entero numerado según el sistema de numeración $2^L - 1$.

Distancia de Hamming

Si queremos detectar d bit erróneos en una palabra de n bits, podemos añadir a cada palabra de n bits d+1 bits predeterminados al final, de forma que quede una palabra de n+d+1 bits con una distancia mínima de Hamming de d+1. De esta manera, si uno recibe una palabra de n+d+1 bits que no encaja con ninguna palabra del código (con una distancia de Hamming x <= d+1 la palabra no pertenece al código) detecta correctamente si es una palabra errónea. Aún más, d o menos errores nunca se convertirán en una palabra válida debido a que la distancia de Hamming entre cada palabra válida es de al menos d+1, y tales errores conducen solamente a las palabras inválidas que se detectan correctamente. Dado un conjunto de m*n bits, podemos detectar x <= d bits errores correctamente usando el mismo método en todas las palabras de n bits. De hecho, podemos detectar un máximo de m*d errores si todas las palabras de n bits son transmitidas con un máximo de d errores.

Códigos de Redundancia Cíclica

El objetivo de una técnica para la detección de erres es permitirle al receptor saber si el mensaje recibido a través de un canal con ruido ha sido corrompido. Para esto, utilizamos Códigos de Redundancia Cíclica.

El CRC es un código de detección de error cuyo cálculo es una larga división de computación en el que se descarta el cociente y el resto se convierte en el resultado, con la importante diferencia de que la aritmética que usamos conforma que el cálculo utilizado es el arrastre de un campo, en este caso los bits. El tamaño del resto es siempre menor que la longitud del divisor, que, por lo tanto, determina el tamaño del resultado. La definición de un CRC especifica el divisor que se utilizará, entre otras cosas. Aunque un CRC se puede construir utilizando cualquier tipo de regla finita, todos los CRC de uso común emplean una base finita binaria, esta base consta de dos elementos, generalmente el 0 y 1.

Implementación

Se implementara la siguiente técnica con los polinomios CRC-16, CRC-CCITT y CRC-12. Utilizaremos lo que son tablas de consulta para mejorar la optimización del código.

Una tabla de consulta o Look-up table es una matriz que reemplaza cómputo de tiempo de ejecución de una operación de indización de matriz simple. El ahorro en términos de tiempo de procesamiento puede ser significante, ya que la recuperación de un valor de la memoria es a menudo más rápido que someterse a una operación de computación "inasequible" o de entrada / salida. Las tablas pueden ser pre calculadas y almacenadas en la memoria de programa estático, calculados como parte de la fase de inicialización de un programa o incluso almacenada en el "hardware" en plataformas específicas de la aplicación. Tablas de consultas también se utilizan ampliamente para validar los valores de entrada, haciendo coincidir contra una lista de elementos válidos (o no válido) en una matriz y, en algunos lenguajes de programación, pueden incluir funciones de puntero (o compensar a las etiquetas) para procesar la entrada correspondientes.

En este caso utilizaremos un código de C proporcionado por Ross N. Williams y modificado por mí mismo para poder obtener las tablas de consulta de los polinomios requeridos para el programa.

```
/**********************************************************************/
/*                            Start of crctable.c                     */
/**********************************************************************/
/*                                                                    */
/* Author  : Ross Williams (ross@guest.adelaide.edu.au.).             */
/* Date    : 3 June 1993.                                             */
/* Version : 1.0.                                                     */
/* Status  : Public domain.                                           */
/*                                                                    */
/* Description : This program writes a CRC lookup table (suitable for */
/* inclusion in a C program) to a designated output file. The program can be */
/* statically configured to produce any table covered by the Rocksoft^tm */
/* Model CRC Algorithm. For more information on the Rocksoft^tm Model CRC */
/* Algorithm, see the document titled "A Painless Guide to CRC Error  */
/* Detection Algorithms" by Ross Williams (ross@guest.adelaide.edu.au.). This */
/* document is likely to be in "ftp.adelaide.edu.au/pub/rocksoft".    */
/*                                                                    */
/* Note: Rocksoft is a trademark of Rocksoft Pty Ltd, Adelaide, Australia. */
/*                                                                    */
/**********************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include "crcmodel.h"

/**********************************************************************/

/* TABLE PARAMETERS                                                   */
/* ================                                                   */
/* The following parameters entirely determine the table to be generated. You */
/* should need to modify only the definitions in this section before running */
/* this program.                                                      */
```

```
/*                                                                          */
/*    TB_FILE  is the name of the output file.                              */
/*    TB_WIDTH is the table width in bytes (either 2 or 4).                 */
/*    TB_POLY  is the "polynomial", which must be TB_WIDTH bytes wide.      */
/*    TB_REVER indicates whether the table is to be reversed (reflected).   */
/*                                                                          */
/* Example:                                                                 */
/*                                                                          */
/*    #define TB_FILE   "crctable.out"                                      */
/*    #define TB_WIDTH  2                                                   */
/*    #define TB_POLY   0x8005L                                             */
/*    #define TB_REVER  TRUE                                                */

#define BITMASK(X) (1L << (X))
#define MASK32 0xFFFFFFFFL
#define LOCAL static

/*****************************************************************************/

LOCAL ulong reflect P_((ulong v,int b));
LOCAL ulong reflect (v,b)
/* Returns the value v with the bottom b [0,32] bits reflected. */
/* Example: reflect(0x3e23L,3) == 0x3e26                        */
ulong v;
int   b;
{
 int   i;
 ulong t = v;
 for (i=0; i<b; i++)
   {
    if (t & 1L)
       v|=  BITMASK((b-1)-i);
    else
       v&= ~BITMASK((b-1)-i);
    t>>=1;
   }
 return v;
}

/*****************************************************************************/

LOCAL ulong widmask P_((p_cm_t));
LOCAL ulong widmask (p_cm)
/* Returns a longword whose value is (2^p_cm->cm_width)-1.     */
/* The trick is to do this portably (e.g. without doing <<32). */
p_cm_t p_cm;
{
 return (((1L<<(p_cm->cm_width-1))-1L)<<1)|1L;
}

/*****************************************************************************/

void cm_ini (p_cm)
p_cm_t p_cm;
{
```

```
 p_cm->cm_reg = p_cm->cm_init;
}

/**********************************************************************/

void cm_nxt (p_cm,ch)
p_cm_t p_cm;
int    ch;
{
 int   i;
 ulong uch  = (ulong) ch;
 ulong topbit = BITMASK(p_cm->cm_width-1);

 if (p_cm->cm_refin) uch = reflect(uch,8);
 p_cm->cm_reg ^= (uch << (p_cm->cm_width-8));
 for (i=0; i<8; i++)
   {
    if (p_cm->cm_reg & topbit)
       p_cm->cm_reg = (p_cm->cm_reg << 1) ^ p_cm->cm_poly;
    else
       p_cm->cm_reg <<= 1;
    p_cm->cm_reg &= widmask(p_cm);
   }
}

/**********************************************************************/

void cm_blk (p_cm,blk_adr,blk_len)
p_cm_t   p_cm;
p_ubyte_ blk_adr;
ulong    blk_len;
{
 while (blk_len--) cm_nxt(p_cm,*blk_adr++);
}

/**********************************************************************/

ulong cm_crc (p_cm)
p_cm_t p_cm;
{
 if (p_cm->cm_refot)
    return p_cm->cm_xorot ^ reflect(p_cm->cm_reg,p_cm->cm_width);
 else
    return p_cm->cm_xorot ^ p_cm->cm_reg;
}

/**********************************************************************/

ulong cm_tab (p_cm,index)
p_cm_t p_cm;
int    index;
{
 int   i;
 ulong r;
 ulong topbit = BITMASK(p_cm->cm_width-1);
```

```
 ulong inbyte = (ulong) index;

 if (p_cm->cm_refin) inbyte = reflect(inbyte,8);
 r = inbyte << (p_cm->cm_width-8);
 for (i=0; i<8; i++)
    if (r & topbit)
       r = (r << 1) ^ p_cm->cm_poly;
    else
       r<<=1;
 if (p_cm->cm_refin) r = reflect(r,p_cm->cm_width);
 return r & widmask(p_cm);
}

#define TB_FILE   "crctable.out"
#define TB_WIDTH  2
#define TB_POLY   0x1021
#define TB_REVER  TRUE

/*****************************************************************************/

/* Miscellaneous definitions. */

#define LOCAL static
FILE *outfile;
#define WR(X) fprintf(outfile,(X))
#define WP(X,Y) fprintf(outfile,(X),(Y))

/*****************************************************************************/

LOCAL void chk_err P_((char *));
LOCAL void chk_err (mess)
/* If mess is non-empty, write it out and abort. Otherwise, check the error   */
/* status of outfile and abort if an error has occurred.                      */
char *mess;
{
 if (mess[0] != 0   ) {printf("%s\n",mess); exit(EXIT_FAILURE);}
 if (ferror(outfile)) {perror("chk_err");   exit(EXIT_FAILURE);}
}

/*****************************************************************************/

LOCAL void chkparam P_((void));
LOCAL void chkparam ()
{
 if ((TB_WIDTH != 2) && (TB_WIDTH != 4))
    chk_err("chkparam: Width parameter is illegal.");
 if ((TB_WIDTH == 2) && (TB_POLY & 0xFFFF0000L))
    chk_err("chkparam: Poly parameter is too wide.");
 if ((TB_REVER != FALSE) && (TB_REVER != TRUE))
    chk_err("chkparam: Reverse parameter is not boolean.");
}

/*****************************************************************************/

LOCAL void gentable P_((void));
```

```
LOCAL void gentable ()
{
 WR("/*****************************************************************/\n");
 WR("/*                                                               */\n");
 WR("/* CRC LOOKUP TABLE                                              */\n");
 WR("/* ================                                              */\n");
 WR("/* The following CRC lookup table was generated automagically    */\n");
 WR("/* by the Rocksoft^tm Model CRC Algorithm Table Generation       */\n");
 WR("/* Program V1.0 using the following model parameters:            */\n");
 WR("/*                                                               */\n");
 WP("/*    Width   : %1lu bytes.                                         */\n",
    (ulong) TB_WIDTH);
 if (TB_WIDTH == 2)
 WP("/*    Poly    : 0x%04lX                                           */\n",
    (ulong) TB_POLY);
 else
 WP("/*    Poly    : 0x%08lXL                                       */\n",
    (ulong) TB_POLY);
 if (TB_REVER)
 WR("/*    Reverse : TRUE.                                             */\n");
 else
 WR("/*    Reverse : FALSE.                                            */\n");
 WR("/*                                                               */\n");
 WR("/* For more information on the Rocksoft^tm Model CRC Algorithm,  */\n");
 WR("/* see the document titled \"A Painless Guide to CRC Error        */\n");
 WR("/* Detection Algorithms\" by Ross Williams                        */\n");
 WR("/* (ross@guest.adelaide.edu.au.). This document is likely to be  */\n");
 WR("/* in the FTP archive \"ftp.adelaide.edu.au/pub/rocksoft\".        */\n");
 WR("/*                                                               */\n");
 WR("/*****************************************************************/\n");
 WR("\n");
 switch (TB_WIDTH)
   {
    case 2: WR("unsigned short crctable[256] =\n{\n"); break;
    case 4: WR("unsigned long  crctable[256] =\n{\n"); break;
    default: chk_err("gentable: TB_WIDTH is invalid.");
   }
 chk_err("");

 {
  int i;
  cm_t cm;
  char *form    = (TB_WIDTH==2) ? "0x%04lX" : "0x%08lXL";
  int   perline = (TB_WIDTH==2) ? 8 : 4;

  cm.cm_width = TB_WIDTH*8;
  cm.cm_poly  = TB_POLY;
  cm.cm_refin = TB_REVER;

  for (i=0; i<256; i++)
    {
     WR(" ");
     WP(form,(ulong) cm_tab(&cm,i));
     if (i != 255) WR(",");
     if (((i+1) % perline) == 0) WR("\n");
```

```
     chk_err("");
    }

 WR("};\n");
 WR("\n");
 WR("/*****************************************************************/\n");
 WR("/*                   End of CRC Lookup Table                     */\n");
 WR("/*****************************************************************/\n");
 WR("");
 chk_err("");
}
}

/*****************************************************************************/

main ()
{
 printf("\n");
 printf("Rocksoft^tm Model CRC Algorithm Table Generation Program V1.0\n");
 printf("------------------------------------------------------------\n");
 printf("Output file is \"%s\".\n",TB_FILE);
 chkparam();
 outfile = fopen(TB_FILE,"w"); chk_err("");
 gentable();
 if (fclose(outfile) != 0)
    chk_err("main: Couldn't close output file.");
 printf("\nSUCCESS: The table has been successfully written.\n");
}

/*****************************************************************************/
/*                             End of crctable.c                             */
/*****************************************************************************/
```

Código de Implementación

```
/**
 *
 * @author Ivan
 * Nombre de la clase: Crc16
 * A continuación se presenta la case Crc16 para obtener el CRC-16,
 * utilizamos la look-up table obtenida anteriormente con el programa
 * de Ross Williams.
 *
 */
public class Crc16 {

    int[] table = {
            0x0000, 0xC0C1, 0xC181, 0x0140, 0xC301, 0x03C0, 0x0280, 0xC241,
            0xC601, 0x06C0, 0x0780, 0xC741, 0x0500, 0xC5C1, 0xC481, 0x0440,
            0xCC01, 0x0CC0, 0x0D80, 0xCD41, 0x0F00, 0xCFC1, 0xCE81, 0x0E40,
            0x0A00, 0xCAC1, 0xCB81, 0x0B40, 0xC901, 0x09C0, 0x0880, 0xC841,
            0xD801, 0x18C0, 0x1980, 0xD941, 0x1B00, 0xDBC1, 0xDA81, 0x1A40,
            0x1E00, 0xDEC1, 0xDF81, 0x1F40, 0xDD01, 0x1DC0, 0x1C80, 0xDC41,
            0x1400, 0xD4C1, 0xD581, 0x1540, 0xD701, 0x17C0, 0x1680, 0xD641,
            0xD201, 0x12C0, 0x1380, 0xD341, 0x1100, 0xD1C1, 0xD081, 0x1040,
```

```
            0xF001, 0x30C0, 0x3180, 0xF141, 0x3300, 0xF3C1, 0xF281, 0x3240,
            0x3600, 0xF6C1, 0xF781, 0x3740, 0xF501, 0x35C0, 0x3480, 0xF441,
            0x3C00, 0xFCC1, 0xFD81, 0x3D40, 0xFF01, 0x3FC0, 0x3E80, 0xFE41,
            0xFA01, 0x3AC0, 0x3B80, 0xFB41, 0x3900, 0xF9C1, 0xF881, 0x3840,
            0x2800, 0xE8C1, 0xE981, 0x2940, 0xEB01, 0x2BC0, 0x2A80, 0xEA41,
            0xEE01, 0x2EC0, 0x2F80, 0xEF41, 0x2D00, 0xEDC1, 0xEC81, 0x2C40,
            0xE401, 0x24C0, 0x2580, 0xE541, 0x2700, 0xE7C1, 0xE681, 0x2640,
            0x2200, 0xE2C1, 0xE381, 0x2340, 0xE101, 0x21C0, 0x2080, 0xE041,
            0xA001, 0x60C0, 0x6180, 0xA141, 0x6300, 0xA3C1, 0xA281, 0x6240,
            0x6600, 0xA6C1, 0xA781, 0x6740, 0xA501, 0x65C0, 0x6480, 0xA441,
            0x6C00, 0xACC1, 0xAD81, 0x6D40, 0xAF01, 0x6FC0, 0x6E80, 0xAE41,
            0xAA01, 0x6AC0, 0x6B80, 0xAB41, 0x6900, 0xA9C1, 0xA881, 0x6840,
            0x7800, 0xB8C1, 0xB981, 0x7940, 0xBB01, 0x7BC0, 0x7A80, 0xBA41,
            0xBE01, 0x7EC0, 0x7F80, 0xBF41, 0x7D00, 0xBDC1, 0xBC81, 0x7C40,
            0xB401, 0x74C0, 0x7580, 0xB541, 0x7700, 0xB7C1, 0xB681, 0x7640,
            0x7200, 0xB2C1, 0xB381, 0x7340, 0xB101, 0x71C0, 0x7080, 0xB041,
            0x5000, 0x90C1, 0x9181, 0x5140, 0x9301, 0x53C0, 0x5280, 0x9241,
            0x9601, 0x56C0, 0x5780, 0x9741, 0x5500, 0x95C1, 0x9481, 0x5440,
            0x9C01, 0x5CC0, 0x5D80, 0x9D41, 0x5F00, 0x9FC1, 0x9E81, 0x5E40,
            0x5A00, 0x9AC1, 0x9B81, 0x5B40, 0x9901, 0x59C0, 0x5880, 0x9841,
            0x8801, 0x48C0, 0x4980, 0x8941, 0x4B00, 0x8BC1, 0x8A81, 0x4A40,
            0x4E00, 0x8EC1, 0x8F81, 0x4F40, 0x8D01, 0x4DC0, 0x4C80, 0x8C41,
            0x4400, 0x84C1, 0x8581, 0x4540, 0x8701, 0x47C0, 0x4680, 0x8641,
            0x8201, 0x42C0, 0x4380, 0x8341, 0x4100, 0x81C1, 0x8081, 0x4040,
        };
    int crc;

    public Crc16(){
        crc = 0x0;
    }
    /**
    * En la siguiente función obtenemos el código CRC aplicando operaciones a
    * nivel de bits para hacer referencia a la tabla.
    *
    */
    public int calculateCRC(byte[] bi){
        byte[] bytes = bi;
        for (byte b : bytes) {
            crc = (crc >>> 8) ^ table[(crc ^ b) & 0xff];
        }
        System.out.println(crc);
        return crc;
    }

}




/**
 *
 * @author Ivan
 * Nombre de la clase: Crc16CCITT
 * A continuación se presenta la case Crc16CCITT para obtener el CRC-CCITT,
```

```
 * utilizamos la look-up table obtenida anteriormente con el programa
 * de Ross Williams.
 */
public class Crc16CCITT {

    int[] table = {
            0x0000, 0x1189, 0x2312, 0x329B, 0x4624, 0x57AD, 0x6536, 0x74BF,
            0x8C48, 0x9DC1, 0xAF5A, 0xBED3, 0xCA6C, 0xDBE5, 0xE97E, 0xF8F7,
            0x1081, 0x0108, 0x3393, 0x221A, 0x56A5, 0x472C, 0x75B7, 0x643E,
            0x9CC9, 0x8D40, 0xBFDB, 0xAE52, 0xDAED, 0xCB64, 0xF9FF, 0xE876,
            0x2102, 0x308B, 0x0210, 0x1399, 0x6726, 0x76AF, 0x4434, 0x55BD,
            0xAD4A, 0xBCC3, 0x8E58, 0x9FD1, 0xEB6E, 0xFAE7, 0xC87C, 0xD9F5,
            0x3183, 0x200A, 0x1291, 0x0318, 0x77A7, 0x662E, 0x54B5, 0x453C,
            0xBDCB, 0xAC42, 0x9ED9, 0x8F50, 0xFBEF, 0xEA66, 0xD8FD, 0xC974,
            0x4204, 0x538D, 0x6116, 0x709F, 0x0420, 0x15A9, 0x2732, 0x36BB,
            0xCE4C, 0xDFC5, 0xED5E, 0xFCD7, 0x8868, 0x99E1, 0xAB7A, 0xBAF3,
            0x5285, 0x430C, 0x7197, 0x601E, 0x14A1, 0x0528, 0x37B3, 0x263A,
            0xDECD, 0xCF44, 0xFDDF, 0xEC56, 0x98E9, 0x8960, 0xBBFB, 0xAA72,
            0x6306, 0x728F, 0x4014, 0x519D, 0x2522, 0x34AB, 0x0630, 0x17B9,
            0xEF4E, 0xFEC7, 0xCC5C, 0xDDD5, 0xA96A, 0xB8E3, 0x8A78, 0x9BF1,
            0x7387, 0x620E, 0x5095, 0x411C, 0x35A3, 0x242A, 0x16B1, 0x0738,
            0xFFCF, 0xEE46, 0xDCDD, 0xCD54, 0xB9EB, 0xA862, 0x9AF9, 0x8B70,
            0x8408, 0x9581, 0xA71A, 0xB693, 0xC22C, 0xD3A5, 0xE13E, 0xF0B7,
            0x0840, 0x19C9, 0x2B52, 0x3ADB, 0x4E64, 0x5FED, 0x6D76, 0x7CFF,
            0x9489, 0x8500, 0xB79B, 0xA612, 0xD2AD, 0xC324, 0xF1BF, 0xE036,
            0x18C1, 0x0948, 0x3BD3, 0x2A5A, 0x5EE5, 0x4F6C, 0x7DF7, 0x6C7E,
            0xA50A, 0xB483, 0x8618, 0x9791, 0xE32E, 0xF2A7, 0xC03C, 0xD1B5,
            0x2942, 0x38CB, 0x0A50, 0x1BD9, 0x6F66, 0x7EEF, 0x4C74, 0x5DFD,
            0xB58B, 0xA402, 0x9699, 0x8710, 0xF3AF, 0xE226, 0xD0BD, 0xC134,
            0x39C3, 0x284A, 0x1AD1, 0x0B58, 0x7FE7, 0x6E6E, 0x5CF5, 0x4D7C,
            0xC60C, 0xD785, 0xE51E, 0xF497, 0x8028, 0x91A1, 0xA33A, 0xB2B3,
            0x4A44, 0x5BCD, 0x6956, 0x78DF, 0x0C60, 0x1DE9, 0x2F72, 0x3EFB,
            0xD68D, 0xC704, 0xF59F, 0xE416, 0x90A9, 0x8120, 0xB3BB, 0xA232,
            0x5AC5, 0x4B4C, 0x79D7, 0x685E, 0x1CE1, 0x0D68, 0x3FF3, 0x2E7A,
            0xE70E, 0xF687, 0xC41C, 0xD595, 0xA12A, 0xB0A3, 0x8238, 0x93B1,
            0x6B46, 0x7ACF, 0x4854, 0x59DD, 0x2D62, 0x3CEB, 0x0E70, 0x1FF9,
            0xF78F, 0xE606, 0xD49D, 0xC514, 0xB1AB, 0xA022, 0x92B9, 0x8330,
            0x7BC7, 0x6A4E, 0x58D5, 0x495C, 0x3DE3, 0x2C6A, 0x1EF1, 0x0F78
        };
    int crc;

    public Crc16CCITT(){
        crc = 0x0;
    }
    /**
    * En la siguiente función obtenemos el código CRC aplicando operaciones a
    * nivel de bits para hacer referencia a la tabla.
    *
    */
    public int calculateCRC(byte[] bi){
        byte[] bytes = bi;
        for (byte b : bytes) {
            crc = (crc >>> 8) ^ table[(crc ^ b) & 0xff];
        }
        return crc;
```

```
    }

}



/**
 *
 * @author Ivan
 * Nombre de la clase: Crc12
 * A continuación se presenta la case Crc12 para obtener el CRC-12,
 * utilizamos la look-up table obtenida anteriormente con el programa
 * de Ross Williams.
 */
public class Crc12 {

    int[] table = {
            0x0000, 0x3DE4, 0x7BC8, 0x462C, 0xF790, 0xCA74, 0x8C58, 0xB1BC,
            0x0F01, 0x32E5, 0x74C9, 0x492D, 0xF891, 0xC575, 0x8359, 0xBEBD,
            0x1E02, 0x23E6, 0x65CA, 0x582E, 0xE992, 0xD476, 0x925A, 0xAFBE,
            0x1103, 0x2CE7, 0x6ACB, 0x572F, 0xE693, 0xDB77, 0x9D5B, 0xA0BF,
            0x3C04, 0x01E0, 0x47CC, 0x7A28, 0xCB94, 0xF670, 0xB05C, 0x8DB8,
            0x3305, 0x0EE1, 0x48CD, 0x7529, 0xC495, 0xF971, 0xBF5D, 0x82B9,
            0x2206, 0x1FE2, 0x59CE, 0x642A, 0xD596, 0xE872, 0xAE5E, 0x93BA,
            0x2D07, 0x10E3, 0x56CF, 0x6B2B, 0xDA97, 0xE773, 0xA15F, 0x9CBB,
            0x7808, 0x45EC, 0x03C0, 0x3E24, 0x8F98, 0xB27C, 0xF450, 0xC9B4,
            0x7709, 0x4AED, 0x0CC1, 0x3125, 0x8099, 0xBD7D, 0xFB51, 0xC6B5,
            0x660A, 0x5BEE, 0x1DC2, 0x2026, 0x919A, 0xAC7E, 0xEA52, 0xD7B6,
            0x690B, 0x54EF, 0x12C3, 0x2F27, 0x9E9B, 0xA37F, 0xE553, 0xD8B7,
            0x440C, 0x79E8, 0x3FC4, 0x0220, 0xB39C, 0x8E78, 0xC854, 0xF5B0,
            0x4B0D, 0x76E9, 0x30C5, 0x0D21, 0xBC9D, 0x8179, 0xC755, 0xFAB1,
            0x5A0E, 0x67EA, 0x21C6, 0x1C22, 0xAD9E, 0x907A, 0xD656, 0xEBB2,
            0x550F, 0x68EB, 0x2EC7, 0x1323, 0xA29F, 0x9F7B, 0xD957, 0xE4B3,
            0xF010, 0xCDF4, 0x8BD8, 0xB63C, 0x0780, 0x3A64, 0x7C48, 0x41AC,
            0xFF11, 0xC2F5, 0x84D9, 0xB93D, 0x0881, 0x3565, 0x7349, 0x4EAD,
            0xEE12, 0xD3F6, 0x95DA, 0xA83E, 0x1982, 0x2466, 0x624A, 0x5FAE,
            0xE113, 0xDCF7, 0x9ADB, 0xA73F, 0x1683, 0x2B67, 0x6D4B, 0x50AF,
            0xCC14, 0xF1F0, 0xB7DC, 0x8A38, 0x3B84, 0x0660, 0x404C, 0x7DA8,
            0xC315, 0xFEF1, 0xB8DD, 0x8539, 0x3485, 0x0961, 0x4F4D, 0x72A9,
            0xD216, 0xEFF2, 0xA9DE, 0x943A, 0x2586, 0x1862, 0x5E4E, 0x63AA,
            0xDD17, 0xE0F3, 0xA6DF, 0x9B3B, 0x2A87, 0x1763, 0x514F, 0x6CAB,
            0x8818, 0xB5FC, 0xF3D0, 0xCE34, 0x7F88, 0x426C, 0x0440, 0x39A4,
            0x8719, 0xBAFD, 0xFCD1, 0xC135, 0x7089, 0x4D6D, 0x0B41, 0x36A5,
            0x961A, 0xABFE, 0xEDD2, 0xD036, 0x618A, 0x5C6E, 0x1A42, 0x27A6,
            0x991B, 0xA4FF, 0xE2D3, 0xDF37, 0x6E8B, 0x536F, 0x1543, 0x28A7,
            0xB41C, 0x89F8, 0xCFD4, 0xF230, 0x438C, 0x7E68, 0x3844, 0x05A0,
            0xBB1D, 0x86F9, 0xC0D5, 0xFD31, 0x4C8D, 0x7169, 0x3745, 0x0AA1,
            0xAA1E, 0x97FA, 0xD1D6, 0xEC32, 0x5D8E, 0x606A, 0x2646, 0x1BA2,
            0xA51F, 0x98FB, 0xDED7, 0xE333, 0x528F, 0x6F6B, 0x2947, 0x14A3
        };
    int crc;

    public Crc12(){
        crc = 0x0;
    }
```

```
    /**
    * En la siguiente función obtenemos el código CRC aplicando operaciones a
    * nivel de bits para hacer referencia a la tabla.
    *
    */
    public int calculateCRC(byte[] bi){
        byte[] bytes = bi;
        for (byte b : bytes) {
            crc = (crc >>> 8) ^ table[(crc ^ b) & 0xff];
        }
        return crc;
    }

}
```

Interfaz y más

```
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.*;

/**
 *
 * @author Ivan
 * Nombre de la clase: GUI
 * Aqui se encuentra nuestro main y la inicialización de nuestros
 * componentes de interfaz.
 */
public class GUI extends JFrame{

    File file;
    byte [] fileData;
    FileInputStream in;
    FileOutputStream out;

    int crc;

    JLabel lChooser, lCrc, lMode;
    JTextField tChooser;
    JRadioButton crc1, crc2, crc3;
    JButton bChooser, bEmisor, bReceptor;
    ButtonGroup optionCrc;
    JFileChooser chooser;

    public GUI(){
        init();
    }

    public void init(){
```

```
crc1 = new JRadioButton ("CRC-16");
crc2 = new JRadioButton ("CRC-CCITT");
crc3 = new JRadioButton ("CRC-12");
lChooser = new JLabel("File");
lCrc = new JLabel("Choose your CRC:");
lMode = new JLabel ("Choose mode:");
tChooser = new JTextField("");
bChooser = new JButton("Browse");
bEmisor = new JButton ("Emisor");
bReceptor = new JButton ("Receptor");
chooser = new JFileChooser();
optionCrc = new ButtonGroup();

optionCrc.add(crc1);
optionCrc.add(crc2);
optionCrc.add(crc3);

this.setLayout(null);
this.setSize(700,400);
lChooser.setLocation(5, 5);
lChooser.setSize(20, 20);
lCrc.setLocation(5, 30);
lCrc.setSize(105, 20);
lMode.setLocation(5, 130);
lMode.setSize(85, 20);
crc1.setLocation(5, 55);
crc1.setSize(70, 20);
crc2.setLocation(5, 80);
crc2.setSize(90, 20);
crc3.setLocation(5, 105);
crc3.setSize(70, 20);
tChooser.setLocation(30, 5);
tChooser.setSize(300, 20);
bChooser.setLocation(335, 5);
bChooser.setSize(90, 20);
bEmisor.setLocation(335, 105);
bEmisor.setSize(75, 20);
bReceptor.setLocation(335, 130);
bReceptor.setSize(90, 20);

this.add(lChooser);
this.add(lCrc);
this.add(lMode);
this.add(crc1);
this.add(crc2);
this.add(crc3);
this.add(tChooser);
this.add(bChooser);
this.add(bEmisor);
this.add(bReceptor);

this.setDefaultCloseOperation(EXIT_ON_CLOSE);
/**
 * Obtenemos los datos del archivo elegido para poder
```

```
         * ser abierto.
         */
        bChooser.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                chooser.showOpenDialog(null);
                tChooser.setText(chooser.getSelectedFile().getPath());
            }
        });
        /**
         * La función principal para la generar el CRC se encuentra
         * aquí.
         *
         * Primero abrimos el archivo y obtenemos los datos de los
         * bits, despues, dependiendo de la elección del usuario,
         * comenzamos con el procedimiento de la generación del
         * código CRC y lo guardamos en un archivo con extensión .crc.
         */
        bEmisor.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent ae) {

                file = new File (chooser.getSelectedFile().getPath());
                fileData = new byte[(int) file.length()];
                try {
                    in = new FileInputStream(file);
                    out = new
FileOutputStream(chooser.getSelectedFile().getParent()+"\\out.crc");
                    in.read(fileData);
                    crc = 0x0;

                    if(crc1.isSelected()){
                        System.out.println(crc);
                        Crc16 crc16 = new Crc16();
                        crc = crc16.calculateCRC(fileData);
                        out.write(crc);
                    }
                    else if(crc2.isSelected()){
                        System.out.println(crc);
                        Crc16CCITT crc16Ccitt = new Crc16CCITT();
                        crc = crc16Ccitt.calculateCRC(fileData);
                        out.write(crc);
                    }
                    else if(crc3.isSelected()){
                        System.out.println(crc);
                        Crc12 crc12 = new Crc12();
                        crc = crc12.calculateCRC(fileData);
                        out.write(crc);
                    }

//                    String content = "";
//                    byte divisor = 0;
//                    for(byte b : fileData)
//                        divisor <<= b << 8;
//                        //content += getBits(b);
//                        //for(int i = 0; i < 8; i++)
                    in.close();
```

```
                    out.close();

                    System.out.println("CRC = " + Integer.toHexString(crc));
                } catch (FileNotFoundException ex) {
                    Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null,
ex);
                } catch (IOException ex) {
                    Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null,
ex);
                }
            }

        });
        /**
         * Función  encargada de saber si el receptor logró recibir
         * el mensaje sin ser corrompido.
         */
        bReceptor.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){

                file = new File (chooser.getSelectedFile().getPath());
                fileData = new byte[(int) file.length()];

                try {
                    in = new FileInputStream(file);
                    in.read(fileData);
                    String ina = "";
                    for(byte b : fileData){
                        //for(int i=0; i<8; i++){
                            ina+=getBits(b);
                            System.out.println(ina);
                        //}
                    }
                } catch (FileNotFoundException ex) {
                    Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null,
ex);
                } catch (IOException ex) {
                    Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null,
ex);
                }

            }
        });

    }
    public void run(){
        setVisible(true);
    }

    public String getBits(byte b){
        String result = "";
        for(int i = 0; i < 8; i++)
            result += (b & (1 << i)) == 0 ? "0" : "1";
        return result;
    }
```

```
    public static void main(String [] args){
        GUI ventana = new GUI();
        ventana.run();
    }

}
```

Conclusiones

El manejo de técnicas para la detección de errores es clave en las Redes de Computadoras para saber si los mensajes son recibidos correctamente. Así mismo, los códigos de redundancia cíclica pueden llegar a ser comprendidos fácilmente para su implementación, sin embargo, la falta de conocimiento en programación a nivel de bits puede complicar el proyecto de varias formas no previstas, alargando el tiempo de la realización de éste.

Las tablas de consulta son sumamente útiles para este tipo de procesos que pueden llegar a ocupar demasiado tiempo de procesador, acortándolos considerablemente.

Las redes de computadoras son sumamente interesantes, por el otro lado la teoría es inmensa y se ocupa bastante tiempo para poder dominarlas.

Bibliografía

www.ross.net/crc/download/crc_v3.txt
es.wikipedia.org/wiki/Lookup_table
https://koding.com/
http://stackoverflow.com/questions/

BENEMERITA UNIVERSIDAD

AUTONOMA DE PUEBLA

REDES DE COMPUTADORAS

Tarea

Examen Parcial 1

Autor:

Omar González Campos

FECHA DE ENTREGA

25/02/2014

INDICE

1. INTRODUCCIÓN EN EL LENGUAGE PHP

PHP es un potente lenguaje, y su intérprete, bien como módulo del servidor web o bien como binario CGI, puede acceder a ficheros, ejecutar comandos o abrir conexiones de red desde el servidor. Estas propiedades hacen que, por omisión, sea inseguro todo lo que se ejecute en un servidor web. PHP está diseñado específicamente para ser un lenguaje más seguro para escribir aplicaciones CGI que Perl or C. Partiendo de un correcto ajuste de opciones de configuración para tiempo de ejecución y en tiempo de compilación, y el uso de prácticas de programación apropiadas, pueden proporcionarle la combinación de libertad y de seguridad que necesita.

Dado que hay muchas vías para ejecutar PHP, existen muchas opciones de configuración para controlar su comportamiento. Al haber una extensa selección de opciones se garantiza poder usar PHP para un gran número de propósitos, pero a la vez significa que existen combinaciones que conllevan una configuración menos segura.

La flexibilidad de configuración de PHP rivaliza igualmente con la flexibilidad de su código. PHP puede ser usado para construir completas aplicaciones de servidor, con toda la potencia de un usuario de consola, o se puede usar sólo desde el lado del servidor implicando un menor riesgo dentro de un entorno controlado.

2. OBJETIVO DEL EXAMEN

El objetivo del examen es diseñar y programar una calculadora de tiempo total de transmisión de un archivo, con algunos algoritmos dados en clase para obtenerla, que son los siguientes:

- Obtener todos los caminos posibles, dada una topología representada en un grafo.
- Calcular la Latencia de cada camino.
  - Calcular los paquetes y el número de paquetes que se dividió el archivo inicial pasando por el camino seleccionado del grado.

- Calcular el Tiempo de Propagación.
- Calcular el Tiempo de Transmisión.

- Calcular El Tiempo de Total de Transmisión del Archivo, dado el tamaño del archivo entre el tamaño de datos de usuario y multiplicando todo por la latencia del mejor camino del grafo.

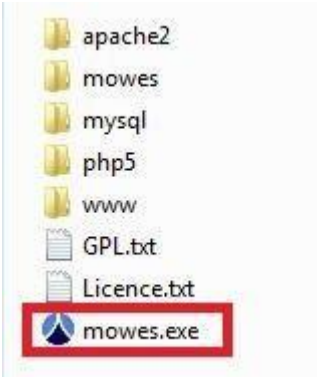
3. MANUAL DEL USUARIO

En esa sección se indicara como ejecutar el programa y como introducir los datos para poder calcular el Tiempo Total de Transferencia de un Archivo.

3.1 Como ejecutar un programa en PHP

Primero localizamos la carpeta MoWes:

Al abrir la Carpeta MoWes localizamos el ejecutable y lo abrimos:



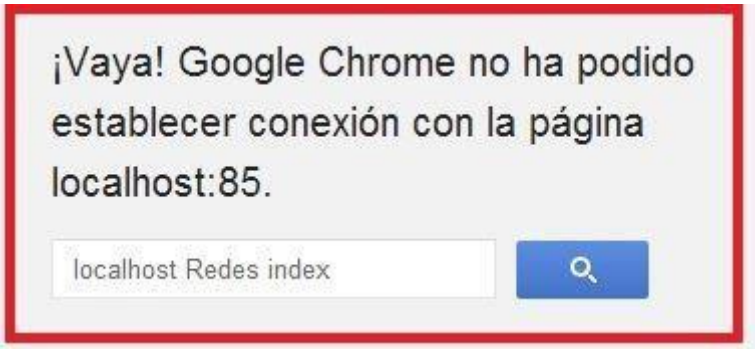
Una vez ejecutado el programa mowes tenemos que esperar que el servicio Sql y el servicio Apache estén listos para usarse:

En el momento en que el servicio Sql y el servicio Apache estén listo para ser usados, abrimos un navegador de internet y colocamos: localhost:85/Redes/index.php

Al pulsar Enter les cargara un contenido similar a este:

```
Caminos:
0 => 4 = 2 [3]: (0-2-4).
--------------------

Paquetes:
Array
(
    [0] => Array
        (
            [0] => 1200
            [1] => 1200
            [2] => 1200
            [3] => 1200
            [4] => 1200
            [5] => 1200
            [6] => 1200
            [7] => 1200
            [8] => 1200
            [9] => 1200
            [10] => 1200
            [11] => 1200
            [12] => 1200
            [13] => 1200
            [14] => 1200
            [15] => 1200
            [16] => 1200
            [17] => 1200
            [18] => 1200
            [19] => 1200
            [20] => 1200
            [21] => 1200
            [22] => 1200
            [23] => 1200
            [24] => 1200
            [25] => 1200
            [26] => 1200
```

Si se omitió alguno de los pasos anteriores o salió mal pasara esto:



En este caso favor de revisar los pasos anteriores, si el problema persiste contacte al encargado del proyecto.

3.2 Como introducir el archivo de datos:

Para cargar el archivo con los datos necesarios para realizar estas operaciones se tenía un formado así:

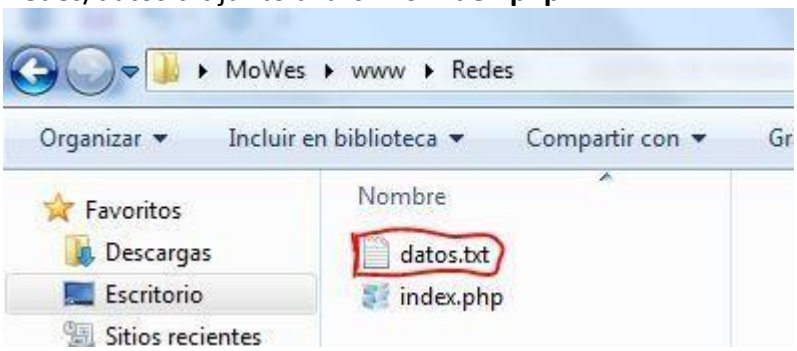
(#Nodos),(#Enlaces),
(Nodo Inicio),(Nodo Fin),

(Nodo Origen),(Nodo Destino),(Distancia en Metros),(V. Transferencia Teórica),(Datos de Control),(Datos de Usuario),

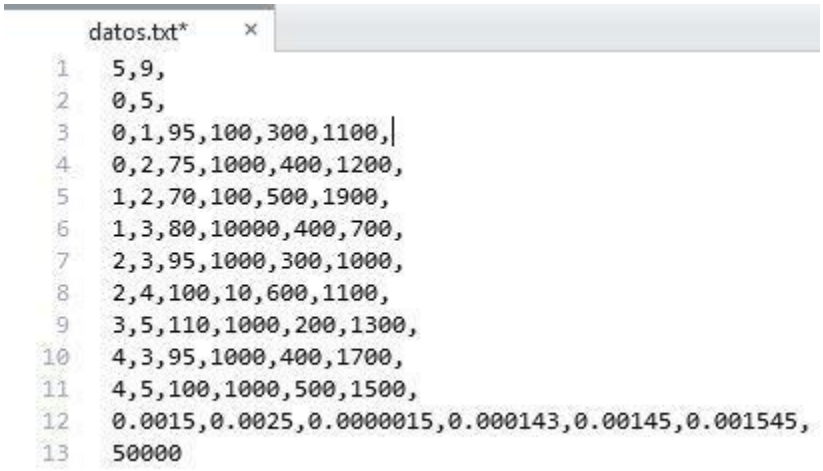
(Tiempos de Cola), (Tiempos de Cola),
(Tamaño del Archivo en Bytes)

La dirección donde se colocara el archivo con nombre **datos.txt** será en

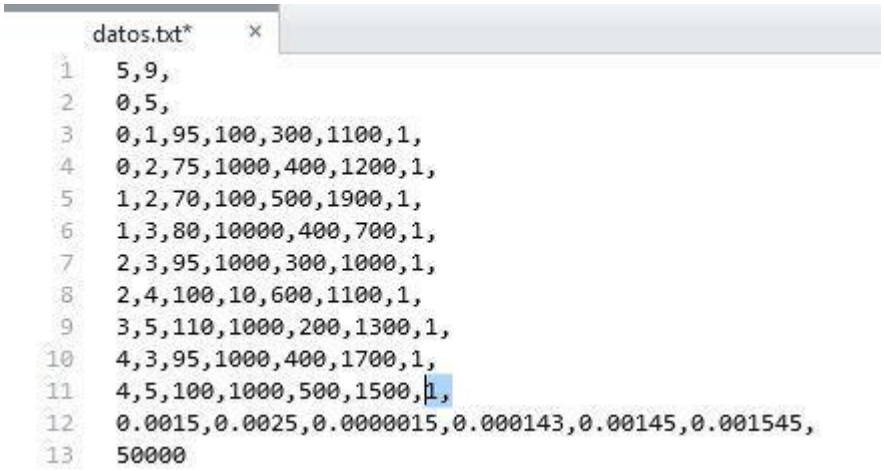
MoWes/www/Redes/datos.txt junto al archivo **index.php**



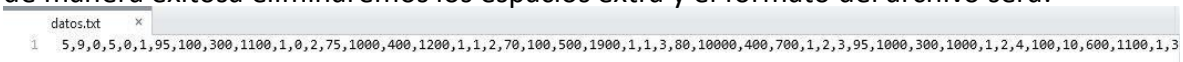
Un ejemplo del formato del archivo seria:



Por problemas de logística al diseñar el algoritmo de recorrido del grafo colocaremos “1” al final de cada camino como se muestra en la sig imagen:



Para la lectura correcta de los datos y no se pierda información y los cálculos se realicen de manera exitosa eliminaremos los espacios extra y el formato del archivo será:



Donde todos los datos quedarían en una sola línea sin espacios ni salto de líneas.

4. FUNCIONES UTILIZADAS

En esta parte explicare las funciones más relevantes que utilice para poder calcular el tiempo de transferencia total de un archivo, mostrando el resultado que regresa usando el ejemplo anterior.

4.1 Funcion Dijkstra:

En la función Dijkstra tenemos varias funciones que nos permiten encontrar un camino dentro de un grafo, la más importante de las funciones son **findShortestPath y updateDistanceAndPrevious** las cuales con ayuda de las demás funciones será posible encontrar el mejor camino:

```
        function findShortestPath($start,$to = null) {
                $this -> startnode = $start;
        $j = 1;
        // En esta parte se asignaran valores en los vértices sea Verdadero,
Falso o '0'
                for ($i=0;$i<$this -> numberOfNodes;$i++) {
                        if ($i == $this -> startnode) {
                                $this -> visited[$i] = TRUE;
                                $this -> distance[$i] = 0;
                        } else {
                                $this -> visited[$i] = FALSE;
                $this -> distance[$i] = isset($this -> map[$this ->
startnode][$i])
                                        ? $this -> map[$this -> startnode][$i]
                                        : $this -> infiniteDistance;
                        }
        $this -> previousNode[$i] = $this -> startnode;
                }

                $maxTries = $this -> numberOfNodes;
                $tries = 0;
        //se realiza la búsqueda del mejor camino llamando a otras funciones
para poder realizarlo desde el nodo inicial hasta el nodo final
                while (in_array(false,$this -> visited,true) && $tries <=
$maxTries) {
```

```
                        $this -> bestPath = $this->findBestPath($this-
>distance,array_keys($this -> visited,false,true));
                        if($to !== null && $this -> bestPath === $to) {
                                break;
                        }
                        $this -> updateDistanceAndPrevious($this ->
bestPath);
                        $this -> visited[$this -> bestPath] = true;
                        $tries++;
                }
        }
```

La función **findShortestPath** manda a llamar a algunas otras funciones para poder realizar el recorrido más efectivo del grafo que es: **findBestPath**

```
function findBestPath($ourDistance, $ourNodesLeft) {
                $bestPath = $this -> infiniteDistance;
                $bestNode = 0;
                for ($i = 0,$m=count($ourNodesLeft); $i < $m; $i++) {
            //hace las comparaciones necesarias sobre los pesos de los caminos
para poder tomar el camino correcto pero
            //en este caso al tener pesos en los caminos de '1' en camino que
realizara sera un recorrido hacia la izquierda
                        if($ourDistance[$ourNodesLeft[$i]] < $bestPath) {
                                $bestPath = $ourDistance[$ourNodesLeft[$i]];
                                $bestNode = $ourNodesLeft[$i];
                }
                }
         //regresa el mejor nodo a tomar
                return $bestNode;
        }
```

Y la función **updateDistanceAndPrevious** que guardara la distancia anterior porque la necesitaremos para mostrar los resultados:

```
function updateDistanceAndPrevious($obp) {
                for ($i=0;$i<$this -> numberOfNodes;$i++) {
                        if(     (isset($this->map[$obp][$i]))
                            &&  (!($this->map[$obp][$i] == $this-
>infiniteDistance) || ($this->map[$obp][$i] == 0 ))
                                &&      (($this->distance[$obp] + $this-
>map[$obp][$i]) < $this -> distance[$i])
                        )
            // guarda las distancias para que la función getResults pueda
```

```
obtenerla
                        {
                                        $this -> distance[$i] = $this ->
distance[$obp] + $this -> map[$obp][$i];
                                        $this -> previousNode[$i] = $obp;
                        }
                }
        }
```

Y una vez realizadas los cálculos por las funciones anteriores ahora se procede a obtener el resultado y como el resultado va de ida para poder mostrarlo correctamente en pantalla tenemos que realizarlo en sentido inverso de lo que se encarga la función **getResults:**

```
function getResults($to = null) {
                $ourShortestPath = array();
                $foo = '';
                for ($i = 0; $i < $this -> numberOfNodes; $i++) {
                        if($to !== null && $to !== $i) {
                                continue;
                        }
                        $ourShortestPath[$i] = array();
                        $endNode = null;
                        $currNode = $i;
                        $ourShortestPath[$i][] = $i;
                        while ($endNode === null || $endNode != $this ->
startnode) {
                                $ourShortestPath[$i][] = $this ->
previousNode[$currNode];

                                $endNode = $this -> previousNode[$currNode];
                                $currNode = $this -> previousNode[$currNode];
                        }

                        $ourShortestPath[$i] =
array_reverse($ourShortestPath[$i]);
                        if ($to === null || $to === $i) {
                        if($this -> distance[$i] >= $this -> infiniteDistance)
{                                                                       } else {

\n",$this -> startnode,$i);
```

```
$foo .= sprintf("no route from %d to %d.
 (%s)."\n" ,
 distance[$i],
 ',$ourShortestPath[$i]));
                        }
                        $foo .= str_repeat('-',20) . "\n";
                                if ($to === $i) {
                                        break;
                                }
                        }
                }
        $this -> numberOfNodesfinal = count($ourShortestPath[$i]);
        $this -> finalpath['1'] = $ourShortestPath[$i];
                return $foo;
        }
```

```
$foo .= sprintf('%d => %d = %d [%d]:
$this -> startnode,$i,$this ->
count($ourShortestPath[$i]),
implode('-
```

4.2 Matrices de Adyacencias:

Para poder asignar muchos valores a una sola arista entre 2 nodos, realice una matriz de adyacencias para cada valor como se muestra siguiente y la función encargada de eso es **printMap.**

```
function printMap(&$map) {
        $placeholder = ' %' . strlen($this -> infiniteDistance) .'d';
        $foo = '';
        for($i=0,$im=count($map);$i<$im;$i++) {
                for ($k=0,$m=$im;$k<$m;$k++) {
                        $foo.= sprintf($placeholder,
isset($map[$i][$k]) ? $map[$i][$k] : $this -> infiniteDistance);
                }
                $foo.= "\n";
        }
        return $foo;
}
```

Al pasar los datos por medio del archivo a la función **printMap** y si lo colocamos en pantalla las matrices quedarían de la siguiente manera:

```
Matriz de Distancias en Metros
 99999    95    75 99999 99999 99999
    95 99999    70    80 99999 99999
    75    70 99999    95   100 99999
 99999    80    95 99999    95   110
 99999 99999   100    95 99999   100
 99999 99999 99999   110   100 99999

Matriz de Velocidad de Transferencia Teorica
 99999   100  1000 99999 99999 99999
   100 99999   100 10000 99999 99999
  1000   100 99999  1000    10 99999
 99999 10000  1000 99999  1000  1000
 99999 99999    10  1000 99999  1000
 99999 99999 99999  1000  1000 99999

Matriz de Datos de Control
 99999   300   400 99999 99999 99999
   300 99999   500   400 99999 99999
   400   500 99999   300   600 99999
 99999   400   300 99999   400   200
 99999 99999   600   400 99999   500
 99999 99999 99999   200   500 99999

Matriz de Datos de Usuario
 99999  1100  1200 99999 99999 99999
  1100 99999  1900   700 99999 99999
  1200  1900 99999  1000  1100 99999
```

4.3 Obtener Datos Según el Camino Obtenido

Para obtener los datos para realizar las operaciones para poder sacar la latencia cree las siguientes funciones que obtienen los datos desde las matrices de adyacencias la primera función que obtiene los **Datos de usuario** es:

```
function datoUsuario(){
    $caminofinal = array();
    $niveles = array();
    for ($j=0;$j<$this -> numberOfNodes;$j++) {
    for ($i=0;$i<$this -> numberOfNodes;$i++) {
    $this -> usuario[$i][$j] = isset($this -> map5[$j][$i])
                                        ? $this -> map5[$j][$i]
                                        : $this -> infiniteDistance;
    }
    }
    for($i=0; $i < $this -> numberOfNodesfinal; $i++ ){
        $caminofinal[$i] = $this -> finalpath['1'][$i];
    }
    for($i=0, $j=1, $n=0; $i < ($this -> numberOfNodesfinal)-1; $i++, $j++,
$n++){
        $datofinal[$n] = $this -> usuario[$caminofinal[$i]][$caminofinal[$j]];
    }

    return $datofinal;
}
```

Que devuelve como resultado teniendo en cuenta los datos del ejemplo anterior:

```
Array
(
    [0] => 1100
    [1] => 700
    [2] => 1300
)
```

Después tenemos la función para sacar los **datos de Distancia** dada por metros:

```
function datoDistance(){
    $caminofinal = array();
    $niveles = array();
    for ($j=0;$j<$this -> numberOfNodes;$j++) {
    for ($i=0;$i<$this -> numberOfNodes;$i++) {
    $this -> distance2[$i][$j] = isset($this -> map2[$j][$i])
                                        ? $this -> map2[$j][$i]
                                        : $this -> infiniteDistance;
    }}
    for($i=0; $i < $this -> numberOfNodesfinal; $i++ ){
        $caminofinal[$i] = $this -> finalpath['1'][$i];
    }
    for($i=0, $j=1, $n=0; $i < ($this -> numberOfNodesfinal)-1; $i++, $j++,
$n++){
        $datofinal[$n] = $this -> distance2[$caminofinal[$i]][$caminofinal[$j]];
    }
    return $datofinal;}
```

Que tiene como resultado:

```
Array
(
    [0] => 95
    [1] => 80
    [2] => 110
)
```

La función que obtiene los datos de Velocidad dada en Mbps y dentro de la función los datos son convertidos a bits es la siguiente:

```
function datoVelocity(){
    $caminofinal = array();
    $niveles = array();
    for ($j=0;$j<$this -> numberOfNodes;$j++) {
    for ($i=0;$i<$this -> numberOfNodes;$i++) {
    $this -> velocity[$i][$j] = isset($this -> map3[$j][$i])
                                        ? $this -> map3[$j][$i]
                                        : $this -> infiniteDistance;
    }
    }
    for($i=0; $i < $this -> numberOfNodesfinal; $i++ ){
        $caminofinal[$i] = $this -> finalpath['1'][$i];
    }
    $Mbps = 1000000;
    for($i=0, $j=1, $n=0; $i < ($this -> numberOfNodesfinal)-1; $i++, $j++,
$n++){
        $datofinal[$n] = ($this -> velocity[$caminofinal[$i]][$caminofinal[$j]])
* $Mbps;
    }

    return $datofinal;
}
```

Que tiene como resultado:

```
Array
(
    [0] => 100000000
    [1] => 10000000000
    [2] => 1000000000
)
```

La función que obtiene los datos de usuario y suma los datos de control dados en bytes y los convierte a bits:

```
function datoControlUsuario(){
    $caminofinal = array();
    $niveles = array();
    for ($j=0;$j<$this -> numberOfNodes;$j++) {
    for ($i=0;$i<$this -> numberOfNodes;$i++) {
    $this -> control[$i][$j] = isset($this -> map4[$j][$i])
                                        ? $this -> map4[$j][$i]
                                        : $this -> infiniteDistance;
    $this -> usuario[$i][$j] = isset($this -> map5[$j][$i])
                                        ? $this -> map5[$j][$i]
                                        : $this -> infiniteDistance;
    }
    }
```

```
    for($i=0; $i < $this -> numberOfNodesfinal; $i++ ){
        $caminofinal[$i] = $this -> finalpath['1'][$i];
    }
    for($i=0, $j=1, $n=0; $i < ($this -> numberOfNodesfinal)-1; $i++, $j++,
$n++){
        $datofinal1[$n] = $this -> usuario[$caminofinal[$i]][$caminofinal[$j]];
    }
    for($i=0, $j=1, $n=0; $i < ($this -> numberOfNodesfinal)-1; $i++, $j++,
$n++){
        $datofinal2[$n] = $this -> control[$caminofinal[$i]][$caminofinal[$j]];
    }
    for($i=0, $n=0, $j=8; $i < ($this -> numberOfNodesfinal)-1; $i++, $n++){
        $datofinal[$n] = ($datofinal1[$n] + $datofinal2[$n]) * $j;
    }
    return $datofinal;
}
```

Que tiene como resultado:

```
Array
(
    [0] => 11200
    [1] => 8800
    [2] => 12000
)
```

y por último la función que asigna los tiempos de cola teniendo en cuenta que el nodo inicial y el nodo final tendrán tiempo de cola de 0:

```
function tiempoCola($colas, $inicio, $final){
    $caminofinal = array();
    $niveles = array();
    $this -> cola = $colas;
    for($i=0; $i < $this -> numberOfNodesfinal; $i++ ){
        $caminofinal[$i] = $this -> finalpath['1'][$i];
    }
```

```
    for($i=0; $i < $this -> numberOfNodesfinal; $i++){
        for($j=0; $j < 600; $j++){
        if( $caminofinal[$i] == $j){
        $datofinal[$i] = number_format($this -> cola[$j],10);
        break;
        }
    }}
    if ($datofinal[0] = $inicio){
        $datofinal[0] = 0;
    }
    if ($datofinal[(count($datofinal))-1] = $final){
        $datofinal[(count($datofinal))-1] = 0;
    }

    return $datofinal;
}
```

4.4 Función de Corte de Paquetes

Para poder sacar la latencia los paquetes se necesitan dividir según los datos de usuario dentro del camino recorrido y mostrara el número de paquetes y los paquetes obtenidos para esa tarea implemente 2 funciones:

```
function recorreNivel($niveles, $paquete, $nivelActual, $resultadoFinal) {


    //Esta es la condición de paro, continuo mientras la cantidad de niveles sea
menor que el nivel actual
    if ($nivelActual < count($niveles)) {
        $resultado2 = dividePaquete($niveles[$nivelActual], $paquete, array());
        //guardamos el resultado de cada paquete dividido en un array por
niveles
        foreach ($resultado2 as $res2)
            $resultadoFinal[$nivelActual][] = $res2;


        //incrementamos el nivel
        $nivelActual++;
        //Recorremos el resultado del paquete divido en el nivel actual para
obtener los nuevos resultados por cada nivel
        //Esto hace el ciclo recursivo
        foreach ($resultado2 as $res)
            $resultadoFinal = recorreNivel($niveles, $res, $nivelActual,
$resultadoFinal);
    }


    return $resultadoFinal;
}



/*
 * Esta función divide un paquete, según la cantidad que se coloque en nivel
 */
```

```php
function dividePaquete($nivel, $paquete, $resultado) {
    if ($paquete > $nivel) {
        $paquete = $paquete - $nivel;
        $resultado[] = $nivel;
        $resultado = dividePaquete($nivel, $paquete, $resultado);
    } else {
        $resultado[] = $paquete;
    }

    return $resultado;
}
```

Como resultado tiene lo siguiente:

```
Array
(
    [0] => Array
        (
            [0] => 1100
            [1] => 1100
            [2] => 1100
            [3] => 1100
            [4] => 600
        )

    [1] => Array
        (
            [0] => 700
            [1] => 400
            [2] => 700
            [3] => 400
            [4] => 700
            [5] => 400
            [6] => 700
            [7] => 400
            [8] => 600
        )

    [2] => Array
        (
            [0] => 700
            [1] => 400
            [2] => 700
            [3] => 400
            [4] => 700
            [5] => 400
            [6] => 700
            [7] => 400
            [8] => 600
        )

)
```

El resultado muestra el número de caminos que recorrió y los paquetes como se dividieron.

4.5 Calcular la Latencia

Teniendo ya todos los datos necesarios se procede a calcular la latencia con la siguiente función:

```
function latencia($resultadoFinal, $datoDistance, $datoControlUsuario,
$datoVelocity, $tiempoCola){
    $VelLuz = 300000000;
    for($i=0; $i < ($this -> numberOfNodesfinal-1); $i++){
        $lat += number_format(count($resultadoFinal[$i])*(($datoDistance[$i] /
$VelLuz) + ($datoControlUsuario[$i] / $datoVelocity[$i]) +
($tiempoCola[($i)+1])),50);
    }

    return($lat);
}
```

Tiene como resultado:

```
Latencia:
0.014470203333333
```

4.6 Calcular el Tiempo Total de Transferencia del Archivo

Y por último teniendo todos los resultados necesarios junto con la latencia procedemos a obtener el Tiempo Total de Transferencia del Archivo con la siguiente función:

```
function tiempototal($paquete, $usuario, $latencia){
    $tiempoTotal = ($paquete / $usuario['0']) * $latencia;

    return $tiempoTotal;
}
```

Y como resultado obtenemos:

```
Tiempo Total de Tranferencia del Archivo:
0.065773651515151521085300601043854840099811553955508
```

BENEMERITA UNIVERSIDAD

AUTONOMA DE PUEBLA

REDES DE COMPUTADORAS

DOCUMENTO TÉCNICO

PRIMER EXAMEN REDES DE COMPUTADORAS

CALCULAR LA LATENCIA Y ENCONTRAR EL CAMINO MÁS ÓPTIMO DE UNA RED

ALUMNO: SALVADOR AMADO SÁNCHEZ TOCHIHUITL

Introducción

Una red de computadoras ayuda a transmitir información entre distintos equipos, estas computadoras estando conectadas tiene un cierto rango de tiempo en el cual operan, con este proyecto calculamos ese tiempo llamado LATENCIA. La Latencia nos ayuda a saber el tiempo que tarda en llegar un paquete de datos y así obtener un TIEMPO TOTAL DE TRANSMISION.

La latencia se calcula con la siguiente formula:

L = tiempo de transmisión + tiempo de propagación + tiempo de cola.

Con esto calculamos la latencia de un nodo a otro. El programa consiste en calcular estos tiempos y encontrar la ruta más óptima, vamos a leer un archivo de texto el cual va a tener la información de entrada para nuestro programa.

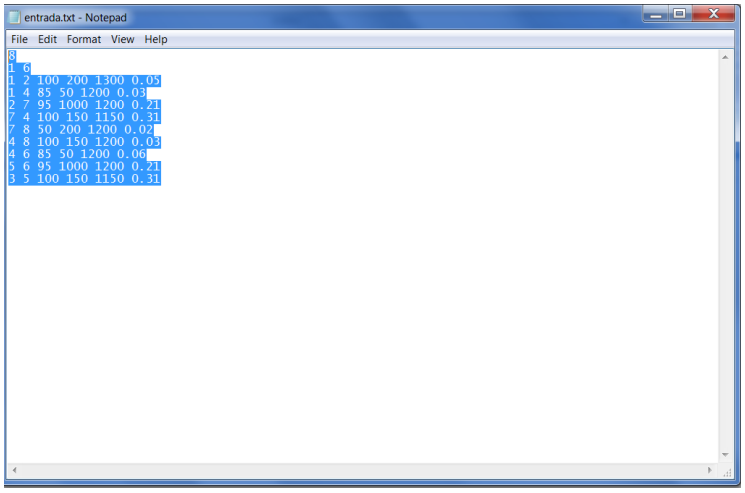
Para calcular el TIEMPO TOTAL DE TRASNMISION se divide el tamaño total del paquete entre los datos de usuario y se multiplica por la suma de todas las latencias del camino más corto encontrado.

Algoritmo Dijkstra

El algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista.

La idea subyacente en este algoritmo consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen, al resto de vértices que componen el grafo, el algoritmo se detiene.

Este algoritmo fue implementado en el lenguaje de programación java para poder calcular la latencia de cada nodo conectado con otro nodo.



Este es el formato de entrada que se va a manejar la primer variable es el numero de nodos que van a existir.

El segundo argumento es el nodo conectado y sus respectivos valores son la distancia, la velocidad de transmisión, los datos de usuario y el tiempo de cola.

```
//Esta clase requiere del uso adicionalmente de la clase Nodo, que va a servir para la cola de prioridad
//y para llevar registro de la distancia mínima desde el origen a un nodo, así como la referencia al nodo inmediatamente anterio

public class Nodo implements Comparable<Nodo> {
    char id;
    int  distancia   = Integer.MAX_VALUE;
    Nodo procedencia = null;
    Nodo(char x, int d, Nodo p) { id=x; distancia=d; procedencia=p; }
    Nodo(char x) { this(x, 0, null); }
    public int compareTo(Nodo tmp) { return this.distancia-tmp.distancia; }
    public boolean equals(Object o) {
        Nodo tmp = (Nodo) o;
        if(tmp.id==this.id) return true;
        return false;
    }
}
```

Esta imagen muestra el código que implementa el nodo el cual nos ayuda a identificar con quien está conectado y cual la latencia que tiene.

```
import java.util.*;
import java.io.*;
public class Grafo {

    char[]  nodos;  // Letras de identificación de nodo
    int[][] grafo;  // Matriz de distancias entre nodos
    String  rutaMasCorta;                          // distancia más corta
     List<Nodo>  listos=null;                       // nodos revisados Dijkstra
    static String NO,ND;
    static int NN;
    static int M[][] = new int[20][20];
    static double latTotal=0.0;
```

Aquí se muestra la declaración del programa principal y aquí se declaran las librerías las variables que se van a usar para poder efectuar las operaciones.

```
// construye el grafo con la serie de identificadores de nodo en una cadena
Grafo(String serieNodos) {
    nodos = serieNodos.toCharArray();
    grafo = new int[nodos.length][nodos.length];
}
```

Este código ayuda a construir una cadena con los nodos existentes para su manipulación

```
// asigna el tamaño de la arista entre dos nodos
public void agregarRuta(char origen, char destino, int distancia) {
    int n1 = posicionNodo(origen);
    int n2 = posicionNodo(destino);
    grafo[n1][n2]=distancia;
    grafo[n2][n1]=distancia;

}
```

Esta función ayuda a agregar los nodos e identificar que nodo está conectado con otro nodo y se le agrega la distancia en este caso la distancia lleva el cálculo de la latencia para poder buscar el camino más óptimo, esta variable no solo lleva la distancia sino que todas las operaciones necesarias para calcular su latencia.

```
// retorna la posición en el arreglo de un nodo específico
private int posicionNodo(char nodo) {
    for(int i=0; i<nodos.length; i++) {
        if(nodos[i]==nodo) return i;
    }
    return -1;
}
```

Esta función ayuda a regresar la posición de un nodo en específico para poder ver si está conectado con algún nodo.

```
// encuentra la ruta más corta desde un nodo origen a un nodo destino
public String encontrarRutaMinimaDijkstra(char inicio, char fin) {
    // calcula la ruta más corta del inicio a los demás
    encontrarRutaMinimaDijkstra(inicio);
    // recupera el nodo final de la lista de terminados
   Nodo tmp = new Nodo(fin);
    if(!listos.contains(tmp)) {
        System.out.println("Error, nodo no alcanzable");
        return " ";
    }
    tmp = listos.get(listos.indexOf(tmp));
    int distancia = tmp.distancia;
    // crea una pila para almacenar la ruta desde el nodo final al ori
    Stack<Nodo> pila = new Stack<Nodo>();
    while(tmp != null) {
        pila.add(tmp);
        tmp = tmp.procedencia;

    }
    String ruta = "";
    // recorre la pila para armar la ruta en el orden correcto
    while(!pila.isEmpty()) ruta+=(pila.pop().id + " ");
    double tampaq = ((4.5*1000*1000*1000)/1200);
    latTotal=tampaq+distancia;
    return "Latencia :" +distancia + " segundos , Ruta: " + ruta;
}
```

Esta es la función mas importante porque es la que verifica si el nodo destino esta en la ruta y si esta entonces empieza a sacar el camino mas optimo guardando los datos en una pila para poder hacer los recorridos y asi entregar un camino sus calores de entrada son el nodo inicio y el nodo final.

```
// encuentra la ruta más corta desde el nodo inicial a todos los demás
public void encontrarRutaMinimaDijkstra(char inicio) {
    Queue<Nodo>   cola = new PriorityQueue<Nodo>(); // cola de prioridad
    Nodo            ni = new Nodo(inicio);          // nodo inicial

    listos = new LinkedList<Nodo>();// lista de nodos ya revisados
    cola.add(ni);                   // Agregar nodo inicial a la cola de prioridad
    while(!cola.isEmpty()) {        // mientras que la cola no esta vacia
        Nodo tmp = cola.poll();     // saca el primer elemento
        listos.add(tmp);            // lo manda a la lista de terminados
        int p = posicionNodo(tmp.id);
        for(int j=0; j<grafo[p].length; j++) {  // revisa los nodos hijos del nodo tmp
            if(grafo[p][j]==0) continue;        // si no hay conexión no lo evalua
            if(estaTerminado(j)) continue;      // si ya fue agregado a la lista de terminados
            Nodo nod = new Nodo(nodos[j],tmp.distancia+grafo[p][j],tmp);
            // si no está en la cola de prioridad, lo agrega
            if(!cola.contains(nod)) {
                cola.add(nod);
                continue;
            }
            // si ya está en la cola de prioridad actualiza la distancia menor
            for(Nodo x: cola) {
                // si la distancia en la cola es mayor que la distancia calculada
                if(x.id==nod.id && x.distancia > nod.distancia) {
                    cola.remove(x); // remueve el nodo de la cola
                    cola.add(nod);  // agrega el nodo con la nueva distancia
                    break;          // no sigue revisando
                }
            }
        }
    }
}
```

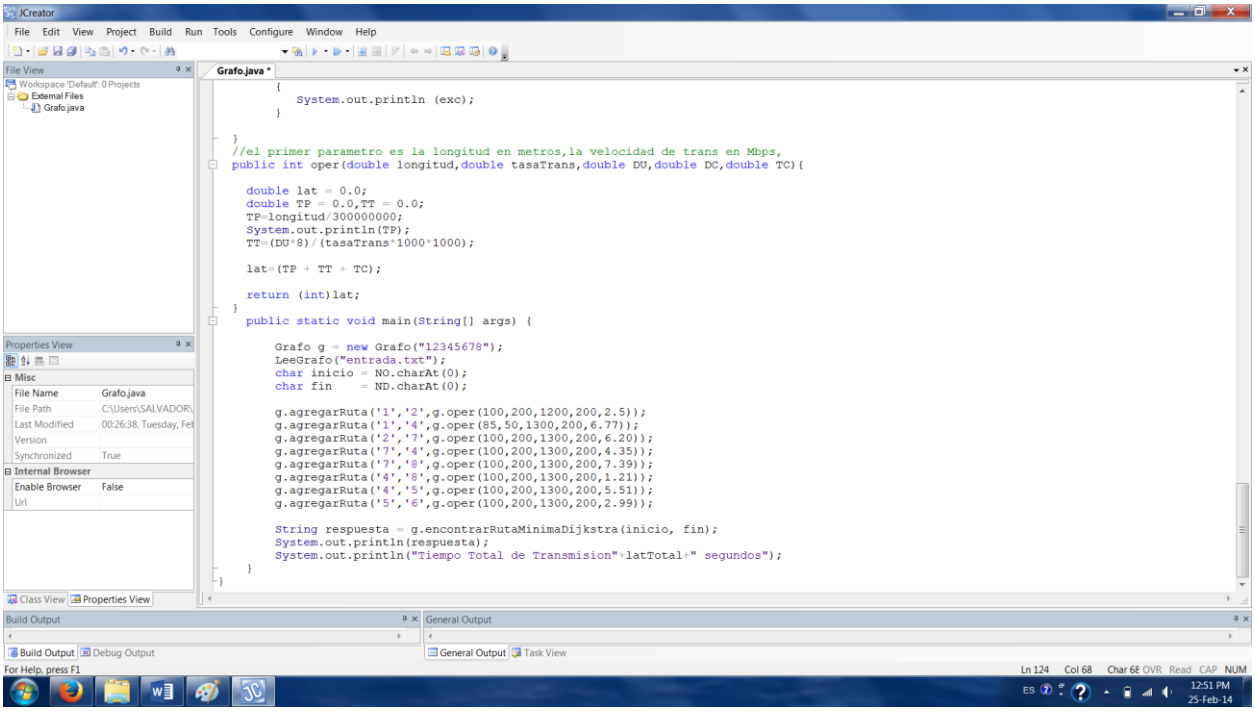
Esta función también calcula los camino que hay desde un nodo inicial pero no llega hasta el nodo final eso lo hace la función de arriba pero necesita de este para poder identificar los camino.

```
// verifica si un nodo ya está en lista de terminados
public boolean estaTerminado(int j) {
     Nodo tmp = new Nodo(nodos[j]);
     return listos.contains(tmp);
}
```

Esta función verifica que los nodos ya hayan sido recorridos correctamente

```
public static void LeeGrafo(String arch) //Lee archivo con los datos del grafo
 {
       FileInputStream fp;//declara una variable para poder leer el archivo de texto
       DataInputStream f;
       String linea = null;
       String token1,token2;// con los tokens se iran leyendo los datos en el archivo .txt
       int token3,token4,token5;
       int i=0,j=0;
       float token6;
       ArrayList a12 = new ArrayList();
       ArrayList<ArrayList<String>> array = new ArrayList<ArrayList<String>>();
       try
           {
               fp = new FileInputStream(arch);
               f = new DataInputStream(fp);
               linea=f.readLine();
               NN=Integer.parseInt(linea);
               System.out.println(" Numero de Nodos: "+NN);
               int a=0;
               // Leemos el archivo linea por linea
               do {
                   linea = f.readLine();
                   if (linea!=null){
                            StringTokenizer tokens=new StringTokenizer(linea);

                            if(a == 0){
                             NO = tokens.nextToken();
                             ND = tokens.nextToken();
                             a++;
                            }

                            else{
```

Con esta función se hace la lectura del archivo de texto que es la entrada pero tuve algunas dificultades para poder asignar los datos hubo errores que no pude resolver entonces la lectura la hice directamente en el método main del programa solo pude leer el archivo origen y destino los nodos conectados también los lei pero no pude asignar correctamente a la función para que los pudiera procesar.



Finalmente aquí se muestra la función que calcula la latencia y envía los datos para sacar el camino más óptimo del grafo.

Aquí está el método principal main que es donde adapte la entrada de los datos ya que con el archivo de texto me trabe en unas cosas, finalmente muestra los resultados de los y muestra el camino más corto , la latencia total y el tiempo total de transmisión expresado en segundos.

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

Materia
Redes de computadoras

Dr. Iván Olmos Pineda

Evaluación del Primer Parcial
"Generación de caminos posibles y cálculo de latencias"

Alumno
Federico Sandoval Zeron

Matricula: 200730273

Introducción

El problema que se ha trabajado consta de lo siguiente, consideremos una red de computadoras armada con N elementos (computadoras y routers), donde además de conocer las conexiones entre componentes tenemos los datos de **distancia** entre cada segmento, **velocidad** y **tamaño del paquete que soporta** trasportar por cada uno de los segmentos de la red, esto dividido en **datos de usuario** y **datos de control**.

Dado la información que tenemos a nuestra disposición nos hemos dado a la tarea de calcular los siguientes datos para la red dada:

1. Todos los caminos (rutas) posibles para enviar un archivo desde un nodo A hasta un nodo B y la *Latencia* de cada uno de ellos en la red.

2. El camino, cuya Latencia sea la mínima entre los nodos A y B, dependiendo del paquete que se envía (división de paquetes).

Problemas

Los problemas que atacaremos en el cálculo de los caminos, dependerá del algoritmo empleado en la solución de esta tarea.

La técnica que utilizaremos para recorrer el grafo es el algoritmo de "Recorrido en amplitud" aunque es costosa en memoria pero efectiva para la obtención de todos los caminos.

Después de haber solucionado el problema de guardar y obtener todos los caminos hay que enfrentarse a la tarea de calcular la *latencia* del camino completo. Esto se puede hacer de diferentes formas, en mi caso explicaré más adelante mi solución.

Después de conocer todas las latencias y caminos, habrá que seleccionar la de menor coste es decir la de la *Latencia* más pequeña

Resolucion.

Usamos esta estructura para llevar el control sobre la transmisión.
Lleva los datos de control Datos de usuario, Longitud del material, Velocidad de transferencia

Esta estructura la usamos para llevar el control de un camino, Nodos, guarda todos los nodos recorridos en éste camino.

Tenemos una clase que se encarga del cálculo de latencias.

Internamente maneja 2 grafos, uno que lleva la relación de transmisiones, Y otro que lleva la conexión de nodos.
Se puede leer un grafo de cualquier flujo de entrada

Calculamos el número de paquetes requerido para éste envío. Si llega a sobrar bytes con el número actual de paquetes creamos un nuevo paquete.

Cambiamos el tamaño del paquete actual del camino, sumando los datos de usuario con los de control, multiplicado por el número de paquetes.

Para éste algoritmo (amplitud) necesitamos saber si podemos agregar un nodo, al camino actual o no, Si no son adyacentes el nodo actual con el candidato entonces no se agrega.
Si llegara a existir el candidato dentro del camino actual entonces tampoco se agrega. De otra forma se agrega al camino. Y se actualiza la latencia

Experimentos
Como experimento y ejemplo utilicé el siguiente grafo (red):

numero nodos
numero de aristas
n origen n destino
numero de router
arcos (ini, fin, distancia(mts), velocida(Mbps), dc(Mbytes), du(Mbytes))
nodo-router tiempo de cola(segundos)
tam paquete (Mbytes)

Datos en
4
3
1 4
2
1 2 200mts 500Mbps 500Mbytes 1000Mbytes
2 3 95mts 10Mbps 100Mbytes 1500Mbyes
3 4 110mts 100Mbps 300Mbytes 200KMytes
2 0.0025s
3 0.53s
1000Mbytes

Resultados
Los resultados arrojados por la entrada antes mencionada fueron los siguientes:

1 -> 2 -> 3 -> 4 | latencia: 996.243

El camino menor (por latencia) es:

1 -> 2 -> 3 -> 4 | latencia: 996.243

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

REDES DE COMPUTADORAS

EXAMEN PARCIAL 2

Comprobación de Redundancia Cíclica (CRC).

Catedrático:

Dr. Ivan Olmos Pineda

Alumno:

Ernesto Daniel Serrano Romero.

07 de Abril del 2014.

Objetivos:

1. Aplicar el método de verificación de redundancia cíclica (CRC) a un mensaje (conjunto de bits) enviado por un emisor.
2. Comprobar la integridad del mensaje enviado, determinando si existen errores en la transmisión, a través de un receptor.

INTRODUCCIÓN

Los códigos de Redundancia Cíclica son ampliamente usados dentro de las Redes de Computadoras. Es un método de control de integridad de datos de fácil implementación. Es el principal método de detección de errores utilizado en las telecomunicaciones.
Las CRC son populares porque su implementación en *hardware* binario es simple, son fáciles de analizar matemáticamente y son particularmente efectivas para detectar errores ocasionados por ruido en los canales de transmisión.
La CRC fue inventada y propuesta por W. Wesley Peterson.

MARCO TEÓRICO

Se basa en representar las cadenas de datos como polinomios. El emisor realiza ciertas operaciones matemáticas antes de enviar los datos. El receptor realizará, a la llegada de la transmisión, una división entre un polinomio convenido (polinomio generador). El residuo de ésta operación determinará si el mensaje transmitido tiene errores o no los tiene.
Existen varios polinomios generadores en la actualidad, aunque los más comunes son los siguientes:

- **CRC-12 (x12 + x11 + x3 + x2 + x + 1):** Usado para transmitir flujos de 6 bits, junto a otros 12 de redundancia.

- **CRC-16 (x16 + x15 + x2 + 1):** Para flujos de 8 bits, con 16 de redundancia. Usado en USA, principalmente.

- **CRC-CCITT (x16 + x12 + x5 + 1):** Para flujos de 8 bits, con 16 de redundancia. Usado en Europa, principalmente.

- **CRC-32 (x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1):** Da una protección extra sobre la que dan los CRC de 16 bits, que suelen dar la suficiente. Se emplea por el comité de estándares de redes locales (IEEE-802) y en algunas aplicaciones del Departamento de Defensa de USA.

Una vez que se ha definido que polinomio generador **G(x)** se usará, es necesario realizar la suma de comprobación:

EMISOR:

- Añadir **r** bits de 0 a **M(x)** (el mensaje), produciendo **B(x)**.
- Dividir **B(x)** por **G(x)** usando división en módulo 2, produciendo un resto **R(x)**.
- Generar **T(x)**, que es el resultado de la resta en módulo 2 de **B(x)** – **R(x)**.
- Transmitir (escribir a fichero) **T(x)**. **T(x)** es divisible por **G(x)**.

RECEPTOR:

- Realizar división en módulo 2 entre **T(x)** (lectura del fichero generado por el emisor) y **G(x)**.
- Si no existen errores de transmisión, el residuo resultante será 0.
- Si hay errores en la transmisión, recibiremos un residuo diferente a 0.

IMPLEMENTACIÓN

El lenguaje de programación utilizado en la implementación del proyecto fue Java. Se consideraron cuatro aspectos principales en la implementación:

- **Lectura del mensaje original (Emisor).**
- **Procesamiento de datos y cálculos (creación de métodos para cálculos en módulo 2).**
- **Transmisión de T(x) (Mensaje de comprobación).**
- **Lectura del mensaje transmitido y verificación de errores (Receptor).**

Lectura del mensaje original (Emisor).

El emisor realizará la lectura del mensaje original contenido en un fichero especificado por el usuario, a través de la clase *FileReader*, contenida en la librería de JAVA *java.io.*Reader.
Los datos se leen en formato *Integer*. A medida que el contenido del fichero es leído, el emisor utiliza el método *toBinaryString* para convertir los datos leídos a un formato binario.

Procesamiento de datos y cálculos (creación de métodos para cálculos en módulo 2).

Una vez leídos los datos de entrada, se crearon los métodos necesarios para realizar división y resta en módulo 2. Para ello se usa un ciclo *while* que censa la longitud del mensaje modificado **B(x)** y toma **r** dígitos de éste, siendo **r** la longitud de **G(x)**, para después aplicar un **or exclusivo (XOR)** a cada elemento de **B(x)** correspondiente a un elemento de **G(x)**.
El algoritmo usado para éste método es el siguiente:

```
int[] dividendo= new int[Gx.length];
int[] aux= new int[Gx.length];
i=0;
j=0;
k=0;
while(i<Bx.length)
{

      while(j<dividendo.length)
      {
          if(i<Bx.length)
          dividendo[j] = Bx[i];
          j++;
          i++;
      }

      for(k=0;k<Gx.length;k++)
      {
          aux[k]= dividendo[k] ^ Gx[k];
      }
      k=0;

      if(i<Bx.length)
      {

      while(aux[k]==0)
          k++;

      j=0;

      while(k<aux.length)
      {
          dividendo[j]= aux[k];
          j++;
          k++;
      }
```

De igual forma, se crea un método para calcular la resta en módulo 2, que opera de manera similar al método anterior.

El algoritmo usado para éste método es el siguiente:

```
int[] Tx= new int[Bx.length];
Tx= Bx;
i=Bx.length-1;
j=aux.length-1;
k=0;

while(j>=lim){

 Tx[i]= Bx[i] ^ aux[j];
 j--;
 i--;
       }
```

Transmisión de T(x) (Mensaje de comprobación).

El emisor realiza la transmisión del mensaje de comprobación **T(x)** generado a través de la escritura a un fichero, haciendo uso de la clase *FileWriter*, contenida en la librería de JAVA *java.io.*Writer.
El fichero generado se guarda con el mismo nombre que el original, pero se le agrega la extensión ".crc". Este fichero será leído por el receptor para la verificación de errores.

Lectura del mensaje transmitido y verificación de errores (Receptor).

El receptor realizará la lectura fichero en formato ".crc", generado por el emisor, a través de las clases *FileReader* y *BufferedReader*, contenidas en la librería de JAVA *java.io.*Reader.
Una vez leído el fichero, el receptor realiza la división en módulo 2 entre **T(x)** y **G(x)** usando el método mencionado anteriormente. Finalmente, se comprueba si el residuo es cero usando el siguiente algoritmo:

```
for(i=0;i<aux2.length;i++){                                    //Se comprueba si el residuo es cero

 if(aux2[i]!=0)
 {
     System.out.print("\n\nEl residuo indica que hubo un error en la transmision. Finalizando programa...");
     System.exit(0);
 }
                          }

 System.out.print("\n\nEl residuo indica que no hubieron errores en la transimision. Finalizando programa...");
```

CONCLUSIONES

La comprobación de redundancia es un código de detección de errores que ha demostrado ser confiable, efectivo y su implementación es relativamente simple. A pesar de ello, y como todo método de detección y corrección de errores, no es un código libre de vulnerabilidades. CRC es útil para detección de errores, pero, en condiciones de seguridad, no podemos confiar en que el CRC puede verificar plenamente que los datos son los correctos en caso de que se hayan producido cambios deliberados y no aleatorios, es decir, sirve para verificar la integridad, pero no para saber si el mensaje es correcto.

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

REDES DE COMPUTADORAS

EXAMEN PARCIAL 1

Topología de una red.

Catedrático:

Dr. Ivan Olmos Pineda

Alumno:

Objetivos:

3. Diseñar la topología de una red para simular la transferencia de paquetes, considerando los conceptos de tiempo de propagación, tiempo de cola, latencia, tamaño de paquetes y tiempo de transferencia.
4. Obtener el tiempo de transferencia de un archivo de un tamaño especificado por el usuario.
5. Conocer el camino más corto de la red (esto es, el que tiene menor latencia).

INTRODUCCIÓN

La definición más clara de una red es la de un sistema de comunicaciones, ya que permite comunicarse con otros usuarios y compartir archivos y periféricos. Es decir, es un sistema de comunicaciones que conecta a varias unidades y que les permite intercambiar información.
Se entiende por red al conjunto interconectado de ordenadores autónomos.
Se dice que dos ordenadores están interconectados, si éstos son capaces de intercambiar información.
El proyecto tiene como fin estudiar y conocer cuáles son los aspectos que determinan el comportamiento de una red, y por tanto, de los paquetes que transitan por ésta.

MARCO TEÓRICO

La cantidad de datos transmitida por una red aumenta debido al número de usuarios de las redes así como a la complejidad de los datos transmitidos
Existen dos variables para mediar la capacidad de transmisión:

- **Tasa de transferencia:** Se refiere a la velocidad con la cual podemos transferir información y se mide en bits/seg. A su vez, esta se divide en:
    - Tasa de transferencia teórica.
    - Tasa de transferencia real.
- **Latencia:** Tiempo que le toma un bit viajar de un extremo de un medio al otro. Depende de tres factores:
    - Tiempo de propagación del bit por el medio.
    - Tiempo de transmisión.
    - Tiempos de espera (tiempo de cola).

Las variables anteriores están relacionadas a través de las siguientes ecuaciones:

Latencia *= Tiempo de propagación + Tiempo de transmisión + Tiempo de cola*
Donde:
Tiempo de propagación *= distancia a recorrer / velocidad de la luz*
Tiempo de transmisión *= tamaño del paquete / tasa de transferencia teórica*

IMPLEMENTACIÓN

El lenguaje de programación utilizado en la implementación del proyecto fue Java. Se consideraron tres aspectos principales en la implementación:

- **Lectura y creación de la red (grafo).**
- **Procesamiento de datos y cálculos.**
- **Algoritmos de búsqueda y de recorrido de un grafo.**

Lectura y creación de la red (grafo).

Para la creación de la red se hizo uso de un grafo ponderado, en donde el peso de cada arista (conexión) se determina en función de los datos proporcionados por el usuario/archivo, tales datos son la distancia de un nodo a otro (dispositivos), velocidad de transferencia, tamaño de paquete y tamaño de datos de usuario. Una vez ingresados los datos, se procede al cálculo de latencias.

Procesamiento de datos y cálculos.

Ésta parte de la implementación usa los datos ya leídos para calcular la latencia de cada arista (conexión) entre dos dispositivos. Debido a la naturaleza de una red, no es posible realizar los cálculos de manera directa, por lo que se diseñó un algoritmo que evalúa el tamaño de los datos de usuario y los compara con el tamaño de paquete. De ser necesario, éste algoritmo

realiza la división de paquetes correspondiente. Para lo anterior se hizo uso de una **Cola**, la cual a su vez hace uso de una **Lista Ligada.** El algoritmo es el siguiente:

```
public static float dividePaquetes(float datUser, float tamMaxDatos,Cola colaAux)
{
    float tam= colaAux.NumElementos();
    float aux= datUser;
    float multip=1;
    if(aux<=tamMaxDatos)
    {
        colaAux.insertaFinal(aux);
        colaAux.borraInicio2();
        multip=colaAux.NumElementos();
        return multip;
    }

    else
    {
    while(aux > tamMaxDatos)
    {
        colaAux.insertaFinal(tamMaxDatos);
        aux-=tamMaxDatos;
    }
    colaAux.insertaFinal(aux);
    colaAux.borraInicio2();
    multip= colaAux.NumElementos();
    return multip;
    }
}
```

Una vez calculadas las latencias de cada conexión se muestra la topología de red y se solicitan los datos para el cálculo del camino más eficiente, como en el siguiente ejemplo:

General Output

```
Topologia de la Red:
1:--> 2, 0.370317ms
2:--> 3, 2.024534ms
3:--> 4, 3.121110ms
El dispositivo 4 no tiene adyacencias(conexiones)


Encontrando ruta mas corta,ingrese datos:

Tamano de Archivo a transferir(en GB):
```

<u>Algoritmos de búsqueda y de recorrido de un grafo.</u>

Para la búsqueda del recorrido con la menor latencia, se implementó el **algoritmo de Dijkstra** sobre la topología de red. Éste algoritmo recibe como entrada el nodo o dispositivo inicial, el

dispositivo final y el tamaño del archivo a transferir. Fue necesario calcular previamente el peso o latencia de cada conexión, ya que esto facilita los cálculos para determinar el camino más eficiente.

Éste es el algoritmo que se implementó:

```
void caminoMinDijkstra(int inicio, int fin,float tamArchivo,float datUser2)
{
    this.inicio=inicio;
    this.fin= fin;
    float d[] = new float[numNodos+1];
    p= new int[numNodos+1];
    boolean s[] = new boolean[numNodos+1];
    int aux;
    NodoCab aux2=iniGrafo;
    NodoL aux3;
    float minimo;
    int v;

    while(aux2.vertice!=inicio)
        aux2=aux2.sigCab;
    aux3=aux2.listA.iniListaAdy;

    //Se llena d
    for(int i=1; i<=numNodos; i++)
    {
        if(i==inicio)
            d[i]=-1;
        else
        {
            if(aux3!=null){
            if(aux3.vertice==i)
                {
                d[i]=aux3.peso;
                aux3=aux3.sigList;
                }
            else
                d[i]=Integer.MAX_VALUE;
                    }
                }
                s[v]=true;
                aux2=iniGrafo;
                while(aux2.vertice!=v)
                    aux2=aux2.sigCab;
                    aux3=null;
                aux3=aux2.listA.iniListaAdy;
                while(aux3!=null)
                {
                    if(s[aux3.vertice]==false && (d[v]+aux3.peso)<d[aux3.vertice])
                    {
                        d[aux3.vertice]=d[v]+aux3.peso;
                        p[aux3.vertice]=v;
                    }
                    aux3=aux3.sigList;
                }

      }
```

```
                else
                    d[i]=Integer.MAX_VALUE;
        }
    }

    //Se llenan p y s
    for(int i=1; i<=numNodos; i++)
    {
        if(i==inicio)
        {
            p[i]=-1;
            s[i]=true;
        }
        else
        {
            p[i]=inicio;
            s[i]=false;
        }
    }

    for(int i=1; i<numNodos; i++)
    {

        minimo=Integer.MAX_VALUE;
        v=0;

        for(int j=1;j<=numNodos;j++)
        {
            //Buscar minimo
            if(d[j]!=-1.0 && d[j]<minimo && s[j]==false)
            {
                minimo=d[j];
                v=j;
```

Para el cálculo de todos los caminos posibles se usó un algoritmo “Backtracking / Fuerza Bruta” con algunas modificaciones. Originalmente, éste algoritmo calcula el camino mínimo entre dos vértices a través del método de Fuerza Bruta, es decir, obtiene todos los caminos posibles entre el par de vértices y compara el costo de cada uno. En nuestro caso, se eliminaron las condiciones de comparación de costo y, siempre que se completaba un camino, se mostraba al usuario y continuaba el proceso. Éste es el método:

```
void caminosPosibles2(int inicio, int fin)
{
    if(borrar==true)
    {
        Colav=new Cola();
        Colav.insertaFinal(inicio);
        marcas=new boolean[numNodos+1];
```

CONCLUSIONES

A pesar de que en la actualidad existe software que realiza cálculos relacionados con el tiempo de transferencia y paquetes en una red (como es el caso de Cisco Packet Tracer), el presente proyecto constituye una oportunidad de aprendizaje para que el estudiante conozca cómo opera una red y sepa como programar el funcionamiento de la misma. También fue posible relacionar conocimientos previos (en especial, teoría de grafos) para la representación de la red.

Examen

Humberto Valencia Toxqui

Redes de Computadoras

Se demostrara el envió de paquetes en una red, calculando todas la rutas posibles posibles con el uso de teoría de grafos y el lenguaje de programación C#

Para el cual de todas las rotas usaremos el algoritmo, El cual contiene el uso de backtracking

Algoritmo en pseudocódigo:

```
1  método DFS( origen,final):
2      marcamos origen como visitado
3      recuperar el path si se llego a final
4          para cada vertice v adyacente a origen en el Grafo:
5              si v no ah sido visitado:
6                 marcamos como visitado v
7                 llamamos recursivamente DFS( v )
8      marcamos origen como no visitado
```

Para la representación del grafo usaremos el siguiente tipo de formato

% NumeroNodos NumeroEnlaces

4 5

% NodoOrigen NodoDestino

1 4

% NodoOrigen NodoDestino Longitud VelocidadTeorica DatosControl DatosUsuario

1 2 20 10 20 30

1 3 10 20 10 20

2 3 40 60 10 50

2 4 80 120 10 30

3 4 5 100 10 30

% TiempoCola

0.0

0.350

0.500

0.0

o

GRAFICACIÓN PRIMAVERA 2014

Practica 2

Objetivo

En este segundo parcial se busca implementar las funciones de translación, rotación y escalamiento esto con el propósito de darle animación a un escenario

3D el cual sin el uso de las funciones que trae opengl por defecto, las figuras sean capaces de moverse de un punto a otro, rotarse respecto a un eje de rotación y crecer respecto a algún escalar.

Esto para saber cómo es que funcionan internamente dichas funciones y tener una idea más general de cómo es que se transforma cada punto para lograr alguna de estas operaciones.

En este parcial se hará un escenario el cual debe de llevar las 3 operaciones básicas antes mencionadas, este escenario se trabajara en el plano 3D (X, Y, Z).

Se estará trabajando como escenario una vista de un cuarto con una ventana que tiene una vista del cielo el cual cambiara de color respecto a la posición del sol y la luna estas se moverán de un lado a otro simulando el paso del tiempo, en el escenario tendrá un televisor en el cual se estará viendo un cubo crecer y decrecer.

Al sol y a la luna se le aplicara translación para que vallan de un lado del escenario al otro extremo, lo que se sintoniza en el televisor se le aplicara el escalamiento (cubo).

Antecedentes

Se necesita saber cómo aplicar las operaciones translación, rotación y escalamiento estas las hemos visto en clase.

También se necesita conocer las primitivas de opengl para dibujar pues nuestro escenario y saber cómo colorear cada figura.

Como se implemento

Se necesita saber y no solo eso sino comprender como es la estructura de nuestro ciclo infinito para poder limpiar pantalla sin afectar a las demás figuras

La estructura básica para animar es:

Ciclo

Operaciones

Dibujar figura

Pintar figura

Sleep

Limpiar pantalla

Ciclo infinito

Primero dibujo mi escenario para esto utilizo estas funciones las cuales son absolutas:

paredes();

ventana();

piso();

mesa();

cuadro();

televisor();

Estos dibujos no se mueven están dados los puntos en 3D sin que se les realice ningún tipo de operación.

En las funciones:

sol(z);

Se hace una translación de esta figura dándole como parámetro la rapidez de con la que se mueve (la cantidad que se moverá cada que se redibuje) estas operaciones se aplican directamente a los puntos ya que no requieren de más matrices

luna(z);

Se hace una translación de esta figura dándole como parámetro la rapidez de con la que se mueve (la cantidad que se moverá cada que se redibuje) estas

operaciones se aplican directamente a los puntos ya que no requieren de más matrices

tele(k);

Esta función dibuja lo que se está visualizando en la televisión (un cubo) que está escalándose a una rapidez de k veces.

ControlMov();

Esta función controla el movimiento de la luna y el sol ya que regresa a su posición original a los cuerpos una vez concluido su recorrido.

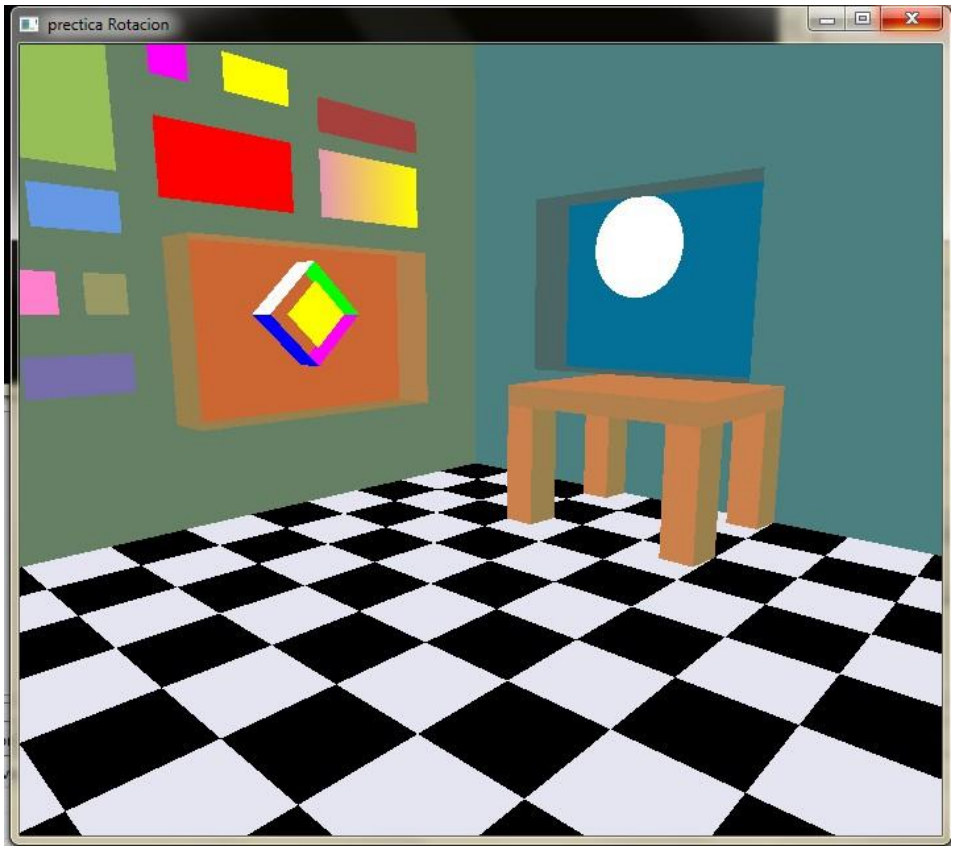
controlC();

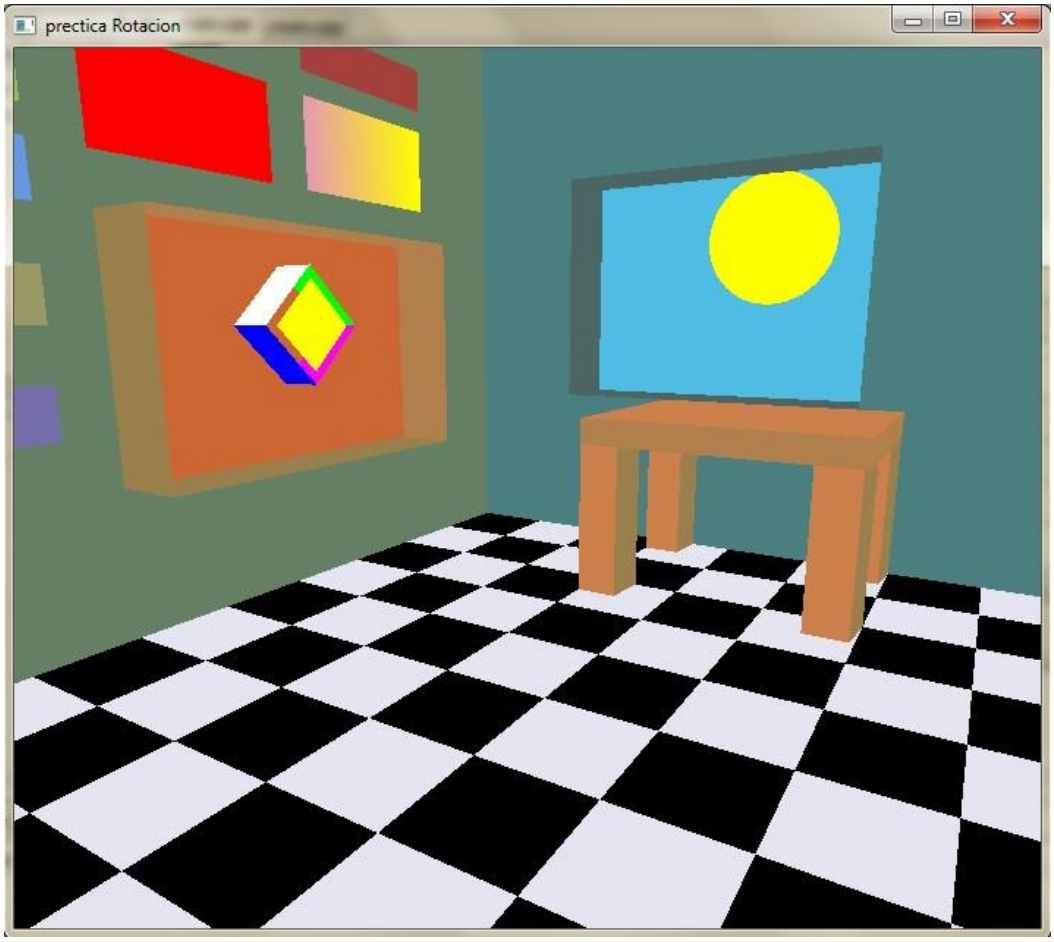
Esta función manda a llamar al cielo según la posición del sol y la luna.

keyboard(unsigned char key, int x, int y);

He implementado esta función que controla la posición del observador mediante el uso del teclado.

Resultados


prectica Rotacion


practica Rotacion

Benemérita Universidad Autónoma de Puebla

Reporte del 3° Proyecto

Graficación

Miguel Ángel Hilario Xelano

Introducción:

La meta de este proyecto es que como estudiante de Graficación ponga en práctica lo estudiado en estos meses de clase, en este caso el trabajo consiste en realizar un escenario en 3D de lo que elijamos.

Desarrollo:

Para lograr nuestro cometido se hizo uso de las librerías de Opengl:

- **corona:** Nos permite utilizar texturas en el proyecto.
- **Gl/glut:** Contiene las instrucciones para inicializar el ambiente gráfico.
- **Glm:** Biblioteca que nos permite cargar los modelos 3D.

También para lograr nuestro cometido se hizo uso de las primitivas de figuras:

- **GL_QUADS:** Función que nos permite definir un cuadrado.
- **glutSolidSphere:** Función que me permite dibujar un círculo.

Entre las funciones de animación usamos:

- **glTranslated:** Permite trasladar una figura a otro punto.
- **glRotatef:** Nos permite rotar la figura en un ángulo.

Carga de Texturas:

- **glm**: Nos permite cargar Modelos hechos en 3d.
- **glmRead**: Nos permite leer la textura
- **glmDraw:** Nos permite dibujar la textura

Iluminación:

- **glEnable(GL_LIGHTING):** Activa la Iluminación en el escenario.
- **glLightfv();** Parametros para modificar el tipo de luz.
- **glDisable(GL_LIGHTING);** Desactiva la iluminación, también puede usarcé para:
    - **glDisable(GL_LIGHT0);** Desactivar la luz que se ocupa

Ahora el escenario que usamos como referencia fue:

Para poder dibujar el escenario lo primero es preparar el ambiente para poder dibujar, lo cual es mediante:

Main():

```
int main(int argc, char **argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
   glutInitWindowPosition(200, 150);
   glutInitWindowSize(700, 500);
   glutCreateWindow("Triangulo a color");
   //if(!LoadTextures()) return 0;
   init();
   glutDisplayFunc(display);
```

```
        glutIdleFunc(idle);     //no hacer nada
   glutMainLoop();
   return 0;
}
```

En esta parte se declara la posición de la ventana, el tamaño, el nombre, que se repita, entre otras cosas.

Lo segundo es inicializar el ambiente:

```
void init()
{
   //Definimos el aspecto o rango de vision en x para el observador
   GLfloat aspect = (GLfloat)(400/400);
   //Limpiamos la pantalla y definimos el color de fondo
   glClearColor(0.0,0.0,0.0,0.0);
   glMatrixMode(GL_PROJECTION);          //Cargamos la matriz de proyeccion
   glLoadIdentity();                     //Cargamos la matriz de identidad
   glOrtho(xmin,xmax,ymin,ymax,zmin,zmax); //Definimos nuestro entorno de trabajo
   gluPerspective(60.0f,aspect,1.0f,400.0f);  //Cargamos un vision en perspectiva
   glMatrixMode(GL_MODELVIEW);           //Cargamos la matriz de modelado
   glLoadIdentity();                     //Cargamos la matriz de identidad
   glEnable(GL_DEPTH_TEST);
   cargar();
   cargarmodelos();
   ancho = 400;
   alto = 400;
   return;
}
```

Esta función permite iniciar todo el ambiente en el cual se estará desarrollando el escenario, con estos dos pasos se ha iniciado el ambiento y ahora podemos dibujar todo lo que querremos.
En este caso se hace uso de la función **glOrtho()** ya que con esta se define un escenario 3D con coordenadas **x,y,z.**

En esta parte también se agregan dos funciones las cuales son **cargar()** y **cargarmodelos()** las cuales explicaremos mas adelante.

Ahora para poder dibujar todo lo que contiene el escenario necesitamos el uso de algunas funciones las cuales son:

Display():

Esta función se encarga de llamar todas las demás que serán dibujadas

```
void display()
{
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

   glPushMatrix();
      glTranslatef(-5,-5,-5);
      Modelos();
      luna();
      contenedor();
      piso();
      poste(440);
      banca();
      gaviotas();
   glPopMatrix();

   glFlush();
   glutSwapBuffers();
   return;
}
```

Modelos();

Esta función nos permite cargar modelos 3D con extensión **.obj** los cuales se pueden bajar de internet de forma gratuita la maner a de cargarlos es igual como las imágenes:

- Cargas la Biblioteca “gml.h” y “gml.cpp”
- Crear una función que cargue las texturas “**Cargar()**”

```
void cargar()
{
  corona::Image* imagen=NULL;
  for(int i=0;i<NTextures;i++)
  {
    imagen = corona::OpenImage(texturefiles[i], corona::PF_R8G8B8A8);
        glGenTextures(1, &textures[i]);
        glBindTexture(GL_TEXTURE_2D, textures[i]);
        gluBuild2DMipmaps( GL_TEXTURE_2D, 4, imagen->getWidth(), imagen->getHeight() ,
        GL_RGBA, GL_UNSIGNED_BYTE, imagen->getPixels() );
        glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST_MIPMAP_LINEAR );
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST_MIPMAP_LINEAR );
        glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S, GL_REPEAT);
        glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T, GL_REPEAT);
        glTexEnvf(GL_TEXTURE_2D,GL_TEXTURE_ENV_MODE,GL_DECAL);
        if(imagen!=NULL) delete imagen;
  }
}
```

Esta función carga los modelos y todos sus parámetros para poder hacer uso de ellos en el entorno grafico.

- Crear una estructura de tipo modelo que contiene un puntero “***identificador**”

```
typedef struct model
{
GLMmodel *identificar;
};
```

Con esta estructura podemos apuntar a el modelo y hacer uso de el.

- Crear un arreglo del tipo estructura creada

```
model cont[NModelos];
```

Contendrá todos los punteros de los modelos que se usaran.

- Crear un arreglo que contendrá los nombres de la estructura

```
char *nombre[]={
   "reja.obj","Jeep.obj","post.obj","banca.obj","GULL.OBJ"
};
```

Contiene el nombre de las estructuras que se cargaran.

- Crear una función que cargue los modelos para hacer uso de ello.

```
void cargarmodelos()
{
```

```
 for(int i=0;i<NModelos;i++)
 {
     cont[i].identificar=glmReadOBJ(nombre[i]);
     if(cont[i].identificar==NULL)
     {
        printf("\n Error al cargar el Modelo \n");
        system("pause");
        exit(0);
     }
 }
}
```

Esta función carga los modelos en el arreglo del tipo estructura donde se guarda la dirección de los modelos cargados.

Una vez hecho esto podemos usar las texturas y cargarlas, si es que no están corruptas o compatibles con el formato.

Luna():

Esta función tiene como tarea dibujar una esfera que hara el papel de “Luna” en nuestro escenario.

Para hacer esto se hizo uso de la función:

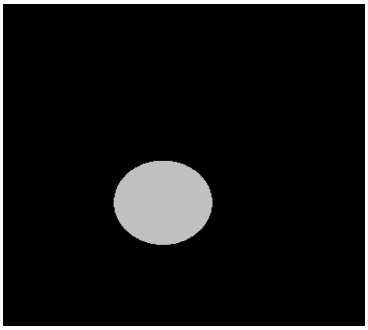
glutSolidSphere(r,100,100);

Esta función lleva como parámetros el radio, y el número de cuadros que la conforman.

Código:

```
void luna()
{
    glColor3f(0.752941,0.752941,0.752941);
     glTranslatef(5.0,5.0,0.0);
     glutSolidSphere(1.0,100,100);
}
```

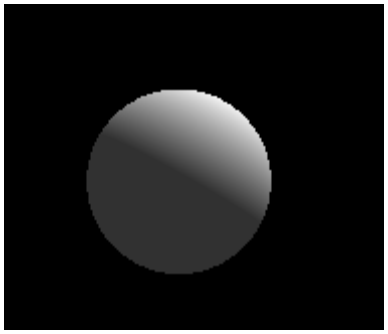
Ese es el código que se usa para crear la luna en el escenario y luce así:



A esta función también le asignamos una iluminación la cual esta codificada de la siguiente manera:

```
float luna[]={5.0,9.0,0.0,0.0}; //Posición de la luz
float luz_AmbienteL[]={0.752941,0.752941,0.752941,1.0}; //Color de la luz aplicada
glLightfv(GL_LIGHT0,GL_POSITION,luna); //Cambio de Parámetros para la luz
glLightfv(GL_LIGHT0,GL_AMBIENT,luz_AmbienteL);
glEnable(GL_LIGHT0);  //Activacion de la luz
        “Codigo para dibujar la luna”
gllDisable (GL_LIGHT0);  //Desactivación de la luz
```

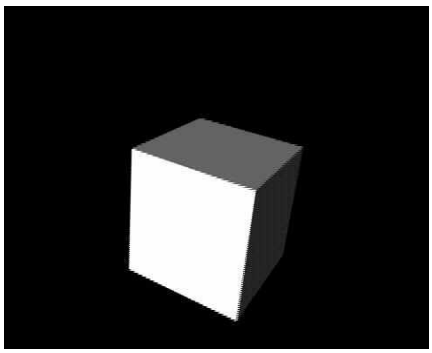
Una vez hecho esto la luna luce así:



Contenedor():

Esta función contiene el codigo para hacer el cubo que hace de contenedor del escenario, el cual no es más que un cubo de 10*10*10 en todos los ejes (x, y, z, -x, -y,-z).

El cubo se ve de la siguiente forma:

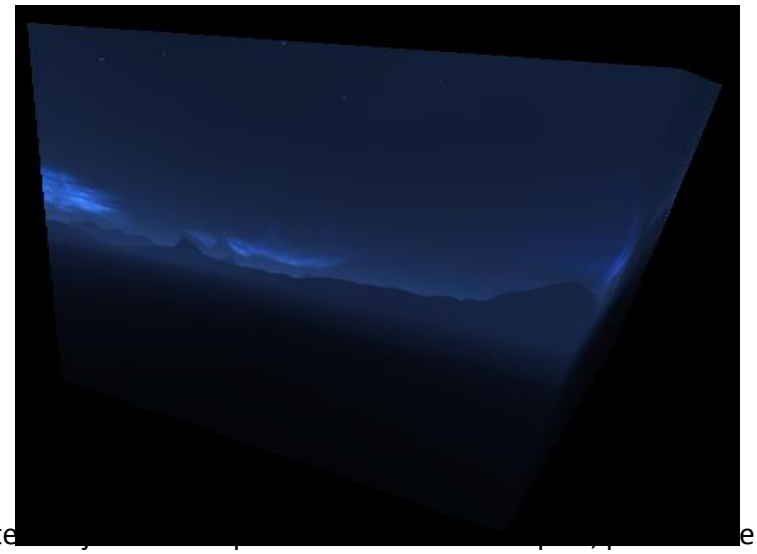


Para texturizar el cubo se hizo uso de las siguientes texturas:

Para poder texturizarlo se usó las funciones:

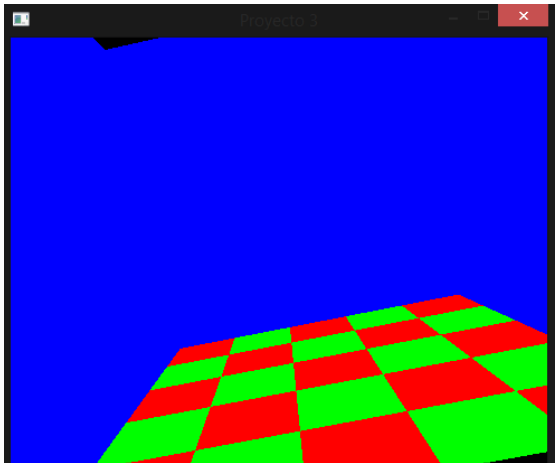
- **glBindTexture();** Con el cual se carga la imagen que se texturizara.
- **glTexCoord2f();** Se define las coordenadas en las que se pondrá la imagen en el cuadro (4 en total).

El cubo texturizado luce de la siguiente manera:



Piso();

Esta función nos permite e hace un tablero de ajedrez el cual es texturizado de diferente forma para lograr hacer que se vea de la siguiente manera:



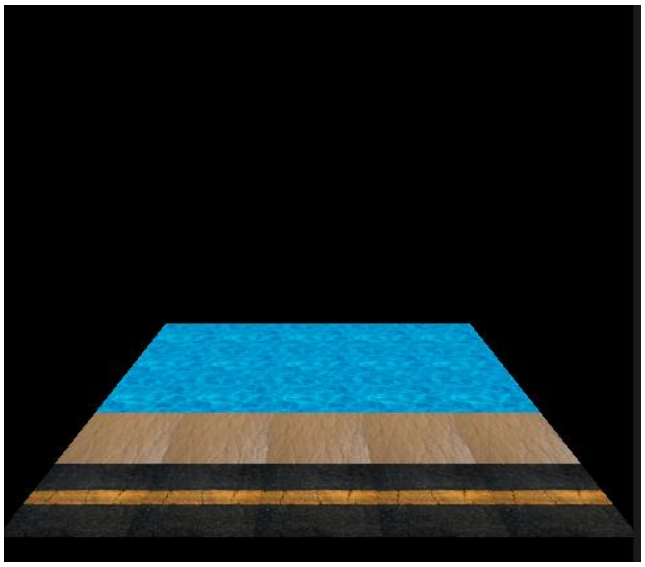
Para texturizar cada cubo se hizo lo mismo que en el contenedor y se usaron las siguientes imágenes,

Arena

Carretera

Agua

Una vez texturizado el piso luce así:



Poste();

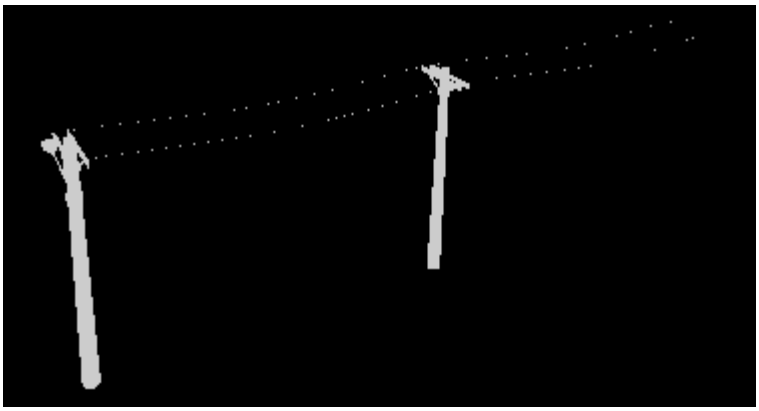
Esta función nos permite dibujar una textura 3d denominada “poste.obj”, para esto se hace uso de la función “**cargar()**” y “**cargarmodelo()**” con las que se hace el llamado a la textura:

Codigo:

```
void poste(int x)
{
```

```
    glColor3f(0.80,0.80,0.80);
    glScalef(0.01,0.01,0.01);
    glTranslatef(x,0,820);
    glPushMatrix();
      glRotatef(90,0,1,0);
      glmDraw(cont[2].identificar, GLM_MATERIAL|GLM_SMOOTH);
    glPopMatrix();
}
```

Con este código se dibuja un poste el cual es escalado a la medida del escenario, trasladado y rotado de manera en la que quede de manera correcta y homogénea al escenario.



Banca();

Esta función nos permite dibujar una banca, la cual es un modelo 3D que fue cargada al igual que "**Poste();**", este modelo necesito de :

glScalef(0.5,0.5,0.5); ***//Escalarla a la mitad de su tamaño.***

glTranslatef(-700,0,-200); ***//Trasladarla a un punto donde se pudiera apreciar.***

glRotatef(300,0,1,0); ***//Rotarla de manera que se viera en donde quería.***

Código de la función:

```
void banca()
{
  glPushMatrix();
    glColor3f(0.647059,0.164706,0.164706);
    glScalef(0.5,0.5,0.5);
    glTranslatef(-700,0,-200);
    glRotatef(300,0,1,0);
    glmDraw(cont[3].identificar, GLM_MATERIAL|GLM_SMOOTH);
  glPopMatrix();
}
```

Dibujo de la banca en el e

Gaviotas();

Esta función nos permite dibujar una o más gaviotas las cuales son modelos 3D cargadas igual que los modelos anteriores, Estas gaviotas tienen una animación que es rotar en el eje y lo cual provoca el efecto de que están volando, para poder realizar esto se hizo lo siguiente en el código:

```
void gaviotas()
{
    glPushMatrix();
      glRotatef(ang,0,1,0);
      glColor3f(1.0,1.0,1.0);
      glScalef(100,100,100);
      glTranslatef(0,4,-4);
      glmDraw(cont[4].identificar, GLM_MATERIAL|GLM_SMOOTH);
    glPopMatrix();
    ang2=ang2+5;;
}
```

La animación de esta parte es con la rotación la cual se hace incrementado el punto de rotación.

Modelos():

En la función modelos también se dibujan otros dos modelos 3D los cuales son Cerca y Coche:

- Cerca(): Esta función dibuja una cerca limitante entre la carretera y la playa, la cual funciona con el siguiente código:

```
void cerca(int x)
{
  glPushMatrix();
    glColor3f(0.752941,0.752941,0.752941);
    glScalef(0.01,0.01,0.01);
    glTranslatef(x,50,800);
    glRotatef(90,0,1,0);
    glmDraw(cont[0].identificar, GLM_MATERIAL|GLM_SMOOTH);
  glPopMatrix();

}
```

Para colocarla en posición y lugar deseado se hace una traslación al punto deseado y una rotación para que quede en el eje de las x y poder formar la cerca divisora.

Dibujo en el Plano:

- **Coche:**

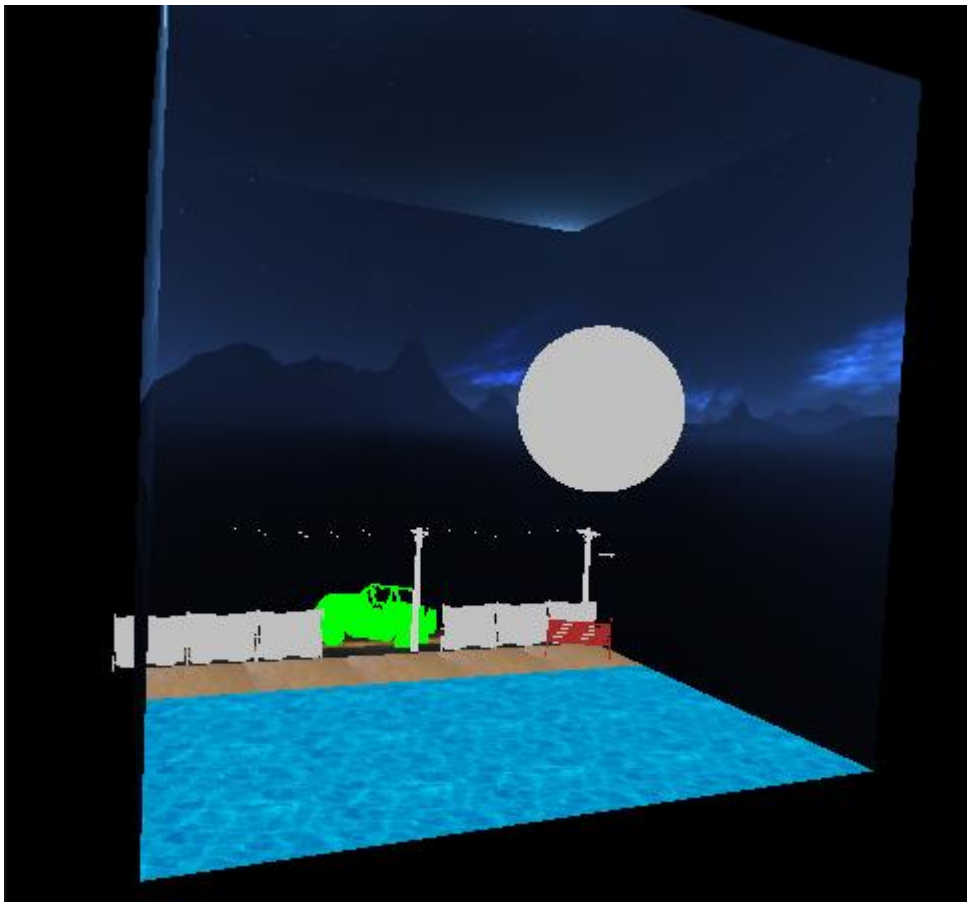
Este se dibuja en la función modelos y lo que hace es cargar al escenario una “Jeep” el cual es un modelo 3D para cargarla se hace uso del siguiente código:

```
glPushMatrix();
    glColor3f(0.0,0.0,0.0);
    glScalef(0.01,0.01,0.01);
    glTranslatef(60,0,1050);
    glPushMatrix();
      glRotatef(150,0,1,1);
      glRotatef(30,0,1,0);
      glRotatef(280,0,0,1);
      glRotatef(10,0,1,0);
      glPushMatrix();
        glTranslatef(-5,aux,0);
        glmDraw(cont[1].identificar, GLM_MATERIAL|GLM_SMOOTH);
      glPopMatrix();
    glPopMatrix();
  glPopMatrix();
  aux=aux-2;
  if(aux==-500)
  {
    aux=800;
  }
```

En este código se hace lo necesario para colocar y trasladar el coche de manera que quede uniforme en el ambiente, cabe mencionar que para poder manipularlo es un poco complicado, este tiene una animación de que recorre toda la carretera en el eje y, para hacer esto se hiso uso de la función “**gltranslatef()**” con un incremento en el eje antes mencionado y así hacer el efecto de re-corrimiento.

Dibujo en el escenario:

Una vez Terminadas cada una de las funciones nos queda el siguiente resultado:



Conclusión:

Este Proyecto- Examen me permitió como estudiante reafirmar mis conocimientos obtenidos en clase y ponerlos en práctica para poder resolver este proyecto, este trabajo me causo solo un problema el cual fue definir los parámetros de iluminación, ya que no encontraba manera para aplicarlos, por lo que solo los aplique en la luna, de ahí en fuera todo lo demás fue sencillo y por lo que me siento satisfecho al terminar este proyecto.

Entre las cosas que realmente me fue interesante aprender fue lo de cargar modelos 3D por lo que también me llevo a desinteresarme en la iluminación y lograr cargar dichos modelos.

BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA

Procesamiento digital de imágenes

Dr. Iván Olmos Pineda

Reporte primer parcial

Filtros puntuales aplicado a regiones

Alumno

Alejandro Rodríguez Vázquez

Primavera 2014

Introduccion

En la materia procesamiento digital de imágenes se ha venido trabajando con las operaciones puntuales para el tratamiento de imágenes digitales en RGB definiendo las funciones he implemetandolas a lo largo de las clases implementándolas en un software con la intención de observar su comportamiento con diferentes imágenes y asi comprender mejor su capacidad y uso en diferentes situaciones, observando su rango de comportamiento en una grafica e imagen por cada filtro ,se programo una ventana que nos reflejara cada filtro,ya sea de aclarado u obscurecimiento donde podemos manejar un parámetro opcional dado por el usuario llamado gamma.

Como 1er examen parcial se nos propuso crear un software que use todos los filtros de operaciones puntuales vistos hasta el momento el cual debe de trabajar ya sea sobre el software que hemos creado o crear uno nuevo y debe cumplir con caracteristicas especificas las cuales serán mencionadas posteriormente el objetivo del sistema es trabajar con los filtros aplicado a diferentes regiones de la imagen obteniendo un resultado optimo en la imagen para el usos especifico de esta, ya sea para corregir o destacar partes importantes de la imagen.

Para cumplir con el objetivo se deben usar las formulas vistas en clase y evaluarlas en el rango que nosotros necesitemos en este caso necesitamos rescrbiir cada función para poder aplicarlas por regiones que es la principal tarea de este software.

DATOS PREVIOS

Como datos previos recalcamos las funciones a usar para la implementación de nuestro software cuyo objetivo es usar los distintos filtros en el rango X que ingrese el usuario.

Usaremos 6 funciones en total las cuales son:

FILTRO DE CORRECCIÓN DE LUZ

Filtro Gamma

FILTROS DE ACLARADO

Filtro Seno

Filtro logaritmo

Filtro Exponencial de aclarado

FILTROS DE OBSCURECIMIENTO

Filtro Coseno

Filtro Exponencial de Obscurecimiento

Las Funciones de cada filtro son las siguientes, rescribiéndolas para su implementación.

GAMMA

$$r' = q\left(\frac{r}{q}\right)^{\gamma}$$

REESCRIBIENDO

$$f(x) = (x_1 - x_0)\left(\frac{(x - x_0)}{x_1 - x_0}\right)^{\alpha} + x_0$$

LOGARITMO

$$x' = A \ln(\alpha x + 1)$$

Donde

$$A = q / \ln(\alpha q + 1)$$

REESCRIBIENDO

$$f(x) = A \cdot \log\left(\alpha\left(x - x_0\right) + 1\right) + x_0$$

$$donde \quad A = \frac{\left(x_1 - x_0\right)}{(\log Q)}$$

$$Q = x_1 - x_0 + 1$$

SENO

$X' = q \sin(\pi x / 2q)$

Reescribiendo

$$f(x) = \left(x_1 - x_0\right) sen\left(\frac{\pi\left(x - x_0\right)}{2 x_1}\right) + x_0$$

EXPONENCIAL

$X' = A(1-e^{-\alpha x/q})$

Donde

$A = q / (1-e^{-\alpha})$

REESCRIBIENDO

$$f(x) = \left(\frac{x_1 - x_0}{1 - e^{-\alpha}} \right) \left(1 - e^{\frac{-\alpha(x - x_0)}{x_1 - x_0}} \right) + x_0$$

COSENO

$$x' = q\left(1 - \cos\left(\frac{\pi x}{2q}\right)\right)$$

REESCRIBIENDO

$$f(x) = (x_1 - x_0)\left[1 - \cos\left(\frac{\pi(x - x_0)}{2(x_1 - x_0)}\right)\right] + x_0$$

EXPONENCIAL

$$x' = A\left(e^{\alpha x / q} - 1\right), \quad \alpha > 0$$

donde:

$$A = q/(e^{\alpha} - 1)$$

REESCRIBIENDO

e

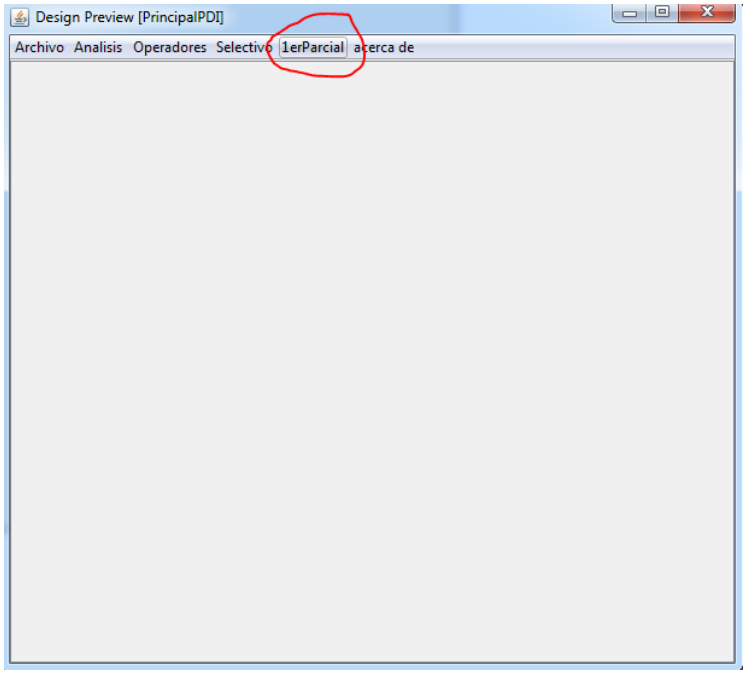
$$f(x) = A\left(e^{\frac{\alpha(x - x_0)}{x_1 - x_0}} - 1\right) + x_0$$

$$A = \frac{x_1 - x_0}{e^{\alpha} - 1}$$

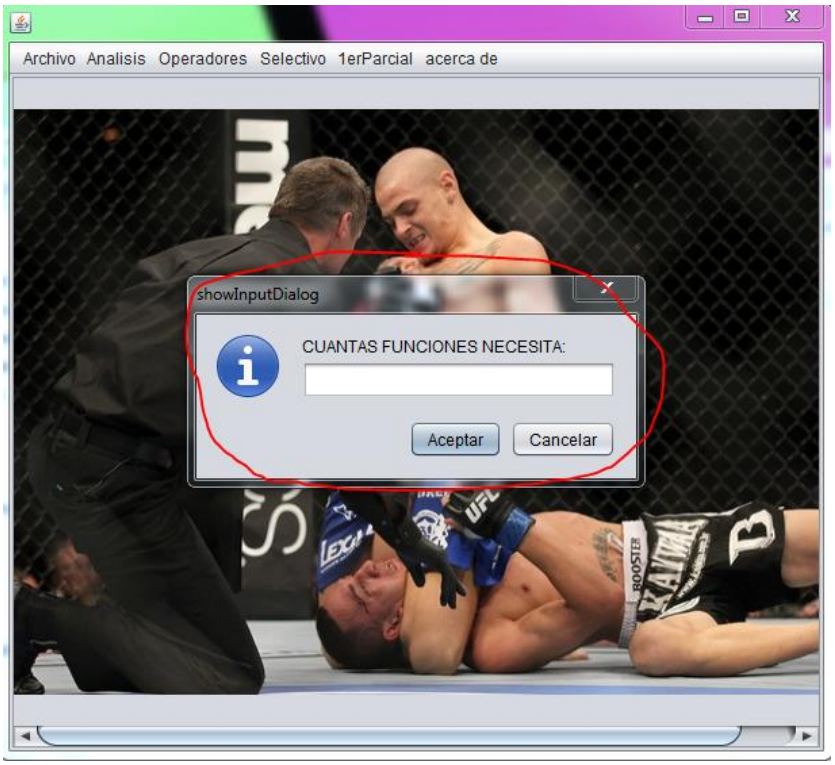
IMPLEMENTACION

Al contar ya con las funciones podemos realizar la programación de funciones para esto seguiremos trabajando con java usando el IDE Netbeans y agregaremos a nuestro software

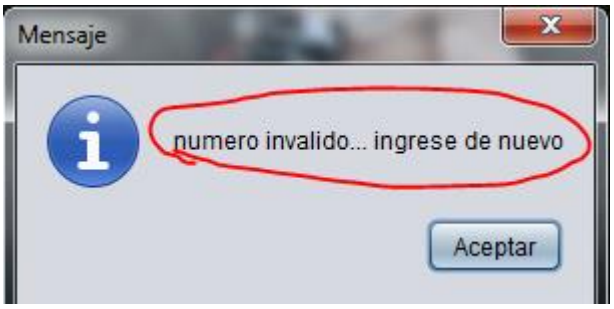
Nuestro software ya cuenta con la opción de seleccionar una nueva imagen lo cual es lo que realizaremos como tarea principal posteriormente se agregó una opción como menuItem llamada 1erparcial la cual nos lanzara una nueva ventana



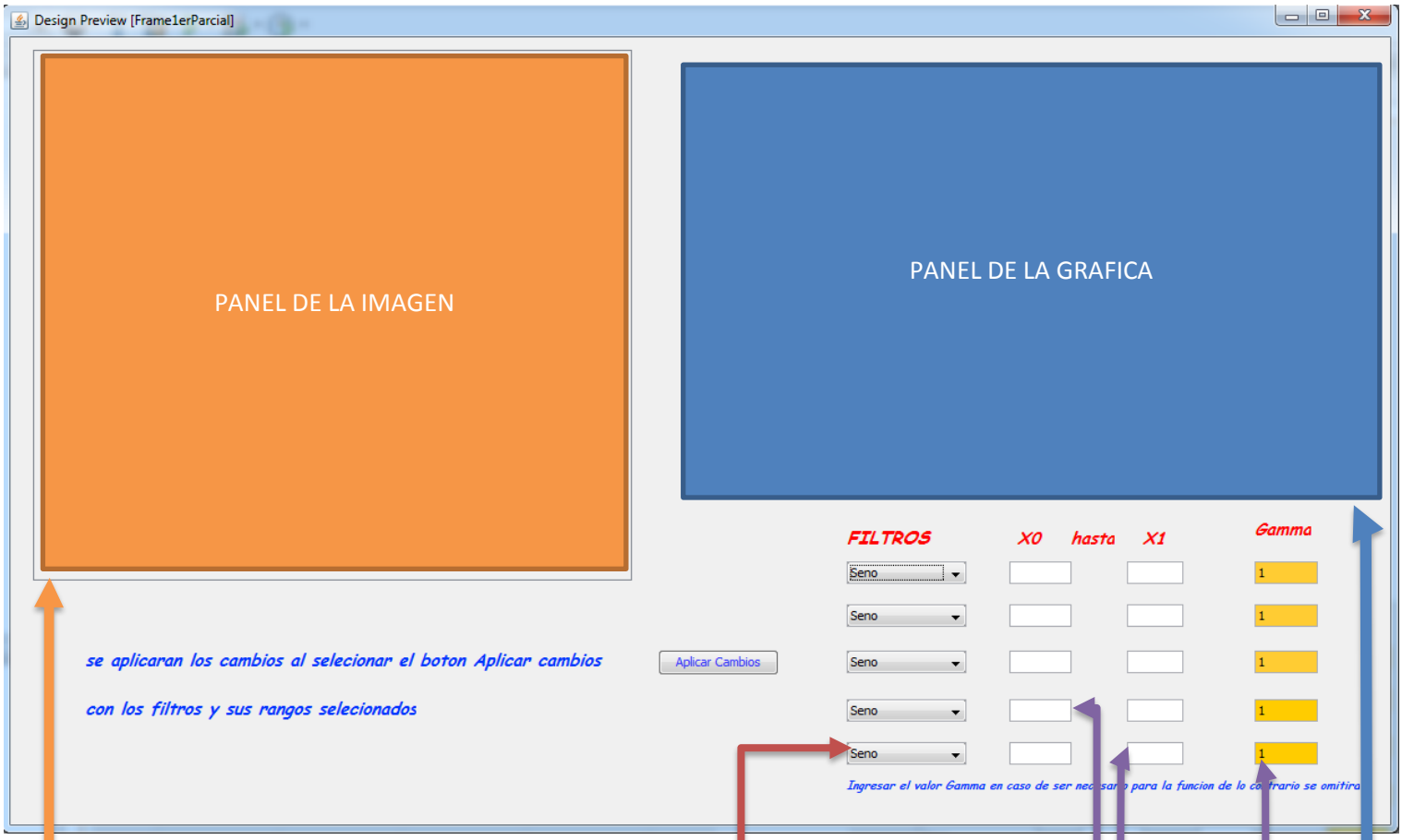
Al seleccionar esta opción nos lanzara una pequeña ventana con la opción de ingresar el numero de funciones que trabajaremos y si no ingresamos un numero en el rango de 1 a 5 nos mandara un mensaje de error ya que lo único que necesitamos es seleccionar el numero de funciones correcto para trabajar



El mensaje de error



DISEÑO DE NUEVO FRAME 1erparcial



Dependiendo el número de funciones ingresado se activaran la misma cantidad de casillas y cada una contiene un menú con los 6 diferentes tipos de filtros que podemos usar

Este panel contendrá la imagen inicialmente contendrá la imagen que se le aplicara el filtro posteriormente se cargara la imagen tratada con los filtros seleccionados

En estos texfields podemos capturar los datos ingresados por el usuario donde las casillas en blanco serán donde ingresaremos el rango y en la casilla amarilla el valor GAMMA

Este panel recibirá una imagen la cual es la imagen de la gráfica que generaremos

Al contar con el diseño procedemos a crear cada una de las funciones la cual manipulara cada pixel cómos e ha venido trabajando pasaremos como parámetro únicamente una imagen a nuestro nuevo frame y en este se realizaran todas las operaciones así como el tratamiento de eventos que se generen.

Se crearon 6 funciones diferentes llamadas

Seno(rango1,rango2)

Coseno(rango1,rango2)

Gamma(rango1,rango2,gamma)

Logaritmo(rango1,rango2,gamma)

Exponencial(rango1,rango2)

ExponencialObsc(rango1,rango2,gamma)

Pasando como parámetros 2 enteros llamados rango1 y rango2 el cual se necesitan para nuestras nuevas funciones y en caso de ser necesario usaremos un parámetro mas llamado Gamma de tipo doublé para no perder precisión.

Estas funciones retornan un tipo BufferedImage ya que a todos se les pasa también una imagen la cual es la que se tratara descomponiéndola primero en sus 3 diferentes Canales posteriormente aplicándole los filtros correspondientes generando una imagen nueva a partir de la ya creada.

Problemas de la implementación y programación

-Filtro sobre filtro

El principal problema al programar estas funciones fue que al trabajar diferentes tipos de filtros nos encontramos inicialmente con que al aplicar independientemente cada función los cambios se guardaban pero al pasar a otro filtro se aplicaba este filtro pero sobre los cambios del filtro anterior y no sobre la imagen original

Dicho de otra manera cada filtro se aplicaba con los cambios del filtro anterior y nos sobre la imagen original afectando así los cambios de nuestra imagen final por que por ejemplo

Aplicaremos 2 filtros, primero seleccionamos seno y luego coseno

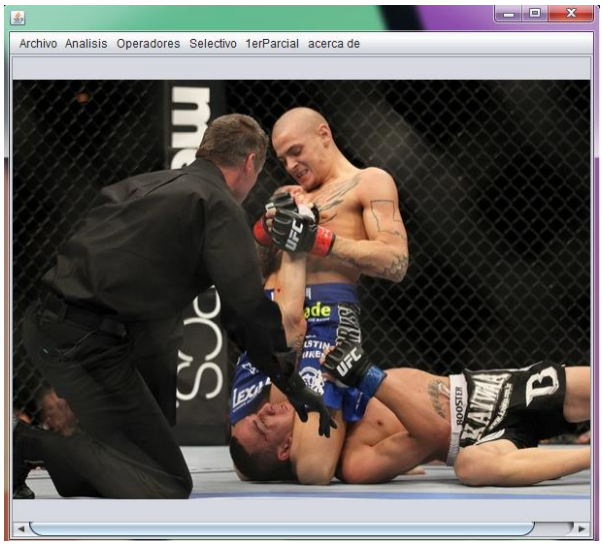
Nuestra imagen original se tratara primero con filtro en sus 3 canales con seno en caso de encontrarse en el rango dado y esta función nos arroja una nueva imagen y como segundo el filtro coseno se aplicara pero ahora lo que pasaba es que se comparaba con la imagen arrojada del primer filtro y esto era un error que modificaba la salida de nuestra imagen

SOLUCION

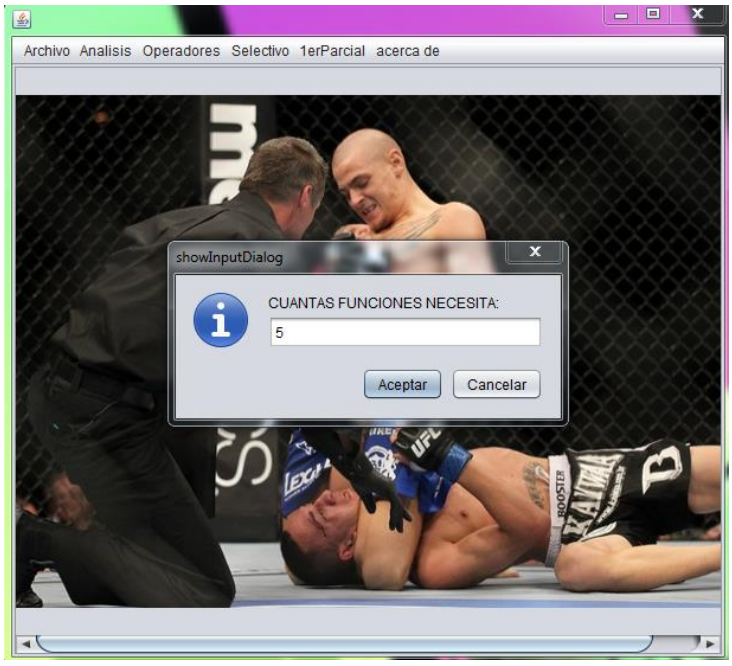
La solución que se genero fue que cada comparación que hará cada función se realizara sobre la imagen original y se guardara en un buffer la imagen nueva generada y teniendo vacío cada pixel de nuestra imagen en caso de no pasar por ninguna función en el rango dado y hasta el momento de dibujarla nuevamente los pixeles se reagrupan usando cada pixel de cada función usada más los pixeles faltantes con los pixeles de la imagen original así de esta forma solo tratamos los pixeles en el rango que necesitamos y los que no se dejan intactos

CORRIENDO LA APLICACIÓN

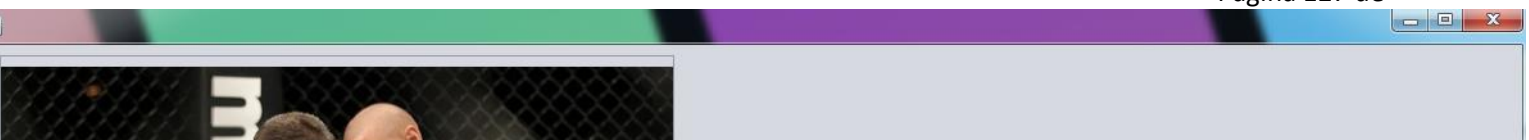
Cargamos la imagen



Como ejemplo usaremos 5 funciones



Se lanza el nuevo frame y ahora ingresamos los datos de nuestros rangos dados



Observamos que se carga por default el texfield de Gamma el valor 1 y este valor se usa en caso de necesitarlo en caso de que no este valor se omite y solo nos enfocamos por checar los valores de nuestro rango x0 y x1

Damos por echo que el usuario es un usuario que conoce del tema y sabe que valores ingresar en el caso de gamma y que función corresponde y cual es el comportamiento de esta así de esta forma podemos ingresar bien los datos que necesitemos

Debido a la cantidad de funciones a elegir damos por echo que el usuario ingresara los rangos correctamente teniendo como ejemplo estas formas señalando la manera correcta de uso y la manera incorrecta

Maneras correctas

Maneras incorrectas

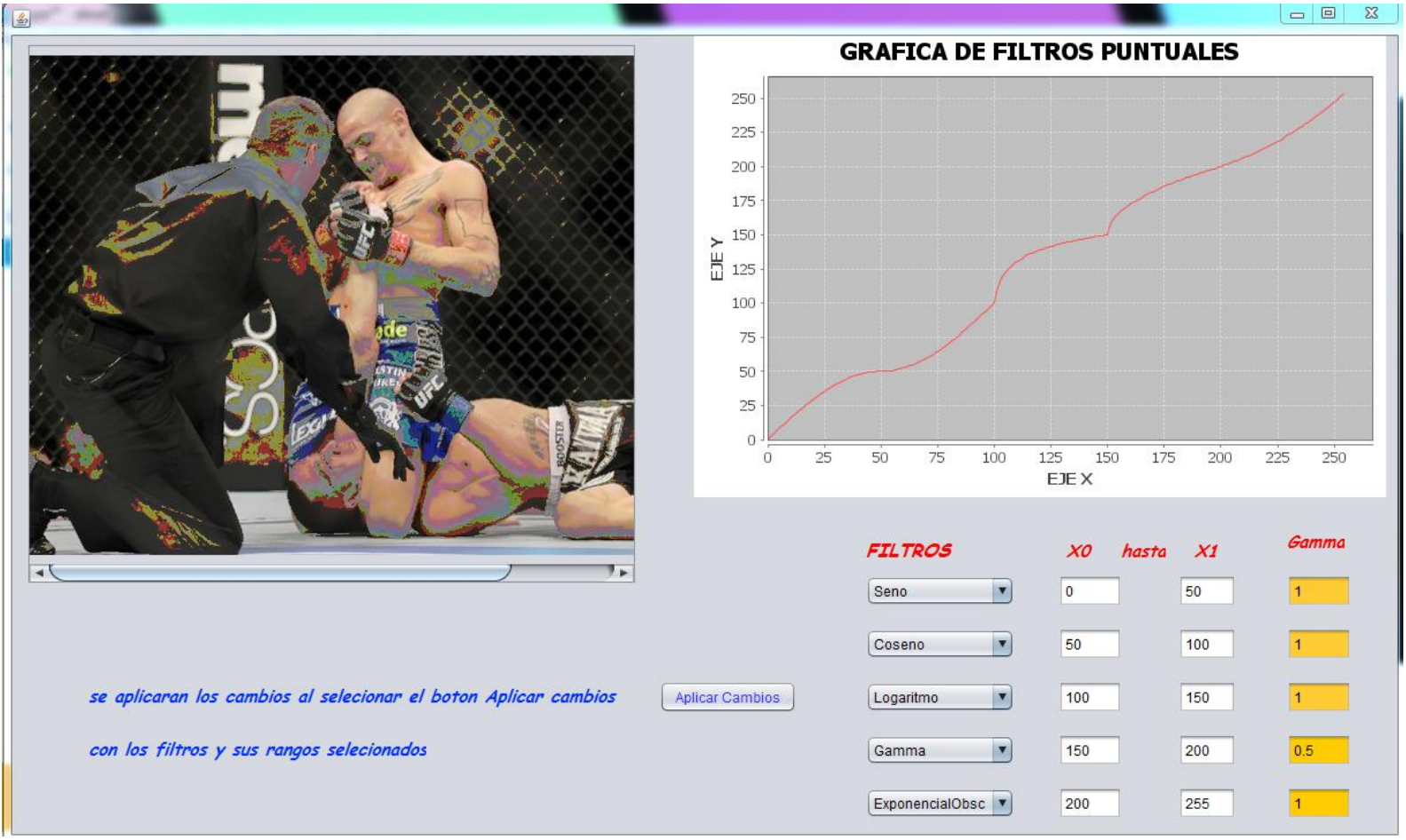
Las maneras incorrectas con las que a continuación ilustramos podemos observar que en los primero 2 filtros estamos usando rangos que usan mismos valores por lo cual se usaran los valores de el filtro 2 omitiendo los valores idénticos del filtro 1.

Otra manera incorrecta es ingresar valores negativos y valores que sobrepasan el rango de 255 ya que este rango lo cubre cualquier canal de RGB y cabe resaltar que solo se capturan números

Al ingresar los datos correctamente solo seleccionamos el botón 

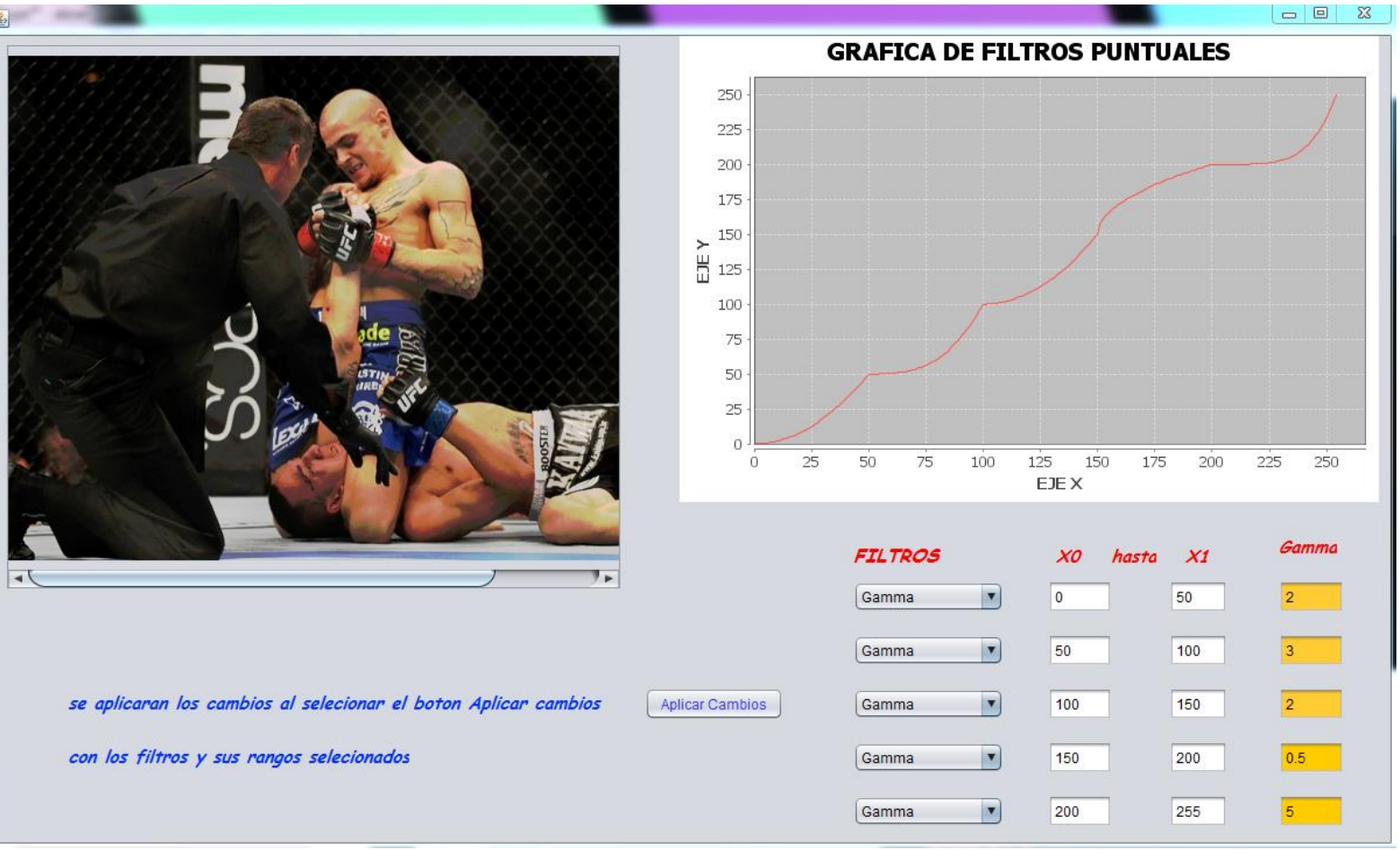
En este caso ingresamos los datos que mostramos a continuación

Y obtenemos una salida

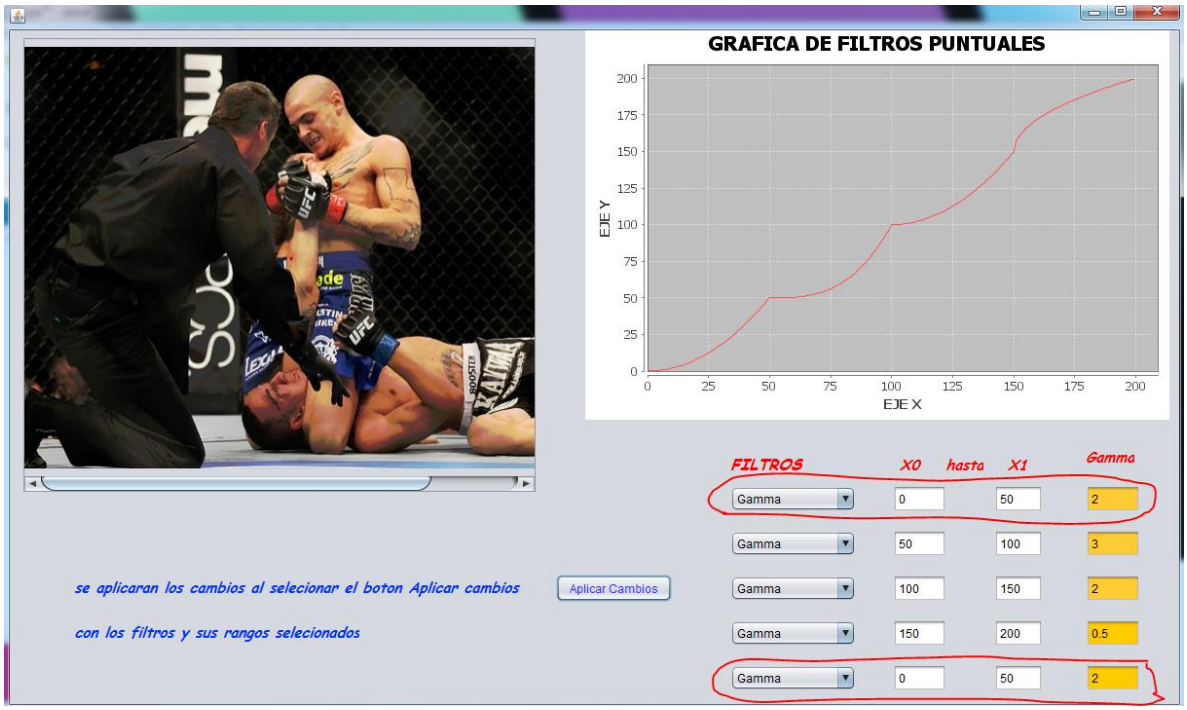


Podemos observar que la imagen se aplican los diferentes filtros y podemos modificar nuevamente el resultado observando los diferentes cambios de la imagen original ya que si seleccionamos nuevamente nuevos filtros tendremos una salida diferente pero los cambios se harán a la imagen original.

Como ejemplo usaremos un tipo de filtro solamente y veremos que se realiza de manera diferente ya que el valor que variamos es el de gamma



Ahora podemos observar que si usamos el mismo valor con el mismo filtro se lleva a cabo la operación lo que pasa es q si se lleva a cabo las 2 operaciones pero como arroja el mismo resultado solo se muestra 1 resultado y como nos faltan valores por cubrir no se muestran y este valor no cubierto es el valor de (200 a 255) el cual no veremos en la gráfica porque este rango no tuvo ningún cambio en la imagen



Conclusiones

Podemos presentar muchas opciones del software representando cada filtro en un rango dado este software se puede ocupar de manera precisa usando los valores correctos de nuestros rangos y el valor de Gamma, ya que estos son los valores que se usan y el resultado depende totalmente de si el valor esta dado de manera correcta para el fin requerido

°

PROCESAMIENTO DIGITAL DE IMÁGENES PRIMAVERA 2014

BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA

FACULTAD DE COMPUTACION

PROCESAMIENTO DIGITAL DE IMÁGENES

“RESCATE DE DOCUMENTOS ANTIGUOS”

PROFESOR: IVAN OLMOS PINEDA

ALLUMNOS: JORGE R. SANCHEZ CASTILLO
DANIEL SORIANO GRANDE

PRIMAVERA 2014

Introducción

Hoy en día existe una gran cantidad de documentos, actas de nacimiento o defunción, certificados, cartas, publicaciones, planillas, documentos administrativos y financieros antiguos o de generaciones pasadas que se encuentran en un estado deplorable, ya que

por cuestiones de tiempo, clima y malos cuidados han perdido claridad en gran parte de su contenido o en algunos casos han perdido parte del material de que están hechos.

El objetivo de este proyecto es recuperar esa información de los documentos antiguos para poder conocer su contenido o bien tener un respaldo con mayor legibilidad digitalizado de dichos documentos. Historiadores y científicos han hecho este tipo de trabajo por años con el fin de conocer la historia, formas de pensar, tradiciones, cultura, y materiales con los que están hechos estos documentos, ahora nosotros presentamos una forma de hacerlo, que en 5 pasos y aplicando algunos filtros de suavizado eliminamos el ruido de estos documentos, recuperando una gran parte del contenido de estos, presentando resultados con el fondo en color blanco y el contenido en color negro.

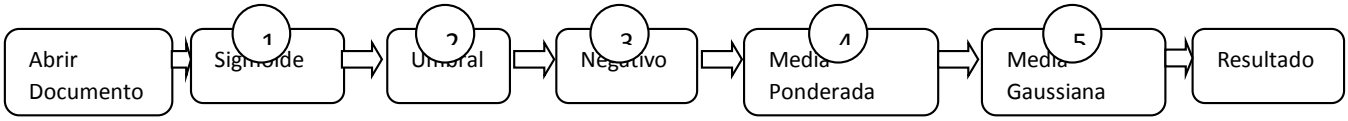
Conceptos utilizados en el Documento

*Pixel***:** Abreviatura de Picture Element, es un único punto en una imagen gráfica

Ruido: El ruido es la distorsión que se manifiesta como pixeles de color o luminosidad incorrectos que afectan el parecido de la imagen con la realidad.

Metodología

Nosotros usamos una metodología dividida en 5 pasos, la cual con lo aprendida en clase y con investigaciones realizadas, vimos que al combinar estos 5 pasos en el orden presentado nos brinda mejores resultados para documentos con buena, regular y mala calidad.



Bien, ahora veamos el porqué se utilizaron estos pasos en ese orden y cómo es que se codifican.

Paso 1: Sigmoide

El filtro de sigmoide nos sirve para resaltar los contrastes en una imagen, es decir que los colores claros sean más claros y los colores obscuros más obscuros, este filtro nos ayudó ya que en varias imágenes el documento se encuentra maltratado y su contenido no era muy legible, así el filtro de sigmoide nos ayuda a remarcar mas dicho contenido.
Su código es el siguiente:

```
for (int x = 0; x < imgsel.getWidth(); x++) {
        for (int y = 0; y < imgsel.getHeight(); y++) {
                Color c = new Color(imgsel.getRGB(x, y));
                float  r = c.getRed();
                float g = c.getGreen();
                float b = c.getBlue();


               A1=(float)((Math.tanh(Math.toRadians(alfa*(r-(q/2)))))+1);

               Float.toString(A1);
               resA1= (float)((q/2)*(A1));


               A2=(float)((Math.tanh(Math.toRadians(alfa*(g-(q/2)))))+1);

                Float.toString(A2);
               resA2= (float)((q/2)*(A2));

                A3=(float)((Math.tanh(Math.toRadians(alfa*(b-(q/2)))))+1);
               Math.toRadians(A3);
                Float.toString(A3);
               resA3= (float)((q/2)*(A3));
```

Paso 2: Umbralización.

El objetivo de este filtro es convertir una imagen a una nueva con solo 2 niveles, es decir, convertir una imagen RGB a una imagen en Blanco y Negro, de manera que los objetos queden separados del fondo, permite eliminar ruido y realzar los bordes a fin de obtener los pixeles en los que se produce un borde.
Se decidió aplicar este filtro después de sigmoide, ya que teniendo nuestra imagen con mas contraste entonces con la Umbralización podíamos separar el contenido del documento con el fondo y así poder obtener lo que nos importa en este proceso.

```
public BufferedImage Umbralizacion(BufferedImage imageny){
     Nueva_Imagen2=new BufferedImage(imageny.getWidth(),imageny.getHeigh
      int Recibegris[]=histograma(imageny);
    CalculaUmbral cal=new CalculaUmbral();
    int umbral=cal.CalcularUmbral(Recibegris);

    int aux;
for(i=1;i<imageny.getWidth();i++){
   for(j=1;j<imageny.getHeight();j++){
     colorAuxiliar=new Color(imageny.getRGB(i, j));
         aux=ConvierteGris(colorAuxiliar);

         ///rojo

        if(aux<umbral){
         Nueva_Imagen2.setRGB(i, j,new Color(255,255,255).getRGB());
         }
        if(aux>umbral){
         Nueva_Imagen2.setRGB(i, j,new Color(0,0,0).getRGB());
         }
       }
   }
return Nueva_Imagen2;
```

Paso 3: Negativo.

El negativo es el inversor de una imagen, es decir, invertir claros y obscuros, como nuestro documento pasó por el proceso de Umbralización el fondo nos quedo negro y las letras en blanco, el objetivo de este proyecto es mostrar los resultados donde el fondo sea blanco y las letras en negro, es por eso que aplicamos el negativo para invertir estos colores.

```
public BufferedImage Negativo(BufferedImage imgsel) {
                BufferedImage bi = null;
                if (imgsel != null) {
                        bi = new BufferedImage(imgsel.getWidth(), imgsel.getHeight(),im
                        for (int x = 0; x < imgsel.getWidth(); x++) {
                                for (int y = 0; y < imgsel.getHeight(); y++) {
                                        Color c = new Color(imgsel.getRGB(x, y));
                                        int r = c.getRed();
                                        int g = c.getGreen();
                                        int b = c.getBlue();
                                        r=255-r;
                                        g=255-g;
                                        b=255-b;
                                        bi.setRGB(x, y, new Color(r, g, b).getRGB());
                                }
```

Paso 4: Media Ponderada.

Ya teniendo nuestra imagen con fondo blanco y contenido en color negro, pasamos a mejorar la calidad de la misma, decidimos aplicar el filtro de media ponderada, ya que el contenido que se tenía era muy poco remarcado y no era legible, además de que presentaba poco ruido. Con la media ponderada logramos remarcar mas el contenido y hacer legible el mismo, además de eliminar en un buen porcentaje el ruido que aun presentaba el documento.

```
public BufferedImage mediaPonderada(BufferedImage img,int a) {
  int[][] matrizr = new int[img.getWidth()][img.getHeight()];
  int[][] matrizg = new int[img.getWidth()][img.getHeight()];
  int[][] matrizb = new int[img.getWidth()][img.getHeight()];
  Color color, color1, color2, color3, color4, color5, color6, color7, c
  int r, g, b, r1, g1, b1, r2, g2, b2, r3, g3, b3, r4, g4, b4, r5, g5, b
  double val=1;
  val=val/(8+a);
  for (int i = 1; i < img.getWidth() - 1; i++) {
      for (int j = 1; j < img.getHeight() - 1; j++) {
          color = new Color(img.getRGB(i, j));
          color1 = new Color(img.getRGB(i - 1, j - 1));
          color2 = new Color(img.getRGB(i - 1, j));
          color3 = new Color(img.getRGB(i - 1, j + 1));
          color4 = new Color(img.getRGB(i, j - 1));
          color5 = new Color(img.getRGB(i, j + 1));
          color6 = new Color(img.getRGB(i + 1, j - 1));
          color7 = new Color(img.getRGB(i + 1, j));
          color8 = new Color(img.getRGB(i + 1, j + 1));
          r = color.getRed();
          g = color.getGreen();
          b = color.getBlue();
          r1 = color1.getRed();
          g1 = color1.getGreen();
          b1 = color1.getBlue();
          r2 = color2.getRed();
          g2 = color2.getGreen();
          b2 = color2.getBlue();
          r3 = color3.getRed();
          g3 = color3.getGreen();
          b3 = color3.getBlue();
          r4 = color4.getRed();
```

Paso 5: Media Gaussiana

La decisión de aplicar el filtro de media Gaussiana fue porque como ya sabemos este filtro se encarga de eliminar el ruido con una magnitud suave y el ruido con magnitud alta lo atenúa, así que solo fue como para detallar nuestro documento antiguo y presentar mejores resultados. Con este filtro se concluyen los pasos de nuestra metodología y mostramos al usuario sus resultados.

```
public BufferedImage mediaGaussianaFuerte(BufferedImage img) {
   int[][]matrizr=new int[img.getWidth()][img.getHeight()];
   int[][]matrizg=new int[img.getWidth()][img.getHeight()];
   int[][]matrizb=new int[img.getWidth()][img.getHeight()];
   Color color,color1,color2,color3,color4,color5,color6,color7,colo
   int r,g,b,r1, g1, b1,r2, g2, b2,r3, g3, b3,r4, g4, b4,r5, g5, b5,
   double val=1;
   val=val/16;
   for (int i = 1; i < img.getWidth()-1; i++) {
       for (int j = 1; j < img.getHeight()-1; j++) {
           color = new Color(img.getRGB(i,j));
           color1 = new Color(img.getRGB(i-1,j-1));
           color2 = new Color(img.getRGB(i-1,j));
           color3 = new Color(img.getRGB(i-1,j+1));
           color4 = new Color(img.getRGB(i,j-1));
           color5 = new Color(img.getRGB(i,j+1));
           color6 = new Color(img.getRGB(i+1,j-1));
           color7 = new Color(img.getRGB(i+1,j));
           color8 = new Color(img.getRGB(i+1,j+1));
           r = color.getRed();
           g = color.getGreen();
           b = color.getBlue();
           r1 = color1.getRed();
           g1 = color1.getGreen();
           b1 = color1.getBlue();
           r2 = color2.getRed();
           g2 = color2.getGreen();
           b2 = color2.getBlue();
```

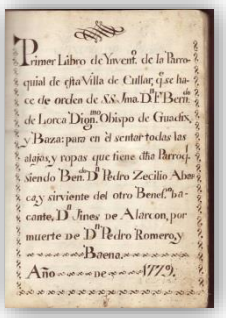
Resultados

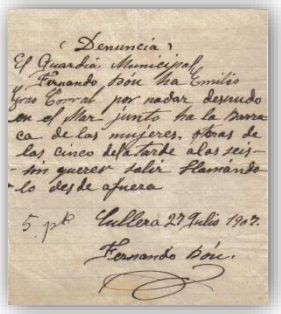
Para el estudio de los resultados de nuestro proyecto se tomó una muestra de 15 documentos antiguos clasificados en 3 categorías:

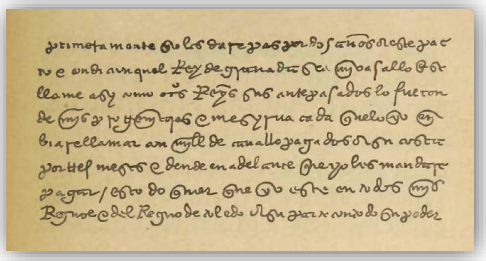
5 Documentos Buenos

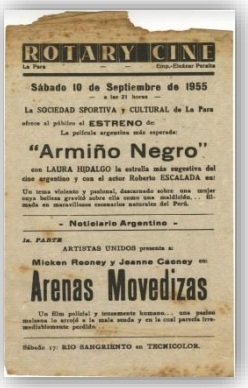
Son aquellos documentos donde:

- El fondo es claro y/o uniforme, lo que dará buenos resultados al momento de la Umbralización.
-Las letras están bien remarcadas garantizando que no se perderá información al momento de aplicarle nuestra metodología
-No hay un cabio brusco en las tonalidades de colores permitiendo que la información sea legible







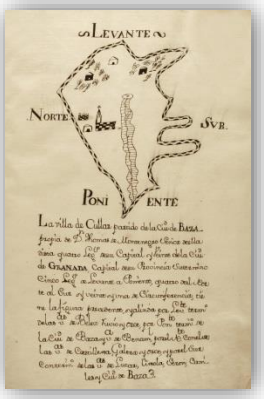

ROTARY CINE
La Para — Emp.-Eleázar Peralta
Sábado 10 de Septiembre de 1955
— a las 21 horas —
La SOCIEDAD SPORTIVA y CULTURAL de La Para
ofrece al público el ESTRENO de:
La película argentina más esperada:
"Armiño Negro"
con LAURA HIDALGO la estrella más augestiva del cine argentino y con el actor Roberto ESCALADA en:
Un tema violento y pasional, desarrollado sobre una sonata cuya belleza gravitó sobre ella como una maldición... El marido en maravillosas escenarios naturales del Perú.
- Noticiario Argentino -
1a. PARTE
ARTISTAS UNIDOS presenta a:
Micken Rooney y Joanne Cagney en:
Arenas Movedizas
Un film policial y totalmente humano... una pasión malsana la arrolló a la mala senda y en la cual pereció irremediablemente perdida.
Sábado 17: RIO SANGRIENTO en TECHNICOLOR.

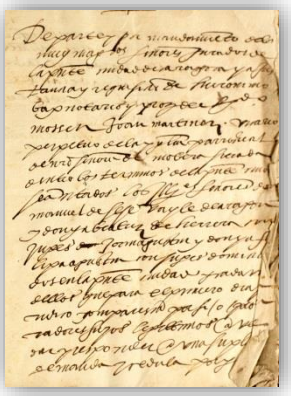

REGISTRO DE LA PROPIEDAD
DE
CARTAGENA.
CANCELACION.
D. Juan de Dios de Grande y Albarez, Doctor Licenciado del Distrito de Cartagena, Juez de primera instancia de la misma del Oeste y Escribano del mismo á
D. Angel Justuste y Batejuz
La obligacion de ciento y un mil cuatrocientos treinta reales cuarenta y seis céntimos, inscrita en el Libro 47 fólio 38 finca queda cancelada en el tomo 1º de hipotecas por orden de fechas fólio 208 en virtud de la escritura otorgada por el acreedor en 16 de Diciembre de 1866 ante el notario D Juan de la Cuerva cuya copia queda archivada en este Registro, en el legajo del 4º trimestre del año actual.
Cartagena 19 de Diciembre de 1866
Honorarios [illegible] Rvn.

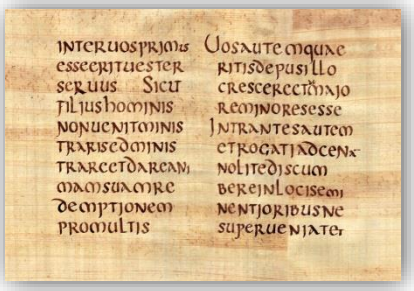
5 Documentos Regulares

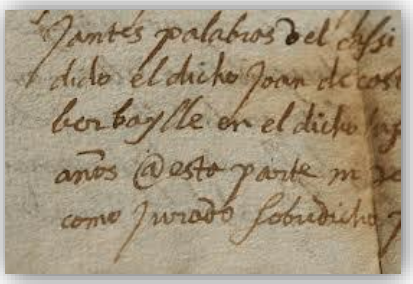
Son aquellos documentos donde:
- El fondo es en algunas zonas claro y/o obscuro lo que podría provocar que en ocasiones con nuestra metodología la información no sea del todo rescatada.
-Algunas las letras están bien remarcadas pero otras soy muy opacas, con las opacas en ocasiones podrían convertirse en fondo blanco, o solo se rescatara fragmentos de estas.
- Empieza a existir un ligero cambio en las tonalidades de colores, es decir, zonas en las que aunque las letras estén remarcadas existe un fondo obscuro lo que llevaría al momento de aplicarle Sigmoide o Umbralizacion perder parcialmente información.


MS 577
Syriac Sertā book script. Mt. Sinai, Egypt, ca. 11th c.


LEVANTE
NORTE
SVR.
PONIENTE
La villa de Cullar frontic de la Ciudad de Baza...
de GRANADA, Capital...




INTERVOSPRIMUS VOSAUTEMQUE
ESSEIRITUESTER RITISDEPUSILLO
SERVUS SICUT CRESCERECTIMAJO
FILIUSHOMINIS REMINOREESSE
NONUENITMINIS INTRANTESAUTEM
TRARISEDMINIS ETROGATIADCENA
TRARECTDARCAM NOLITEDISCUM
MANSUAMRE BEREINLOCISEOM
DEMPTIONEOM NENTIORIBUSNE
PROMULTIS SUPERUENIATE



5 Documentos Malos

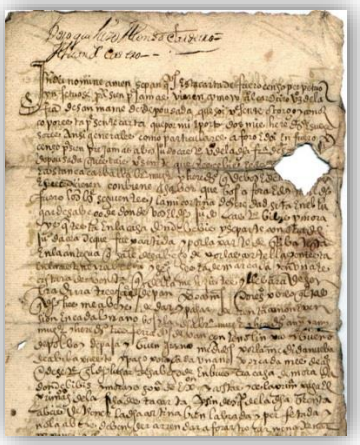
Son aquellos documentos donde:

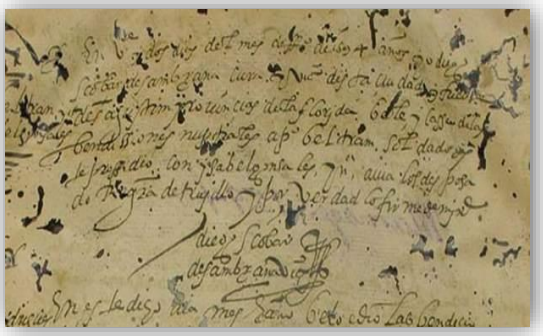
- El fondo en zonas es muy obscuro, ya sea por el mal estado del papel del documento o por manchas de tinta y/o sellos

-Las letras o son muy claras ,o muy obscuras

-Existe un cambio brusco en las tonalidades de colores, es decir, letras bien marcadas en fondo obscuro, o letras opacas en fondo claro llevaría a que al aplicarle nuestra metodología se perdiera la mayor parte de información en esas zonas.

-En las manchas obscuras nuestra metodología no sabría si tomarla como información valiosa o como parte del fondo del documento.





Aplicando la Metodología

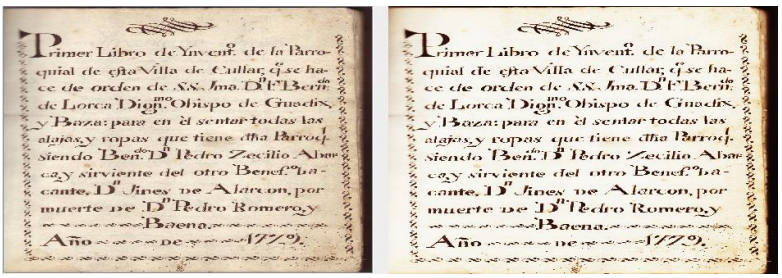
A continuación se muestra como las imágenes son procesadas por la metodología propuesta en este documento y el resultado final.

Documentos Buenos

Documento 1

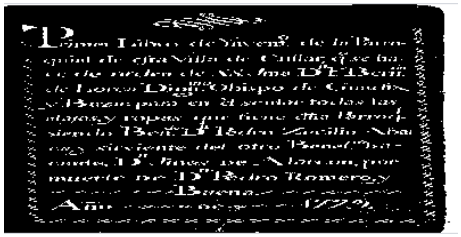
Original ---> Sigmoide

Se puede notar que el fondo se volvió más claro y las letras fueron obscurecidas



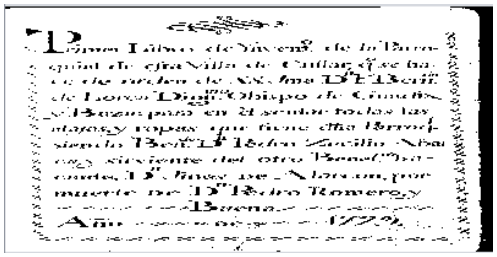
Umbralización

Resulto una imagen en blanco y negro permitiendo dejar la imagen en un fondo negro.



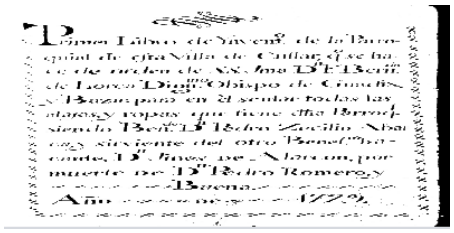
Negativo

Resulto una imagen en fondo blanco, pero la información que un poco pixeliada.



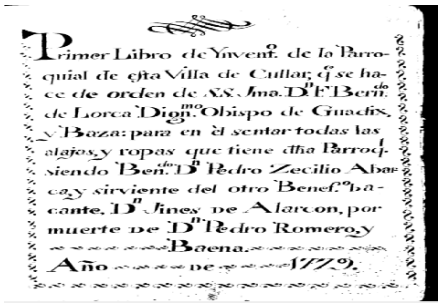
Media Ponderada

Se suavizo la imagen eliminando un poco el ruido.



Media Gaussiana-Resultado

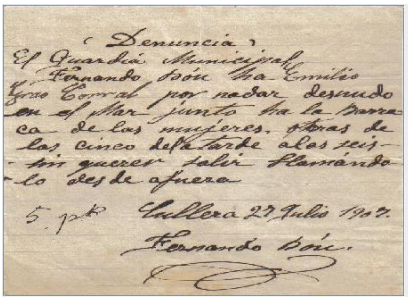
Se recuperó toda la información y aunque existe un borde negro no afecta a la información.

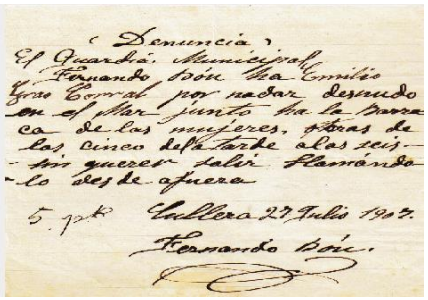

Primer Libro de Vnvent.s de la Parro-
quial de esta Villa de Cullar, q.e se ha-
ce de orden de S.S. Ima. D.n F.co Ber.n
de Lorca Dign.mo Obispo de Guadix,
Baza: para en él sentar todas las
alajos, y ropas que tiene dha Parroq.l
siendo Ben.do D.n Pedro Zecilio Aba-
ca, y sirviente del otro Benef.o va-
cante, D.n Jines de Alarcon, por
muerte de D.n Pedro Romero, y
Baena.
Año de 1779.

Documento 2

Original ---> Sigmoide

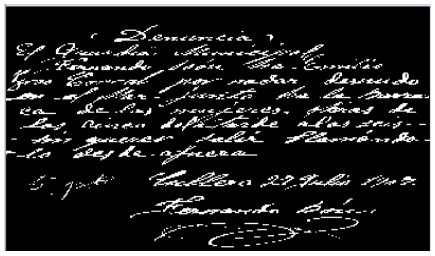
Se puede notar que el fondo se volvió más claro aunque no uniforme pero las letras fueron obscurecidas que es lo que deseamos


(Denuncia)
El Guarda Municipal
Fernando Boix Sra Emilio
Gros Corral por nadar desnudo
en el Mar junto ha la Barra
ca de las mujeres, horas de
las cinco de la tarde ahas seis
sin querer salir llamando
lo desde afuera
5 pts Cullera 27 Julio 1903.
Fernando Boix.


(Denuncia)
El Guarda Municipal
Fernando Boix Sra Emilio
Gros Corral por nadar desnudo
en el Mar junto ha la Barra
ca de las mujeres, horas de
las cinco de la tarde ahas seis
sin querer salir llamando
lo desde afuera
5 pts Cullera 27 Julio 1903.
Fernando Boix.

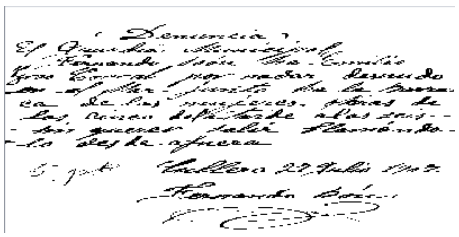
Umbralización

Al momento de umbralizar se eliminó las manchas amarillentas, así como también el fondo claro



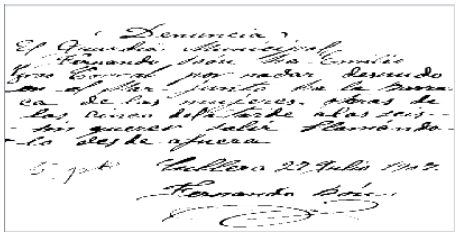
Negativo

La imagen ahora es binaria, pero se puede notar como se ha perdido un poco de información desde el paso anterior



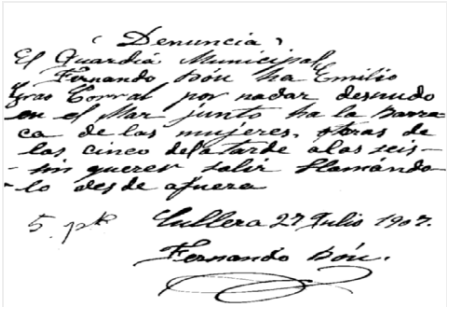
Media Ponderada

Se suavizo la imagen así eliminando un poco de ruido y aumentando el grosor de las letras, pero no es muy legible



Media Gaussiana-Resultado

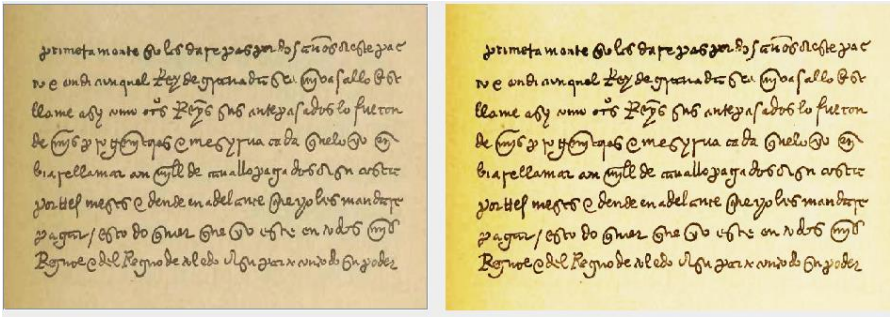
Se recuperó la información que se había perdido ,las letras son más distinguibles que en el paso anterior



Documento 3

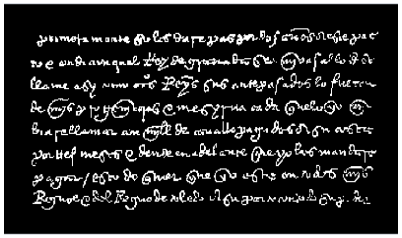
Original ---> Sigmoide

Se puede notar que el fondo se volvió amarillento claro en una zona, pero claro uniforme en lo restante del documento, las letras fueron obscurecidas que es lo que deseamos



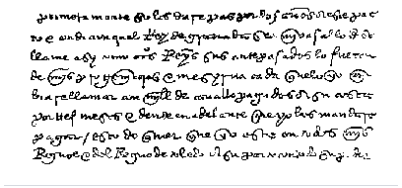
Umbralización

El documento solo se encuentra en blanco y negro, la información se ve bien pero no sabemos si la recuperamos en su totalidad



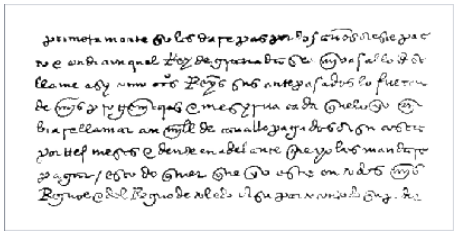
Negativo

Se puede ver que la información se recuperó completa, pero hay zonas en las que es opaca.



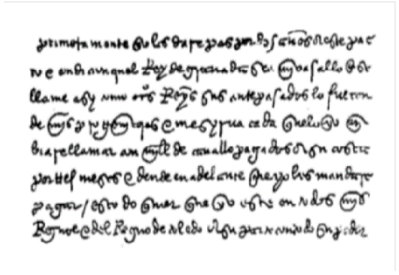
Media Ponderada

Este paso tendría que ser omitido porque volvió al documento más opaco, pero como se le está aplicando una metodología en general no se puede hacer mucho**.**



Media Gaussiana-Resultado

El resultado fue bueno, se recuperó la información y el fondo blanco con las letras negras rescatando lo que el paso anterior produjo.



Documento 4

Original ---> Sigmoide

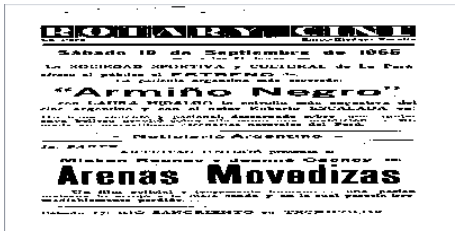
Se puede notar que el fondo se volvió amarillento claro en algunas zonas, claro uniforme en lo restante del documento, las letras fueron obscurecidas que es lo que deseamos

Umbralización

Se eliminó esas manchas amarillentas aunque aún presenta un poco de ruido arriba**.**

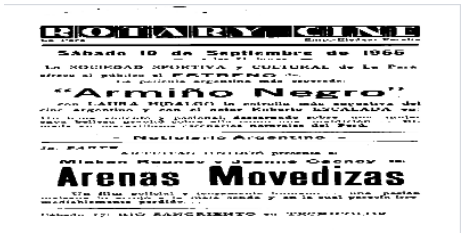
Negativo

Se puede apreciar bien la información, así también el ruido del paso anterior



Media Ponderada

Se suavizo la imagen tratando de eliminar el poco de ruido existente pero no se logró.



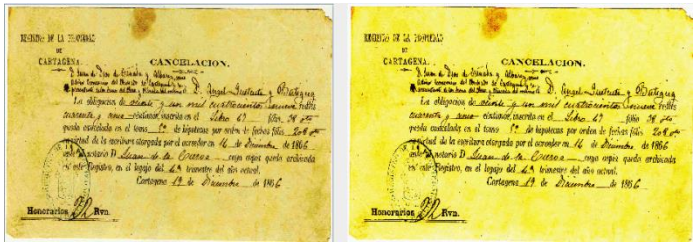
Media Gaussiana-Resultado

Se recuperó la información, aunque también unas manchas en la parte superior pero que no afectan al documento

Documento 5

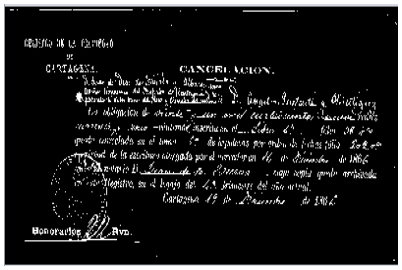
Original-→ Sigmoide

Se presentan zonas en las que el fondo es casi blanco y en otras con un fondo amarillo.



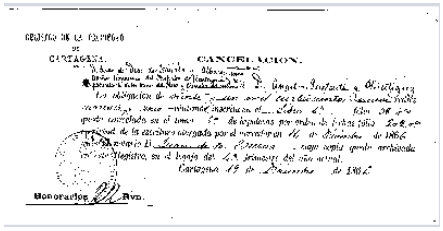
Umbralización

La información se recuperó aunque al parecer está un poco pixeliada



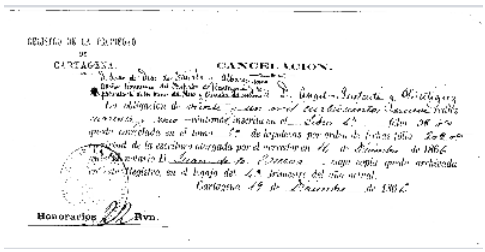
Negativo

Se nota que la información se recuperó como lo deseado



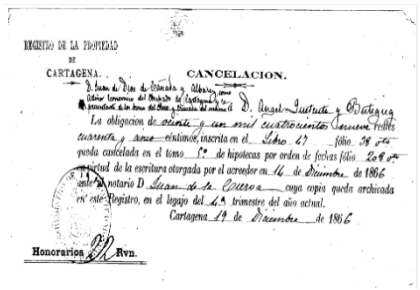
Media Ponderada

Se suavizo eliminando el poco ruido que tenia



Media Gaussiana-Resultado

Las letras se ven un poco más marcadas pero, se obtuvo ruido en la parte superior



Documentos Regulares

Documento 1

Original→Sigmoide

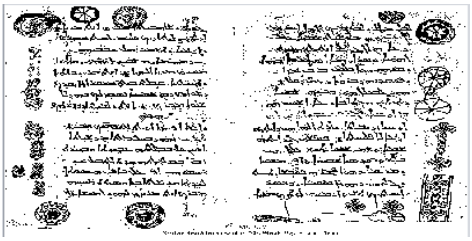
El fondo se volvió claro permitiendo hacer notorio donde se encuentran zonas obscuras no deseables

Umbralización

Se eliminaron esas zonas amarillentas obscuras que se apreciaban en el paso anterior, dejando un poco de ruido en los extremos y el centro del documento

Negativo

Se hace más notorio el ruido detectado en el paso anterior



Media Ponderada

Suavizando la imagen se eliminó el ruido presente

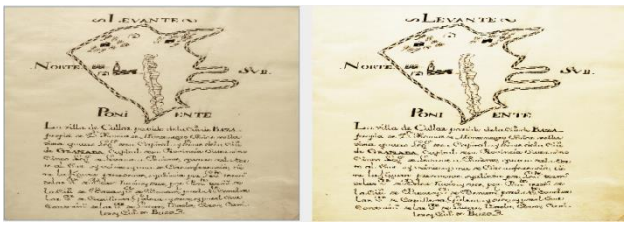
Media Gaussiana-Resultado

Al querer tener más definidas las letras del documento se produjo un ruido que se había eliminado en el paso anterior

Documento 2

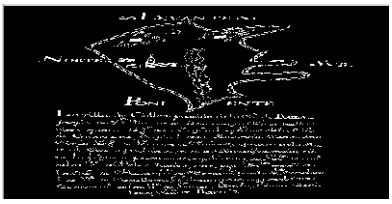
Original→Sigmoide

El fondo se aclaró y la información se preservo



Umbralización

Se eliminó el fondo claro, la información no se puede saber si está completa

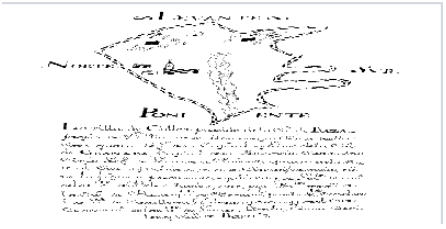


Negativo

La información esta recuperada como se desea, pero tiene que seguir aun el proceso

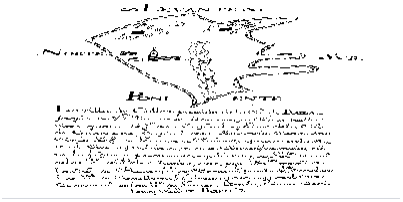
Media Ponderada

La imagen suavizada hizo que la información se volviera opaca haciéndola poca legible



Media Gaussiana-Resultado

Las letras tienen un color más obscuro arreglando el problema del paso anterior.



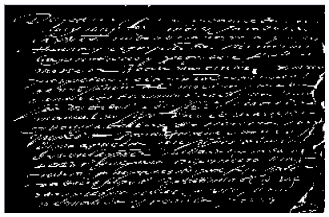
Documento 3

Original→Sigmoide

El fondo es casi blanco a excepción de una zona en la parte derecha.

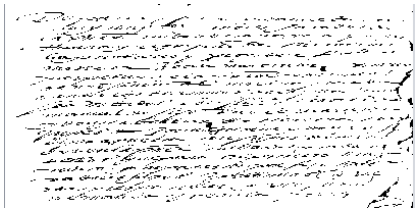
Umbralización

Parece ser que la zona derecha donde no estaba del todo clara en el paso anterior desapareció**.**



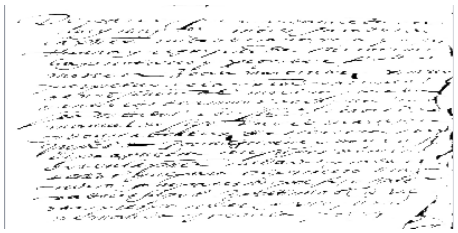
Negativo

Se presenta un poco de ruido y pequeñas zonas obscuras



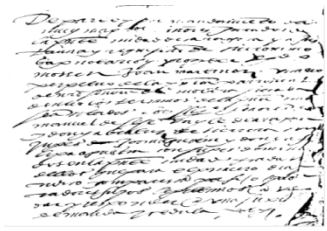
Media Ponderada

El intento de eliminar el ruido no funciono provocando además que las letras se vuelvan opacas



Media Gaussiana-Resultado

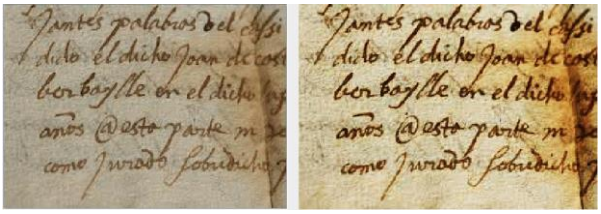
Resaltando el contenido se puede ver también que el ruido se eliminó en pequeñas zonas, aunque no en todo el documento



Documento 4

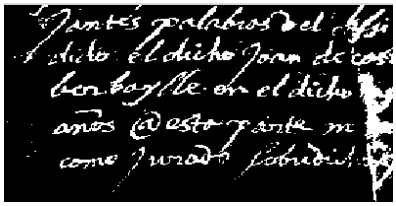
Original→sigmoide

Se obtuvo una información más remarcada y un fondo mas claro



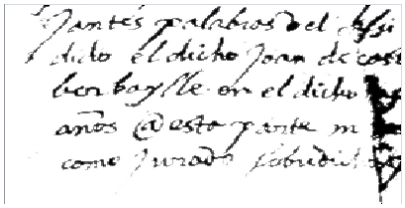
Umbralización

Se aprecia una mancha blanca en el lado derecho aunque no afecta la información, también algunas letras están incompletas



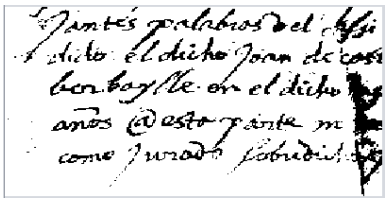
Negativo

Se hace notorio un poco de ruido y las letras han perdido un poco su forma



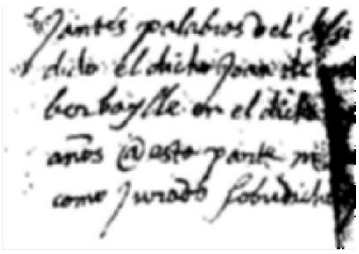
Media Ponderada

Se empieza a recuperar un poco la forma las letras



Media Gaussiana-Resultado

Se recuperó el texto, aunque un poco de ruido no se elimino



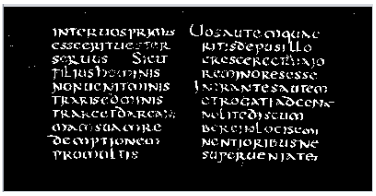
Documento 5

Original →Sigmoide

El fondo se volvió mas claro

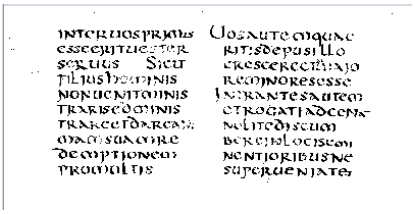
Umbralización

Se aprecia que se recuperó toda la información



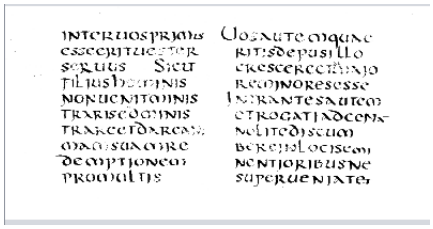
Negativo

El documento es el deseado pero aún tiene un poco de ruido en la parte izquierda y arriba



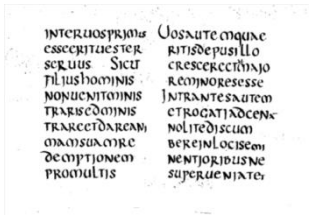
Media Ponderada

Se eliminó el poco ruido que existía



Media Gaussiana-Resultado

El resultado es bueno, se recuperó al información y no hay presencia de ruido



Documentos Malos

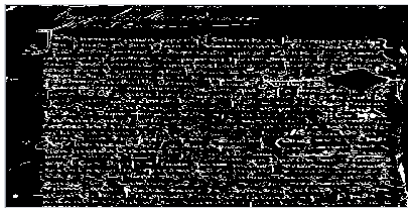
Documento 1

Original→Sigmoide

Se aclaró el fondo eliminando un poco las manchas obscuras

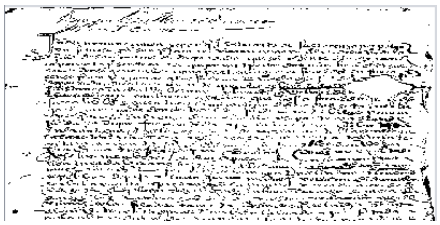
Umbralización

Se puede apreciar donde existe ruido que podría provocar la pérdida de información



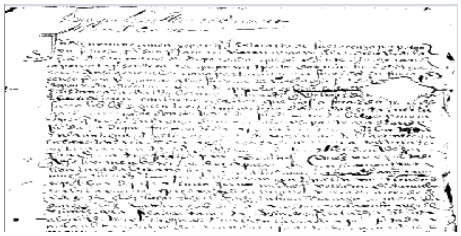
Negativo

Se ve ruido en los extremos, en la parte central la información esta opaca



Media Ponderada

Suavizar elimino un poco de ruido en la parte superior derecha pero en el resto del documento no, las letras se opacaron demasiado



Media Gaussiana-Resultado

Al recuperar la información perdida en el paso anterior provoco que existiera el mismo ruido presente desde el inicio del proceso

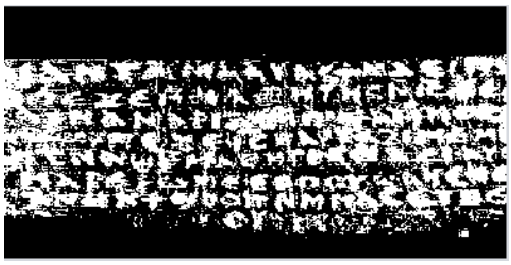
Documento 2

Original→Sigmoide

Se resalta la información aunque el fondo no es del todo claro

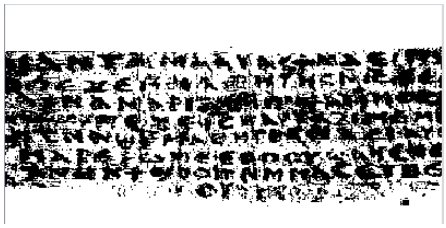
Umbralización

Se ve el ruido existente en la parte derecha e izquierda del documento



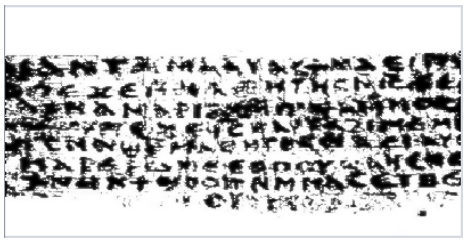
Negativo

Las manchas negras en no permiten que sea legible la información



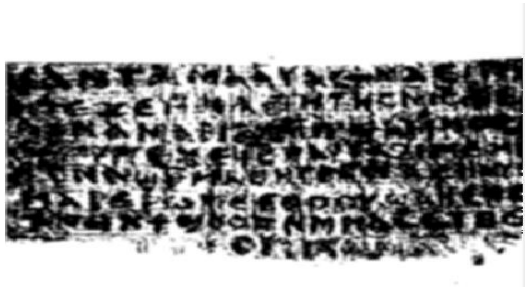
Media Ponderada

Se eliminan las manchas negras pero no del todo



Media Gaussiana-Resultado

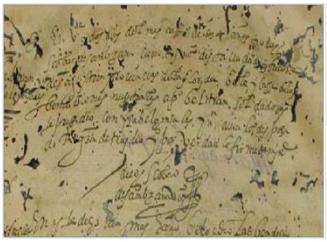
El querer resaltar la información no fue bueno porque el documento se oscureció mucho a comparación con el paso anterior



Documento 3

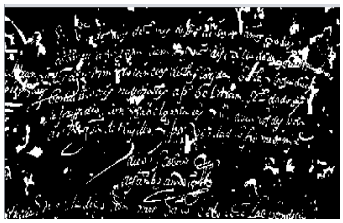
Original→Sigmoide

La información se ve más resaltada, el fondo claro pero unas zonas negras que no hacer ver bien el documento



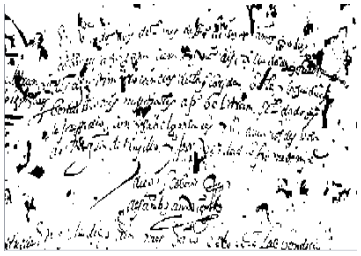

Umbralización

Se aprecia la información, así como también zonas blancas que son ruido



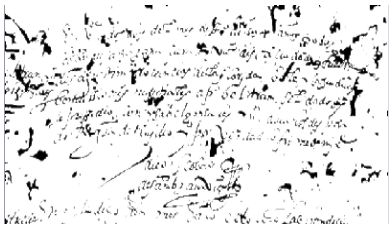
Negativo

Se nota perdida de información y el ruido presente



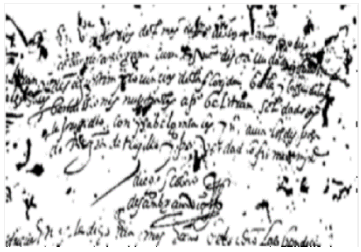
Media Ponderada

No se eliminó el ruido pero si parte de la información



Media Gaussiana-Resultado

La información se recuperó pero el ruido no se pudo eliminar



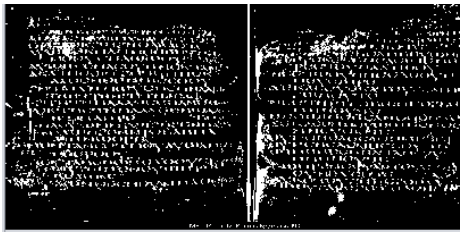
Documento 4

Original→Sigmoide

Se resaltó la información pero se ve zonas obscuras que tal vez no puedan ser eliminadas

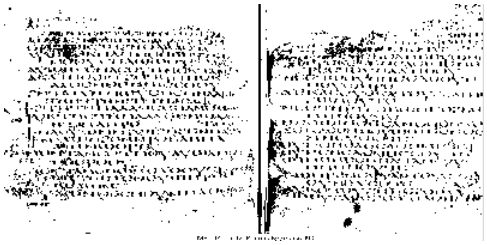
Umbralización

Se eliminaron las zonas obscuras, no sabemos si completamente pero, podemos suponer que esas manchas blancas son ruido



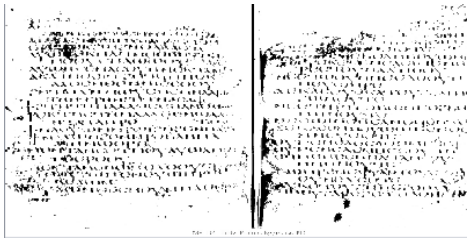
Negativo

Existe ruido en la parte izquierda y central del documento



Media Ponderada

Se eliminó un poco de ruido en la parte izquierda del documento, las letras son muy opacas



Media Gaussiana-Resultado

Las letras están perfectamente remarcadas pero, hay mucho ruido.
En los pasos anteriores se fue eliminando el ruido pero aquí se originó más.

Documento 5

Original→Sigmoide

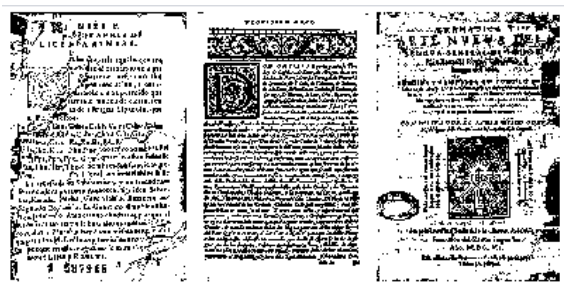
La información se ve resaltada pero el fondo en zonas es claro y en otros obscuros

Umbralización

El texto es muy pequeño, las zonas blancas no sabemos si es parte de la información o ruido

Negativo

Hay mucho ruido en la parte derecha e izquierda del documento



Media Ponderada

No se eliminó el ruido pero, la información se opaco

Media Gaussiana-Resultado

Resaltando las letras el documento se oscureció demasiado

Conclusión

La tarea de rescatar la información en documentos antiguos no es sencilla, engloba muchos aspectos como son la calidad del papel del documento, la cantidad de información que se desea rescatar, los cambios de tonalidad.

Haciendo uso de los algoritmos existentes para el tratamiento de imágenes digitales de aclarado, obscurecimiento, contraste, brillo, detección de bordes, etc. Creamos una metodología para la recuperación de documentos antiguos que funciona perfectamente en documentos donde tienen un fondo claro y/o la información está en su mayor parte perfectamente definida; En aquellos donde la información es poco opaca y en el fondo existe un ligero cambio de tonalidad los resultados varían. En cambio en documentos donde en su fondo existen muchos cambios de tonalidad es muy difícil obtener los resultados deseados.

Así, queda la tarea de seguir encontrando nuevas formas en las que los algoritmos existentes sean de mayor utilidad o permitan que originen nuevas técnicas que sean superiores a estas.

“Nuestra historia es un patrimonio que no debemos perder, que debemos hacer el mayor esfuerzo para mantenerlo vivo, para el futuro nuestro y de futura generaciones”

Bibliografía:

http://www.masadelante.com/faqs/pixel
dialnet.unirioja.es/descarga/articulo/3869409.pdf
http://repositorio.bib.upct.es/dspace/bitstream/10317/1811/1/pfc3161.pdf
http://www.mediamarkt.es/mp/article/Ruido-de-la-imagen,976504.html
http://isa.umh.es/asignaturas/crss/temasvision/t05.pdf
http://tierradecullar.blogspot.mx/
www.cullera.es/es/policialocal/documentos-antiguos

blog.myheritage.es/tag/descifrar-escrituras-antiguas/
www.museodelapara.gov.ar/nuestrahiostoriaenimagenes.html
http://www.republica.com/2013/02/12/desarrollan-un-software-capaz-de-reconstruir-lenguas-antiguas_613091/
tierradecullar.blogspot.com/
www.sofiaoriginals.com
librosygrabados.blogspot.com/2014_02_01_archive.html
www.barriodemovera.com/movera_004.htm
http://cgbuxan.blogspot.mx/2013/01/documentos-arboles-libros-demasiados.html
www.upf.edu/asia/projectes/che/che16.htm
www.komvos.edu.gr/glwssa/odigos/thema_d10/d_10_thema.htm
www.delcampe.net › Todo Delcampe.net › Documentos antiguos

Benemérita Universidad

Autónoma de Puebla

Munive Morales Luis

Gerardo

[PRIMER EXAMEN PARCIAL]

Proceso Digital de Imágenes-Aplicación de filtros regionales

ANTECEDENTES:

Como se ha visto anteriormente se le pueden aplicar diferentes filtros a una imagen, ya sean de obscurecimiento, aclarado o mixtos, por otro lado también se pueden aplicar los llamados “Filtros Regionales” a la imagen.

¿Qué son los filtros regionales?

Sol filtros de obscurecimiento, aclarado o mixtos que se le aplican a una imagen, sin embargo esto se hace en un rango determinado, es decir dependiendo del valor que tenga el pixel se le aplicara un filtro especifico.

Los filtros que se le pueden aplicar a una imagen se reducen a ecuaciones matemáticas, las cuales modifican el valor del pixel en sus 3 canales (rojo, verde, azul), estas ecuaciones tienen que cumplir con ciertas características, en el caso específico de que un solo filtro (f(x)) se le aplique a una imagen el filtro debe cumplir con:

- f(0)=0
- f(255)=255
- f(a)=c; donde 0< c <255

En el caso de un filtro regional f(g) comprendido entre los rangos x0-x1, f(g) debe cumplir con:

- f(x0)=x0
- f(x1)=x1
- f(b)=k; donde x0 < k < x1

Es importante mencionar que los filtros que se le aplican a una imagen se pueden “transformar” a filtros regionales, esto lo veremos a continuación.

DESARROLLO:

Los filtros que se transformaran a regionales son los que hemos estado ocupando anteriormente:

seno, coseno, corrección gamma, exponencial, exponencial obscurecimiento y logarítmico.

FILTRO LOGARITMICA

La función se define como:

$x' = A \ln(\alpha x + 1), \alpha > 1, x \in [0, q]$ (q normalmente toma el valor de 255)

$A = q / \ln(\alpha q + 1)$

En esta función al usuario se le pide que proporcione el valor de α, este valor no debe de ser cero.

Para que esta fórmula pueda ser aplicada en un intervalo x0,x1 se le tienen que hacer unas modificaciones:

Ajustaremos algunos parámetros de la fórmula:

q=x1

Q= x1-x0+1

La función queda como:

F(x)= A ln(α(x-x0) +1)+x0; donde A = q-x0 / ln(αQ)

FILTRO SENO

La función se define como:

X' = q sin(πx / 2q), donde x es el valor del pixel en alguno de sus canales.

Esta función no le pide ningún valor al usuario. Para que esta fórmula pueda ser aplicada en un intervalo x0,x1 se le tienen que hacer unas modificaciones:

Ajustaremos algunos parámetros de la fórmula:

q=x1

Q= x1-x0+1

La función queda como:

F(x') = (q-x0)*sin(π(x-x0) / 2(q-x0))

FUNCION EXPONENCIAL

La función se define como:

X' = A(1-e-αx/q), donde α ∈ [0, q] y A = q / (1-e-α)

En esta función al usuario se le pide que proporcione el valor de α, este valor no debe de ser cero.

Para que esta fórmula pueda ser aplicada en un intervalo x0,x1 se le tienen que hacer unas modificaciones.

Ajustaremos algunos parámetros de la fórmula:

q=x1

Q= x1-x0+1

La función queda como:

$F(X') = A(1-e^{-\alpha(x-x0)/q-x0})$, donde $\alpha \in [0, q]$; donde $A = q-x0 / (1-e^{-\alpha})$

FUNCION COSENO

La función coseno se define como:

$$x' = q\left(1 - \cos\left(\frac{\pi x}{2q}\right)\right)$$

Esta función no le pide ningún valor al usuario. Para que esta fórmula pueda ser aplicada en un intervalo x0,x1 se le tienen que hacer unas modificaciones:

Para que esta fórmula pueda ser aplicada en un intervalo x0,x1 se le tienen que hacer unas modificaciones.

Ajustaremos algunos parámetros de la fórmula:

q=x1

Q= x1-x0+1

$F(X^1)=(q-x0)[1-Cos(\pi(x-x0)/2(q-x0))]+x0$

FUNCIÓN EXPONENCIAL DE OBSCURECIMIENTO:

La función se define como:

$$x' = A\left(e^{\alpha x/q} - 1\right), \quad \alpha > 0$$

donde:

$$A = q/(e^{\alpha} - 1)$$

En esta función al usuario se le pide que proporcione el valor de α, este valor no debe de ser cero.

Para que esta fórmula pueda ser aplicada en un intervalo x0,x1 se le tienen que hacer unas modificaciones.

Ajustaremos algunos parámetros de la fórmula:

q=x1

Q= x1-x0+1

$F(x')= A (e^{\alpha(x-x0)/q-x0})+ x0$; donde $A= q-x0/e^{\alpha}-1$

CORRECCIÓN GAMMA

La función se define como:

$$r' = q\left(\frac{r}{q}\right)^{\gamma}$$

En esta función al usuario se le pide que proporcione el valor de γ, este valor no debe de ser cero.

Para que esta fórmula pueda ser aplicada en un intervalo x0,x1 se le tienen que hacer unas modificaciones:

Ajustaremos algunos parámetros de la fórmula:

q=x1

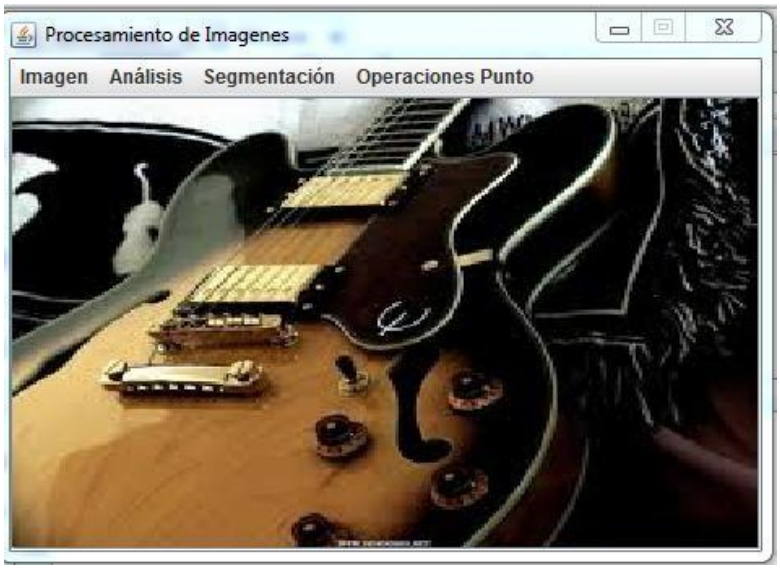
Q= x1-x0+1

F(X')= (q-x0)(x-x0/q-x0)^{r}+x0

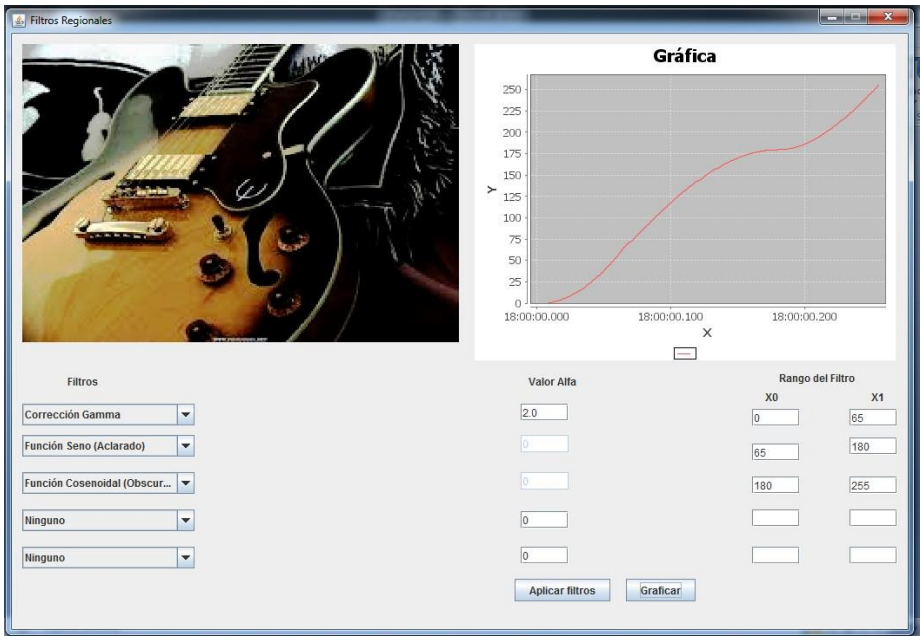
Una vez que se han modificado las fórmulas correspondientes, se programan dentro de la clase de OpPunto, para así poder aplicarlas a la imagen deseada, de la misma forma se programan las mismas funciones para la clase grafica, para así poder construir la gráfica final de la imagen.

RESULTADOS:

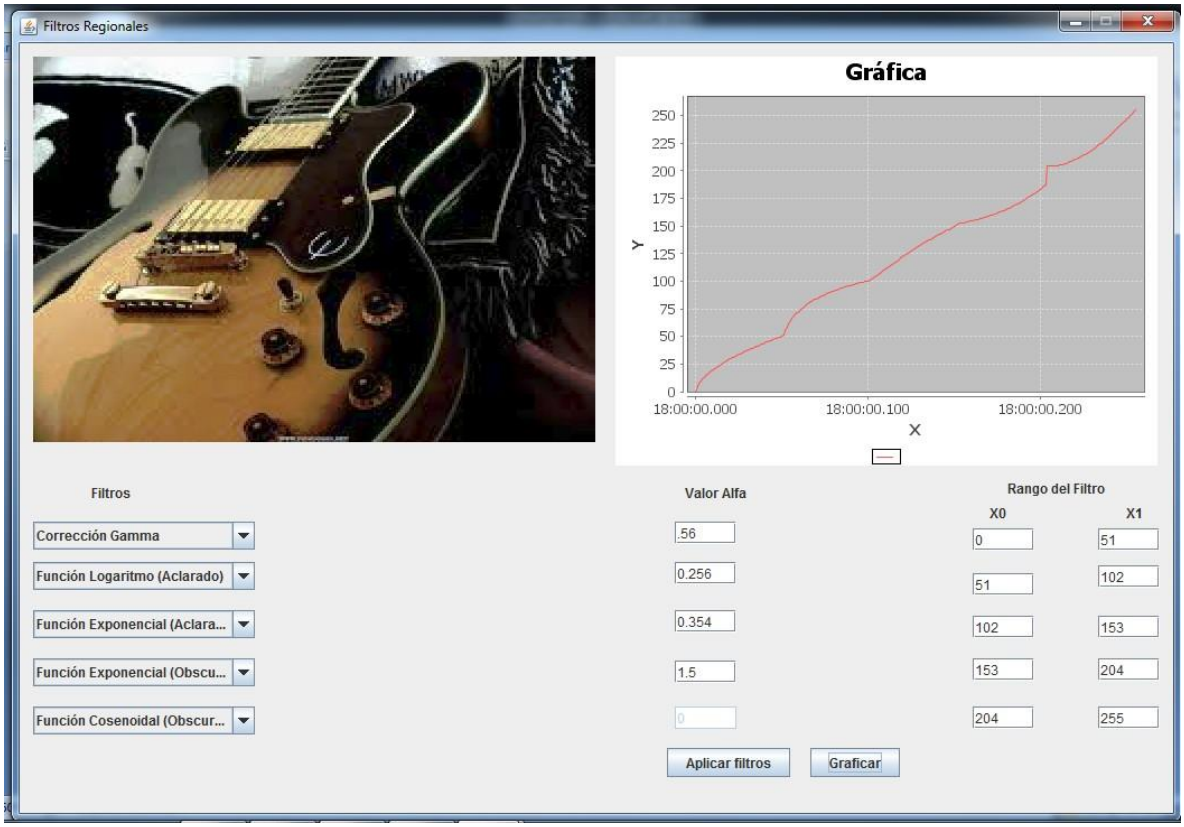
A una sola imagen se le aplicarán diferentes filtros regionales y se mostrarán los resultados


Procesamiento de Imagenes
Imagen
Análisis
Segmentación
Operaciones Punto

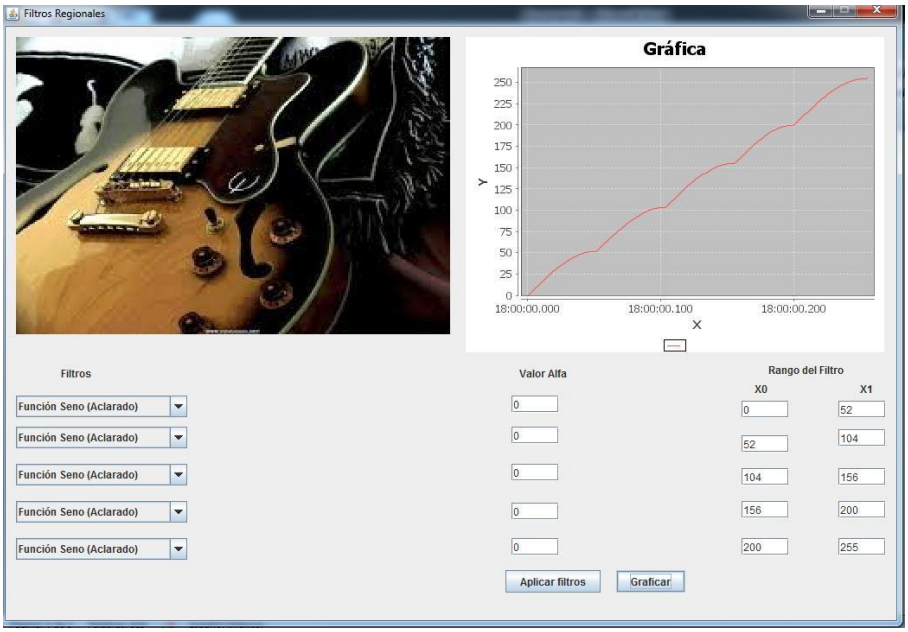
Aplicando 3 filtros a la imagen



Aplicando 5 filtros a la imagen.



Aplicando 5 veces el mismo filtro a la imagen



CONCLUSIONES:

Podemos concluir que la aplicación de los filtros regionales es compleja en su implementación debido a que el filtro debe de actuar exclusivamente en los rangos definidos, al igual que en los filtros comunes el valor resultante debe encontrarse entre 0 y 255,.

Al aplicar filtros regionales a una imagen se puede “manipular” esta imagen de una manera más detallada, modificando los rangos, el filtro, etc. hasta obtener l resultado deseado

REFERECNCIAS:

http://docs.oracle.com/javase/6/docs/api/java/awt/image/BufferedImage.html

http://docs.oracle.com/javase/7/docs/api/java/awt/Color.html

http://iolmos.cs.buap.mx/pdi/Sesion4_Operaciones_Punto.pdf

http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/package-summary.html

http://mmartin.cs.buap.mx/pdi/proy/Proy-PID-P1-prim-2012.pdf

Benemérita Universidad Autónoma de Puebla

Imágenes panorámicas

Materia: Proceso Digital de Imágenes

Munive Morales Luis Gerardo

Sánchez Hernández Diego

Acevedo Carrera Ignacio

14

INTRODUCCION:

Las fotografías panorámicas muestran un área amplia del campo visual. Para ello se unen una serie de imágenes tomadas desde un mismo punto de vista. El proceso general se puede describir en tres pasos fundamentales: captura de las imágenes, registrado y unión.

De estos pasos, el más importante y complejo es el de registrado de imágenes. Este proceso consiste en encontrar una transformación que, aplicada a una imagen, permita mapear sus puntos comunes en la otra. Es decir, ante dos imágenes, el registrado consistirá en buscar las zonas comunes que representen la misma información y, a partir de ellas, encontrar una transformación que permita unir ambas imágenes.

DESARROLLO

El proceso de creación de fotografías panorámicas se puede descomponer en varios pasos:

i. *Adquisición de las imágenes*. Para un mejor resultado es conveniente que las imágenes se tomen desde un mismo punto de vista. En este paso hay que tener cuidado con parámetros de la cámara como la exposición y el balance de blancos. Cuanto más uniformes sean las imágenes mejor resultado final se obtendrá.

ii. *Pre procesamiento de las imágenes*. Transformaciones orientadas a facilitar el proceso de unión de las imágenes.

iii. *Registrado de imágenes*. Búsqueda de las zonas comunes en las imágenes. En este paso se debe encontrar el área de superposición de las imágenes y calcular la transformación que permita su unión. Más adelante se detallarán los aspectos relacionados con este paso.

iv. *Unión de las imágenes*. Procesamiento necesario para unir las imágenes de acuerdo a la transformación calculada en el apartado anterior. Durante esta etapa se pueden llevar a cabo varios ajustes orientados a disminuir lo máximo posible las diferencias de las imágenes en la zona de superposición, de modo que la transición entre ellas sea lo más suave posible.

El registrado de imágenes es el proceso de superponer dos o más imágenes obtenidas en diferentes instantes de tiempo, desde diferentes puntos de vista y/o con diferentes sensores. Es decir, se trata de localizar zonas u objetos que representan la misma información en imágenes diferentes. La inexistencia de una correspondencia verdadera otorga al registrado de imágenes la categoría de problema.

La mayoría de métodos de registrado consisten en los siguientes cuatro

- *Detección de características*. Detección de forma automática o manual de características descriptivas de las imágenes.

- *Emparejamiento de características*. Se establece la correspondencia entre las características detectadas en las imágenes.

- *Estimación del modelo de transformación*. Se estiman los parámetros de las funciones de mapeo, que alinean ambas imágenes. Estos parámetros se calculan considerando los emparejamientos establecidos anteriormente.

- *Transformación*. La imagen se transforma mediante las funciones de mapeo (Homografia).

En la Figura 2.5 podemos observar estos pasos. En la fila superior se lleva a cabo la detección de características, en este caso esquinas. En la fila intermedia se pueden observar los emparejamientos marcados con números. Finalmente, en la fila inferior se calcula el modelo de transformación y se aplica.

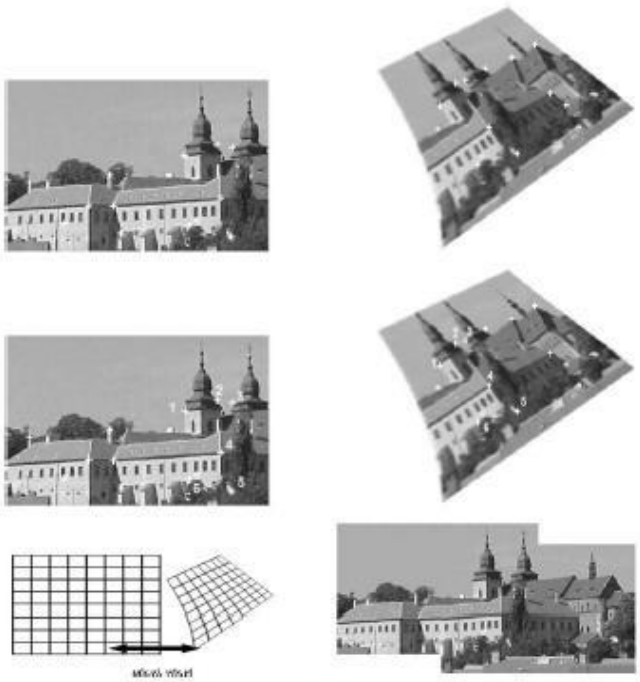


Figura 2.5: Pasos del registrado de imágenes

La fase de correspondencia es, en definitiva, una etapa fundamental en la que, dependiendo del criterio adoptado para la construcción de emparejamientos, la transformación se verá afectada

por la calidad de las correspondencias. Cuanto mejor sea la correspondencia, más próxima será la estimación al valor óptimo.

CREACION DE UNA IMAGEN PANORAMICA:

Imagen 1

Imagen 2

Emparejamiento SURF

Inliers RANSAC 1

Inliers RANSAC 2

Imágenes alineadas según una homografía

Correspondencias entre imágenes. El objetivo de esta etapa es encontrar todas las imágenes con correspondencias, es decir, que se superpongan. Los conjuntos conexos de imágenes serán, posteriormente, las panorámicas.

Para conseguir un buen resultado, basta con comparar cada imagen con un número reducido de imágenes que se superponen. En el paso anterior se han identificado imágenes con un gran número de emparejamientos entre ellas.

Primero se utiliza RANSAC para seleccionar un conjunto de inliers que son compatibles con una homografía entre las imágenes. En la Figura 2.11 se puede observar cómo, para dos imágenes con un gran número de correspondencias, se aplica RANASC para computar la homografía, de acuerdo al conjunto de inlieers.

Después se aplica un modelo probalístico para verificar la correspondencia. La idea del modelo de verificación es comparar las probabilidades de que el conjunto de inliers/outliers fuese generado por una correspondencia entre imágenes correcta o por una falsa.

Una vez establecidas las correspondencias por parejas entre las imágenes, se puede encontrar secuencias de panorámicas como conjuntos de imágenes con correspondencias. Esto permite reconocer varias panorámicas en un conjunto de imágenes, y rechazar las imágenes ruido sin correspondencia

Algoritmos utilizados:

SURF (Speeded Up Robust Features)

SURF es un “método para extraer características invariantes distintivas de imágenes que pueden ser usadas para llevar a cabo un *matching* fiable entre distintas vistas de un objeto o escena” Estas características locales (o puntos) son invariantes a escalado y rotación, y parcialmente invariantes a cambios en iluminación y punto de vista.

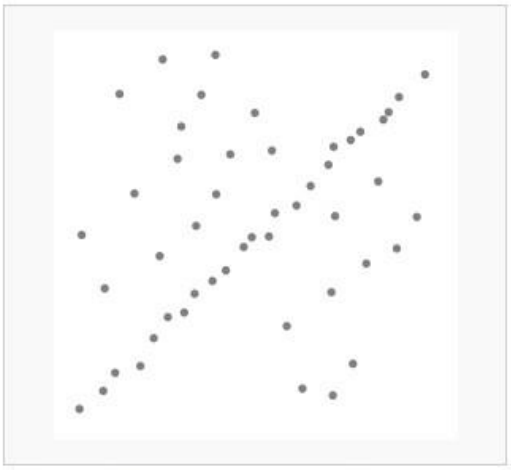
RANSAC es una abreviatura de "Consenso muestra aleatoria". Se trata de un método iterativo para calcular los parámetros de un modelo matemático de un conjunto de datos observados que contiene valores atípicos . Se trata de un algoritmo no determinista en el sentido de que produce un resultado razonable sólo con una cierta probabilidad, con esta probabilidad cada vez mayor a

medida que más iteraciones se permiten. El algoritmo fue publicado por primera vez por Fischler y Bolles en SRI International en 1981.

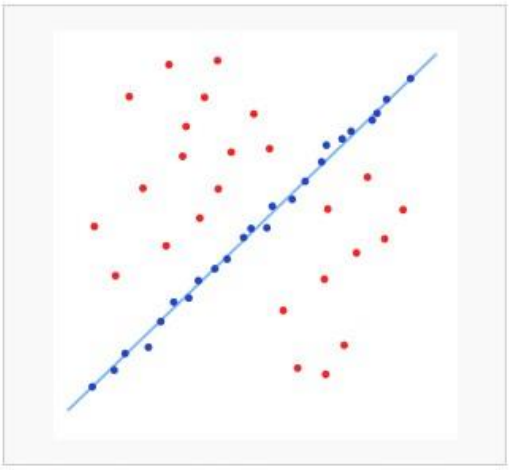
Un supuesto básico es que los datos consisten en "inliers", es decir, los datos cuya distribución puede ser explicada por un conjunto de parámetros del modelo, aunque puede estar sujeto al ruido, y los "valores atípicos", que son datos que no encajan en el modelo. Los valores extremos pueden llegar, por ejemplo, de valores extremos del ruido o de las mediciones erróneas o hipótesis erróneas acerca de la interpretación de los datos. RANSAC también supone que, dada

una (generalmente pequeño) conjunto de inliers, existe un procedimiento que puede estimar los parámetros de un modelo que explica de manera óptima o se adapte a estos datos.

Ejemplo:



Un conjunto de datos con muchos valores atípicos para los que una línea tiene que ser instalado.



Línea Equipado con RANSAC; valores atípicos tienen ninguna influencia en el resultado.

CONCLUCIONES

Este proyecto tiene como principal objetivo evaluar imágenes y seleccionar métodos de alineamiento que nos permitan trabajar con pixeles coincidentes de las imágenes, es por eso que mediante la aplicación de los algoritmos surf y ransac se han obtenido resultados bastante convincentes y un emparejamiento de pixeles estable ante diferente unión de imágenes.

Pudimos conocer la importancia de implementar nuevos algoritmos con los cuales nos llevaron a emplear técnicas de emparejamiento y crear imágenes panorámicas atreves de mosaicos.

REFERENCIAS

http://neithan.weebly.com/uploads/5/2/8/0/52807/memoria_-_final.pdf

Segundo examen Parcial

Procesamiento de documentos antiguos

27/03/2014

Mun
ive
Mor
ales
Luis
Ger
ardo
Sam
uel
Her
nan
dez
Per
ez

rtyuiopas sdfghjklzx

cvbnmq cvbnmq

wertyuiopasdfghjklzxcvbnmqwertyuio
pasdfghjklzxcvbnmqwertyuiopasdfghj
klzxcvbnmqwertyuiopasdfghjklzxcvbn
mqwertyuiopasdfghjklzxcvbnmqwerty
uiopasdfghjklzxcvbnmqwertyuiopasdf
ghjklzxcvbnmqwertyuiopasdfghjklzxc
vbnmqwertyuiopasdfghjklzxcvbnmrty
uiopasdfghjklzxcvbnmqwertyuiopasdf
ghjklzxcvbnmqwertyuiopasdfghjklzxc

INTRODUCCIÓN:

Las técnicas de tratamiento de imágenes vistas anteriormente se aplicaran en imágenes de documentos antiguos, con el propósito de rescatar la mayor información posible, eliminando el "ruido" (manchas, marcas de pliegues, etc.) encontrado en la imagen.

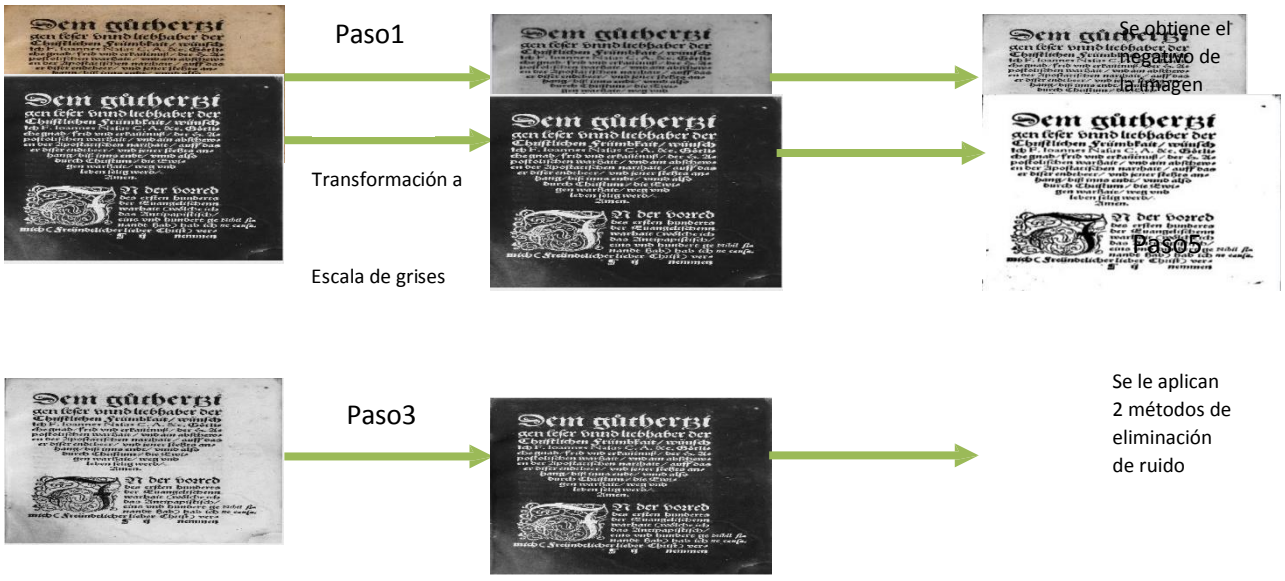
Esto se logrará a través de la aplicación de técnicas de procesamiento de imágenes a través de un algoritmo.

Las técnicas a utilizar en este proceso son:

- Transformación a Escala de Grises
- Umbralización
- Obtención de negativo
- Sigmoide
- Método de Otsu
- Media ponderada y media geométrica

DESARROLLO:

El tratamiento de documentos antiguos se llevará a cabo de la siguiente forma:



Paso2

Se umbraliza la imagen

Paso4

Se aplica un filtro sigmoide a la imagen

Paso6

Finalmente se aplica el método de otsu

Paso 1

El primer paso para el tratamiento de documentos antiguos es transformar la imagen a escala de grises, de esta manera solo trabajaremos con un solo canal, lo que facilitará el manejo de la imagen.

La escala de grises consistió en obtener la media de los 3 canales de la imagen. Esto es se obtuvo el valor de cada canal, se sumaron todos y el resultado se dividió entre 3, y el nuevo valor se le agrego a los 3 canales de la imagen.

Paso 2

Una vez obtenida la escala de grises se procede a umbralizar la imagen, este proceso se logra a través de la binarización de la imagen, primero se había determinado que la función fuese la quien determinara el valor de la umbralización, sin embargo, en la mayoría de los casos se perdía información, por lo que se decidió que este valor fuese brindado desde el interior, tras una serie de pruebas e investigación en diferentes artículos se determino que el valor del umbral fuese 160, con este valor no se perdía información.

Paso 3

Posteriormente se obtuvo el negativo de la imagen, esto se hace porque de esta manera el fondo de la imagen se ve “más uniforme” y lo escrito en el documento se resalta mucho más.

Paso 4

Una vez obtenido el negativo se aplica un filtro sigmoide el cual obscurecerá mas el fondo y aclarará el texto del documento, dándonos como resultado una mejor imagen

Paso 5

El siguiente paso a realizar es aplicar es la técnica de suavizado de media ponderada, con esa técnica se eliminará el ruido que tenga la imagen, posteriormente se aplica otra técnica de suavizado, media geométrica, que de igual manera elimina el ruido de la imagen, un dató importante es que al aplicar estas técnicas de suavizado se notó que, no sólo eliminaban el ruido de la imagen, si no que al mismo tiempo engrosaban y definían mas el texto del documento. Este paso no es muy notorio a

simple vista ya que se esta trabajando con el negativo de la imagen, pero sus resultados se pueden observar cuando se le aplica el método de Otsu posteriormente

Media ponderada

Este método consiste en eliminar el ruido de la imagen, esto nos ayuda a eliminar manchas y marcas en la imagen modificando el valor de los pixeles, el nuevo valor del pixel se obtiene a través de la siguiente fórmula

$$P'[i,j] = \bar{M}_{MP}(P[i,j]) \otimes \bar{P}_3[i,j] = \frac{1}{8+P[i,j]} \begin{bmatrix} 1 & 1 & 1 \\ 1 & P[i,j] & 1 \\ 1 & 1 & 1 \end{bmatrix} \otimes \bar{P}_3[i,j]$$

Media Geométrica

Este método tiene el mismo objetivo, eliminar el ruido de la imagen, solo que este se realiza a través de la norma del vector. El nuevo valor del pixel se obtiene a través de la siguiente fórmula

$$X = \{x_1, \ldots x_n\}, \quad \| X \| = \sqrt[n]{x_1 x_2 \cdots x_n}$$

$$P'[i,j] = \sqrt[9]{\prod_{x=-1}^{1} \prod_{y=-1}^{1} P[i+x, j+y]}$$

Paso 6

Finalmente se aplica el método de Otsu a la imagen, este es un método de segmentación po umbralización.

El método de Otsu es un procedimiento no paramétrico que selecciona el umbral óptimo maximizando la varianza entre clases mediante una búsqueda exhaustiva. La Principal ventaja de este método es la buena respuesta del método frente a la mayoría en situaciones del mundo real (imágenes ruidosas, con histogramas planos, mal iluminadas…).

Descripción

Partimos de una imagen en niveles de gris con *N* píxels y *L* posibles niveles diferentesy la probabilidad de ocurrencia del nivel de gris *i* en la imagen:

$$p_i = \frac{f_i}{N}$$

fi → Frecuencia de repetición del nivel de gris *i-ésimo* con *i = 1,2,…,L.*

En el caso particular de umbralización en dos niveles (binarización), los píxels se dividen en dos clases → ***C1*** y ***C2***, con niveles de gris *[1,2,…,t]* y *[t+1,t+2,…,L]* respectivamente, donde las distribuciones de probabilidad de ambas clases son:

$$C_1 : \quad \frac{p_1}{\omega_1(t)}, \ldots, \frac{p_t}{\omega_1(t)}$$
$$C_2 : \quad \frac{p_{t+1}}{\omega_2(t)}, \frac{p_{t+2}}{\omega_2(t)}, \ldots, \frac{p_L}{\omega_2(t)}$$

Las medias para cada una de las clases se definen como:

$$\mu_1 = \sum_{i=1}^{t} \frac{i.p_i}{\omega_1(t)} \quad \mu_2 = \sum_{i=t+1}^{L} \frac{i.p_i}{\omega_2(t)}$$

La intensidad media total de la imagen se define, siendo fácil demostrar así mismo:

$$\omega_1.\mu_1 + \omega_2.\mu_2 = \mu_T \quad \omega_1 + \omega_2 = 1$$

Haciendo uso de un análisis discriminante, Otsu definió la varianza entre clases de una imagen umbralizada como:

$$\boxed{\sigma_B^2 = \omega_1.(\mu_1 - \mu_T)^2 + \omega_2.(\mu_2 - \mu_T)^2}$$

La idea es ahora encontrar el umbral, *t,* que maximice la varianza (Otsu demostró que este era el umbral óptimo):

$\boxed{t^* = \underset{t}{Max}\{\sigma_B^2(t)\}}$ dónde: $1 \le t \le L$

En este caso, al existir *M* clases, existirán *M-1* umbrales distintos, generalizando el caso particular anteriormente descrito. Por tanto, en este caso habremos de obtener el conjunto multinivel que maximice la varianza entre clases de la forma:

$$\boxed{\{t_1^*, t_2^*, ..., t_{M-1}^*\} = \underset{t_1 t_2 ... t_{M-1}}{Max}\{\sigma_B^2(t_1, t_2, ..., t_{M-1})\}} \quad 1 \le t_1 < ... < t_{M-1} < L$$

Dónde:

$$\boxed{\sigma_B^2 = \sum_{k=1}^{M} \omega_k.(\mu_k - \mu_T)^2} \quad \omega_k = \sum_{i \in C_k} p_i \quad \mu_k = \sum_{i \in C_k} \frac{i.p_i}{\omega_k}$$

RESULTADOS:

En las siguientes tablas se muestran los resultados al aplicar el proceso paso a paso a las imágenes, las primera 2 son consideradas buenas imágenes, la tercera es regular y la última es mala.

Se considera buena imagen a aquella cuyo fondo es uniforme y las letras están bien definidas, la imagen regular es aquella que cuenta con un par de manchas y sombras con el texto definido y por último se considera imagen mala a aquella que tiene demasiadas manchas, colores, dobleces

Como el paso 5 no es notable a simple vista, su resultado se muestra en conjunto con el 6

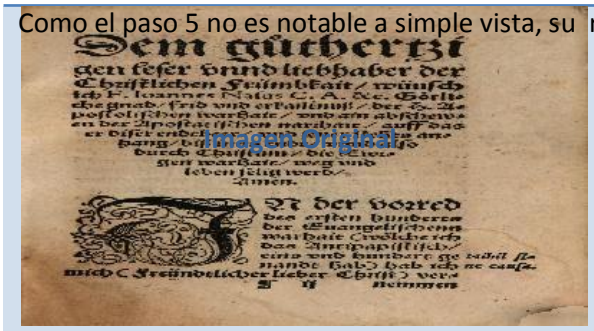
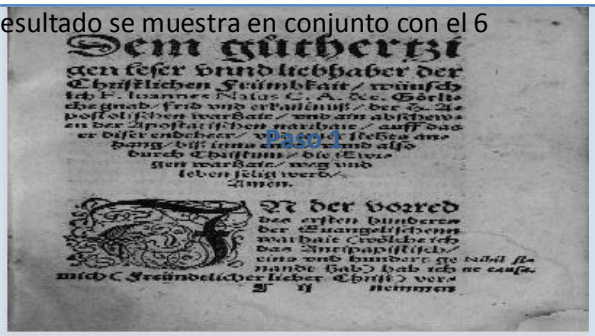
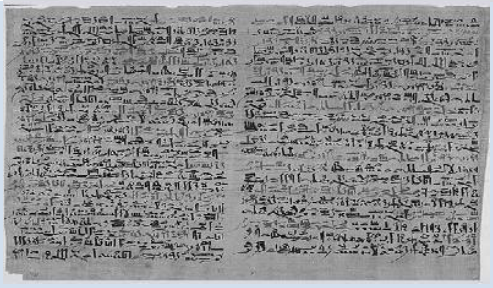


Imagen Original

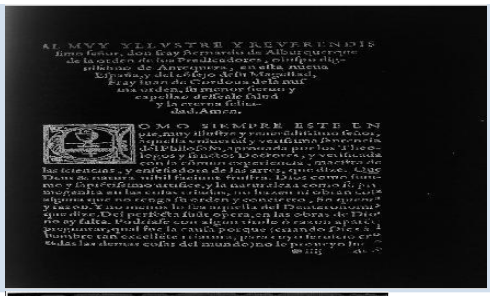

Paso 1

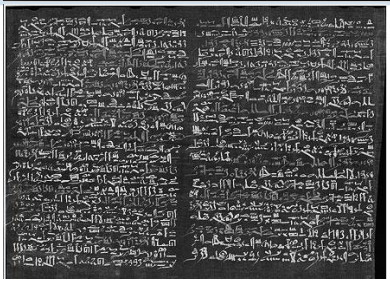


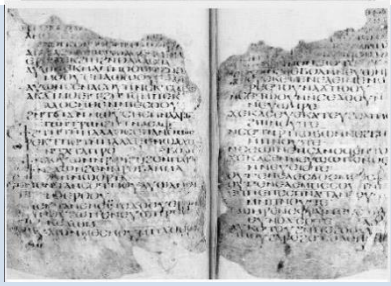
Paso 2

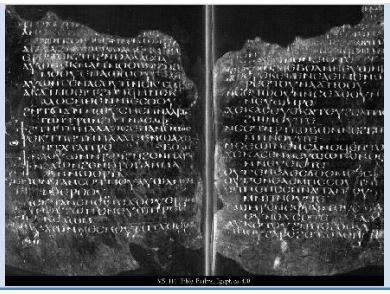

Dem güthertzi
gen leser vnd liebhaber der
Christlichen Frümbkait / wünsch
ich F. Iouanes Natus C. A. &c. Gnad
che gnad / frid vnd erbarmnüß / der H.
postolischen warhaite / vnd am abschew
en der Apostatischen narrhaite / auff das
er diser endeleer / vnd jener fleschea am
hang / biß inns ende / vnd also
durch Christum die ewig
gen warhaite / erlangt
leben selig werd.
Amen.
N der vorred
des ersten hunderts
der Euangelischenn
warhaite Gnädigcleich
das Antipapistisch /
eins vnd hundert gr Nihil fa
mandt hab / hab ich ne causa.
mich (Freündlicher lieber Christ) vers
nemmen


Dem güthertzi
gen leser vnd liebhaber der
Christlichen Frümbkait / wünsch
ich F. Iouanes Natus C. A. &c. Gnad
che gnad / frid vnd erbarmnüß / der H.
postolischen warhaite / vnd am abschew
en der Apostatischen narrhaite / auff das
er diser endeleer / vnd jener fleschea am
hang / biß inns ende / vnd also
durch Christum die ewig
gen warhaite / weg vnd
leben selig werd.
Amen.
N der vorred
des ersten hunderts
der Euangelischenn
warhaite Gnädigcleich
das Antipapistisch /
eins vnd hundert gr Nihil fa
mandt hab / hab ich ne causa.
mich (Freündlicher lieber Christ) vers
nemmen



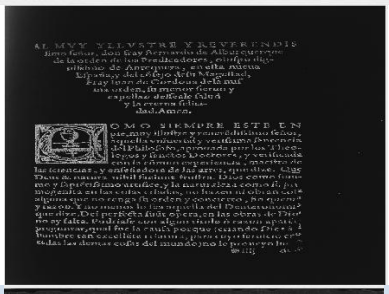


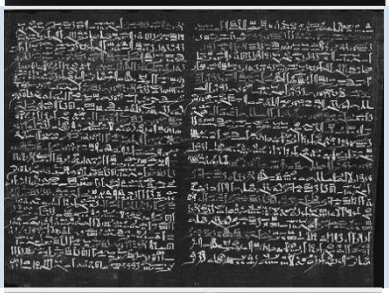


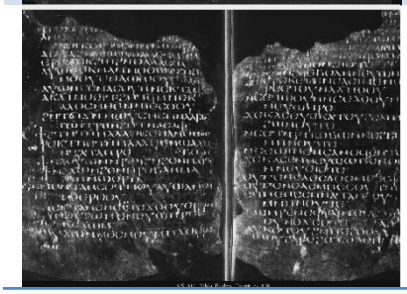



Dem güthertzi
gen leser vnnd liebhaber der
Christlichen Frümbkait/ wünsch
ich F. Ioannes Natus C. A. &c. Göttli
che gnad/ frid vnd erbarmuß/ ber &. Ap
postolischen warhait/ vnd am abschew
en der Apostatischen narrhait/ auff das
er diser endecheer/ vnd jener flechta an
hang/ biß inns ende/ vmb alß
durch Christum/ die Ewig
gen warhaite/ weeg vnd
leben sälig werd/
Amen.
Pared4


Dem güthertzi
gen leser vnnd liebhaber der
Christlichen Frümbkait/ wünsch
ich F. Ioannes Natus C. A. &c. Göttli
che gnad/ frid vnd erbarmuß/ ber &. Ap
postolischen warhait/ vnd am abschew
en der Apostatischen narrhait/ auff das
er diser endecheer/ vnd jener flechta an
hang/ biß inns ende/ vmb alß
durch Christum/ die Ewig
gen warhaite/ weeg vnd
leben sälig werd/
Amen.
Paso 5 y 6







CONCLUSIONES:

El tratamiento de documentos antiguos es un proceso delicado, el programa no elimina del todo el ruido cuando la imagen esta muy deteriorada o cuando las manchas son muy notorias, ya que el

programa lo toma como texto, sin embargo los resultados que se obtienen, son buenas yo que se logra rescatar bastante información del documento.

Cabe mencionar que el resultado también depende de la calidad de la imagen inicial, ya que si esta es de baja calidad el resultado será de la misma manera

REFERECNCIAS:

http://docs.oracle.com/javase/6/docs/api/java/awt/image/BufferedImage.html

http://docs.oracle.com/javase/7/docs/api/java/awt/Color.html

http://iolmos.cs.buap.mx/pdi/Sesion4_Operaciones_Punto.pdf

http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/package-

summary.html

http://mmartin.cs.buap.mx/pdi/proy/Proy-PID-P1-prim-2012.pdf

http://www.uabcs.mx/geologia/geo_bajamx/pr/Mehl_Fundamentos_PR.pdf

http://iaci.unq.edu.ar/materias/vision/archivos/apuntes/Segmentaci%C3%B3n%20por%20umbrali zaci%C3%B3n%20-%20M%C3%A9todo%20de%20Otsu.pdf

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

PROCESAMIENTO DIGITAL DE IMAGENES

PROF. IVÁN OLMOS PINEDA Aplicación

de Filtros Puntuales aplicado a Regiones

DANIEL SORIANO GRANDE

PRIMAVERA 2014

Introducción:

El proceso de filtrado es el conjunto de técnicas englobadas dentro del pre procesamiento de imágenes cuyo objetivo fundamental es obtener, a partir de una imagen origen, otra final cuyo resultado sea más adecuado para una aplicación específica mejorando ciertas características de la misma que posibilite efectuar operaciones del procesado sobre ella.

Conceptos Desarrollados:

Filtro:

Se considera un filtro como una operación que se aplica a los píxeles de una imagen digital para optimizarla, enfatizar cierta información o conseguir un efecto especial en ella.

Operaciones orientadas a punto:

Modifican los valores de los píxeles

No es necesario considerar los valores de los píxeles vecinos

Una operación punto sobre una imagen I se define como:
Una función f: I Æ l', tal que f(x) = y

Algoritmo Básico de la Operación Punto

Sea R ⊆ I, donde R[i1… i2, j1, …, j2]

El algoritmo básico de transformación de R bajo f se define como:

```
for(i=i1; i <= i2; i++)
for(j=j1; j <= j2; j++)
R'[i,j] = f(I[i,j])
```

Los operadores punto con los que se van a trabajar son:

1. Filtro de corrección gamma
2. Filtro de Aclarado Logarítmica
3. Filtro de Aclarado Seno
4. Filtro de Aclarado Exponencial
5. Filtro de obscurecimiento Cosenoidal
6. Filtro de obscurecimiento Exponencial

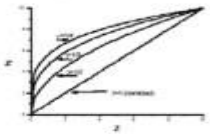
Las funciones de cada filtro son:

Filtro de corrección gamma

$$r' = q\left(\frac{r}{q}\right)^{\gamma}$$

γ es un real positivo, r $\in$ [0, q], r' $\in$ [0, q]

- Si $\gamma \in (0,1]$ → la imagen será aclarada



Filtro de Aclarado Logarítmica

$$x' = A \ln(\alpha x + 1),\ \alpha > 1,\ x \in [0, q]$$

(q normalmente toma el valor de 255)

Filtro de Aclarado Seno

$$X' = q \sin(\pi x / 2q)$$

Filtro de Aclarado Exponencial

$$X' = A(1-e^{-\alpha x/q}),\ \text{donde } \alpha \in [0, q]$$

A se define como

$$A = q / (1-e^{-\alpha})$$

Filtro de obscurecimiento Cosenoidal

$$x' = q\left(1 - \cos\left(\frac{\pi x}{2q}\right)\right)$$

Filtro de obscurecimiento Exponencial

$$x' = A\left(e^{\alpha x/q} - 1\right), \quad \alpha > 0$$

donde:

$$A = q/(e^{\alpha} - 1)$$

Los filtros anteriormente definidos operan en un rango de [0,255] y son aplicados a todos los pixeles de la imagen.

Pero también, los filtros se pueden diseñar para operar por regiones dentro de la imagen.

El objetivo es poder realizar esto, es decir, dependiendo de la posición de cada segmento, se lograrán efectos de aclarado / obscurecimiento.

Por tal razón tenemos que modificar las funciones de los filtros para que operen en un rango [x0, x1], así también el valor de cuantificación q será modificado.

Las funciones de los filtros quedan como:

1. Filtro de corrección gamma

( ) ( )

Donde [ ]

2. Filtro de Aclarado Logarítmica

(( ))

Donde [ ]

(( ))

3. Filtro de Aclarado Seno

( ) (————

( ) ( ))

Donde [ ]

4. Filtro de Aclarado Exponencial

( )

( ) ( )

Donde [ ]

5. Filtro de obscurecimiento Cosenoidal

Donde [ ]

( ) ((

( ) ( )))

6. Filtro de obscurecimiento Exponencial

( )
( )

Donde [ ] (——)

Experimentos:

Haremos un programa que permita realizar filtros a una imagen aplicado a regiones.

Se desarrolló un programa en netbeans y lo que hace es:

1. Cargar una imagen y mostrarla en pantalla.

```
//Creamos un nuevo cuadro de diálogo para seleccionar imagen
    JFileChooser selector=new JFileChooser();
    //Le damos un título
    selector.setDialogTitle("Seleccione una imagen");
    //Filtramos los tipos de archivos
```

```
FileNameExtensionFilter filtroImagen = new FileNameExtensionFilter("JPG & BMP", "jpg", "bmp");
selector.setFileFilter(filtroImagen);
//Abrimos el cuadro de diálog
int flag=selector.showOpenDialog(null);
//Comprobamos que pulse en aceptar
if(flag==JFileChooser.APPROVE_OPTION){
    try {
        //Devuelve el fichero seleccionado
```

```
            File imagenSeleccionada=selector.getSelectedFile();
            //Asignamos a la variable bmp la imagen leida
           imagen= ImageIO.read(imagenSeleccionada);
        } catch (IOException e) {
        }
            jLabel1_Imagen.setIcon(new ImageIcon(imagen));
    }
}
}
```

2.Creamos una variable para guardar la copia de la imagen

```
BufferedImage Nueva_Imagen=new BufferedImage Imagen_Resultado=new
BufferedImage(imagen.getWidth(),imagen.getHeight(),imagen.getType());
```

3.Para aplicar cualquier filtro en un rango [x0,x1] se hace lo siguiente

-Creamos variables para recuperar los colores de los canales

```
int r,g,b;
Color colorAuxiliar;
```

-Recorremos la imagen

```
for(int i=0;i<imagen.getWidth();i++){
    for(int j=0;j<imagen.getHeight();j++){
        colorAuxiliar=new Color(imagen.getRGB(i, j));


            r=colorAuxiliar.getRed();
           g=colorAuxiliar.getGreen();
```

b=colorAuxiliar.getBlue();
Asignamos los colores a la imagen

-Verificamos si el valor del canal se encuentra en el rango[x0,x1],Si es que si se aplica el filtro

```
if(canal<=x1){
    if(canal>=x0){
    //Aplicar filtro
    }
```

Asignamos los nuevos valores a la imagen.

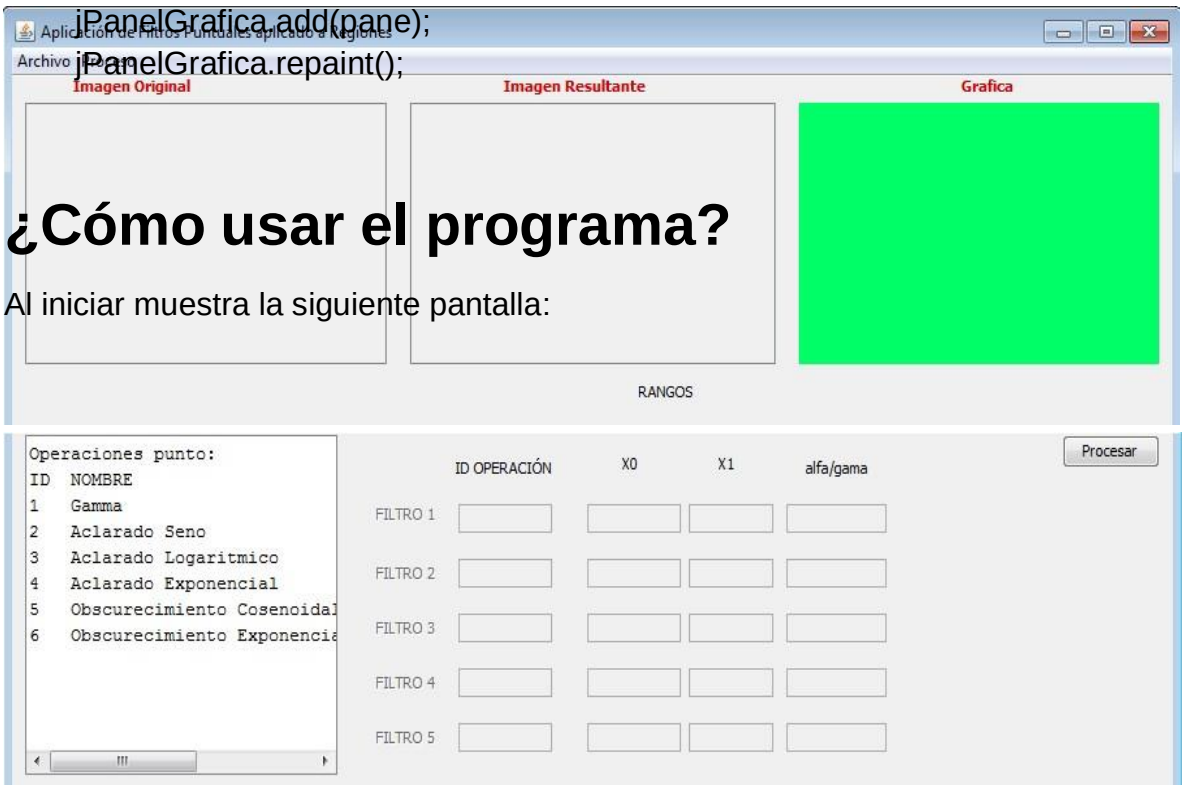
Imagen_Resultado.setRGB(i, j,new Color(r,g,b).getRGB

4.Mostramos en pantalla la imagen después de aplicarle los filtros

```
jLabelIdentidad.setIcon(new ImageIcon(Nueva_Imagen));
```

5.Mandamos a graficar la función composición

```
//dibujamos grafica
    Plot plot= new Plot();
        ChartPanel pane=new ChartPanel(plot.plotear(a));
        pane.setSize(jPanelGrafica.getSize());
        pane.setVisible(true);

        jPanelGrafica.removeAll();
        jPanelGrafica.add(pane);
        jPanelGrafica.repaint();
```



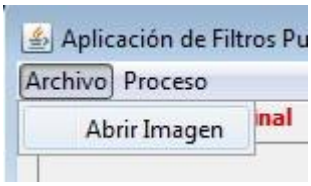
¿Cómo usar el programa?

Al iniciar muestra la siguiente pantalla:

Paso 1:

Abrir una imagen.

Dar clic en Archivo → Abrir imagen



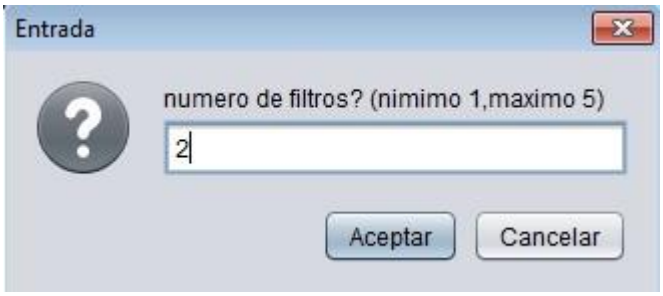
Seleccionar la imagen con la que se desee trabajar, esta se mostrara en el lado izquierdo

Paso 2:

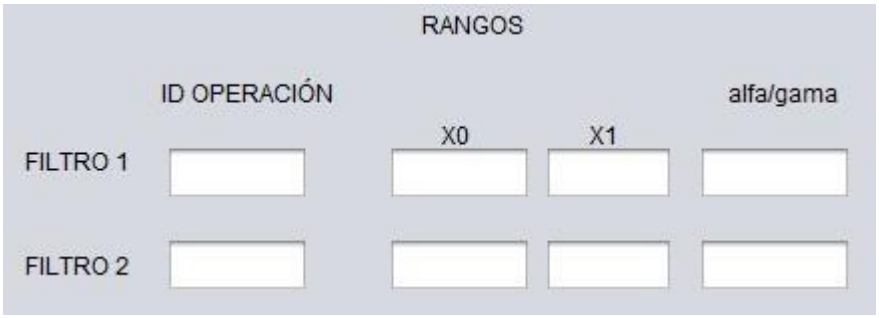
Aplicar el filtrado por regiones.

Dar clic en Proceso →Filtrado por Regiones

Aparecerá una ventana en la cual pondremos el número de filtros que deseamos aplicar el mínimo es de 1 y máximo de 5.Damos clic en aceptar

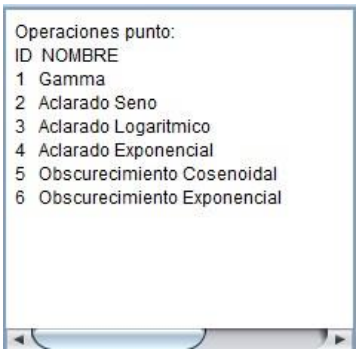

Entrada
numero de filtros? (nimimo 1,maximo 5)
2
Aceptar
Cancelar

Dependiendo del número de filtros se activaran las casillas



Paso 3.

Poner en ID OPERACIÓN el número de filtro con respecto a la ventana



En los rangos x0, x1 poner los valores, teniendo en cuenta que entre todos los filtros se abarca el rango [0,255].Además que dependiendo del filtro será necesario ingresar un valor alfa/gamma.

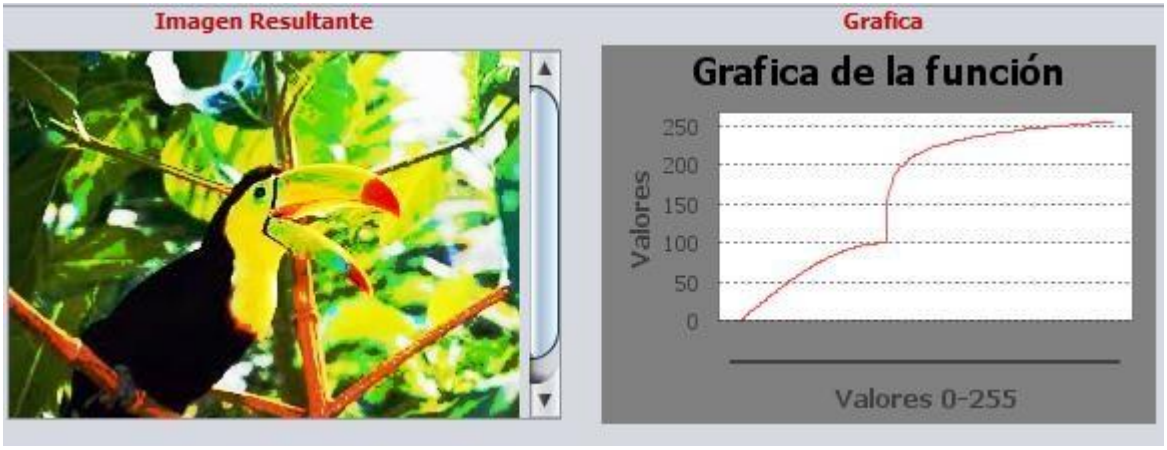
Paso 4.

Ver resultado y grafica

Dar clic en el botón

Dependiendo del número de filtros se activaran las casillas

Aparecerá la imagen resultante, así como también la grafica



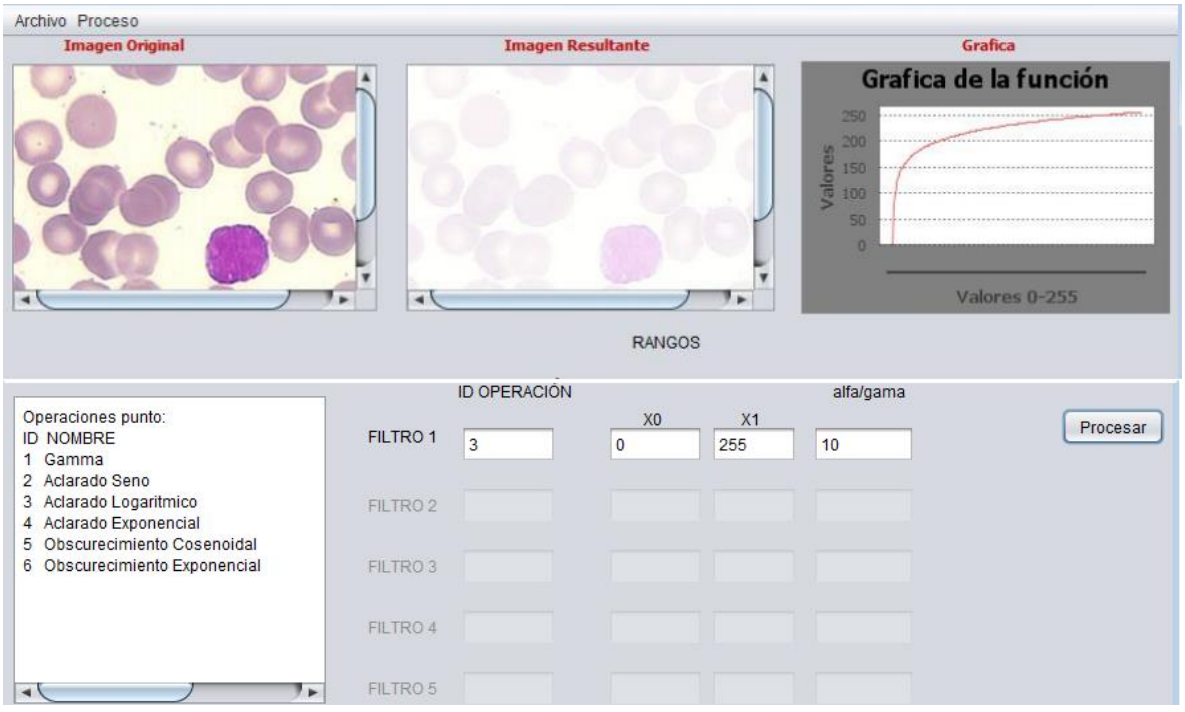
Paso 5.

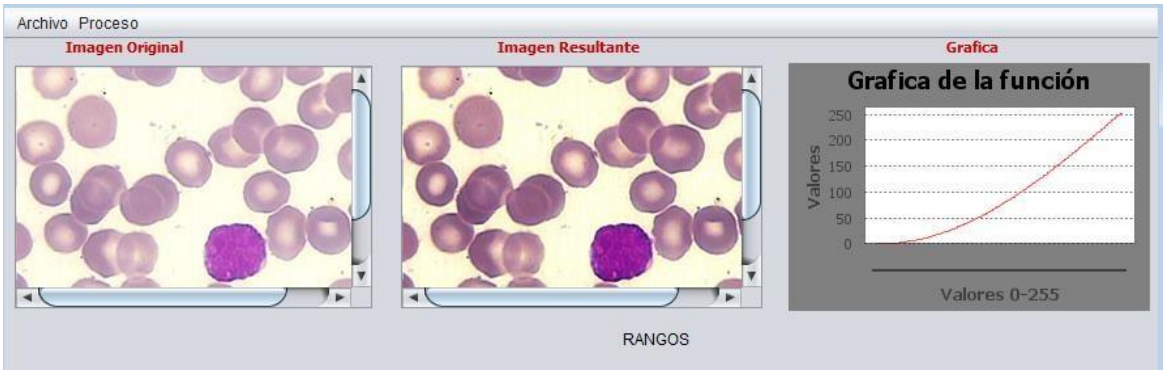
-Si desea salir de la aplicación dar clic en el botón  de la ventana

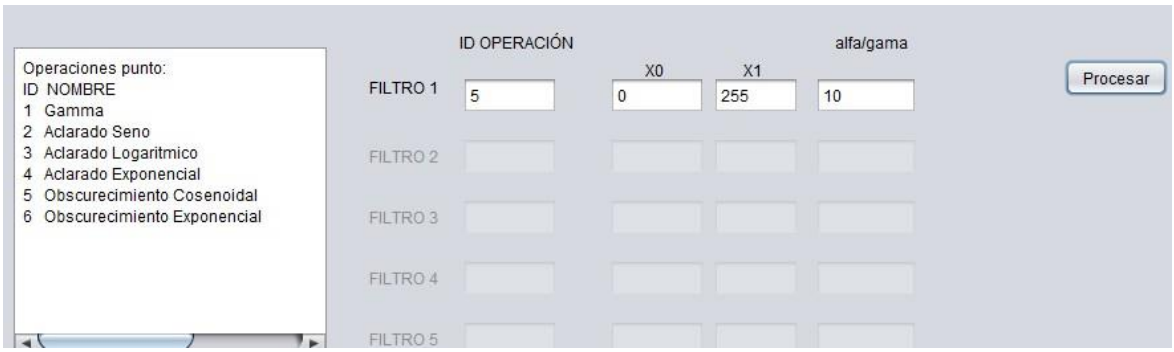
-Si desea cambiar los valores de los filtros (ID, Rangos) solo basta en ingresar los nuevos valores y dar clic en 

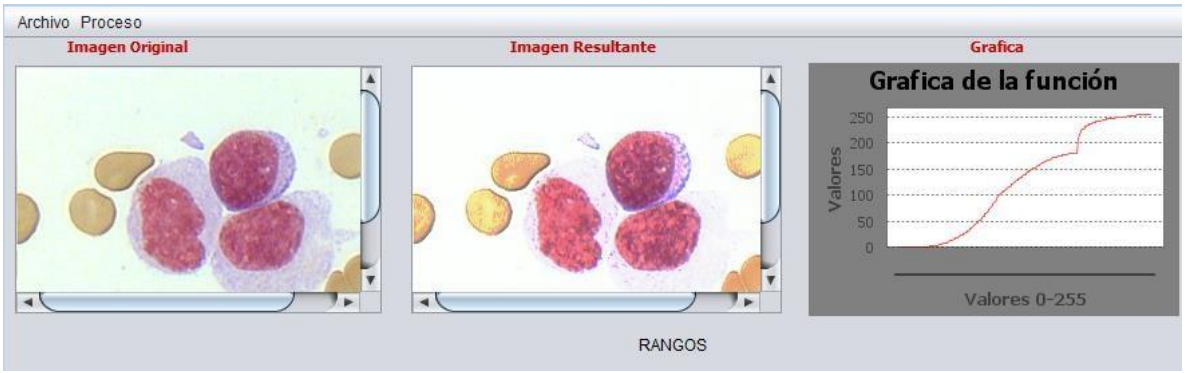
-Si desea utilizar otra imagen, repita el proceso desde el Paso 1

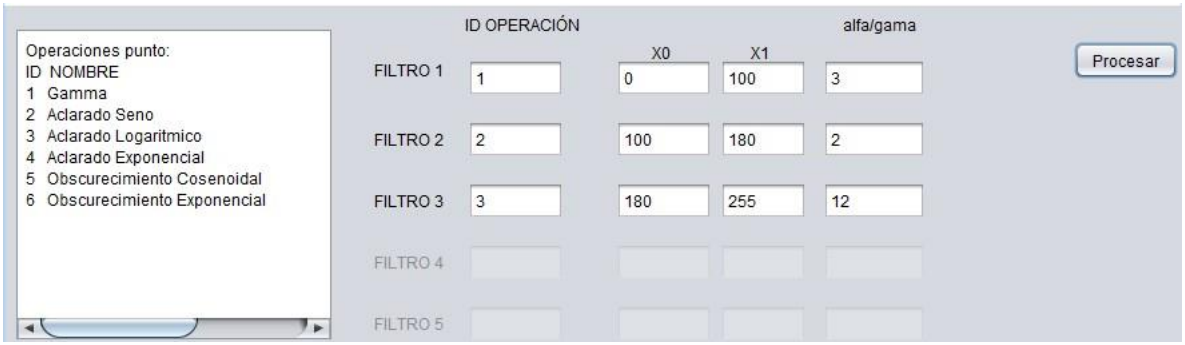
Ejemplos:




Archivo Proceso
Imagen Original
Imagen Resultante
Grafica
Grafica de la función
250
200
150
100
50
0
Valores
Valores 0-255
RANGOS


Operaciones punto:
ID NOMBRE
1 Gamma
2 Aclarado Seno
3 Aclarado Logaritmico
4 Aclarado Exponencial
5 Obscurecimiento Cosenoidal
6 Obscurecimiento Exponencial
ID OPERACIÓN
X0
X1
alfa/gama
FILTRO 1
5
0
255
10
FILTRO 2
FILTRO 3
FILTRO 4
FILTRO 5
Procesar


Archivo Proceso
Imagen Original
Imagen Resultante
Grafica
Grafica de la función
250
200
150
100
50
0
Valores
Valores 0-255
RANGOS


Operaciones punto:
ID NOMBRE
1 Gamma
2 Aclarado Seno
3 Aclarado Logaritmico
4 Aclarado Exponencial
5 Obscurecimiento Cosenoidal
6 Obscurecimiento Exponencial
ID OPERACIÓN
X0
X1
alfa/gama
FILTRO 1
1
0
100
3
FILTRO 2
2
100
180
2
FILTRO 3
3
180
255
12
FILTRO 4
FILTRO 5
Procesar

Bibliografía:

http://www.escet.urjc.es/~visiona/tema2.pdf
*www.thewebfoto.com/3-accesorios/303-los-**filtros***
http://mmartin.cs.buap.mx/notas/PDI-MM-Rev.2013.pdf

FACULTAD de CIENCIAS de la COMPUTACIÓN

REPORTE TECNICO:

Combinación de Operadores Punto por Regiones (canales RGB).

Autor: Víctor Hugo Cañete Olivares

Fecha: 20 de Febrero del 2014

Profesor: Iván Olmos Pineda

1.1. Justificación

El Proyecto se llevó a cabo por razones escolares para el 1er Proyecto, en la materia de Procesamiento digital de Imágenes, donde se propuso realizar dicho proyecto para facilitar analizar cualquier imagen digital al aplicar varios

filtros como es Gamma, Seno, Logaritmo, Exponencial Aclarado, Cosenoidal, Exponencial Obscurecimiento que nos permite Aplicar por Regiones en cada uno de los canales RGB en la imagen.

1.2. Antecedente

La tecnología a avanzando a gran escala y un ejemplo de ello es este proyecto, antes los diagnósticos no eran tan precisos dado que las imágenes no eran digitales y no se hacia un tratamiento tan manejable y agradable es decir era uy difícil sin embargo ahora que existe la digitalización se pueden aplicar filtros por medio de operadores punto a punto que aplicamos en este proyecto.

1.3. Desarrollo

En este proyecto se aplicaran los siguientes filtros como Gamma, Seno, Logaritmo, Exponencial Aclarado, Cosenoidal, Exponencial Obscurecimiento y para aplicarlos tenemos que generalizarlos para poderlos utilizar en cualquier rango de x0 a x1, claro con la restricción principal de que la gama de colores es de 0 a 255 en cada canal RGB y las presentaremos a continuación.

X0=(0,255)

X1=(0,255) donde x0<x1

Gamma
 	
$(x1 - x0)\left(\frac{(X-x0)}{(x1-x0)}\right)^{\propto} + x0$
	r= colorAux.getRed();
	g= colorAux.getGreen();
	b= colorAux.getBlue();
	if(r>=x0 && r<=x1){
	r=(x1-x0)*(Math.pow(((double)(colorAux.getRed()-x0)/(double)(x1-x0)),gamma))+x0;
	}
	if(g>=x0 && g<=x1){
	g=(x1-x0)*(Math.pow(((double)(colorAux.getGreen()-x0)/(double)(x1-x0)),gamma))+x0;
	}
	if(b>=x0 && b<=x1){
	b=(x1-x0)*(Math.pow(((double)(colorAux.getBlue()-x0)/(double)(x1-x0)),gamma))+x0;
	}

Seno	r= colorAux.getRed();
	g= colorAux.getGreen();
	b= colorAux.getBlue();

$(x1 - x0) * \left(\sin\left(\frac{\pi(X - x0)}{2 * (x1 - x0)}\right)\right) + x0$	if(r>=x0 && r<=x1){
	r=((x1-x0)*(Math.sin(Math.PI*(r-x0)/(2*(x1-x0))))+x0);
	}
	if(g>=x0 && g<=x1){
	g=((x1-x0)*(Math.sin(Math.PI*(g-x0)/(2*(x1-x0))))+x0);
	}
	if(b>=x0 && b<=x1){
	b=((x1-x0)*(Math.sin(Math.PI*(b-x0)/(2*(x1-x0))))+x0);
	}

Logaritmo
$\left(\frac{x1 - x0}{\log(x1 - x0 + 1)}\right)(\log((X - x0) + 1)) + x0$
	r= colorAux.getRed();
	g= colorAux.getGreen();
	b= colorAux.getBlue();
	if(r>=x0 && r<=x1){
	r=((x1-x0)/Math.log(x1-x0+1))*(Math.log((r-x0)+1))+x0;
	}
	if(g>=x0 && g<=x1){
	g=((x1-x0)/Math.log(x1-x0+1))*(Math.log((g-x0)+1))+x0;
	}
	if(b>=x0 && b<=x1){
	b=$((x1 - x0)/Math.log(x1 - x0 + 1)) * (Math.log((b - x0) + 1)) + x0$;
	}

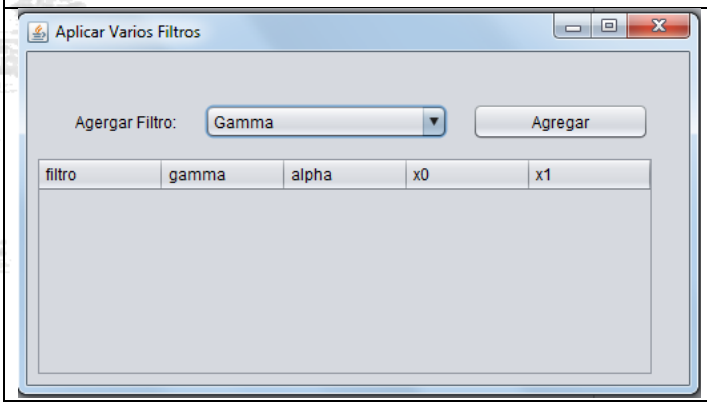
Exponencial A
$\left(\frac{x1 - x0}{1 - e^{\alpha}}\right)\left(\frac{1 - e^{-alpha*(X-x0)}}{x1 - x0}\right) + x0$
	r= colorAux.getRed();
	g= colorAux.getGreen();
	b= colorAux.getBlue();
	if(r>=x0 && r<=x1){
	r=(((x1-x0)/(1-Math.exp(-alpha)))*(1-Math.exp((-alpha*(r-x0))/(x1-x0))))+x0;
	}
	if(g>=x0 && g<=x1){
	g=(((x1-x0)/(1-Math.exp(-alpha)))*(1-Math.exp((-alpha*(g-x0))/(x1-x0))))+x0;
	}
	if(b>=x0 && b<=x1){
	b=(((x1-x0)/(1-Math.exp(-alpha)))*(1-Math.exp((-alpha*(b-x0))/(x1-x0))))+x0;
	}

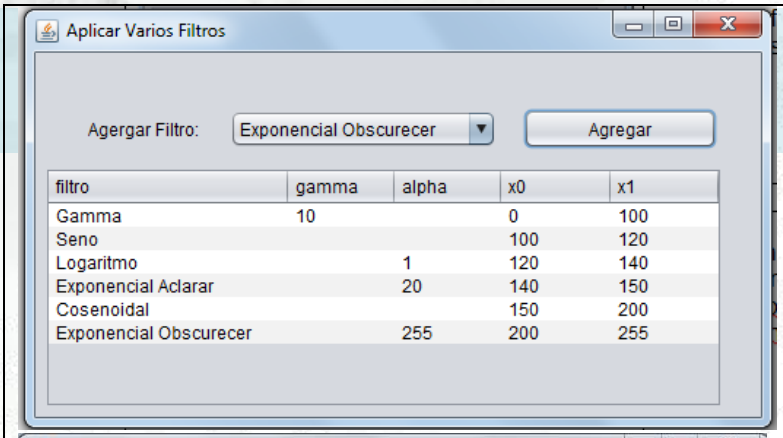
Cosenoidal	r= colorAux.getRed();
	g= colorAux.getGreen();
	b= colorAux.getBlue();
	if(r>=x0 && r<=x1){

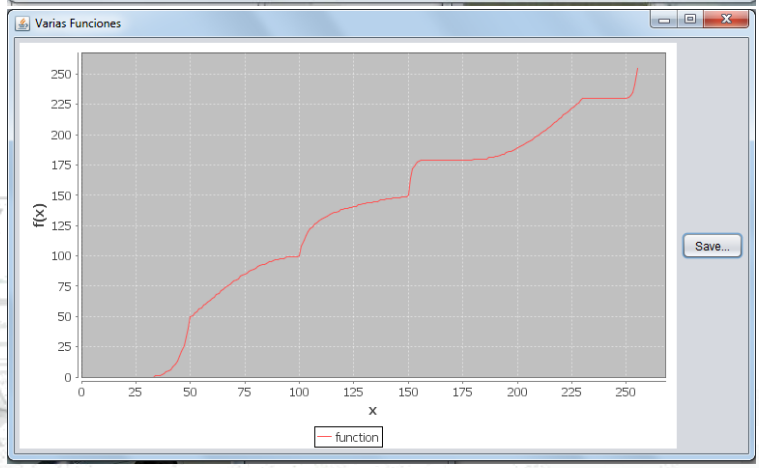
$(x1 - x0)\left(1 - \cos\left(\frac{\pi(X - x0)}{2 * (x1 - x0)}\right)\right) + x0$	r=(x1-x0)*(1-Math.cos((Math.PI*(r-x0))/(2*(x1-x0))))+x0;
	}
	if(g>=x0 && g<=x1){
	g=(x1-x0)*(1-Math.cos((Math.PI*(g-x0))/(2*(x1-x0))))+x0;
	}
	if(b>=x0 && b<=x1){
	b=(x1-x0)*(1-Math.cos((Math.PI*(b-x0))/(2*(x1-x0))))+x0;
	}

Exponencial O
 	
((x1-x0)/(Math.exp(alpha)-1))*(Math.exp((alpha*(r-x0))/(x1-x0))-1)+x0
	r= colorAux.getRed();
	g= colorAux.getGreen();
	b= colorAux.getBlue();
	if(r>=x0 && r<=x1){
	r=((x1-x0)/(Math.exp(alpha)-1))*(Math.exp((alpha*(r-x0))/(x1-x0))-1)+x0;
	}
	if(g>=x0 && g<=x1){
	g=((x1-x0)/(Math.exp(alpha)-1))*(Math.exp((alpha*(g-x0))/(x1-x0))-1)+x0;
	}
	if(b>=x0 && b<=x1){
	b=((x1-x0)/(Math.exp(alpha)-1))*(Math.exp((alpha*(b-x0))/(x1-x0))-1)+x0;
	}

Utilizamos la IDE *NetBeans IDE Version 7.1.2* , En donde tenemos un ventana (jFrame) de la siguiente manera.

	Aquí tenemos el frame con tres menús (Archivo, operadores punto y ver) y un Etiqueta, en donde se mostrara el nombre de la imagen y otra para la imagen, El menú de operador punto, se encuentra la opción de Aplicar Varios que se encuentra en la pestaña de operadores punto. Al dar clic te presentara la siguiente ventana.
	En esta ventana puedes seleccionar como máximo 6 filtros distintos en diferentes rangos y que los puedes combinar hasta Aplicar 10 filtros ya sea distintos o Duplicables. Como por Ejemplo:

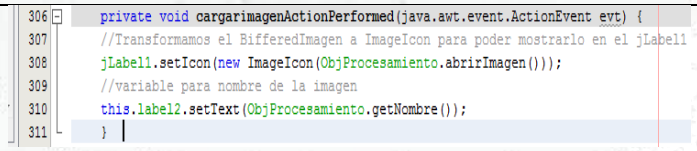




Esta implementación del Proyecto nos arrojara inmediatamente estas 2 ventanas: una con la imagen modificada y otra con la gráfica de las funciones implementadas de cada filtro en la imagen, cuando finalicemos en el rango con 255.

Al ir agregando cada filtro en un determinado rango le envía a la clase Procesamiento que se encarga de aplicar la función en cada canal RGB en el rango de X0 aX1 la procesa y puedes agregar máximo 5 filtros, al ir aplicando filtros los guarda los datos como el valor del pixel, punto inicial, punto final y ya sea gamma o alfa ya sea el caso del filtro y estos valores los guarda en un arreglo que le llame art y la envió a la clase se genera la gráfica en la que voy corriendo el arreglo y valorando su función correspondiente y guardo x y f(x) en un arreglo para al último solo recorrerlo y graficar los puntos que son de 0 a 255.

Para realizar la gráfica se utiliza las bibliotecas jfree-chart, en las que

	se evalúa cada filtro en su determinado rango como se ve en esta imagen.
	En la ceja “archivo” esta “Cargar Imagen” en donde se lleva a cabo la selección de la imagen de tipo “jpg” y “bmp” y utilizando la librería de java “image” posicionamos la imagen en la etiqueta de nuestro Frame.

Conclusion

Este proyecto aplica muy bien con los operadores punto a punto aplicando una combinación de 6 filtros (Gamma, Seno, Logaritmo, Exponencial Aclarado, Cosenoidal, Exponencial Obscurecimiento) en distintos rango desde 0 a 255 al aplicarlos a cada canal, nos da la posibilidad de aclarar u obscurecer a voluntad propia en el rango que quieras con alfa y gamma manejables para manipulación y así poder manejar a tu consideración para examinar y mejorar tu imagen digital.

2. Bibliografía.

1. Api de Imagen de java. En línea: http://docs.oracle.com/javase/7/docs/api/java/awt/Image.html

2. Uso de las cajas de texto

http://www.javerosanonimos.com/2009/02/uso-de-las-cajas-de-texto.html

3. Blog spot de Procesamiento digital de imágenes.

http://algoimagen.blogspot.mx/2013/08/java-abrir-imagen-leer-pixeles-y-pasar.html

4. Tutorial de java: Métodos para imágenes.

http://www.fismat.umich.mx/computacion/notas/parte15/cap15-8.html

FACULTAD de CIENCIAS de la COMPUTACIÓN

REPORTE TECNICO:

Combinación de Operadores Punto por Regiones (canales RGB).

Autor: Víctor Hugo Cañete Olivares

Fecha: 20 de Febrero del 2014

Profesor: Iván Olmos Pineda

2.1. Justificación

El Proyecto se llevó a cabo por razones escolares para el 1er Proyecto, en la materia de Procesamiento digital de Imágenes, donde se propuso realizar dicho proyecto para facilitar analizar cualquier imagen digital al aplicar varios

filtros como es Gamma, Seno, Logaritmo, Exponencial Aclarado, Cosenoidal, Exponencial Obscurecimiento que nos permite Aplicar por Regiones en cada uno de los canales RGB en la imagen.

2.2. Antecedente

La tecnología a avanzando a gran escala y un ejemplo de ello es este proyecto, antes los diagnósticos no eran tan precisos dado que las imágenes no eran digitales y no se hacia un tratamiento tan manejable y agradable es decir era uy difícil sin embargo ahora que existe la digitalización se pueden aplicar filtros por medio de operadores punto a punto que aplicamos en este proyecto.

2.3. Desarrollo

En este proyecto se aplicaran los siguientes filtros como Gamma, Seno, Logaritmo, Exponencial Aclarado, Cosenoidal, Exponencial Obscurecimiento y para aplicarlos tenemos que generalizarlos para poderlos utilizar en cualquier rango de x0 a x1, claro con la restricción principal de que la gama de colores es de 0 a 255 en cada canal RGB y las presentaremos a continuación.

X0=(0,255)

X1=(0,255) donde x0<x1

Gamma
 	
$(x1 - x0)\left(\frac{(X-x0)}{(x1-x0)}\right)^{\propto} + x0$
	r= colorAux.getRed();
	g= colorAux.getGreen();
	b= colorAux.getBlue();
	if(r>=x0 && r<=x1){
	r=(x1-x0)*(Math.pow(((double)(colorAux.getRed()-x0)/(double)(x1-x0)),gamma))+x0;
	}
	if(g>=x0 && g<=x1){
	g=(x1-x0)*(Math.pow(((double)(colorAux.getGreen()-x0)/(double)(x1-x0)),gamma))+x0;
	}
	if(b>=x0 && b<=x1){
	b=(x1-x0)*(Math.pow(((double)(colorAux.getBlue()-x0)/(double)(x1-x0)),gamma))+x0;
	}

Seno	r= colorAux.getRed();
	g= colorAux.getGreen();
	b= colorAux.getBlue();

$(x1 - x0) * \left(\sin\left(\frac{\pi(X - x0)}{2 * (x1 - x0)}\right)\right) + x0$	if(r>=x0 && r<=x1){
	r=((x1-x0)*(Math.sin(Math.PI*(r-x0)/(2*(x1-x0))))+x0);
	}
	if(g>=x0 && g<=x1){
	g=((x1-x0)*(Math.sin(Math.PI*(g-x0)/(2*(x1-x0))))+x0);
	}
	if(b>=x0 && b<=x1){
	b=((x1-x0)*(Math.sin(Math.PI*(b-x0)/(2*(x1-x0))))+x0);
	}

Logaritmo
$\left(\frac{x1 - x0}{\log(x1 - x0 + 1)}\right)(\log((X - x0) + 1)) + x0$
	r= colorAux.getRed();
	g= colorAux.getGreen();
	b= colorAux.getBlue();
	if(r>=x0 && r<=x1){
	r=((x1-x0)/Math.log(x1-x0+1))*(Math.log((r-x0)+1))+x0;
	}
	if(g>=x0 && g<=x1){
	g=((x1-x0)/Math.log(x1-x0+1))*(Math.log((g-x0)+1))+x0;
	}
	if(b>=x0 && b<=x1){
	b=$((x1 - x0)/Math.log(x1 - x0 + 1)) * (Math.log((b - x0) + 1)) + x0$;
	}

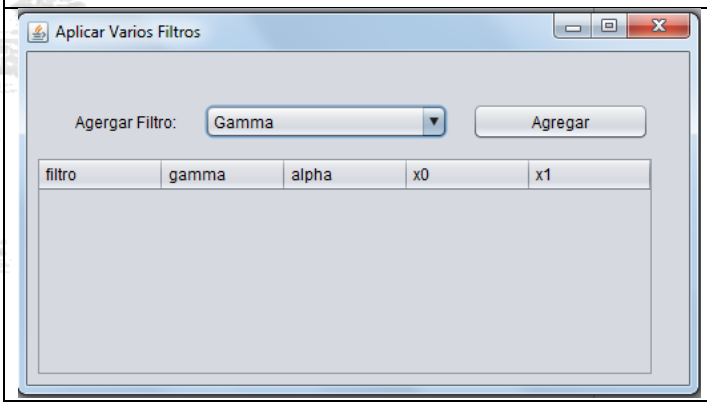
Exponencial A
$\left(\frac{x1 - x0}{1 - e^{\alpha}}\right)\left(\frac{1 - e^{-alpha*(X-x0)}}{x1 - x0}\right) + x0$
	r= colorAux.getRed();
	g= colorAux.getGreen();
	b= colorAux.getBlue();
	if(r>=x0 && r<=x1){
	r=(((x1-x0)/(1-Math.exp(-alpha)))*(1-Math.exp((-alpha*(r-x0))/(x1-x0))))+x0;
	}
	if(g>=x0 && g<=x1){
	g=(((x1-x0)/(1-Math.exp(-alpha)))*(1-Math.exp((-alpha*(g-x0))/(x1-x0))))+x0;
	}
	if(b>=x0 && b<=x1){
	b=(((x1-x0)/(1-Math.exp(-alpha)))*(1-Math.exp((-alpha*(b-x0))/(x1-x0))))+x0;
	}

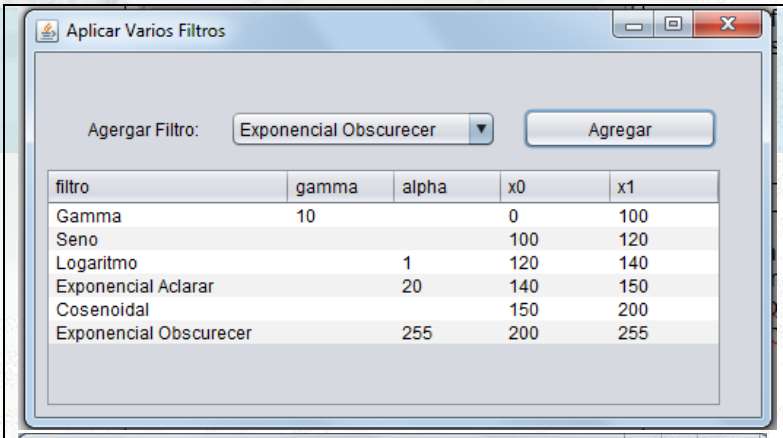
Cosenoidal	r= colorAux.getRed();
	g= colorAux.getGreen();
	b= colorAux.getBlue();
	if(r>=x0 && r<=x1){

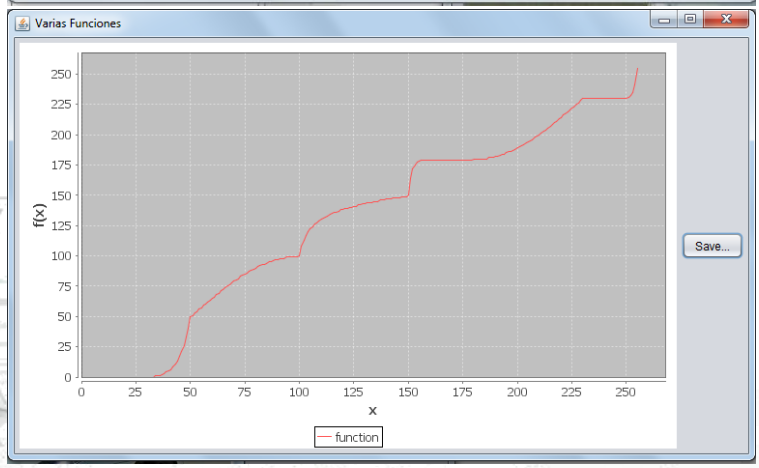
$(x1 - x0)\left(1 - \cos\left(\frac{\pi(X - x0)}{2 * (x1 - x0)}\right)\right) + x0$	r=(x1-x0)*(1-Math.cos((Math.PI*(r-x0))/(2*(x1-x0))))+x0;
	}
	if(g>=x0 && g<=x1){
	g=(x1-x0)*(1-Math.cos((Math.PI*(g-x0))/(2*(x1-x0))))+x0;
	}
	if(b>=x0 && b<=x1){
	b=(x1-x0)*(1-Math.cos((Math.PI*(b-x0))/(2*(x1-x0))))+x0;
	}

Exponencial O
 	
((x1-x0)/(Math.exp(alpha)-1))*(Math.exp((alpha*(r-x0))/(x1-x0))-1)+x0
	r= colorAux.getRed();
	g= colorAux.getGreen();
	b= colorAux.getBlue();
	if(r>=x0 && r<=x1){
	r=((x1-x0)/(Math.exp(alpha)-1))*(Math.exp((alpha*(r-x0))/(x1-x0))-1)+x0;
	}
	if(g>=x0 && g<=x1){
	g=((x1-x0)/(Math.exp(alpha)-1))*(Math.exp((alpha*(g-x0))/(x1-x0))-1)+x0;
	}
	if(b>=x0 && b<=x1){
	b=((x1-x0)/(Math.exp(alpha)-1))*(Math.exp((alpha*(b-x0))/(x1-x0))-1)+x0;
	}

Utilizamos la IDE *NetBeans IDE Version 7.1.2* , En donde tenemos un ventana (jFrame) de la siguiente manera.

	Aquí tenemos el frame con tres menús (Archivo, operadores punto y ver) y un Etiqueta, en donde se mostrara el nombre de la imagen y otra para la imagen, El menú de operador punto, se encuentra la opción de Aplicar Varios que se encuentra en la pestaña de operadores punto. Al dar clic te presentara la siguiente ventana.
	En esta ventana puedes seleccionar como máximo 6 filtros distintos en diferentes rangos y que los puedes combinar hasta Aplicar 10 filtros ya sea distintos o Duplicables. Como por Ejemplo:

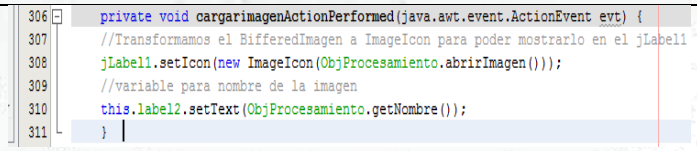




Esta implementación del Proyecto nos arrojara inmediatamente estas 2 ventanas: una con la imagen modificada y otra con la gráfica de las funciones implementadas de cada filtro en la imagen, cuando finalicemos en el rango con 255.

Al ir agregando cada filtro en un determinado rango le envía a la clase Procesamiento que se encarga de aplicar la función en cada canal RGB en el rango de X0 aX1 la procesa y puedes agregar máximo 5 filtros, al ir aplicando filtros los guarda los datos como el valor del pixel, punto inicial, punto final y ya sea gamma o alfa ya sea el caso del filtro y estos valores los guarda en un arreglo que le llame art y la envió a la clase se genera la gráfica en la que voy corriendo el arreglo y valorando su función correspondiente y guardo x y f(x) en un arreglo para al último solo recorrerlo y graficar los puntos que son de 0 a 255.

Para realizar la gráfica se utiliza las bibliotecas jfree-chart, en las que

	se evalúa cada filtro en su determinado rango como se ve en esta imagen.
	En la ceja “archivo” esta “Cargar Imagen” en donde se lleva a cabo la selección de la imagen de tipo “jpg” y “bmp” y utilizando la librería de java “image” posicionamos la imagen en la etiqueta de nuestro Frame.

Conclusion

Este proyecto aplica muy bien con los operadores punto a punto aplicando una combinación de 6 filtros (Gamma, Seno, Logaritmo, Exponencial Aclarado, Cosenoidal, Exponencial Obscurecimiento) en distintos rango desde 0 a 255 al aplicarlos a cada canal, nos da la posibilidad de aclarar u obscurecer a voluntad propia en el rango que quieras con alfa y gamma manejables para manipulación y así poder manejar a tu consideración para examinar y mejorar tu imagen digital.

3. Bibliografía.

5. Api de Imagen de java. En línea: http://docs.oracle.com/javase/7/docs/api/java/awt/Image.html

6. Uso de las cajas de texto

http://www.javerosanonimos.com/2009/02/uso-de-las-cajas-de-texto.html

7. Blog spot de Procesamiento digital de imágenes.

http://algoimagen.blogspot.mx/2013/08/java-abrir-imagen-leer-pixeles-y-pasar.html

8. Tutorial de java: Métodos para imágenes.

http://www.fismat.umich.mx/computacion/notas/parte15/cap15-8.html

REPORTE TECNICO:

PRE PROCESAMIENTO DE BINARIZACION PARA LA RECUPERACIÓN DE DOCUMENTOS ANTIGUOS

Autores: Víctor Hugo Cañete Olivares

Eduardo Cantoran Flores

Fecha: 27 de Marzo de 2014

Profesor: Iván Olmos Pineda

Resumen

El objetivo del documento pre-procesamiento es facilitar el reconocimiento de texto de documentos. El análisis de documentos históricos parece ser un gran reto porque la mayoría de esos documentos son ruidosos y presentes muchas degradaciones. En este trabajo se propone un marco de procesamiento previo para un gran conjunto de datos de documentos históricos. El pre-procesamiento que se realiza es efectivamente a mejorar el aspecto de los documentos con la binarizacion más efectiva de la imagen digital del documento antiguo por medio del método de sauvola optimizado para un marco pre-procesamiento automático de documentos históricos que parece ser un gran reto.

Introducción

Binarización de documentos históricos es considerada como el paso crucial de procesamiento previo. Cuanto mejor sea la binarización más fácil los pasos de segmentación y reconocimiento son. Con el fin de comparar la eficiencia de binarización.El método de binarización tiene que ejecutarse de forma automática y un resultado eficiente.

Algoritmo De Pre- Procesamiento

1.- Convertir la imagen a escala de grises

$$I'(x,y) = 0.2989 \cdot I_R(x,y) + 0.5870 \cdot I_G(x,y) + 0.1140 \cdot I_B(x,y)$$

2.- Aplicar Umbralización (Binarizacion) por el método de Sauvola.

La umbralizacion es una técnica de segmentación simple y eficiente que permite separar los pixel de la imagen en escala de grises en dos categorías a partir de un una ventana o borde umbral. El umbral puede ser función de la posición *p(r,c),* de la vecindad *N(r,c)* y de la intensidad *I(r,c)* actual del pixel.

$$T(x,y) = m(x,y)\left[1 + k\left(\frac{s(x,y)}{R} - 1\right)\right]$$

Donde

T(x, y) = Es valor del umbral calculado para cada pixel en la posición (x, y).

K y R = Son contantes que modifican el valor del umbral obtenido que pueden variar de acuerdo a imagen particular.

m(x, y) = media aritmética

s(x, y) = desviación estándar

m(x, y) y s(x, y) se calculan en una ventana de w x w con centro en (x, y)

3.- Binarizacion final

```
If(I(x,y)>T(x, y)){
I(x,y)=1;
}else
I(x, y)=0;
```

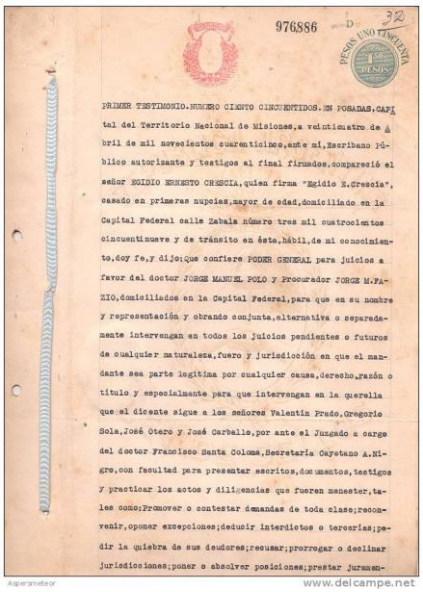
4.- Limpieza de ventana

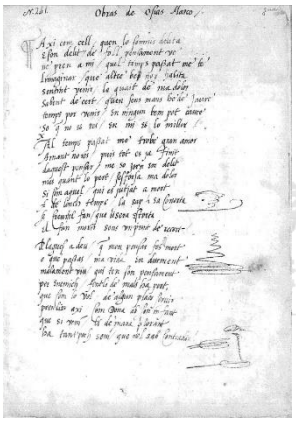
Se limpia los marcos que se crearon por el uso de las ventanas en el método de Sauvola.

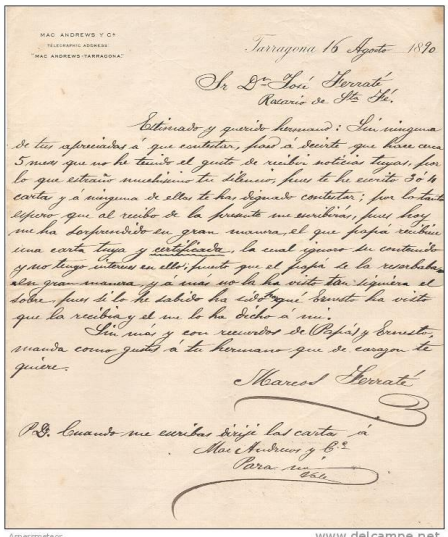
Implementación

Para implementar el método se hizo uso de una pequeña biblioteca de imágenes

Tomadas del buscador Google Imágenes y algunas paginas que se dedican a la venta de documentos antiguos, y algunos de bibliotecas digitales.







MAC ANDREWS Y Cª
TELEGRAPHIC ADDRESS
"MAC ANDREWS-TARRAGONA"

Tarragona 15 Agosto 1890

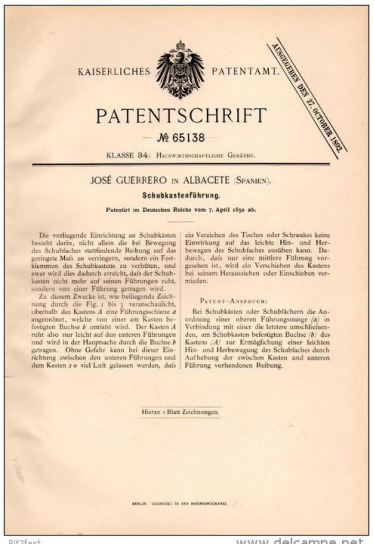
Sr. Dn José Ferrate
Katanis de Pla Sr.

Estimado y querido hermano: Sin ninguna de tus apreciadas á que contestar, pero á decirte que hace cerca 5 meses que no he tenido el gusto de recibir noticias tuyas, por lo que estraño muchísimo te dilemas, pues te he escrito 3 ó 4 cartas y á ninguna de ellas te has dignado contestar; por lo tanto espero que al recibo de la presente me escribas, pues hoy me ha sorprendido en gran manera, el que papá recibiera una carta tuya y entregarse la cual ignoro su contenido y no tengo interes en ello, punto que el papá te la recordaba con gran manera, y á mas no la ha visto tan alegre, el dolor pues de le he sabido ha desaparecido, deseando la vista que la recibas y el me le ha dicho á mi.

Sin mas y con recuerdos de Papá y Ernesto quanda como gusto á tu hermano que se cuyan de quiere.

Marcos Ferrate

P.D. Cuando me escribas dirijé las carta á Mac Andrews y Cª Tarragona



KAISERLICHES PATENTAMT.

AUSGEGEBEN DEN 1. OKTOBER 1892.

PATENTSCHRIFT

№ 65138

KLASSE 84. Hauswirthschaftliche Geräthe.

JOSÉ GUERRERO IN ALBACETE (SPANIEN).

Schubkastenführung.

Patentirt im Deutschen Reiche vom 7. April 1892 ab.

Die vorliegende Einrichtung an Schubkästen besteht darin, nicht allein die bei Bewegung des Schubfaches stattfindende Reibung auf das geringste Maß zu verringern, sondern ein Festklemmen des Schubkastens zu verhüten, und zwar wird dies dadurch erreicht, daß der Schubkasten nicht mehr auf seinen Führungen ruht, sondern von einer Führung getragen wird.

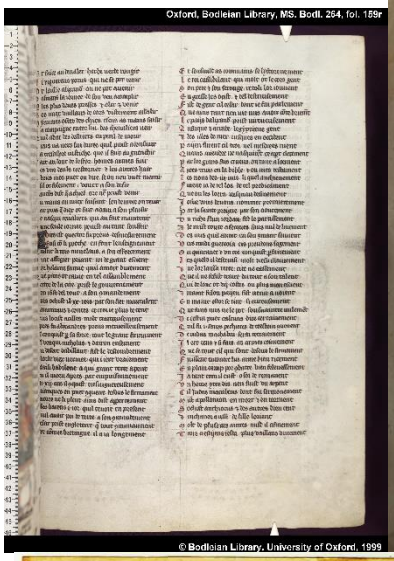
Zu diesem Zwecke ist, wie beiliegende Zeichnung durch die Fig. 1 bis 3 veranschaulicht, oberhalb des Kastens A eine Führungsschiene a angeordnet, welche von einer am Kasten befestigten Buchse b umfaßt wird. Der Kasten A ruht also nur leicht auf den unteren Führungen und wird in der Hauptsache durch die Buchse b getragen. Ohne Gefahr kann bei dieser Einrichtung zwischen den unteren Führungen und dem Kasten c e viel Luft gelassen werden, daß ein Verziehen des Tisches oder Schrankes keine Einwirkung auf das leichte Hin- und Herbewegen des Schubfaches ausüben kann. Dadurch, daß nur eine mittlere Führung vorgesehen ist, wird ein Verschieben des Kastens bei seinem Herausziehen oder Einschieben vermieden.

Patent-Anspruch:

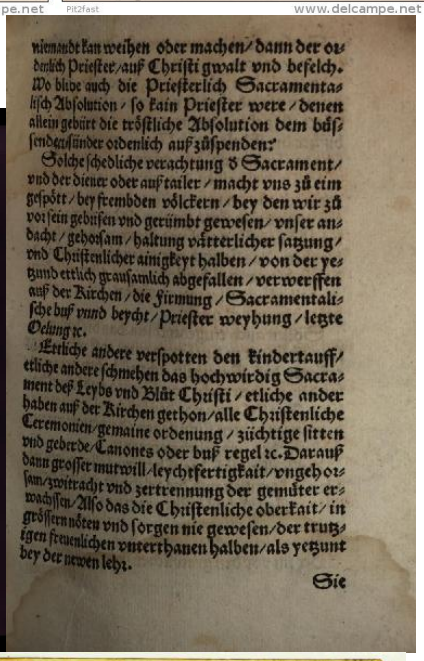
Bei Schubkästen oder Schubfächern die Anordnung einer oberen Führungsstange (a) in Verbindung mit einer die letztere umschließenden, am Schubkasten befestigten Buchse (b) des Kastens (A) zur Ermöglichung einer leichten Hin- und Herbewegung des Schubfaches durch Aufhebung der zwischen Kasten und unterer Führung vorhandenen Reibung.

Hierzu 1 Blatt Zeichnungen.

BERLIN. GEDRUCKT IN DER REICHSDRUCKEREI.



Oxford, Bodleian Library, MS. Bodl. 264, fol. 159r



niemandt kan weihen oder machen/ dann der ordenlich Priester/ auff Christi gwalt vnd befelch. Wo blybe auch die Priesterlich Sacramentalisch Absolution/ so kain Priester were/ denen allein gebürt die tröstliche Absolution dem büssenden sünder ordenlich auffzüspenden:

Solche schedliche verachtung d Sacrament/ vnd der diener oder auffsatzler/ macht vns zů eim gespött/ bey frembden völckern/ bey den wir zů vor ein gebutsen vnd gerümbt gewesen/ vnser andacht/ gehorsam/ haltung vätterlicher satzung/ vnd Christenlicher ainigkeyt halben/ von der yetzund etlich grausamlich abgefallen/ verworffen auff der Kirchen/ die Firmung/ Sacramentalische buß vnnd beycht/ Priester weyhung/ letzte Oelung ꝛc.

Etliche andere verspotten den kindertauff/ etliche andere schmehen das hochwirdig Sacrament deß Leybs vnd Blůt Christi/ etliche anderhaben auff der Kirchen gethon/ alle Christenliche Ceremonien/ gemaine ordenung/ züchtige sitten vnd gebende Canones oder buß regel ꝛc. Darauff dann grosser mutwill/ leychtfertigkait/ vngehorsam/ zwitracht vnd zertrennung der gemüter erwachssen. Also das die Christenliche oberkait/ in grössern nöten vnd sorgen nie gewesen/ der trutzigen freuenlichen vnerthanen halben/ als yetzunt bey der newen lehr.

Sie

REGISTRO DE LA PROPIEDAD
DE
CARTAGENA.

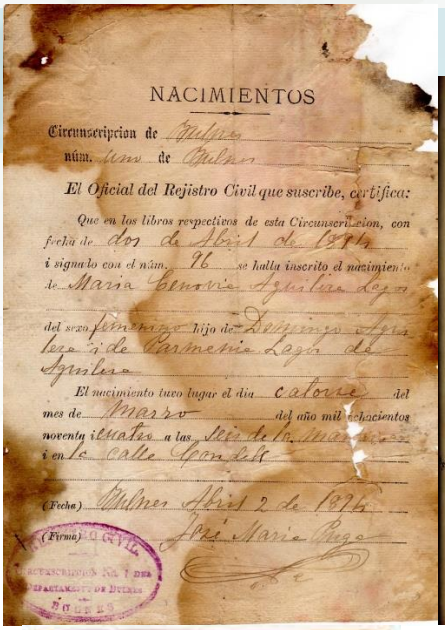
CANCELACION.

D. Juan de Dios de Granada y Alvarez, como Admor Economico del Obispado de Cartagena, &c. presentante de los bienes del Clero y tenedor del mismo á D. Angel Guirado y Batiguez

La obligacion de veinte y un mil cuatrocientos treinta reales corriente y cinco céntimos, inscrita en el Libro 47 fólio 38 vto queda cancelada en el tomo 1º de hipotecas por orden de fechas fólio 208 vto en virtud de la escritura otorgada por el acreedor en 16 de Diciembre de 1866 ante el notario D Juan de la Cuerva cuya copia queda archivada en este Registro, en el legajo del 4º trimestre del año actual.

Cartagena 19 de Diciembre de 1866

Honorarios Rvn.



NACIMIENTOS

Circunscripcion de Gijon

núm. Uno de Gijon

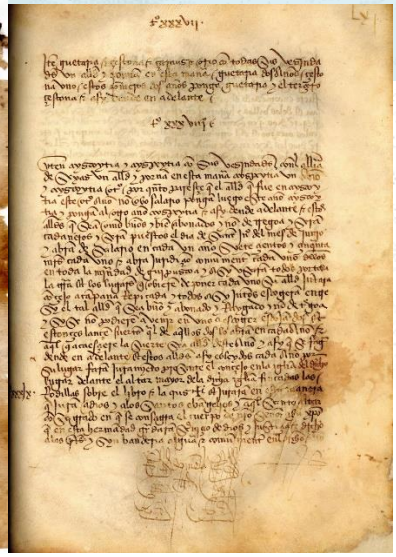
El Oficial del Rejistro Civil que suscribe, certifica:

Que en los libros respectivos de esta Circunscripcion, con fecha de don de Abril de 1876 i signado con el núm. 16 se halla inscrito el nacimiento de María Remedios Agüero Lago del sexo femenino hijo de Domingo Agüero Lago i de Carmen de Lago de Agüero

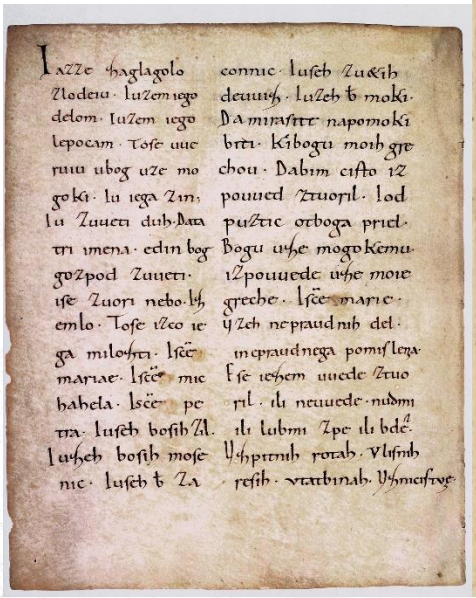
El nacimiento tuvo lugar el dia catorce del mes de Marzo del año mil ochocientos noventy i cuatro a las Seis de la Mañana i en la calle Corrida

(Fecha) Gijon Abril 2 de 1876

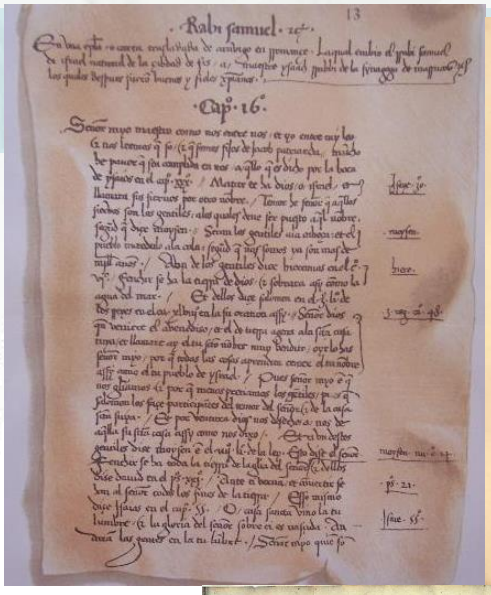
(Firma) José María Rego



Del Baptismo de Christo. 55. 101

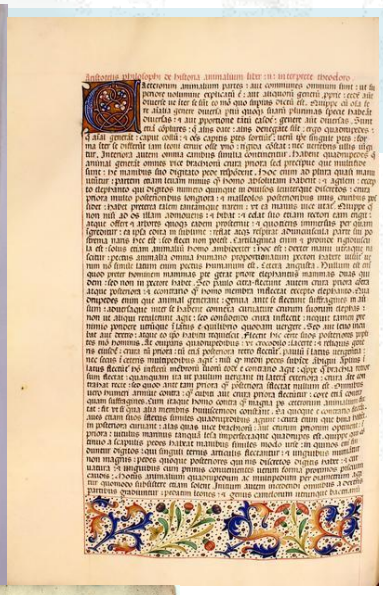


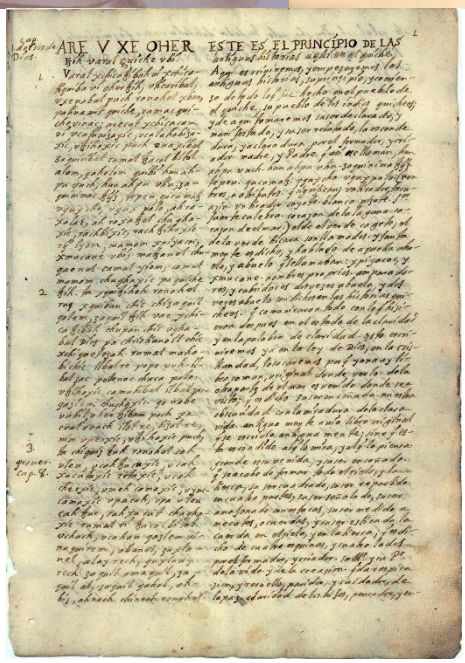
Izzze haglagolo connic luseh zu&ih
zlodeiu luzem iego deuuich luzeh b moki
delom luzem iego Da minuite napomoki
lepocam tose uue brti kibogu moih gre
ruiu ubog uze mo chou Dabim cisto iz
goki lu iega zim pouued ztuoril lod
lu zuueti dub.bra puztic otboga priel
tri imena edin bog Bogu uske mogo kemu
gozpod zuueti izpouuede uske moie
ise zuori nebo.bh greche lsce marie
emlo tose izco ie yzeh nepraudnih del
ga milosti lsce nepraudnega pomislena
mariae lsce mie lse iehem uuede ztuo
habela lsce pe ril lu neuuede nudm
tra luseh bosih zl. ili lubm zpe ili bde
luseh bosih mose Vsptrnih rotab vlsnih
nic luseh b za resih vtarbinab vsmesivge



Rabi samuel. 12.

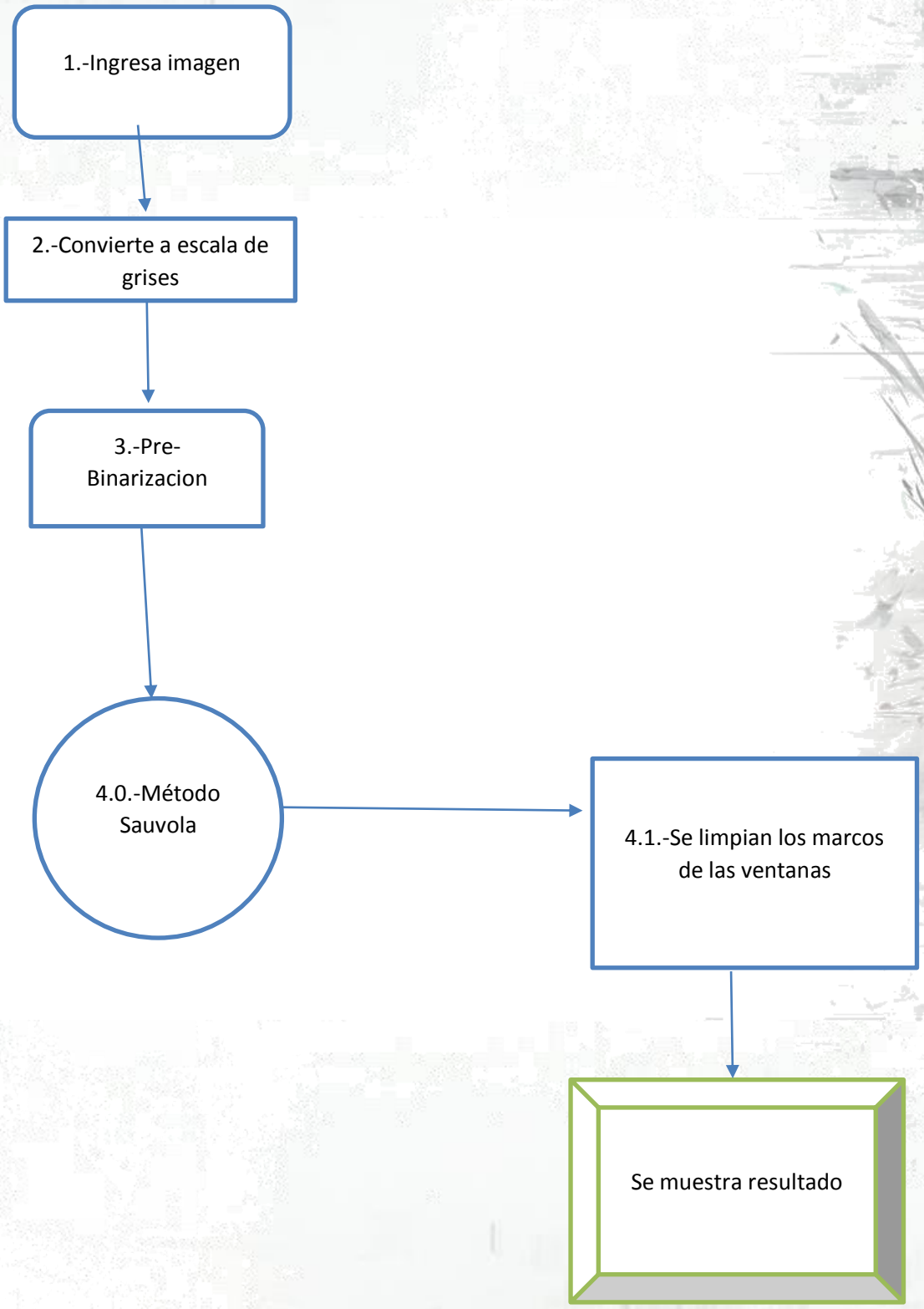
Capº. 16.





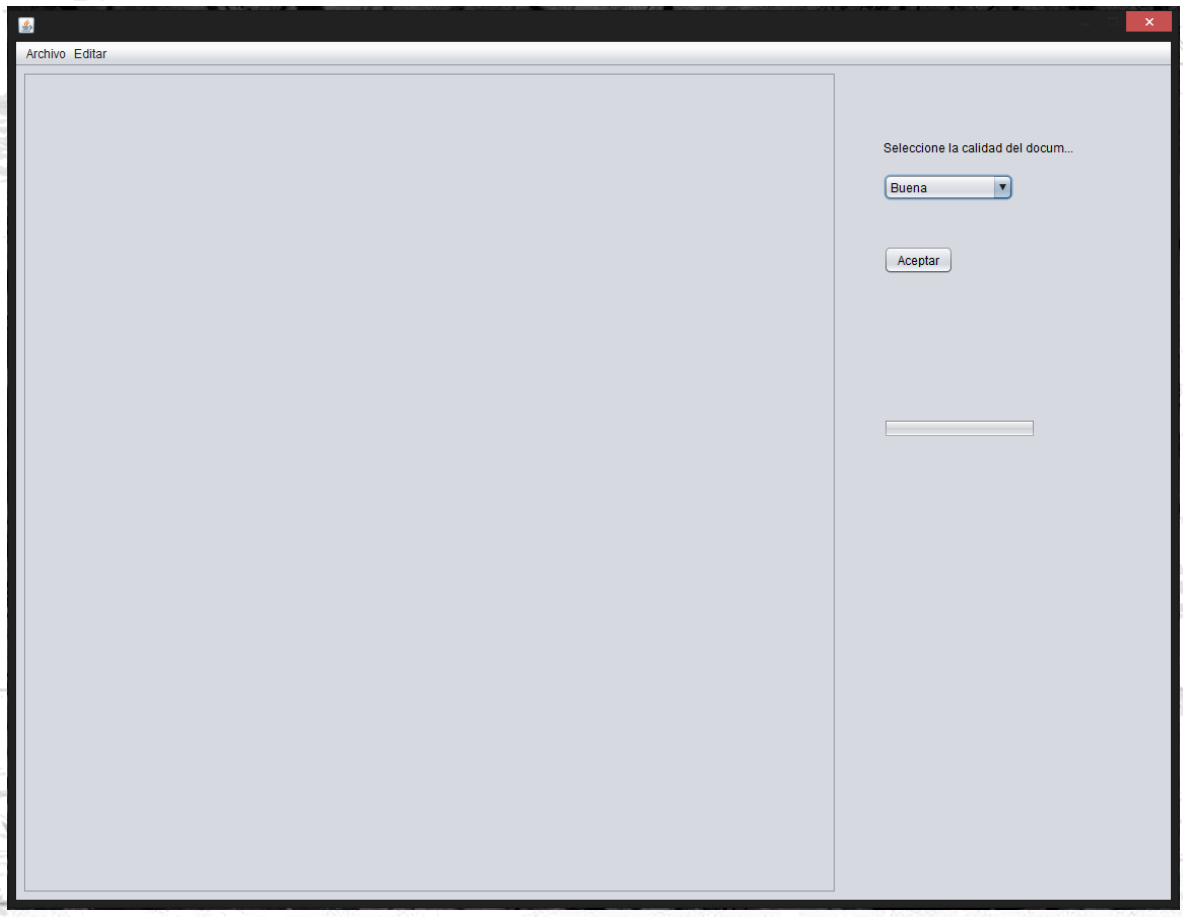
ARE V XE OHER ESTE ES EL PRINCIPIO DE LAS

El método consta de 4 pasos fundamentales:



El Programa cuenta con una interfaz muy sencilla para el usuario, la cual tiene un botón que ejecuta todos los métodos de forma automática, lo único que se pide al usuario es que elija el tipo de imagen que está ingresando ya sea:

- Buena
- Regular
- Mala
- Muy Mala
- Más que muy mala



Ya que en el documento donde fue basado este método no menciona cual es la fórmula para poder calcular k, R se llevó una investigación un poco más profunda y se encontró que para poder obtener los mejores valores para estas constantes de la formula se utiliza lo que se conoce como algoritmo genético, además muestran los valores

obtenidos y considerados como óptimos los cuales están en la siguiente tabla:

Tabla 2. Resultados promedio obtenidos luego de la optimización. Fuente: Autores

Variable	Valor
k	$0{,}570 \pm 0{,}489$
R	$13{,}451 \pm 20{,}060$
w	10
S_n	$0{,}947 \pm 0{,}046$
$1 - S_p$	$0{,}046 \pm 0{,}065$
Área ROC	0.907
t	$16{,}5\ s$

(Molina-Cortés, 2011)

En base a esto y algunas pruebas realizadas se decidió tomar valores para k que van desde 0.2 hasta 0.6 dependiendo de la calidad de la imagen; y para R se calculó en base a la media aritmética de la imagen elevada al cuadrado con una ventada w=15.

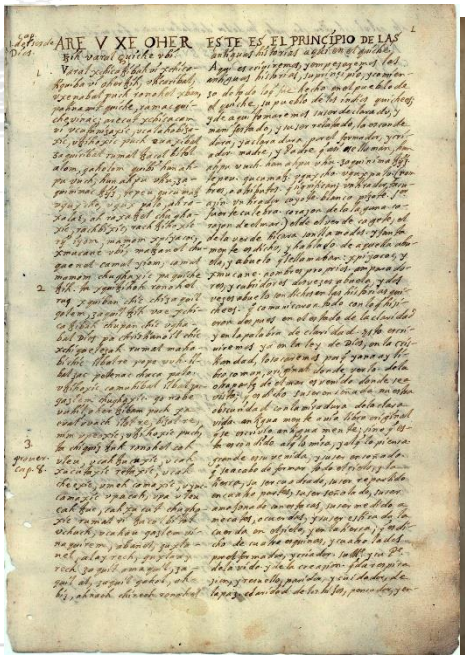
Y para optimizar tiempo se hicieron unas pequeñas modificaciones al método, las cuales se basan en la media de la imagen y se hace una pre-binarizacion que se explicara más adelante.

1. Ingresa la imagen

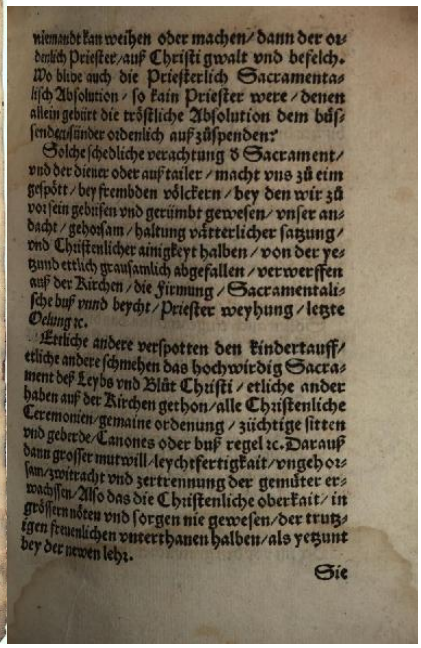
Para ingresar la imagen al programa solo basta con ir al menú “Archivo” y seleccionar abrir o presionar “Ctrl+O” y navegar en búsqueda de la imagen.

Una vez cargada la imagen el usuario puede hacer que el programa se ejecute automáticamente eligiendo la calidad de la imagen y presionando el botón “Aceptar” o puede hacerlo de modo manual para ver como son los pasos para llegar la imagen final, solo tiene que ir al menú “Editar” y seguir los pasos antes mencionados.

Para demostrar resultados tomaremos dos imágenes de la base y mostraremos sus resultados las cuáles serán las siguientes:



Popol_vuh.jpg



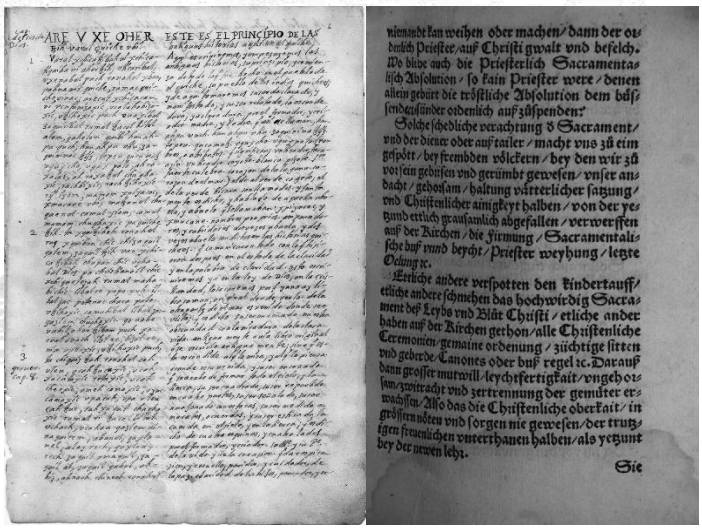
otsu-input-0.4.jpg

2. Conversión a escala de grises

En este paso se lee pixel por pixel y se toman los valores de los tres canales y se les aplica la siguiente formula

$$I'(x,y) = 0.2989 \cdot I_R(x,y) + 0.5870 \cdot I_G(x,y) + 0.1140 \cdot I_B(x,y)$$

Para asignarle el valor resultante al P(x,y).



niemandt kan weihen oder machen/ dann der or-
denlich Priester/ auß Christi gwalt vnd befelch.
Wo blibe auch die Priesterlich Sacramenta-
lisch Absolution/ so kain Priester were/ denen
allein gebürt die tröstliche Absolution dem bäs-
sten sünder ordenlich auß zůspenden:

Solche schedliche verachtung d Sacrament/
vnd der diener oder auß teiler/ macht vns zů ein
gespött/ bey frembden völckern/ bey den wir zů
vor sein gedrisen vnd gerümbt gewesen/ vnser an-
dacht/ gedosam/ haltung vätterlicher satzung/
vnd Christenlicher ainigkeyt halben/ von der ye
zuvnd ertrich grausamlich abgefallen/ verwerffen
auf der Kirchen/ die Firmung/ Sacramentali-
sche buß vnnd beycht/ Priester weyhung/ letzte
Oelung ꝛc.

Etliche andere verspotten den Kindertauff/
etliche andere schmehen das hochwirdig Sacra-
ment deß Leybs vnd Blůt Christi/ etliche ander
haben auf der Kirchen gethon/ alle Christenliche
Ceremonien/ gemaine ordenung/ züchtige sitten
vnd geberde/ Canones oder buß regel ꝛc. Darauß
dann grosser murwill/ leychfertigkait/ vngeho-
sam/ zwitracht vnd zertrennung der gemüter er-
wachsen/ Also das die Christenliche oberkait/ in
grössern nöten vnd sorgen nie gewesen/ der trutz-
igen freuenlichen vnterthanen halben/ als yetzunt
bey der newen lehr.

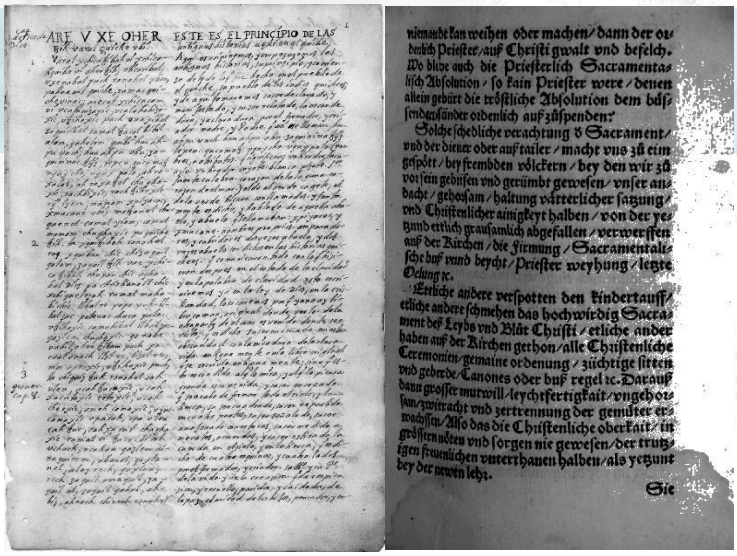
Sie

3. Pre-binarizacion

Con el fin de reducir tiempos ya que entre más grande sea la imagen el tiempo que se tarda este método es mayor, así que se decidió hacer una pre-binarizacion la cual consiste en usar un filtro regional en este caso el filtro Cosenoidal para saturar lo que son los colores oscuros y queden lo más cercanos a el valor 0 (0-70), tomando la media de la imagen tenemos que:

V=Media+ Media/2

Se recorre la imagen comparando el valor del pixel con el valor V, si el Pixel(x,y)>V entonces se le asigna el color blanco o 255 .



4. Sauvola

Como ya se mencionó antes este método se basa en la fórmula:

$$T(x,y) = m(x,y)\left[1 + k\left(\frac{s(x,y)}{R} - 1\right)\right]$$

Donde m es la media de la ventana y s es la desviación estándar la cual como sabemos tiene un alto costo computacional así que para ahorrar un poco se modificó el método y haciendo la pre-binarizacion la cual nos va a ayudar a reducir el número de pixeles a los cuales tenemos que aplicarles la formula, el algoritmo para esto es el siguiente:

Si P(x,y)!=0 && P(x,y)!=255

{

Sauvola();

}

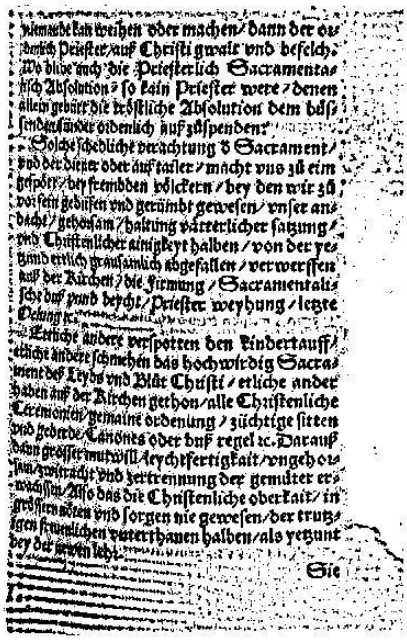
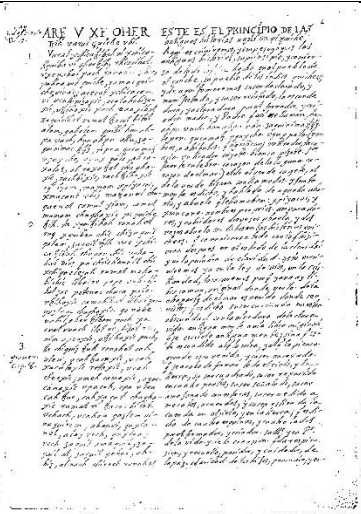
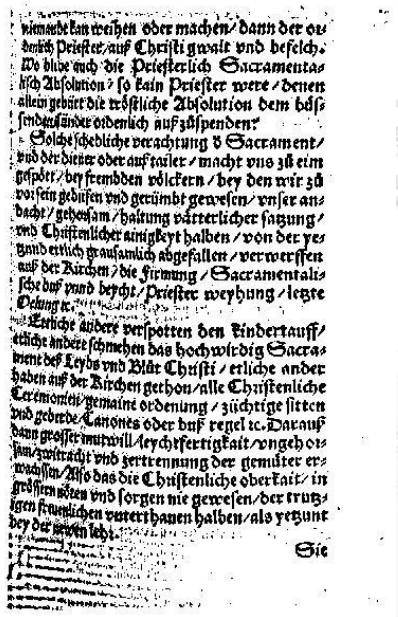
Sino{

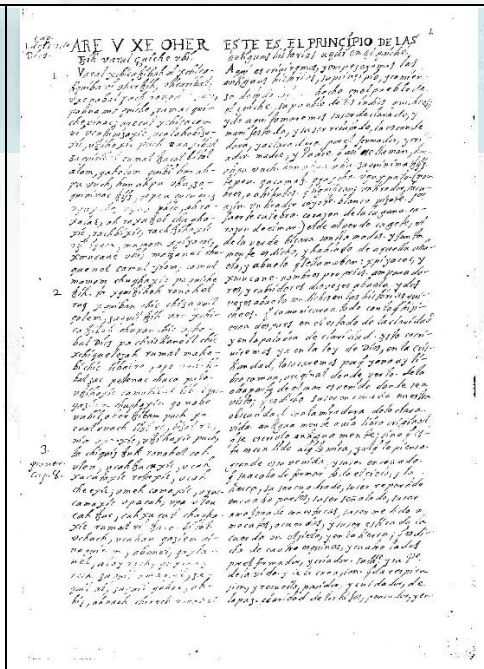
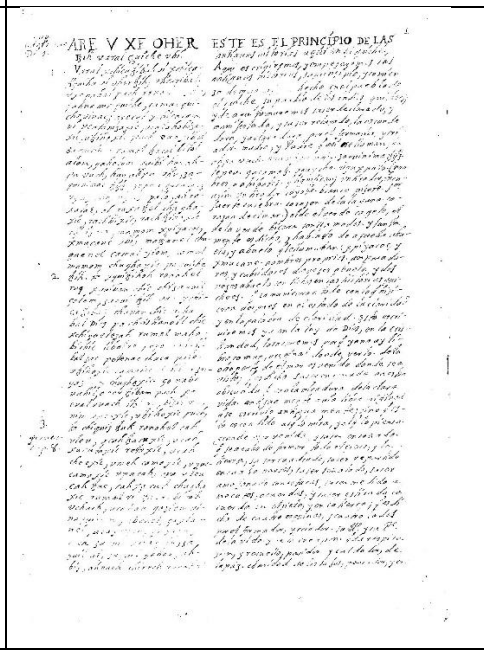
El pixel no se modifica}

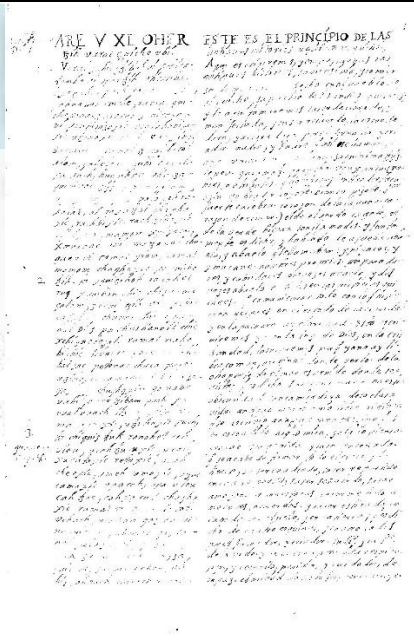
Esto tomando en cuenta que se hizo una saturación del color negro y del color blanco, para que las letras quedaran lo más cercano a 0 y algunos colores claros del fondo quedaran en 255 y así dejar lo que se puede considerar como ruido y el método de Sauvola se encargue de quitarlo.

Dentro del método de Sauvola se encuentra la parque que se encarga de quitar el marco de la ventana usada en el método en el cual recorre toda la imagen en búsqueda de pixeles negros y blancos si no lo es lo convierte en color blanco para así tener una imagen 100% binarizada.

Resultados.

	Popol_vuh.jpg	otsu-input-0.4.jpg
Buena		
Regular		

Mala		niemandt kan weihen oder machen/ dann der or denlich Priester/ auf Christi gwalt vnd befelch. We blibe auch die Priesterlich Sacramentas lisch Absolution/ so kain Priester were/ denen allein gehört die trostliche Absolution dem bös sendengsünder ordenlich auf zůspenden: Solche schedliche verachtung ů Sacrament/ vnd der diener oder auf tauler/ macht vns zů eim gespött/ bey frembden völckern/ bey den wir zů vor sein gönsen vnd gerümbt gewesen/ vnser an dacht/ gehorsam/ haltung vätterlicher satzung/ vnd Christenlicher ainigkeyt halben/ von der ye zu nd etlich grausamlich abgefallen/ verworffen auf der Kirchen/ die Firmung/ Sacramentali sche duf vnnd beycht/ Priester weyhung/ letzte Oelung ꝛc. Etliche andere verspotten den kindertauff/ etliche andere schmehen das hochwirdig Sacra ment deß Leybs vnd Blůt Christi/ etliche ander haben auf der Kirchen gethon/ alle Christenliche Ceremonien/ gemaine ordenung/ züchtige sitten vnd gebrde/ Canones oder duf regel ꝛc. Darauf dann grosser mutwill/ leychfertigkait/ vngeho sam/ zwitracht vnd zertrennung der gemüter er wachssen/ Also das die Christenliche oberkait/ in grösten nöten vnd sorgen nie gewesen/ der trutz igen freuenlichen vnterthanen halben/ als yetzunt bey der newen lehr. Sie
Muy Mala		niemandt kan weihen oder machen/ dann der or denlich Priester/ auf Christi gwalt vnd befelch. We blibe auch die Priesterlich Sacramentas lisch Absolution/ so kain Priester were/ denen allein gehört die trostliche Absolution dem bös sendengsünder ordenlich auf zůspenden: Solche schedliche verachtung ů Sacrament/ vnd der diener oder auf tauler/ macht vns zů eim gespött/ bey frembden völckern/ bey den wir zů vor sein gönsen vnd gerümbt gewesen/ vnser an dacht/ gehorsam/ haltung vätterlicher satzung/ vnd Christenlicher ainigkeyt halben/ von der ye zu nd etlich grausamlich abgefallen/ verworffen auf der Kirchen/ die Firmung/ Sacramentali sche duf vnnd beycht/ Priester weyhung/ letzte Oelung ꝛc. Etliche andere verspotten den kindertauff/ etliche andere schmehen das hochwirdig Sacra ment deß Leybs vnd Blůt Christi/ etliche ander haben auf der Kirchen gethon/ alle Christenliche Ceremonien/ gemaine ordenung/ züchtige sitten vnd gebrde/ Canones oder duf regel ꝛc. Darauf dann grosser mutwill/ leychfertigkait/ vngeho sam/ zwitracht vnd zertrennung der gemüter er wachssen/ Also das die Christenliche oberkait/ in grösten nöten vnd sorgen nie gewesen/ der trutz igen freuenlichen vnterthanen halben/ als yetzunt bey der newen lehr. Sie

Mas que muy mala		niemandt kan weihen oder machen/ dann der ordenlich Priester/ auf Christi gwalt vnd befelch. Wo bliebe auch die Priesterlich Sacramentalisch Absolution/ so kain Priester were/ denen allein gebürt die tröstliche Absolution dem büssenden sünder ordenlich auf zůspenden? Solche schedliche verachtung d Sacrament/ vnd der diener oder auftailer/ macht vns zů ainem gespött/ bey frembden völckern/ bey den wir zů vorsein gebuten vnd gerümbt gewesen/ vnser andacht/ gehorsam/ haltung vätterlicher satzung/ vnd Christenlicher ainigkeyt halben/ von der yetzund etlich grausamlich abgefallen/ verwerffen auf der Kirchen/ die firmung/ Sacramentalische buss vnnd beycht/ Priester weyhung/ letzte Ölung ꝛc. Etliche andere verspotten den kindertauff/ etliche andere schmehen das hochwirdig Sacrament des Leybs vnd Blůt Christi/ etliche ander haben auf der Kirchen gethon/ alle Christenliche Ceremonien/ gemaine ordenung/ zůchtige sitten vnd gebrede/ Canones oder buss regel ꝛc. Darauß dann grosser mutwill/ leychtfertigkait/ vngehorsam/ zwitracht vnd zertrennung der gemüter erwachssen/ Also das die Christenliche oberkait/ in gröstern nöten vnd sorgen nie gewesen/ der trutzigen freuenlichen vnterthanen halben/ als yetzunt bey der newen lehr. Sie

Conclusiones

En conclusión podemos observar que el método es muy efectivo en cuanto a quitar el ruido en la imagen solo es cuestión de seleccionar la K con la que se tiene que trabajar ya que seleccionar una K diferente a la que corresponde trae resultados con demasiado ruido o si la K es muy grande puede eliminar pedazos de letras o hasta palabras.

Sería muy interesante e importante el implementar un método para que todo sea automatizado y el usuario no tenga que preocuparse por los valores y sea la misma maquina quien se encargue de determinarlos y así poder obtener el mejor resultado.

Otra posible solución podría ser el uso de algún filtro de suavizado o “smooth” que aparte de limpiar el poco ruido que deja este método también pueda rellenar esos pedazos que puedan llegar a faltar en la letra para poder tener una imagen mejor preparada para OCR.

Referencias

- Jeyson Molina-Cortés,Alejandro Restrepo-Martínez,John W. Branch-Bedoya **Optimización de la Segmentación Local de Sauvola Aplicada a la Detección de Defectos Superficiales en Escenas con Iluminación No Homogénea**

- Ines Ben Messaoud **A Design of a Preprocessing Framework for Large Database of Historical Documents**
- J. Sauvola*, M. PietikaKinen **Adaptive document image binarization**

Molina-Cortés, J. (Diciembre de 2011). Obtenido de Scielo: http://www.scielo.org.co/scielo.php?pid=S0123-77992011000200004&script=sci_arttext

Imágenes tomadas de:

Google Imágenes

delcampe.es

http://www.todocoleccion.net/

http://www.cervantesvirtual.com/

en 1

user

2015

Contenido

en 1

Introducción

En esta practica utitilazaremos los filtros ya antes vistos como son:

- Seno
- Exponencial
- Logaritmo
- Gamma

Los cuales nos sirven para poder aclarar una imagen cada uno con una intensidad diferente en algunos como es el caso del Exponencial y de Gamma se le pide al usuario que proporcione el valor de una variable ya sea alpha o gamma en sus respectivos casos, ademas integramos los filtros:

- Cosenoidal
- Oscurecimiento Exponencial

Los cuales nos sirven para poder oscurecer una imagen, tambien cada uno tiene su nivel de oscurecimiento en el caso del Exponencial se le pide al usuario que proporcione el valor de alpha el cual entre mas grande mas ocura resultara la imagen.

Repasando las formulas de los filtros tenemos que:

- Logaritmo
  - $A \ln(\alpha x + 1)$

Donde A es: A $\ln(\alpha q + 1)$, y Alfa se considera como una constante = 1

- Función Seno
  - $q \sin\left(\frac{\pi x}{2q}\right)$
- Función exponencial
  - $A\ (1 - e^{-\alpha x/q})$

Donde A es: $\frac{q}{1-e^{-\alpha}}$ y alfa está dado por el usuario

- Gamma
  - $r' = q\left(\frac{p}{q}\right)^{\gamma}$

Donde γ esta dado por el usuario.

Filtros de Oscurecimiento

- Cosenoidal
  - $q * (1 - \cos(\frac{\pi x}{2}))$
- Exponencial Osc

- $A * (e^{\frac{\alpha x}{q}} - 1)$
- $A = q/(e^{\alpha} - 1)$

Las formulas anteriores nos sirven para evaluar y modificar las imágenes en el intervalo [0,255] el cual pertenece a los valores que puede tomar cada canal (R,G,B), pero para esta practica el reto es modificar las funciones de tal manera que el usuario pueda proponer un intervalo que se encuentre entre [0,255] para cualquier funcion y con esto pueda hacer una selección de varios filtros y aplicarlos al mismo tiempo para obtener un mejor resultado en la imagen.

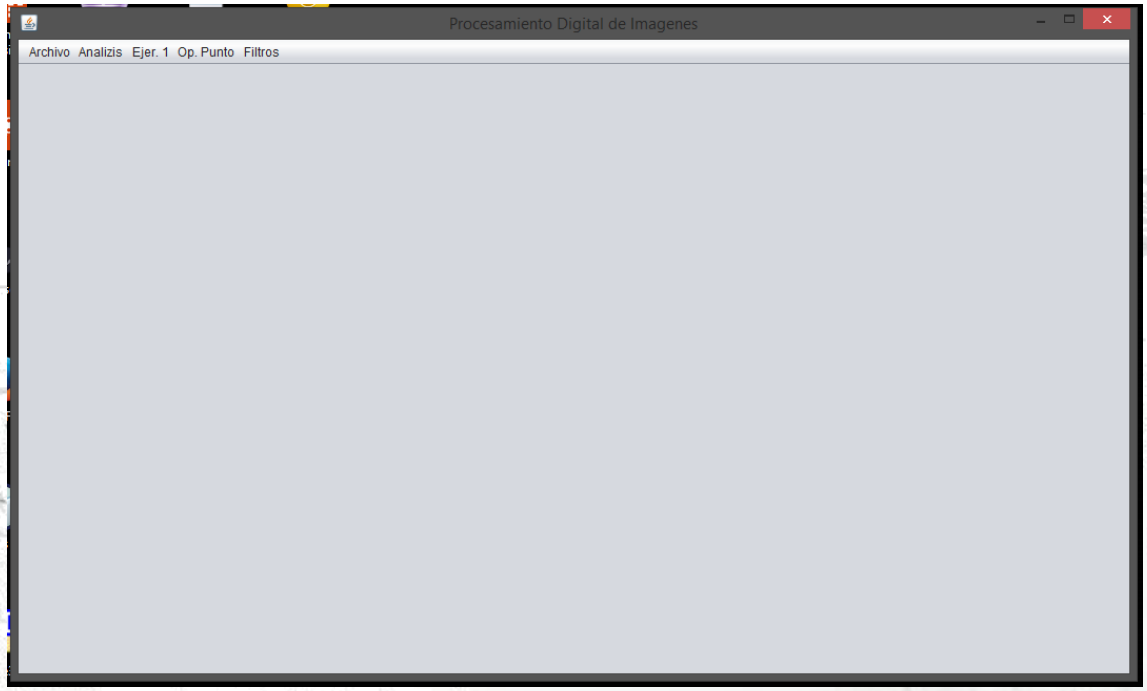
El usuario podra elegir entre 1 y 5 filtros maximo para su aplicación, una ves seleccionados y lllenados los campos necesarios se procese a procesar la imagen al final de todo se muestra la imagen resultante acompañada de una grafica que muestra como fue alterada la imagen dada las funciones y rangos seleccionados.

Desarrollo

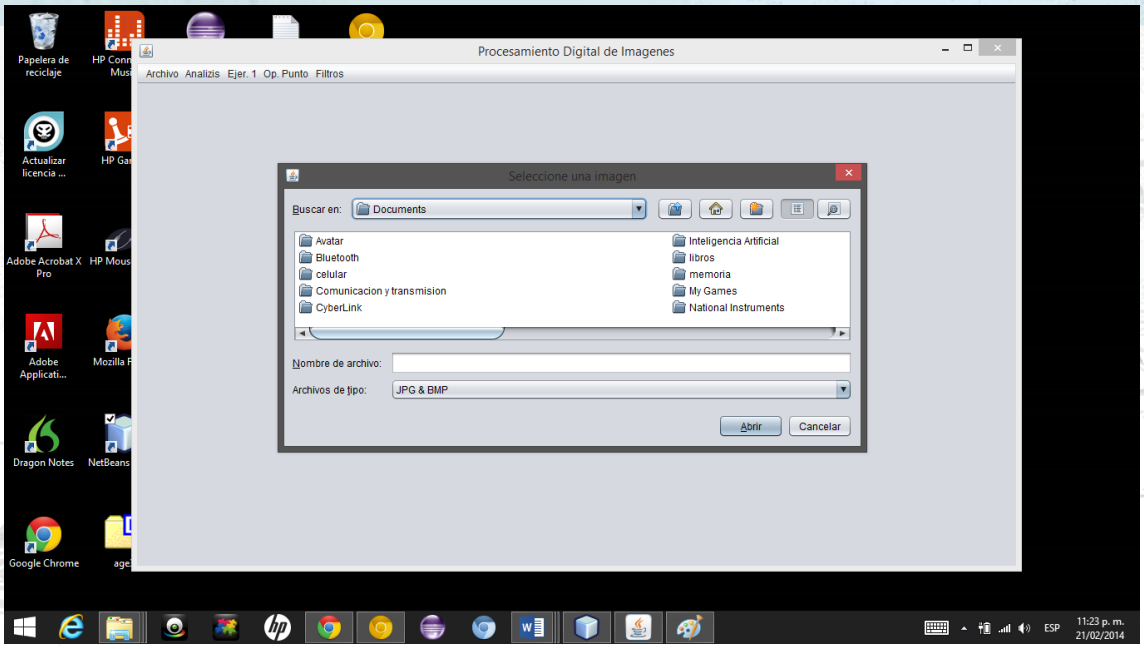
La implementacion de los filtros esta hecha en java y la IDE Netbeans la cual se puede descargar desde la pagina oficial en donde la puedes encontrar en sus diferentes versiones dependiendo el sistema operativo que ocupes:

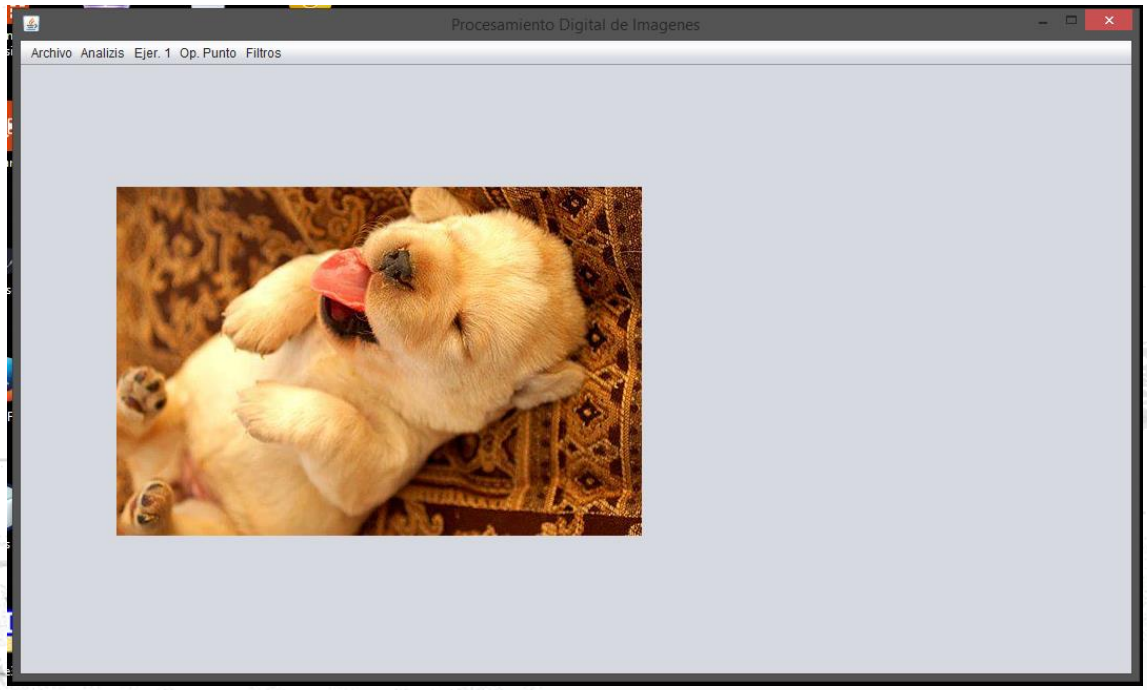
https://netbeans.org/

El proyecto consta de una Ventana principal en la cual podemos visualizar la imagen y en un menu que se encuentra ubicado en la parte superior de la ventana podemos encontrar un lista de operaciones que se le puede hacer a la imagen.



Para poder cargar una imagen podemos hacerlo de dos maneras presionando Ctrl+O o simplemente dando click al menu Archivo->Abrir, se navega hasta la carpeta deseada y se selecciona una imagen para mostrarla.





Implementacion de filtros regionales

Para poder implementar los filtros se tuvo que hacer un análisis matemático para poder asegurar que las funciones funcionen en el rango que el usuario desea ya que las formulas tal y como fueron presentadas al principio solo sirven para el rango [0,255].

Tras el análisis matemático se llegó a la conclusión que las formulas quedan de esta manera:

- Logaritmo
  - $A \ln(\alpha(x - x_0) + 1) + x_0$

Donde A es: $\frac{q-x_0}{Log\ Q}$ y $Q = x_1 - x_0 + 1$ $y\ q = x_1$ Alfa se considera como una constante = 1

- Función Seno
  - $(q - x_0) \sin\left(\frac{\pi(x-x_0)}{2(q-x_0)}\right) + x_0$
- Función exponencial
  - $A\ (1 - e^{-\frac{\alpha(x-x_0)}{q-x_0}})$

Donde A es: $\frac{q-x_0}{1-e^{-\alpha}}$ y alfa está dado por el usuario

- Gamma
  - $r' = (q - x_0)\left(\frac{x-x_0}{q-x_0}\right)^{\gamma} + x_0$

Donde γ esta dado por el usuario.

Filtros de Oscurecimiento

- Cosenoidal
  - $(q - x_0) * \left(1 - \cos\left(\frac{\pi(x-x_0)}{2(q-x_0)}\right)\right) + x_0$
- Exponencial Osc
  - $A * \left(e^{\frac{\alpha(x-x_0)}{q-x_0}} - 1\right) + x_0$
  - $A = (q - x_0)/(e^{\alpha} - 1)$

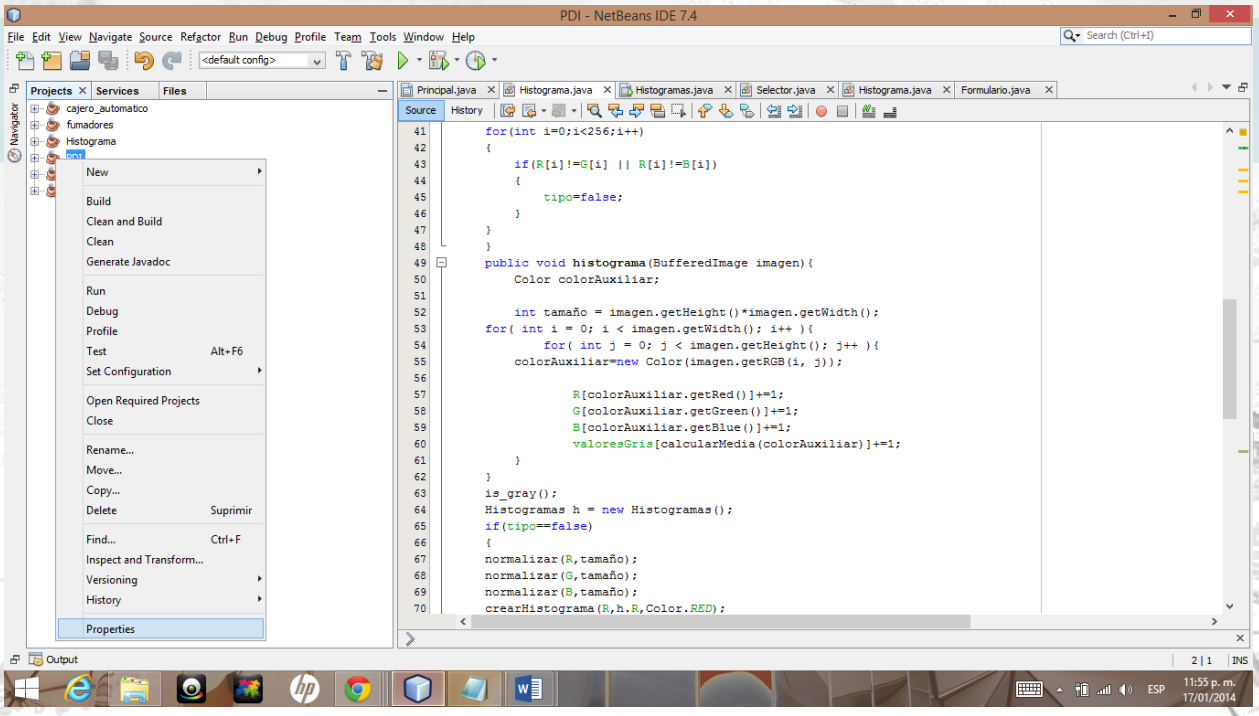
En teoría para poder aplicar los códigos solo se tiene que hacer lo siguiente:

```
Desde i=0 hasta largo de la imagen {
        Desde j=0 hasta ancho de la imagen {
                Obtener color;
                Separar Color en R,G,B;
                //procedemos a usar la formula de la función para cada color
                CanalR=F(R);
                CanalG=F(G);
                CanalB=F(B);
                //Se asigna un nuevo color al pixel
                Pixel= nuevo color(R,G,B);
        }
}
```

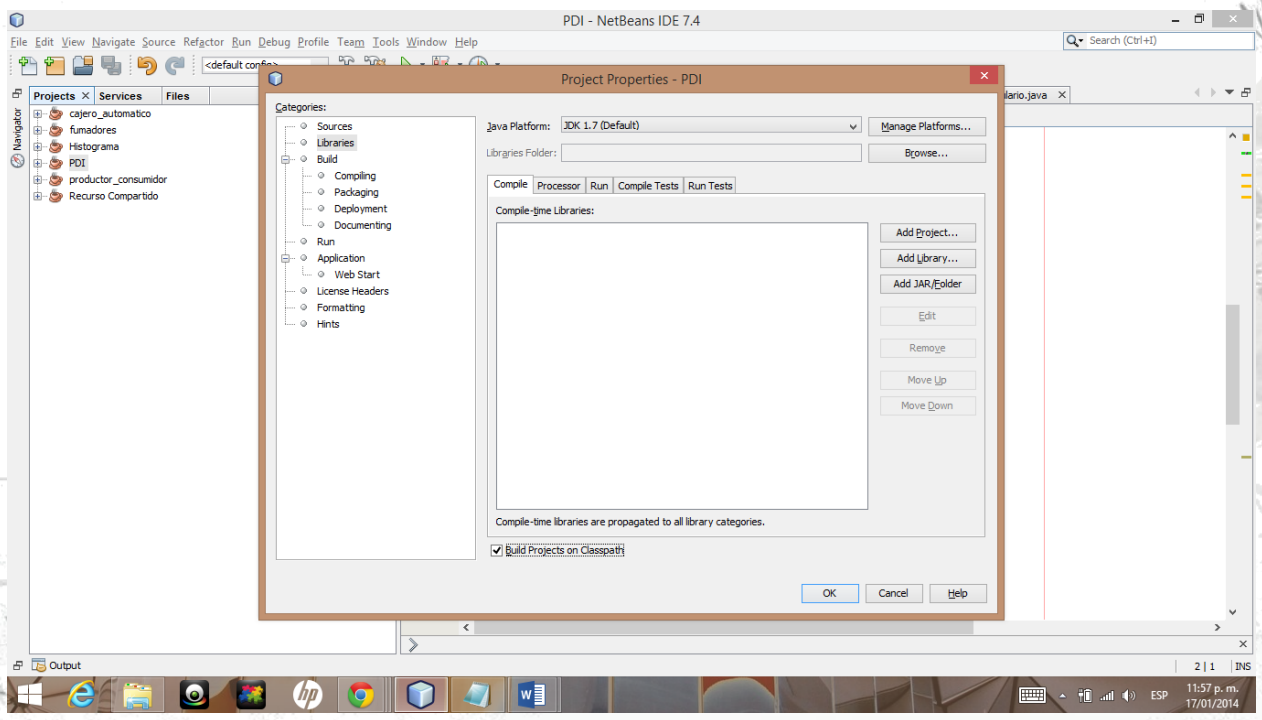
Para poder crear la gráfica de la función se manda un arreglo como parámetro a cada función en el cual se evalúa la F() para así poder tener una gráfica de las múltiples modificaciones que se le hicieron a la imagen.

Para crear la grafica se necesito de las librerias **jfreechart** y **jcommon** las cuales podemos descargar de la pagina oficial http://sourceforge.net/projects/jfreechart/files/.

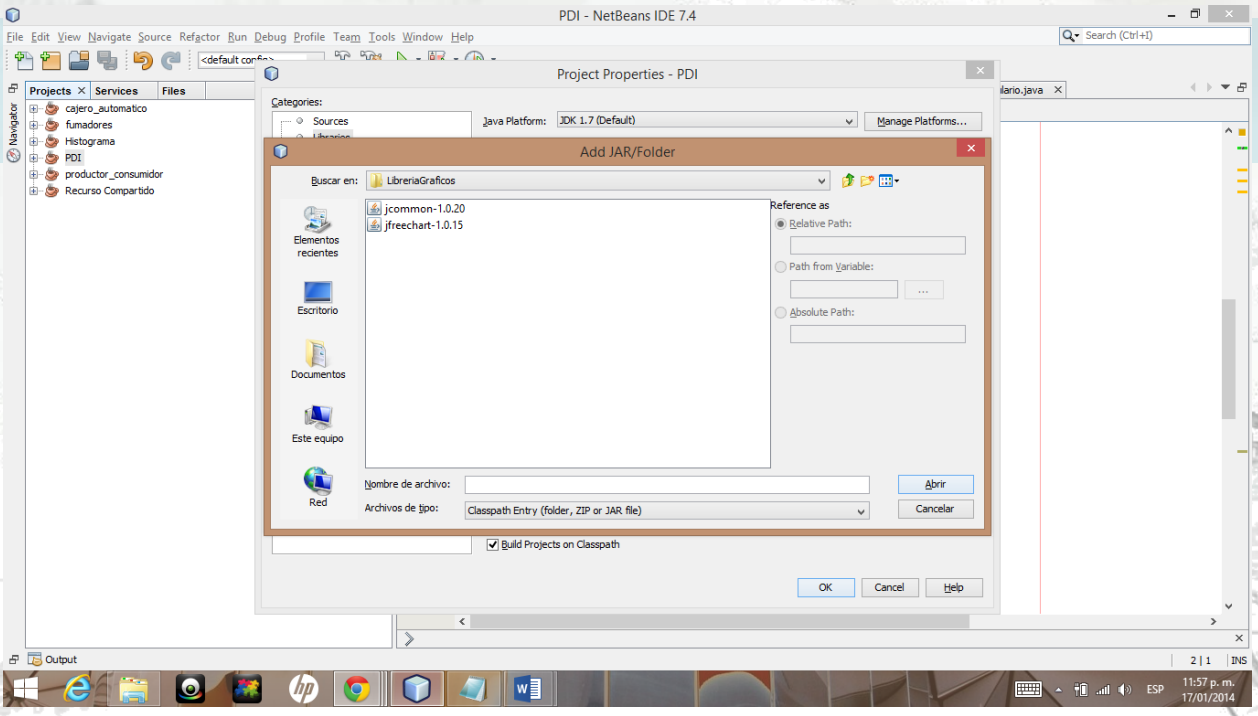
Una vez descargadas lo único que debemos hacer es dar click derecho en nuestro proyecto y dirigirnos a propiedades



Despues nos dirigimos a Libraries y seleccionamos “Add JAR/Folder”



Buscamos nuestras librerias que descargamos, las seleccionamos y le damos aceptar

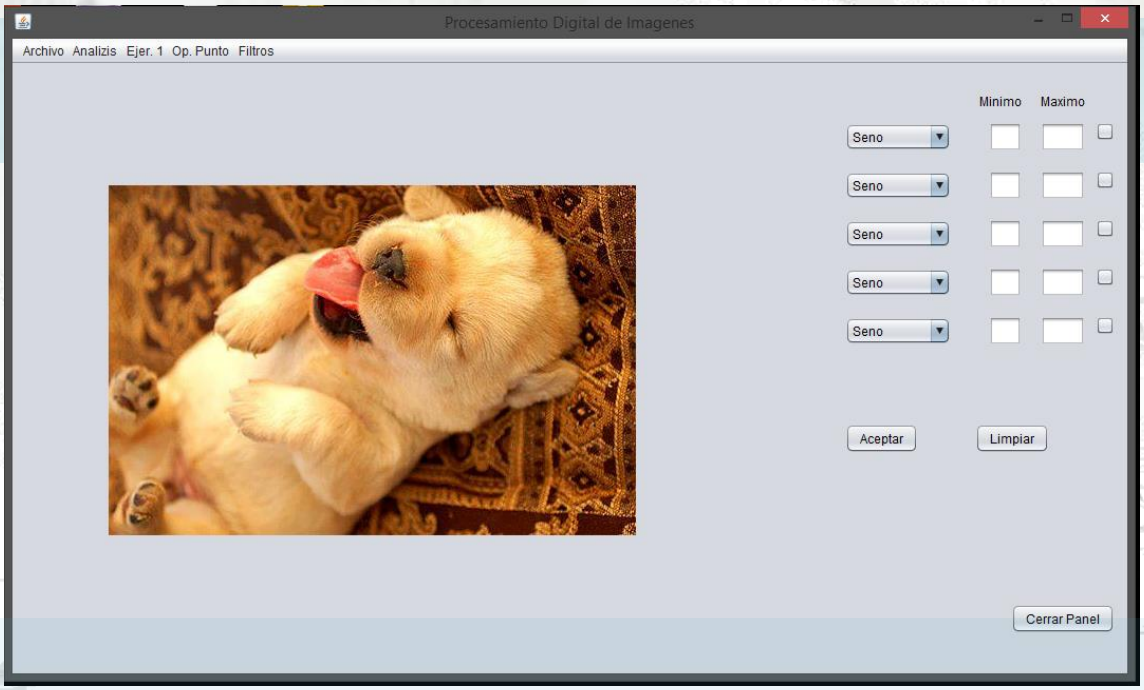


Y con esto tenemos agregadas las librerias al proyecto ya solo haria falta seleccionar un tipo de grafico e importarlo por ejemplo:

```
import org.jfree.chart.ChartFactory;

import org.jfree.chart.ChartPanel;

import org.jfree.chart.JFreeChart;

import org.jfree.chart.plot.CategoryPlot;

import org.jfree.chart.plot.PlotOrientation;

import org.jfree.chart.plot.XYPlot;

import org.jfree.chart.renderer.category.BarRenderer;

import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;

import org.jfree.data.category.DefaultCategoryDataset;

import org.jfree.data.xy.XYSeries;

import org.jfree.data.xy.XYSeriesCollection;
```

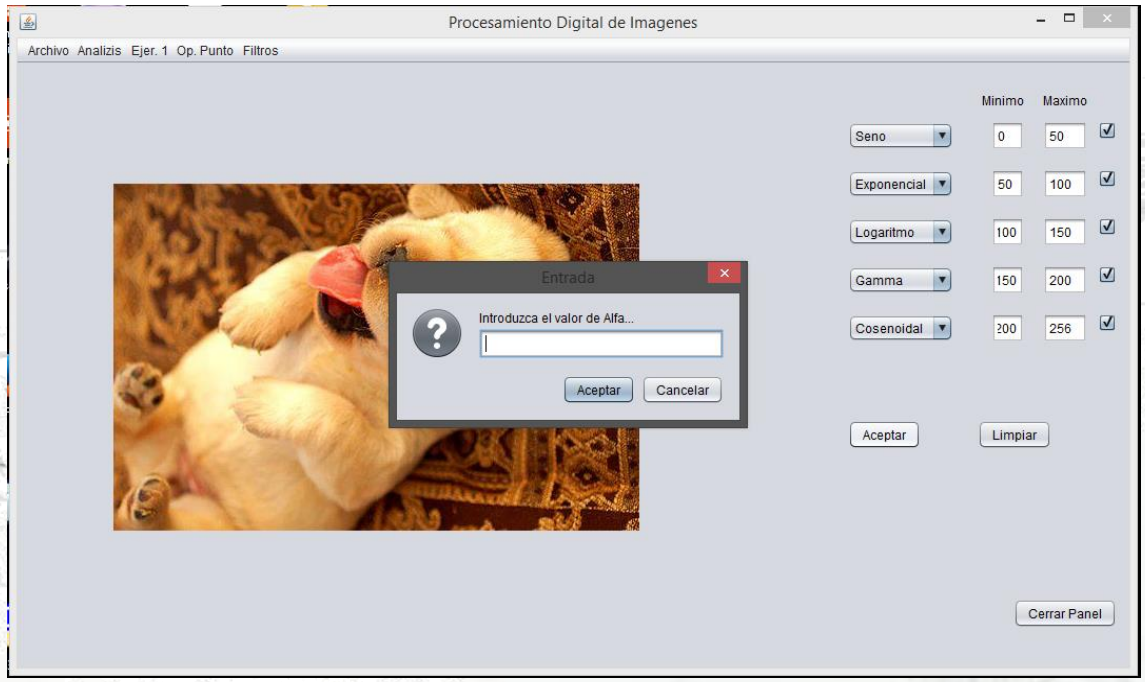
Uso de los filtros

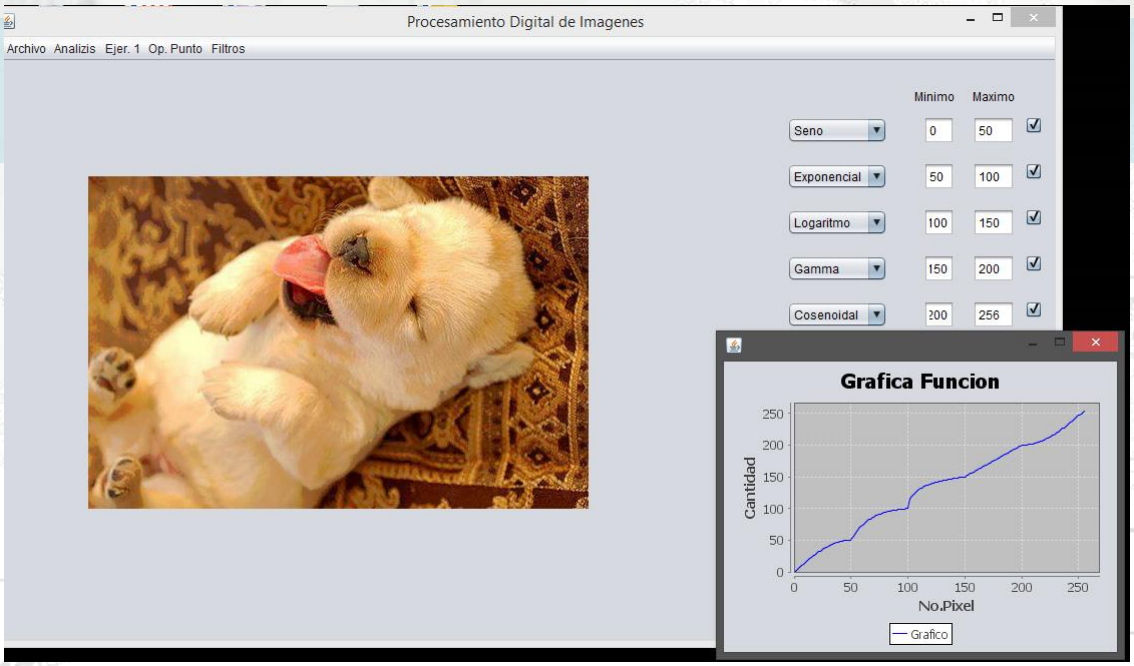
La ventana cuenta con un Panel el cual esta Invisible pero se puede activar llendo al menu Filtros->Regionales, si el panel ya esta visible lo vuelve invisible y nos queda algo asi:

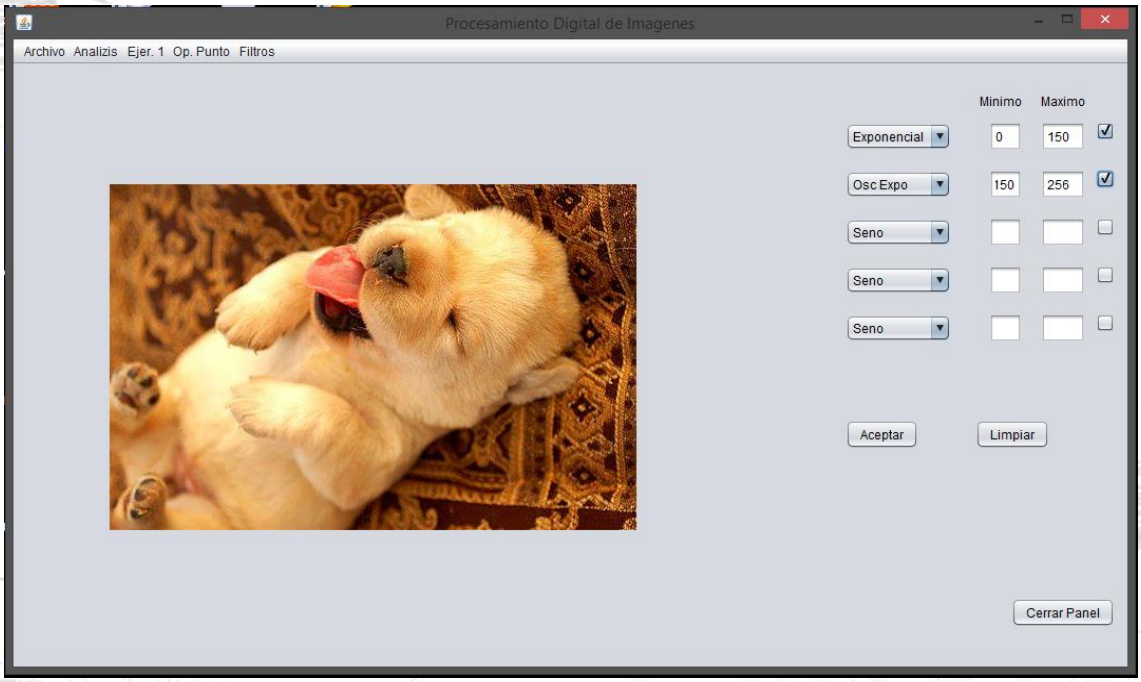


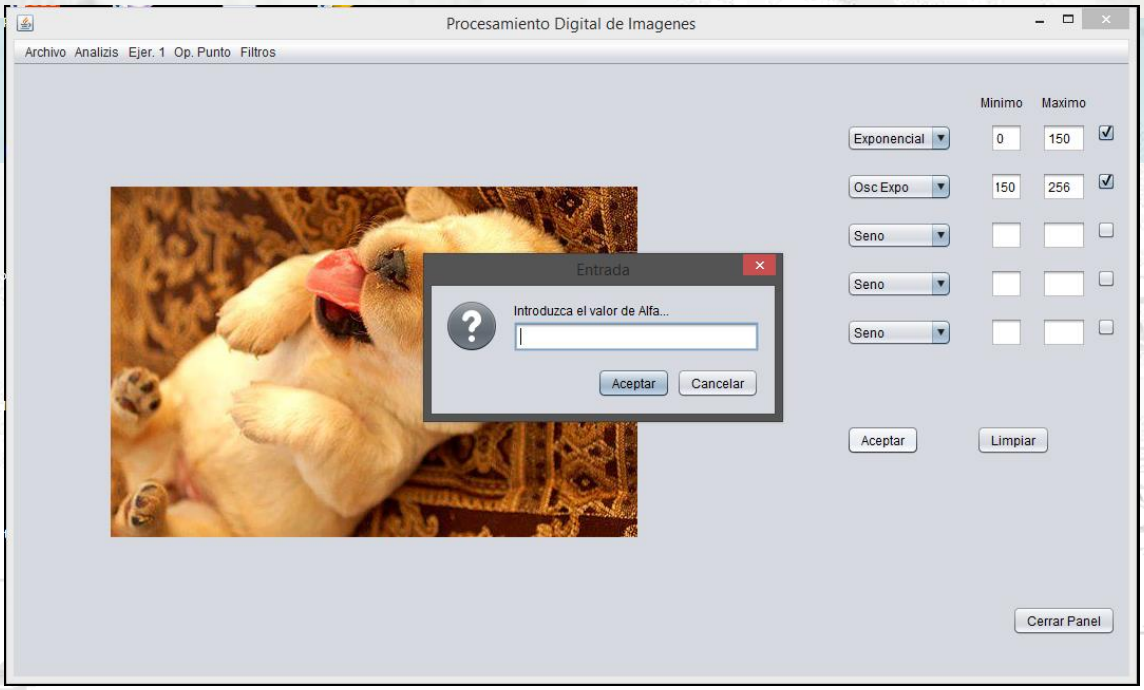
Para poder seleccionar aplicar un filtro es necesario seleccionar un filtro del JComboBox y en los JTextField poner los rangos en que se quiera que funcione y por ultimo marcar los checkboxs correspondientes y presionar el boton aceptar, en dado caso de que el usuario se equivoque al ingresar los datos puede usaar el boton limpiar para poner todos los campos vacios.

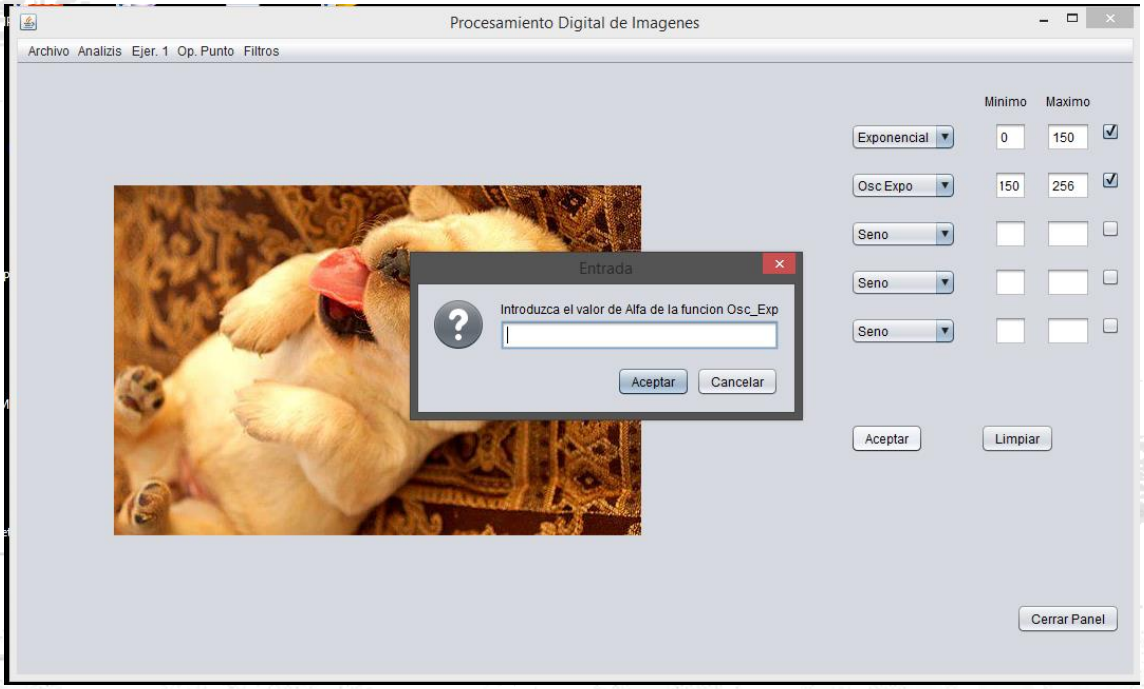
Resultados

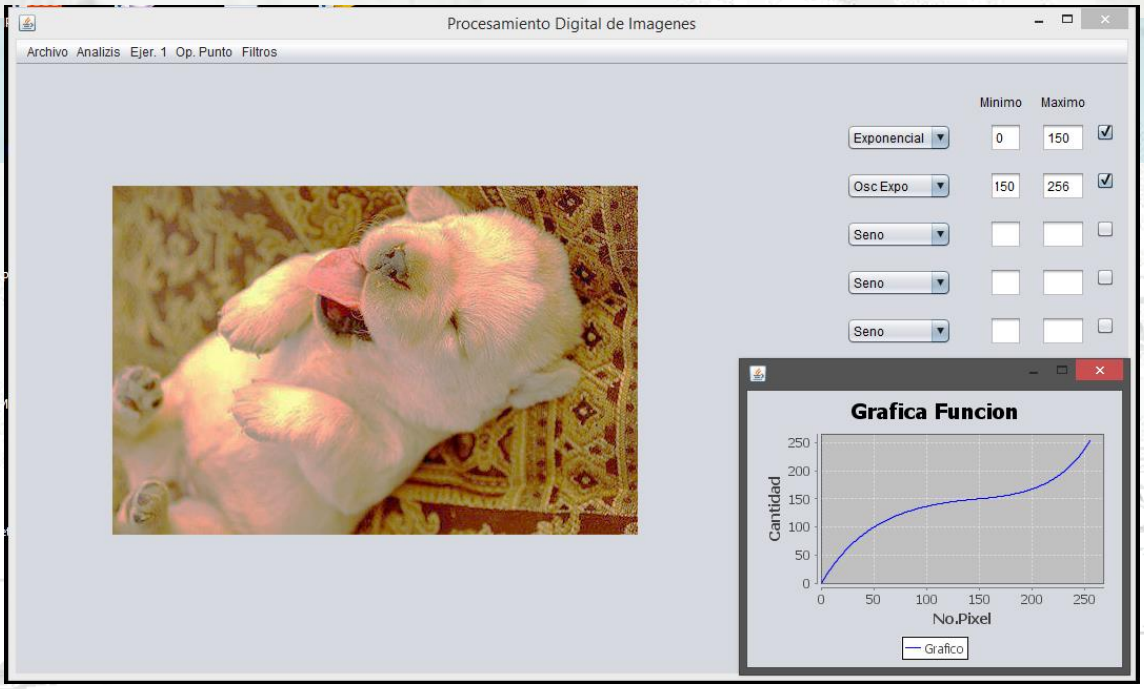



Procesamiento Digital de Imagenes
Archivo Analizis Ejer. 1 Op. Punto Filtros
Minimo Maximo
Seno 0 50
Exponencial 50 100
Logaritmo 100 150
Gamma 150 200
Cosenoidal 200 256
Grafica Funcion
Cantidad
No.Pixel
Grafico


Procesamiento Digital de Imagenes
Archivo Analizis Ejer. 1 Op. Punto Filtros
Minimo Maximo
Exponencial 0 150
Osc Expo 150 256
Seno
Seno
Seno
Aceptar
Limpiar
Cerrar Panel


Procesamiento Digital de Imagenes
Archivo Analizis Ejer. 1 Op. Punto Filtros
Minimo
Maximo
Exponencial
0
150
Osc Expo
150
256
Seno
Seno
Seno
Entrada
Introduzca el valor de Alfa...
Aceptar
Cancelar
Aceptar
Limpiar
Cerrar Panel


Procesamiento Digital de Imagenes
Archivo Analizis Ejer. 1 Op. Punto Filtros
Minimo
Maximo
Exponencial
0
150
Osc Expo
150
256
Seno
Seno
Seno
Entrada
Introduzca el valor de Alfa de la funcion Osc_Exp
Aceptar
Cancelar
Aceptar
Limpiar
Cerrar Panel



Conclusión

Se puede concluir que los filtros regionales son demasiado útiles cuando se sabe que es lo que necesita la imagen además se debe de saber el rango exacto donde se debe aplicar dicho filtro para que así se pueda obtener el mejor resultado, en otras palabras el que use el programa debe de tener noción de los comportamientos de cada filtro y por ende saber cual es el adecuado para su imagen.

FACULTAD de CIENCIAS de la COMPUTACIÓN

en 1

Materia:

"Procesamiento Digital de Imágenes"

Primavera 2014

Profesor:

Dr. Iván Olmos Pineda

Integrantes:

Cesar Manuel Rodríguez Mendiola 2011

Jesús Max Melchor Jiménez 201125498

FECHA: 27 de abril del 2014

1 Introducción

En este proyecto se implementara el uso de la interpolación en las imágenes digitales, estas interpolaciones se utilizan para crear nuevos pixeles en la imagen digital, esto es para darle un acercamiento a la imagen (Zoom), además de que existen una gran variedad de interpolaciones de datos, en este proyecto solo se implementaran tres tipos de interpolación, y se demostrara cual es la interpolación más efectiva.

2 Desarrollo

Para el desarrollo de la práctica, la cual está basada en el lenguaje orientado a objetos JAVA , se hizo uso de las librerías que ya vienen por defecto en el lenguaje, algunas de estas son la librería de “image”, la cual nos permite cargar una imagen en nuestro programa, también se hizo uso de la librería “math” para el cálculo de las funciones exponenciales y operaciones matemáticas.

De las librerías que usamos y que ya vienen incluidas en el lenguaje son:

1. java.awt.Color;
2. java.awt.image.BufferedImage;
3. java.awt.Graphics;
4. java.awt.Image;
5. java.io.IOException;
6. java.util.logging.Level;
7. java.util.logging.Logger;
8. javax.swing.ImageIcon;
9. javax.swing.JPanel;
10. java.awt.image.ImageObserver.WIDTH;
11. javax.swing.JOptionPane;
12. java.math.*;

Las librerías anteriores fueron usadas para el procesamiento, manejo y manipulación de los canales RGB de las imágenes, para ello se necesitó una variable de tipo color la cual nos la ofrece la librería 1, esta variable nos permite obtener el valor de los canales de un pixel, la librería 2 nos permite cargar una imagen de cualquier tipo de extensión a una variable de tipo buffer para su

respectivo manejo, así como las librerías anteriores las librerías de la 3 a la 11 nos permiten el uso de diferentes variables para darle una mayor comodidad grafica al usuario, y por último la librería 12 es la que nos permite hacer una gran variedad de cálculos matemáticos, así como el uso de constantes ya definidas en esta librería. Además de las funciones ya dichas anteriormente también se usaron las funciones incluidas por la librería javax.swing.ImageIcon para el manejo de las imágenes, así como apertura, cierre y manipulación de sus canales. Además también se hizo uso de las funciones:

JOptionPane.showMessageDialog (null,"String")

JOptionPane.showInputDialog ("String")

La primera se usó para advertirle al usuario que tipo de interpolación estaba a punto de utilizar, también se usó en el caso de que el usuario no haya seleccionado ninguna interpolación, y la segunda se implementó para pedirle al usuario que tipo de interpolación que le agrade más.

Para poder aplicar los filtros a las imágenes, primero se tiene que crear una nueva variable de tipo bufferimage en la cual contendremos la nueva imagen la interpolada, el tamaño de nuestra nueva imagen va a depender del tipo de factor de aumento que proporcione el usuario, después se tiene que recorrer toda la imagen pixel a pixel, e ir almacenando estos valores en la nueva imagen de tal forma que los valores que aún no conocemos queden libres para su próximo calculo ,después de tener la nueva imagen, se recorre pixel a pixel los pixeles vacíos de esta imagen y se hace el cálculo de estos pixeles dependiendo del tipo de interpolación que el usuario haya escogido, al obtener el valor del pixel por interpolación este se guarda en su dicha posición, la imagen resultante es una imagen a escala de grises, ya que es más fácil el manejo de un solo dato y no el de los tres canales (RGB).

Los tipos de interpolación con los que cuenta práctica son:

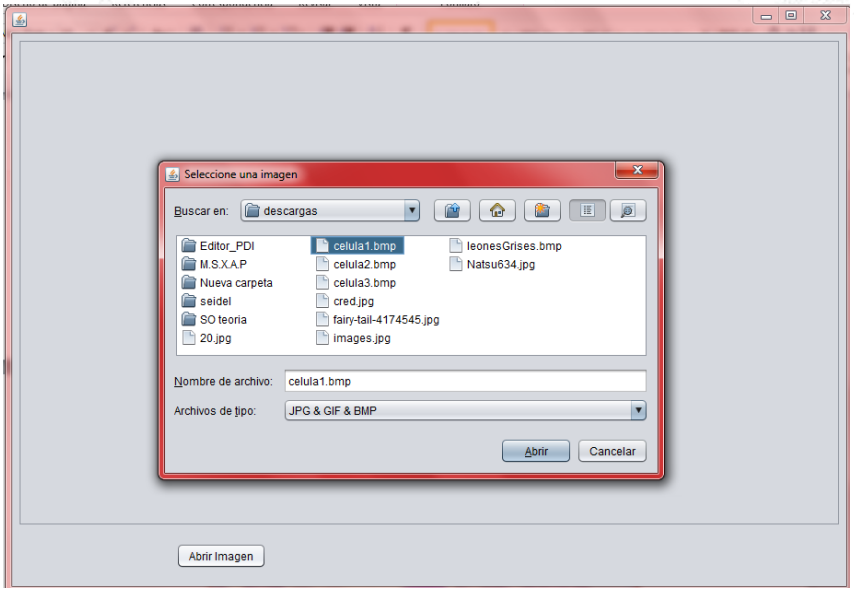
Interpolación	Funciones
LINEAL	$Y = Y0 + \frac{Y1 - Y0}{X1 - X0}(X - X0)$
CUADRATICA	$P1 = aX^2 + bX0 + c$ E(x0,x1) $P2 = a1X^2 + b1X0 + c$ E(x1,x2) $P3 = a2X^2 + b2X0 + c$ E(x2,xn) Dependiendo del intervalo de x la interpolación se calculara con alguno de los polinomios anteriores
CUBICA	$P1 = aX^3 + bX^20 + cX + d$ E(x0, x1) $P2 = a1X^3 + b1X^20 + cx + d$ E(x1, x2) $P3 = a2X^3 + b2X^20 + cX +$d E(x2,x3) $P4 = a3X^3 + b3X^20 + cX + d$ E(x3,xn) Dependiendo del intervalo de x la interpolación se calculara con alguno de los polinomios anteriores

En si en las interpolaciones anteriores los polinomios de interpolación se hallaron tomando en cuenta el número de pixel que estábamos trabando y dichos polinomios se igualaban al valor del pixel, además para que la interpolación no tuviera puntos de quiebre, a estos polinomios se les calculo la primera y segunda derivada para poder encontrar estos puntos de quiebre , ya obtenidos las ecuaciones de los polinomios, dichas ecuaciones se igualaban a el siguiente polinomio, esto para poder calcular las variables no conocidas, con el método de gauss Jordán se

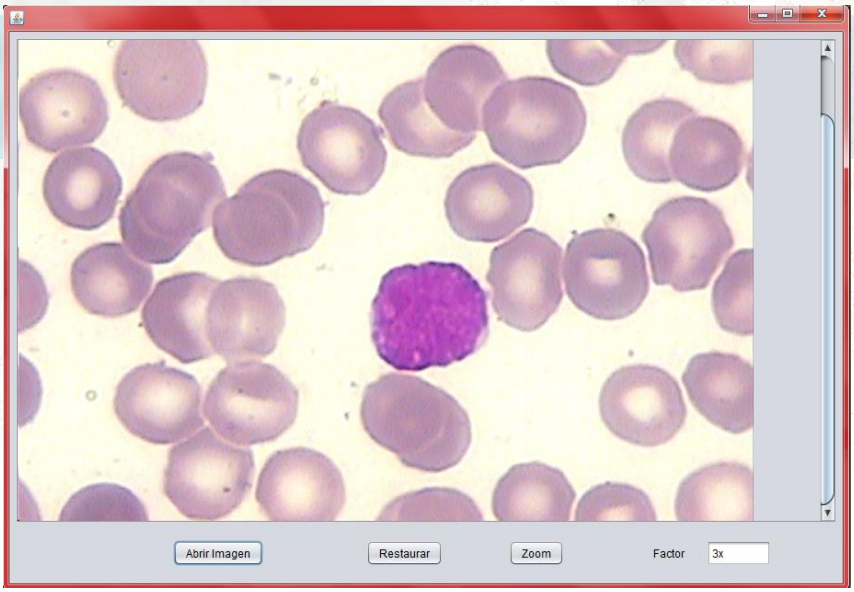
calcularon las incógnitas (a, b, c y d) desconocidas y ya obtenidas estas se las asignamos a los polinomios descritos anterior mente.

3 Resultados

Bueno ya dicho lo anterior, para el buen uso de la aplicación, lo primero que se tiene que hacer es presionar en la parte de abrir imagen y seleccionar la imagen que más nos agrade.

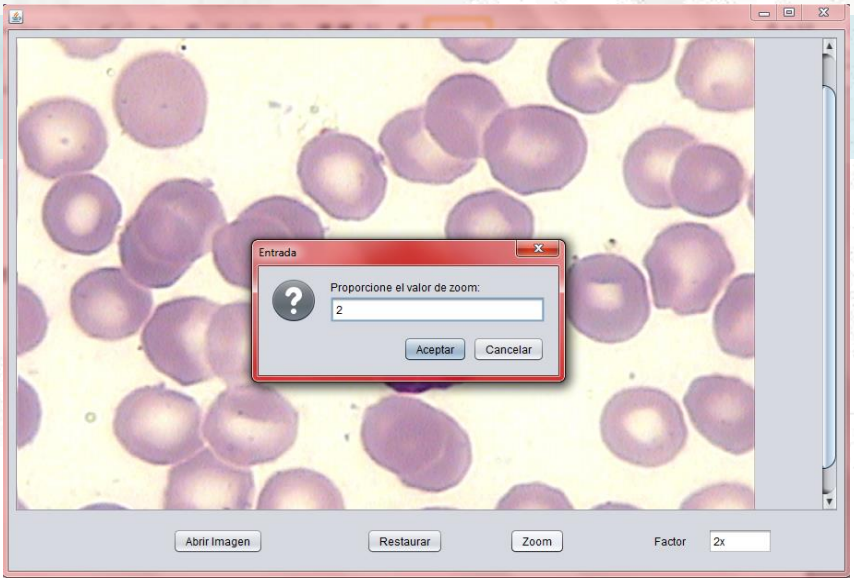


Después de haber seleccionado la imagen preferida por el usuario, esta se mostrara en una pequeña ventana con otros botones, dichos botones son para restaurar la imagen a su estado normal (sin interpolar), y el segundo botón es para aplicar el zoom (interpolación) a la imagen, y también se muestra una caja de texto para mostrarle al usuario que factor de aumento fue el que utilizo.

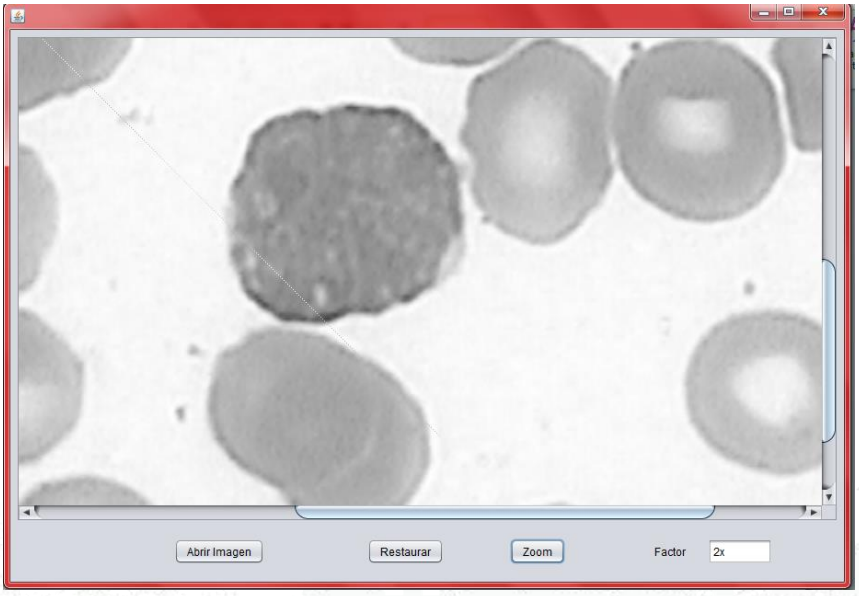


Después de haber seleccionado la imagen el usuario puede hacer uso de la interpolación, presionando el botón que dice zoom, este botón le pedirá al usuario el factor de aumento de la imagen, entre más grande sea el factor de aumento de la imagen, los datos interpolados serán más difíciles de calcular, esto se bebe a que el zoom digital no muestra valores que están en la imagen si no que estos son creados digitalmente, y como consecuencia de esto, la imagen va perdiendo contraste y claridad.

Como se muestra en la siguiente imagen, el programa le pide al usuario el factor de aumento en este caso se usó un factor de aumento 2x.



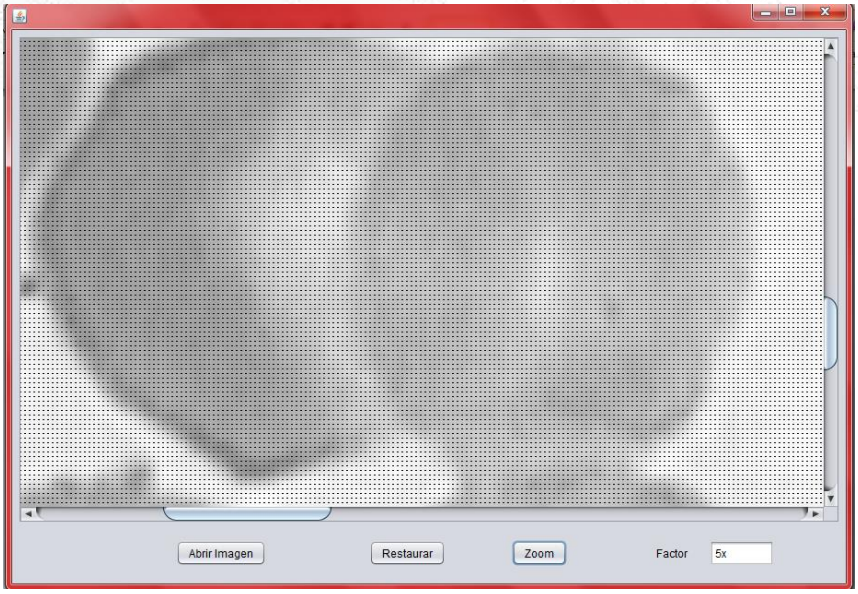
Los resultados de aplicar un factor de aumento 2x ala imagen feuron los siguientes.



Como se pudo observar en la imagen anterior el factor de aumnto 2x no proboca mucha perdida de informacion y nos permite ver el gran aumento de la imagen, y como podemos ver la imagen resultante es a escala de grises, no solo se pude aplicar un factor 2x ala imagen, sino que se podria aplicar un factor de aumento hasta de 40x, pero notariamos la gran perdiada de

imformacion. Tambien podemos ver que en la imagen se muestra en la caja de texto el factor de aumento que se utilizo.

En la siguiente imagen se muestra un ejemplo de aumento de factor 8x, para poder mostra que la perdida de informacion no es mucha.



4 Conclusiones

Se llegó a la conclusión de que el filtro utilizado en esta práctica no es muy sutil en el caso de interpolar datos, ya que en los resultados mostrados en este reporte no muestran que el filtro empieza a perder calidad después de aplicarle un factor de aumento 5x, también como pudimos observar el filtro es muy bueno en cuestión de que el aumento que se necesite no se ha demasiado y también se observó que dicho filtro de interpolación es de mejor calidad que el filtro de interpolación factor 2x y que el filtro lineal.

También llagamos a la conclusión de que este filtro de interpolación es mucho más costoso computacionalmente, ya que los cálculos que debe generar son para cada pixel que es generado por la interpolacion.

5 Bibliografía

http://www.imh.es/es/comunicacion/dokumentazio-irekia/manuales/curso-de-tratamiento-de-imagenes-con-gimp/referencemanual-all-pages

http://www.revista.unam.mx/vol.6/num5/art50/may_art50.pdf ----------

http://www.dzoom.org.es/zoom-optico-y-zoom-digital-cual-es-mejor/

http://algoimagen.blogspot.mx/2013/05/biblioteca-de-clases-para-tratamiento.html

http://algoimagen.blogspot.mx/2013/02/zoom-interactivo-en-picturebox.html#uds-search-results

http://algoimagen.blogspot.com.es/2013/10/java-pequena-aplicacion-para.html

http://ocw.usal.es/ensenanzas-tecnicas/herramientas-informaticas-para-el-geoprocesado

http://212.128.130.23/eduCommons/ensenanzas-tecnicas/procesamiento-avanzado-de-imagenes-digitales/contenidos/Tema2.pdf

http://es.wikipedia.org/wiki/Zoom_digital

http://www.secyt.frba.utn.edu.ar/gia/introd-al-proc-de-imagenes.PDF ----------

http://riunet.upv.es/handle/10251/12011

http://www.dte.us.es/ing_inf/trat_voz/Practicas/Practica2.pdf ----------

http://verona.fi-p.unam.mx/boris/practicas/Practica5.pdf

http://www.alumnos.inf.utfsm.cl/~vpena/ramos/ili286/presentacionCC2.pdf

http://bioingenieria.edu.ar/postgrado/index.php?option=com_content&view=article&id=261

http://www.us.es/ger/estudios/grados/plan_226/asignatura_2260022

http://www.jc-mouse.net/ingenieria-de-sistemas/esteganografia-lsb-en-java-proyecto-completo

http://jc-mouse.blogspot.mx/2011/02/procesamiento-de-imagenes-en-java2d-los.html

http://swing-facil.blogspot.mx/2012/03/escalar-imagen-con-swing-zoom-una.html

http://atobeto-eremita.blogspot.mx/2009/09/tratamiento-de-imagenes-en-java-parte-1.html

http://www.chuidiang.com/java/ejemplos/Panoramica/panoramica.php

https://www.google.com.mx/search?q=interpolacion+aplicada+a+imagenes&oq=interpolacion+aplicada+a+imagenes&aqs=chrome..69i57.15677j0j8&sourceid=chrome&es_sm=0&ie=UTF-8

http://blogvecindad.com/ampliar-el-tamano-de-una-imagen-por-interpolacion/

http://es.wikipedia.org/wiki/Interpolaci%C3%B3n_(fotograf%C3%ADa)

http://www.fernandogago.es/interpolacion.html

http://papeldeperiodico.com/2012/12/29/ampliando-fotos/

http://adobexpert.com/tecnicas-de-interpolacion-o-remuestreo-en-photoshop/#.U1CzBlV5NS4

http://es.scribd.com/doc/56680574/interpolacion

http://www.emezeta.com/articulos/interpolacion-de-imagenes

http://aicitel.files.wordpress.com/2012/03/tema_8.pdf ----------

http://www.alumnos.inf.utfsm.cl/~vpena/ramos/ili286/presentacionCC2.pdf

http://es.wikipedia.org/wiki/Interpolaci%C3%B3n

http://www.cambridgeincolour.com/tutorials/image-interpolation.htm

http://gva1.dec.usc.es/~antonio/docencia/2005tci/teoria/P2ModifGeome.pdf---------

http://en.wikipedia.org/wiki/Bicubic_interpolation

en.wikipedia.org/wiki/Nearest-neighbor_interpolation

http://worldofdesign.foroactivo.com/t498-interpolacion-de-imagenes

http://www.comolahice.com/2012/08/remuestrea-imagenes-con-la-interpolacion-adecuada/

http://alereimondo.no-ip.org/OpenCV/uploads/41/tema4.pdf ----------

http://www.alammi.info/2congreso/memorias/Documentos/martes/TRANSFORMGEOMETRICAS.pdf ---------

http://catarina.udlap.mx/u_dl_a/tales/documentos/mel/gonzalez_g_ra/capitulo2.pdf -------

http://algoimagen.blogspot.mx/2013/02/tratamiento-de-imagenes-parte-viii.html

http://alfaroymurga.blogspot.mx/2009_06_14_archive.html

http://es.scribd.com/doc/146195059/2reporte-interpolacion-bicubica

http://books.google.com.mx/books?id=h4Gj8GuwPVkC&pg=PA24&lpg=PA24&dq=como+funciona+la+interpolacion+bicubica&source=bl&ots=Cq3A9hYFbA&sig=6LP9yhGUDu38b3-tTppKKa3CjBU&hl=es-419&sa=X&ei=1gdSU8yeCIma2AXFw4DoCg&ved=0CIMBEOgBMAk#v=onepage&q=como%20funciona%20la%20interpolacion%20bicubica&f=false

http://leopard-bsod.blogspot.mx/2012/11/seleccion-de-area-de-una-imagen-con-java.html

http://sepwww.stanford.edu/public/docs/sep107/paper_html/node20.html

http://www.engr.mun.ca/~baxter/Publications/ImageZooming.pdf ----------

http://www2.uah.es/libretics/concurso2014/files2014/Trabajos/Ampl%EDa%20ese%20reflejo%20y%20compensa%20la%20imagen%20quiz%E1s%20podamos%20ver%20la%20cara%20del%20asesino.pdf ----------

[Seleccionar fecha]

en 1

[Escriba el subtítulo del documento]

user

en 1

Introducción

En esta práctica se implementa el uso de filtros regionales, estos filtros se utilizan en ciertos rangos para los canales de la imagen , esto quiere decir que solo se modificaran los valores que se encuentren dentro de estos rangos y los que no estén en estos rangos no se les modifica nada.

Desarrollo

Para la práctica de filtros regionales, la cual está basada en el lenguaje orientado a objetos JAVA , se hizo uso de las librerías que ya vienen por defecto en el lenguaje, algunas de estas son la librería de “image”, la cual nos permite cargar una imagen en nuestro programa, también se hizo uso de la librería “math” para el cálculo de las funciones trigonométricas.

De las librerías que usamos y que ya vienen incluidas en el lenguaje son:

```
java.awt.Color;

java.awt.image.BufferedImage;

java.awt.Graphics;

java.awt.Image;

java.io.IOException;

java.util.logging.Level;

java.util.logging.Logger;

javax.swing.ImageIcon;

javax.swing.JPanel;

java.awt.image.ImageObserver.WIDTH;
```

```
javax.swing.JOptionPane;

java.math.*;
```

Las librerías anteriores fueron usadas para el procesamiento, manejo y manipulación de los canales RGB de las imágenes.

Para poder aplicar los filtros a las imágenes, primero se tiene que recorrer toda la imagen pixel a pixel, e ir descomponiendo el pixel en los tres canales, rojo, verde, y azul, una vez descompuesto el pixel en los tres canales el usuario elige el o los filtros que desee aplicarle a la imagen, y el rango en el cual desea aplicar este. Dependiendo del o los filtros que escoja el usuario la o las funciones de cada filtro se les aplicara a cada uno de los canales de la imagen dentro del rango proporcionado por el usuario.

Los filtros y las funciones con las que cuenta la práctica son:

Filtro	Filtros puntuales	Filtros regionales
Aclarado Gamma	$f(x) = q\left(\frac{x}{q}\right)^{\gamma}$	$f(x) = (q' - x0)\left(\frac{x - x0}{q' - x0}\right)^{\gamma}$
Aclarado Logaritmo	$f(x) = A\log(\alpha x + 1)$	$f(x) = A\log(\alpha(x - x0) + 1)$
Aclarado Seno	$f(x) = q\,\text{sen}\left(\frac{\pi x}{2q}\right)$	$f(x) = (q - x0)\,\text{sen}\left(\frac{\pi(x - x0)}{2(q - x0)}\right) + x0$
Aclarado Exponencial	$f(x) = A(1 - e^{-\frac{\alpha x}{q}})$	$f(x) = A\left(1 - e^{\frac{-\alpha(x-x0)}{(q'-x0)}}\right) + x0$
Oscurecimiento Exponencial	$f(x) = A(e^{\frac{\alpha x}{q}} - 1)$	$f(x) = A\left(e^{\frac{\alpha(x-x0)}{(q'-x0)}} - 1\right) + x0$
Oscurecimiento Cosenoidal	$f(x) = q(1 - \cos\left(\frac{\pi x}{2q}\right))$	$f(x) = (q' - x0)(1 - \cos\left(\frac{\pi(x - x0)}{2(q' - x0)}\right)) + x0$

Para poder pasar de la forma de filtros puntuales a filtros regionales, se implemento el uso de operaciones algebraicas, y fueron evaluadas en los extremos de los canales de tal forma que F(0)=0 y F(255)=255.Cada formula tiene distintos valores tanto para q ,α ,γ ,A.

En la práctica también se implemento el uso de una clase para graficar las funciones de los filtros anteriores, dicha clase recibe un arreglo de 256 datos para poder graficar las funciones, este arreglo se inicializa de tal forma que todos los valores de 0 a 255 son evaluados en la función identidad, esto nos sirve para que en dado caso que el usuario dejara un intervalo abierto de filtro a filtro la función graficada no quedara discontinua en un intervalo. También se hizo uso de una variable de tipo entero como contador de filtros, ya que el usuario solo podrá seleccionar de 1 a 5 filtros, esta variable nos permite saber la cantidad de filtros seleccionados y en que posiciones se encuentran.

Además se hizo uso de las funciones:

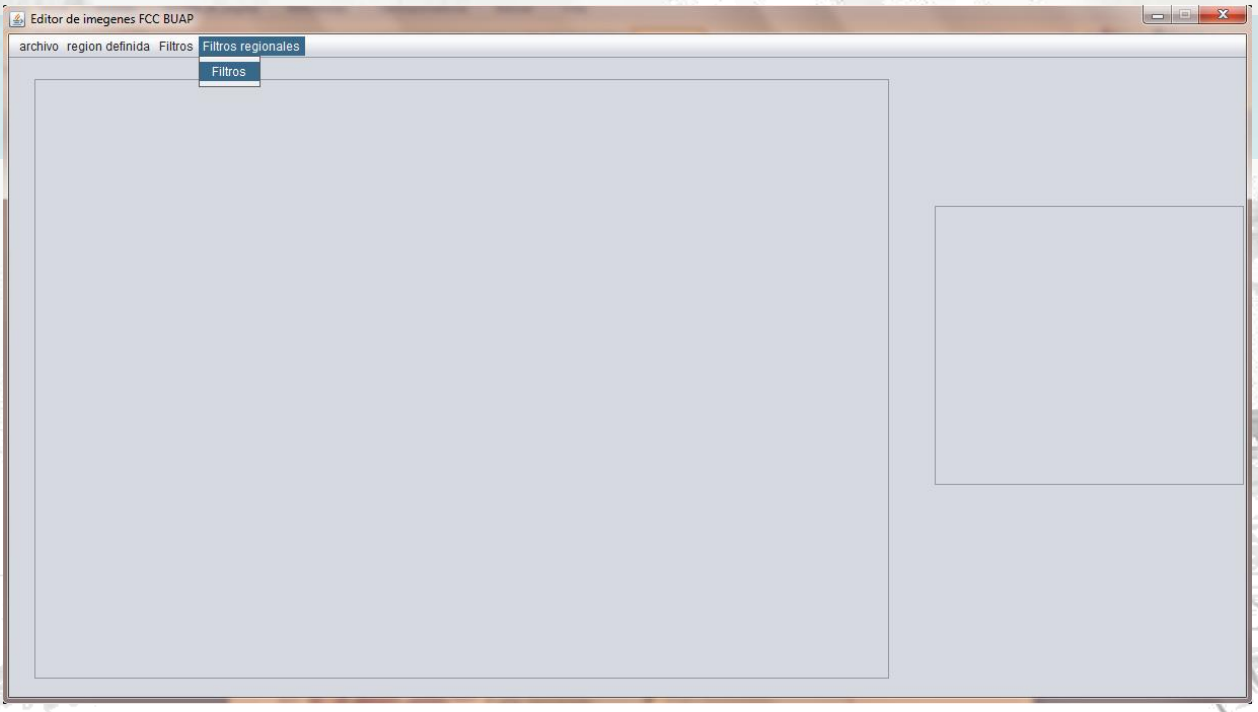
JOptionPane.showMessageDialog (null,"String")

JOptionPane.showInputDialog ("String")

La primera se huso para advertirle al usuario que había sobrepasado la cantidad de filtros permitidos por la aplicación, también se uso en el caso de que el usuario no haya seleccionado ningún filtro, y la segunda se implemento para pedirle al usuario α o γ dependiendo de qué filtro haya seleccionado.

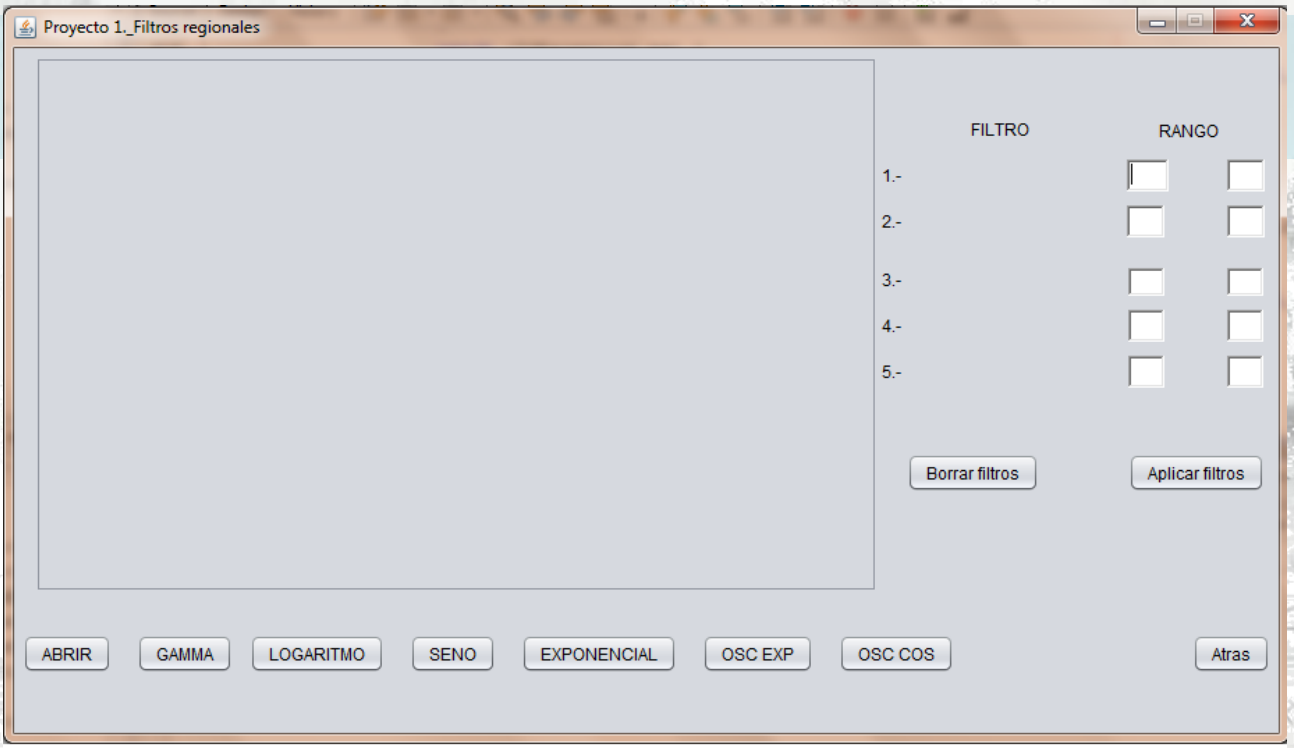
Además de las funciones ya dichas anteriormente también se usaron las funciones incluidas por la librería javax.swing.ImageIcon para el manejo de las imágenes, así como apertura, cierre y manipulación de sus canales.

Bueno ya dicho lo anterior, para el buen uso de la aplicación, lo primero que se tiene que hacer es presionar en la parte que dice filtros regionales, ahí aparcera una opción que dice filtros, para ello le damos clic en cima de este.

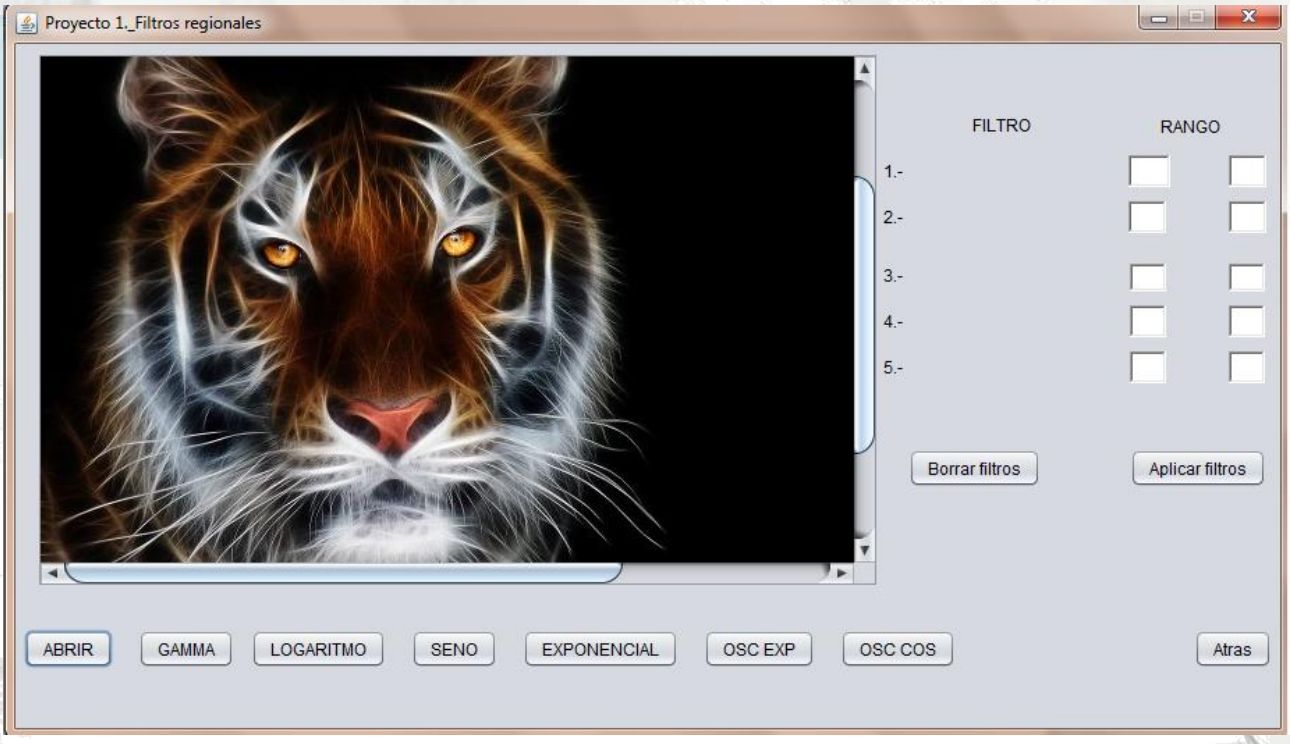


Resultados

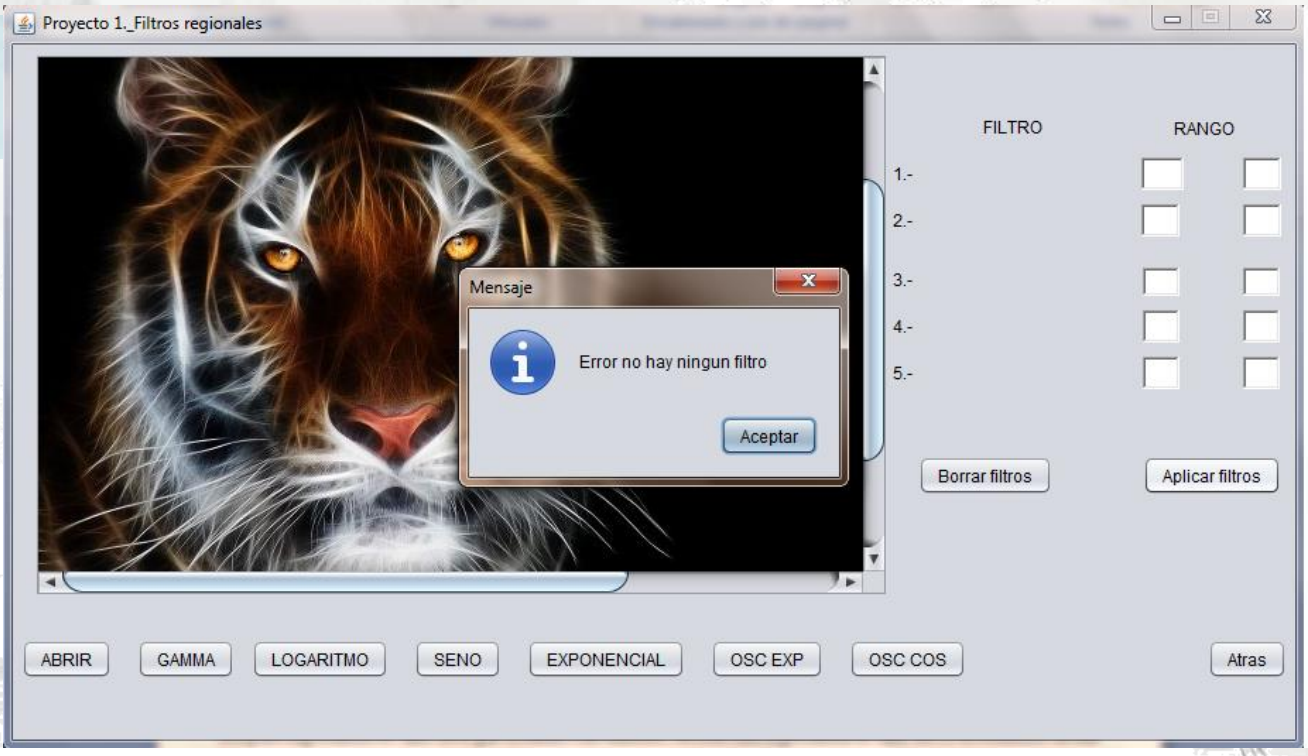
Después de haber seleccionado filtros, la aplicación muestra una ventana con botones , cuadros de texto y una pequeña ventana en el centro con la cual podremos visualizar la imagen ,los botones hacen referencia a los filtros y alas acciones como abrir, atrás , aplicar filtro, borrar filtros.



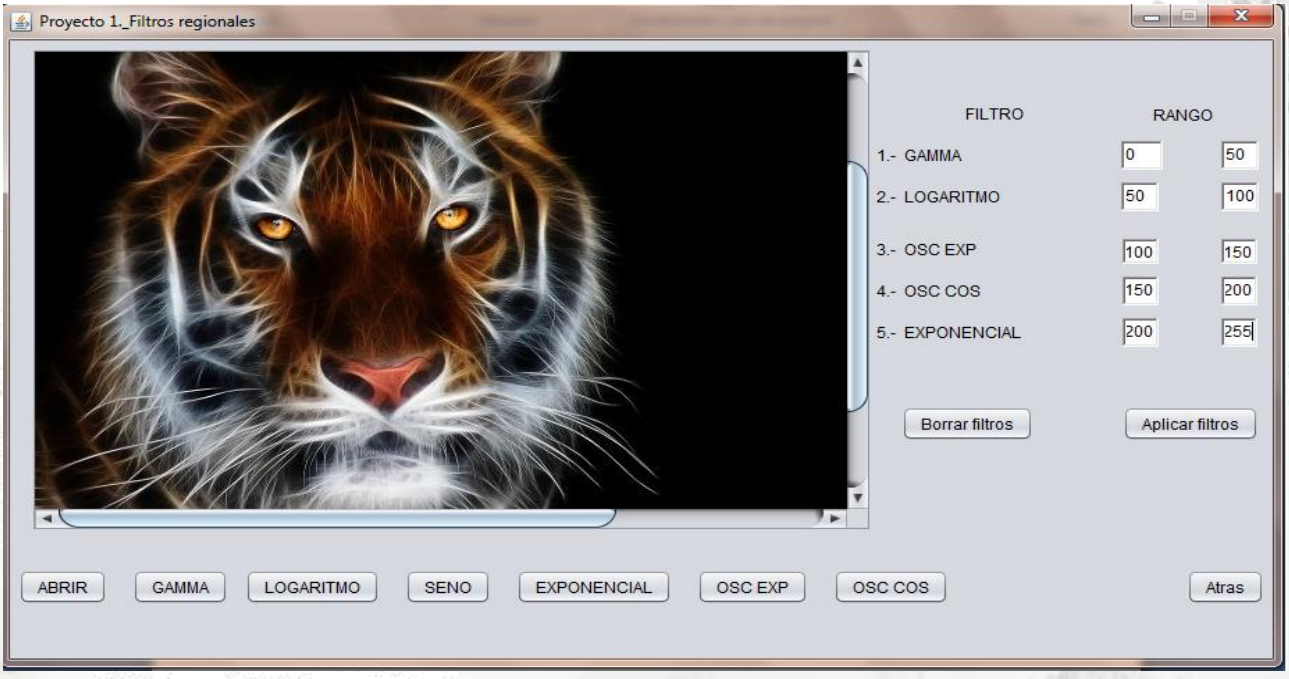
Después de visualizar la ventana anterior el usuario podrá hacer el manejo de los filtros de pendiendo de su necesidad, para ello lo primero que tiene que hacer es dar clic en el botón abrir, el cual nos mostrara una pequeña ventana con la cual podremos buscar la imagen que se desea modificar.



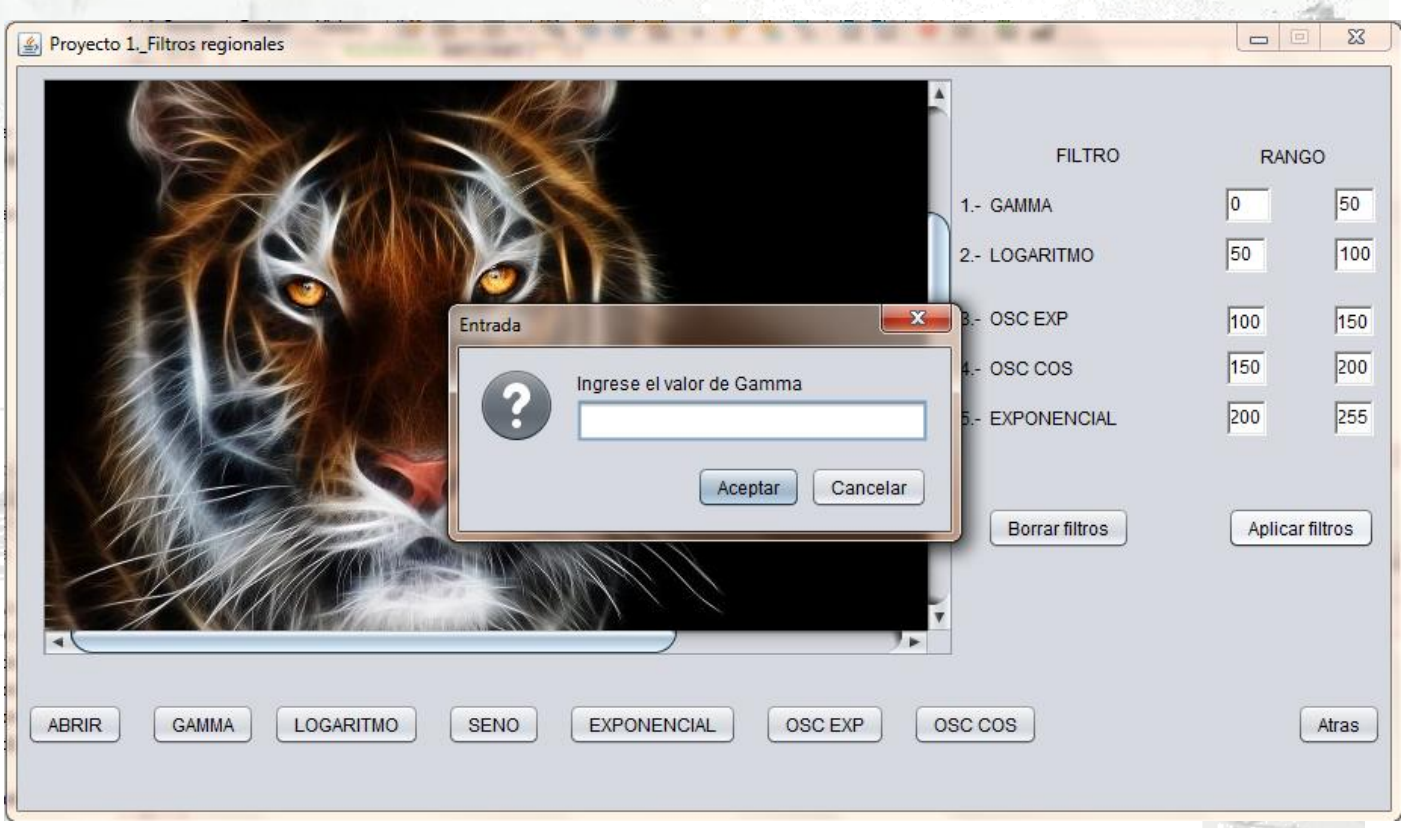
Después de que el usuario haya seleccionado la imagen de su preferencia, el usuario podrá seleccionar los filtros que desee, en dado caso que el usuario seleccione mas de 5 filtros o no seleccione ninguno, le aparcera una ventana con una advertencia.



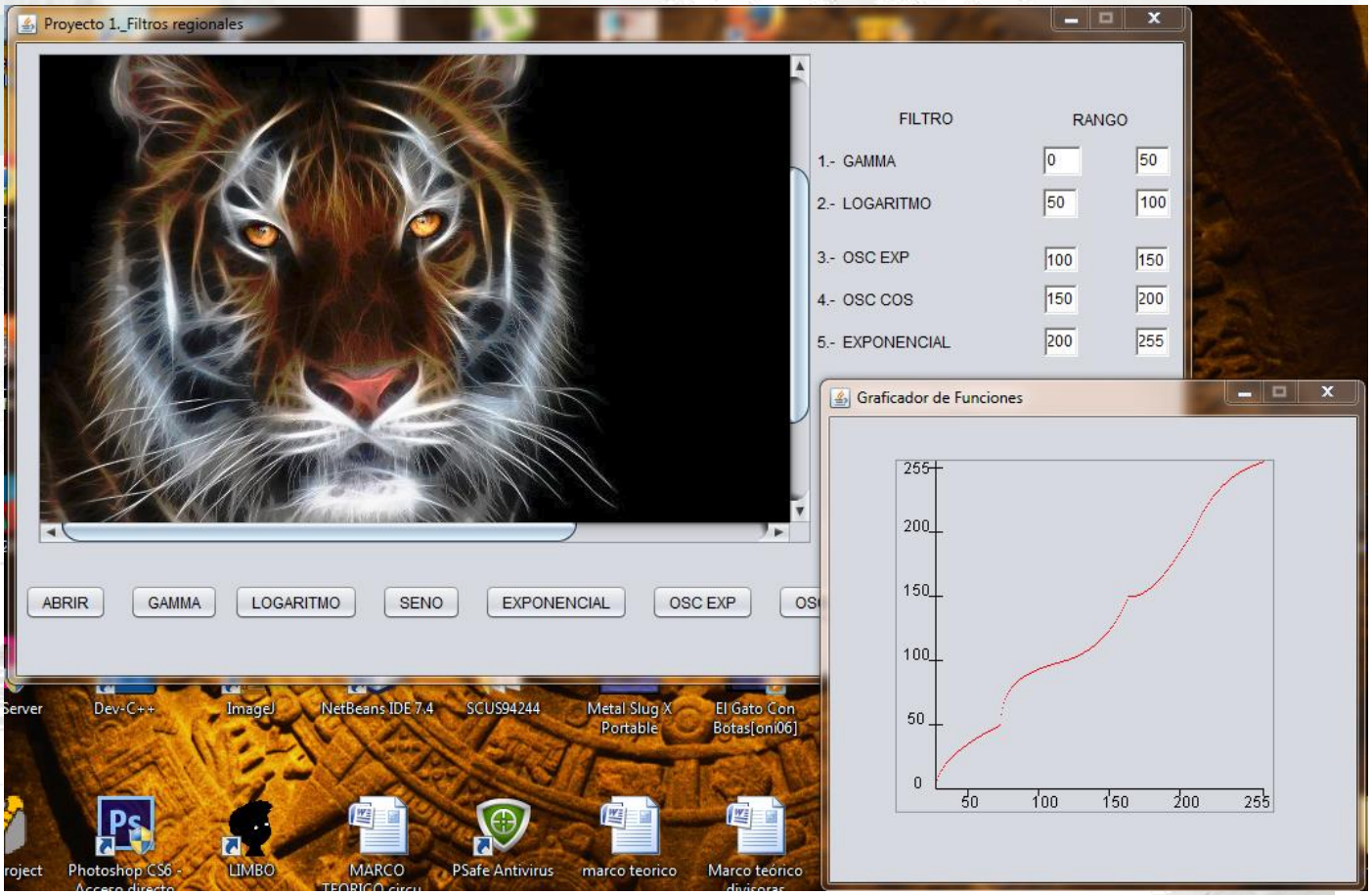
Para poder aplicar los filtros el usuario deberá presionar los botones con los filtros que desee, y ala derecha del filtro seleccionado deberá proporcionar el rango en el cual desea aplicar dicho filtro, por ejemplo:



Después de haber seleccionado los filtros y de haber llenado los campos de rangos, podrá darle clic al botón aplicar filtros, esta acción nos mostrara la ventana siguiente, como se ha mencionado antes dependiendo del filtro nos mostrara una ventana donde nos pide el valor de α o γ, por ejemplo:



Por ultimo cuando el usuario haya terminado de haber ingresados los datos correspondientes los filtros se irán aplicando a la respectiva imagen, después de estos se visualizara una nueva imagen acompañada de una pequeña ventana con la gráfica de las funciones.



Conclusión

Gracias a los filtros regionales las imágenes pueden ser retocadas en ciertos rangos, sin alterar por completo toda la imagen y estos filtros ayudan a un mejor manejo para el balance entre zonas oscuras y zonas calaras, ya que no todos los filtros de oscurecimiento o aclarado lo hacen con la misma intensidad.

Bibliografía

http://www.cs.buap.mx/~iolmos/pdi/

http://www.cs.buap.mx/~mmartin/notas/PDI-MM-Rev.2013.pdf

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Procesamiento Digital de Imágenes

Proyecto:

Aplicación de filtros de aclarado

Alumno:

Everardo Antonio Mendoza
200813080

Introducción

Un filtro es una clase especial de herramienta diseñada para tomar como entrada un pixel, aplicarle un algoritmo matemático, y devolverlo en un formato modificado.
El filtrado consiste en aplicar una transformación (*filtro*) a una imagen digital completa, o a una parte de ella. En el procesamiento digital de imágenes es común el aclarado u obscurecimiento de imágenes mediante la implementación de algunas funciones matemáticas; se podría aumentar en 1 cada uno de los pixeles esperando que la imagen sea más clara (recordar que entre mayor sea el valor del pixel, mas se acercara al color blanco, siendo255 el valor máximo que puede tomar el mismo), o al contrario; se podría esperar quela imagen sea más obscura reduciendo en 1el valor de los pixeles de nuestra imagen.

Entre los filtros más comunes encontramos: Corrección gamma, logarítmica, exponencial, seno y cosenoidal y que serán aplicados en el desarrollo de este proyecto.

Desarrollo

El este proyecto se utilizo el lenguaje de programación Java para implementar cada uno de los algoritmos de filtros que se listan a continuación: gamma, logarítmica, exponencial, seno y cosenoidal.

El programa puede cargar una imagen previamente guardada en la computadora, a través del menú ubicado en la parte superior izquierda de la interfaz. Se puede acceder a este menú con la combinación de teclas Alt+A.

En la interfaz de nuestro programa se muestra la imagen original y la imagen que se va creando a partir de la aplicación de cada filtro para poder comparar los resultados y comprobar si el rango que se esta indicando para aplicar el filtro esta coincidiendo con lo que se esta ejecutando en el programa.

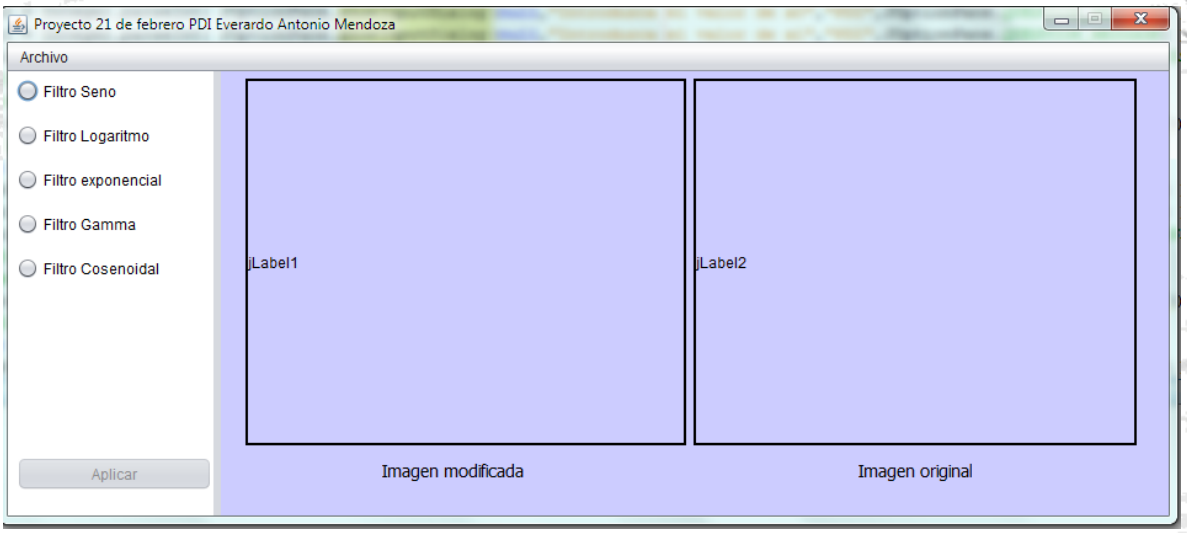


Figura 1. Interfaz del programa

En el lado izquierdo se sitúan los filtros disponibles y que se pueden aplicar a la imagen seleccionada previamente y cargada en la aplicación.

Al utilizar este programa se asume que el usuario tendrá conocimiento de los filtros por lo tanto no se hace una comprobación de valores x0, x1, gamma o alpha, según cada caso, por lo tanto si se ingresan valores equivocados, es probable que el programa no arroje ningún resultado; cabe aclarar que para este tipo de errores el programa simplemente no mostrara resultados, pero se podrá seguir haciendo uso de el y no se congelara en la ejecución.

Se ha evaluado el sistema con distintas pruebas y en todas ellas se han obtenido muy buenos resultados.

Figura 2.Ejemplo de uso.

En la figura 2 se puede ver la aplicación del filtro Seno usando los valores de x0=0 y x1=150, y como se puede ver en la imagen se obtiene un resultado de aclarado muy exacto.

En el correo enviado, se adjunta el código fuente para la comprobación de cada uno de los algoritmos que esta aplicación es capaz de utilizar. Se pretende agregar otros filtros a esta aplicación, para hacerla mas completa.

Al realizar este proyecto se puede observar los distintos efectos que tienen los filtros en una imagen en particular y que si se combinan estos filtros se pueden obtener resultados muy útiles para la aplicación en distintas áreas no solo de la informática.

BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA

FACULTAD DE COMPUTACION

PROCESAMIENTO DIGITAL DE IMÁGENES

“OCULTACION DE DATOS EN IMÁGENES DIGITALES (ESTEGANOGRAFIA)”

PROFESOR: IVAN OLMOS PINEDA

ALLUMNOS: JORGE R. SANCHEZ CASTILLO
DANIEL SORIANO GRANDE

PRIMAVERA 2014

Introducción:

El ser humano siempre ha tenido secretos de muy diversa índole, y ha buscado mecanismos para mantenerlos fuera del alcance de miradas indiscretas. Esta necesidad tiene hoy día una mayor relevancia en la protección de las comunicaciones digitales, donde la investigación y desarrollo de nuevas técnicas ha avanzado notoriamente en las últimas décadas para evitar, o minimizar, ataques de revelación, supresión o alteración de la información intercambiada entre diferentes actores.

Un pilar fundamental en este proceso ha consistido en apoyar la seguridad de los procedimientos implementados en el uso de la criptografía. Esta ciencia tiene entre sus virtudes facilitar servicios de autenticidad, confidencialidad e integridad de la información, pero su problema fundamental recae en la visibilidad de su uso, es decir, las comunicaciones cifradas pueden ser detectadas. De la necesidad de desarrollar nuevos mecanismos para complementar a esta ciencia surge el interés de uso de la esteganografia ya que esta se encargara de ocultar la existencia de la misma comunicación.

El objetivo de este proyecto es usar la esteganografia empleando una técnica donde el medio portador será una imagen digital de tipo .bmp; permitiendo dos cosas:

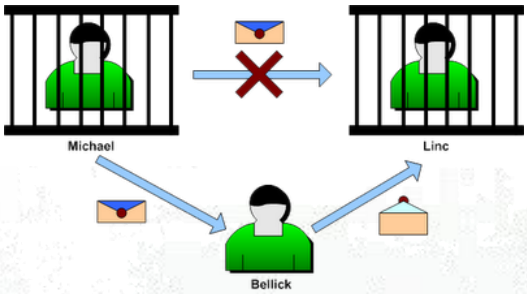
- ocultar un mensaje en la imagen
-mostrar el mensaje que tiene la imagen

Marco Teórico:

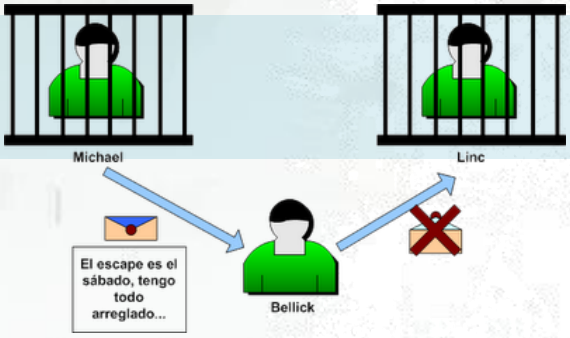
La esteganografía es una solución al clásico problema del prisionero (J. Simpson, 1983), el cual es:

¿Cómo pueden comunicarse dos prisioneros (e. g. para acordar un plan de fuga) si están en celdas separadas y todos los mensajes que intercambian pasan a través de un guardián?

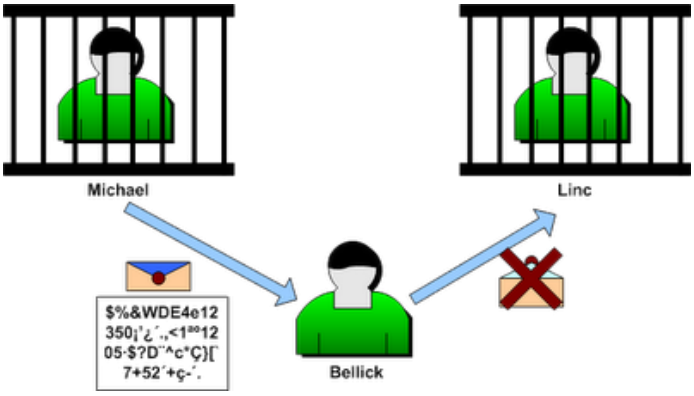
Para hacerlo más interesante a los prisioneros les llamaremos Michael y Linc, mientras que el guardián se llamará Bellick.



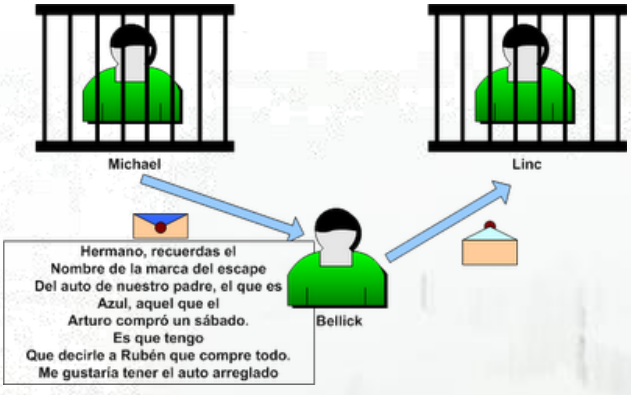
De esta forma, si Michael le manda a Linc el mensaje: "El escape es el sábado tengo todo arreglado...", Bellick descubrirá su plan de escape, lo cual provocaría la destrucción del mensaje



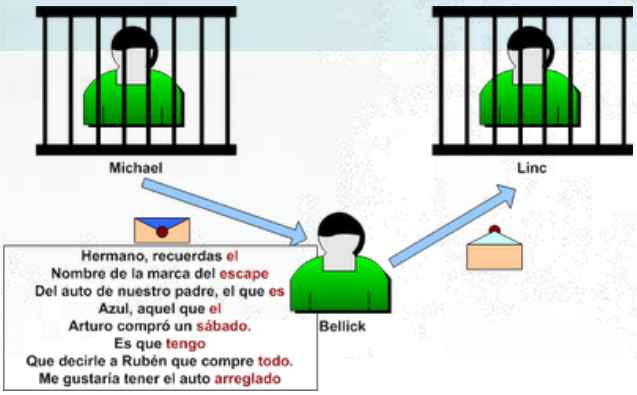
Pero como todos sabemos que Michael es un erudito y conoce de la criptografía entonces lo que hace es cifrar el mensaje que le envía a Linc (claro esta!, Linc conoce como descifrar el mensaje) para que Bellic no lo entienda. Pero Bellick al ver el mensaje cifrado, sospecha y lo destruye, evitando nuevamente su escape.



Otra alternativa que puede tomar Michael es utilizar la esteganografía para esconder el mensaje, pero ahora lo haría de una manera tal que dicho mensaje se escondería dentro de otro mensaje menos importante, es decir un medio portador inocuo. Así, Michael le manda a Linc el siguiente mensaje:



Bellick al leer el mensaje piensa que no hay nada extraño y lo deja pasar a Linc. Sin embargo, esta nota tiene un mensaje oculto el cual se puede observar en la siguiente imagen



Con la esteganografía el guardia inspecciona mensajes aparentemente inocuos que contienen un canal subliminal muy útil para los prisioneros.
Se pueden observar distintos actores implicados en el campo de la esteganografía:

• **Objeto contenedor:**
Se trata de la entidad que se emplea para portar el mensaje oculto. Acudiendo al ejemplo de los mensajes sobre el cuero cabelludo, el objeto contenedor es el esclavo en sí.
• **Estego-objeto:**
Se trata del objeto contenedor más el mensaje encubierto. Siguiendo con el ejemplo, se trata del esclavo una vez se ha escrito en su cuero cabelludo el mensaje y se le ha dejado crecer el pelo.
• **Adversario:**
Son todos aquellos entes a los que se trata de ocultar la información encubierta. En el ejemplo de la prisión, se trata del guardia que entrega los mensajes a uno y otro prisionero. Este adversario puede ser pasivo o activo. Un adversario pasivo sospecha que se puede estar produciendo una comunicación encubierta y trata de descubrir el algoritmo que se extrae del estego-objeto, pero no trata de modificar dicho objeto. Un adversario activo, además de tratar de hallar el algoritmo de comunicación encubierta, modifica el estego-objeto con el fin de corromper cualquier intento de mensajería subliminal.
• **Estegoanálisis:**
Ciencia que estudia la detección (ataques pasivos) y/o anulación (ataques activos) de información oculta en distintas tapaderas, así como la posibilidad de localizar la información útil dentro de la misma (existencia y tamaño).

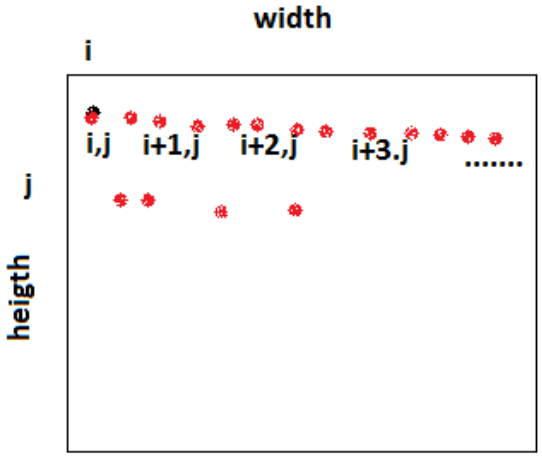
Teniendo en cuenta que pueden existir adversarios activos, una buena técnica esteganográfica debe ser robusta ante distorsiones, ya sean accidentales o fruto de la interacción de un adversario activo.

Metodología:

El método que se empleo es **LSB** (*Least Significant Bit*).
Este consiste en aprovechar el bit menos significativo de cada byte para guardar información en él.

Método tradicional:

El método tradicional de LSB es ir tomando los pixeles de la imagen de manera secuencial.

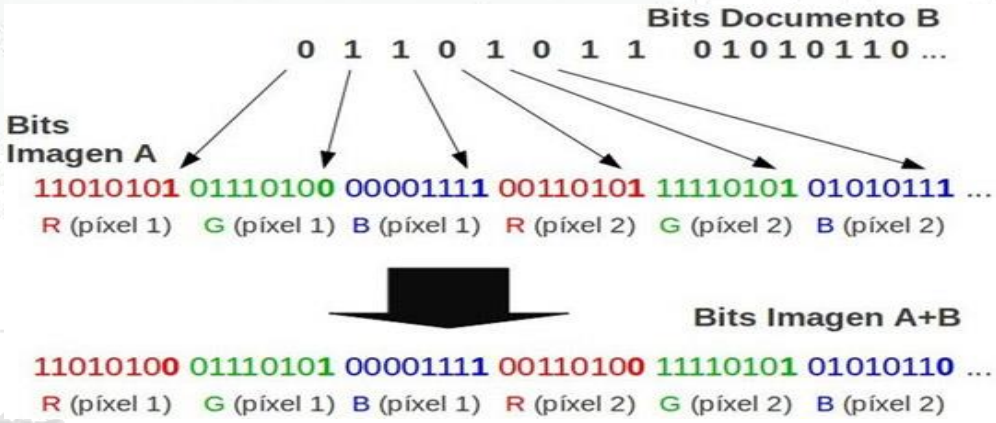


El siguiente código muestra el método ya implementado:

```
For(i=0; i<width; i++){
For(j=0; j<height; j++){

            //extraer el valor del pixel
        color=imagen.getRGB(i,j);

        //extraer el valor de cada canal
            color.getRed();
            color.getGreen();
            color.getBlue();
}
}
```

Para hacer más comprensible la técnica vamos utilizar un ejemplo basándonos en una imagen (llamada "A") en la que queremos ocultar un documento de texto (llamado "B"), creando la imagen "A+B".

Para utilizar esta técnica, lo que se debe hacer es almacenar todos los bits del documento B en los bits menos significativos de cada uno de los colores que componen los píxeles de la imagen A, sustituyendo los de la propia imagen por los del documento. Al ser los bits que proporcionan menos información de color al píxel, los cambios realizados en los colores de la imagen no serán apreciables por el ojo humano

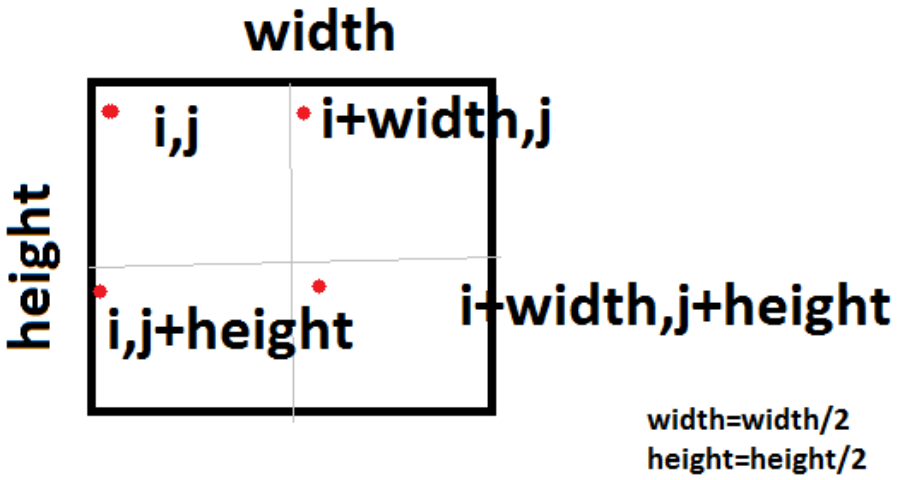


Tal y como se puede apreciar en la imagen anterior, vemos como los bits menos significativos de la imagen A+B contendrán todos los bits del documento B.

Una vez se tiene la imagen A+B, para poder recuperar el archivo oculto, únicamente deberemos acceder a cada uno de los píxeles de la imagen y volcar el contenido del bit menos significativo de cada uno de ellos en un nuevo archivo, obteniendo así el documento B tal y como estaba antes de ser ocultado.

Método Mejorado:

El método mejorado de LSB que nosotros proponemos es ir tomando los pixeles de la imagen de la siguiente manera:



El siguiente código muestra el método ya implementado:

```
W=Width/2
H=Height/2
For(i=0; i<W; i++){
For(j=0; j<H; j++){

        //extraer el valor del pixel
    Color1=imagen.getRGB( i , j);
    Color2=imagen.getRGB( i+W , j );
    Color3=imagen.getRGB( i , j+H );
    Color4=imagen.getRGB( i+W , j+H);

    //extraer el valor de cada canal
       Color1.getRed();
       Color1.getGreen();
       Color1.getBlue();
       Color2.getRed();
       Color2.getGreen();
       Color2.getBlue();
       Color3.getRed();
       Color3.getGreen();
       Color3.getBlue();
       Color4.getRed();
       Color4.getGreen();
       Color4.getBlue();
```

Restricciones:

Un requisito que requiere este método (aplicado a imágenes) es que el archivo a ocultar tiene que ser tres veces más pequeño que el número total de píxeles de la imagen portadora (en la que ocultaremos el archivo). Esto es debido a que para cada 3 bits del documento que queremos ocultar, necesitaremos un píxel de la imagen (suponiendo que el píxel está formado por los tres bytes de RGB).

Por último comentar que para que tanto el método tradicional como el mejorado que nosotros proponemos sean efectivos una vez se ha ocultado el archivo en la imagen, esta no puede ser editada, ya que si se vuelve a codificar/comprimir, la información de los bits cambiaría y por tanto no se podría recuperar el archivo.

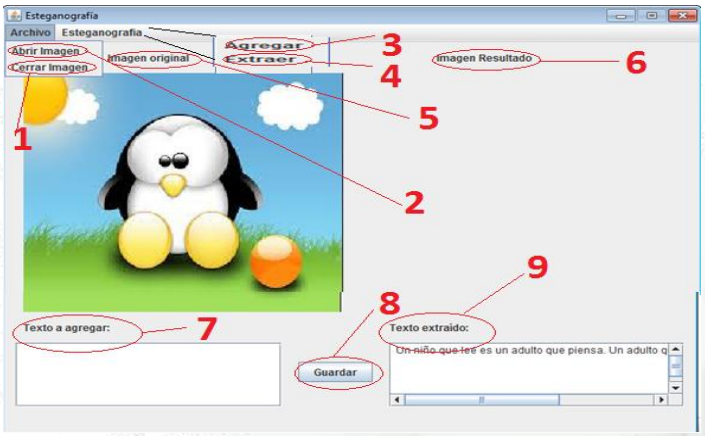
Resultados:

Para las pruebas se utilizaran las siguientes imágenes con extensión .bmp. Esta extensión es de mucha importancia ya que puede guardar imágenes de 24 bits además de que puede darse compresión sin pérdida de calidad (en nuestro caso de información).





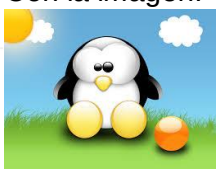
Interfaz del programa:



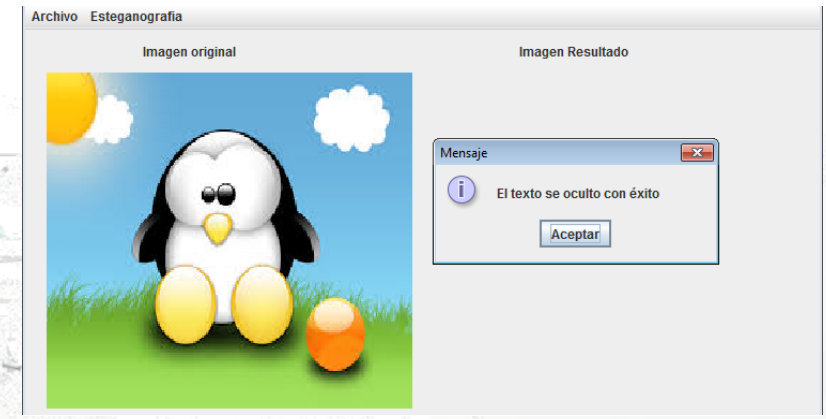
1. **Cerrar Imagen:** Cierra el proyecto actual y abre un nuevo.
2. **Abrir Imagen:** Abre una imagen con la cual queremos trabajar.
3. **Agregar:** Agrega el texto a la imagen (hace el proceso de esteganografia)
4. **Extraer:** Extrae de la imagen actual el texto que tiene oculto.
5. **Imagen Original:** muestra la imagen con la cual vamos a guardar el texto o en otro caso la imagen que le queremos extraer el texto que anteriormente le agregamos.
6. **Imagen Resultado:** Muestra la imagen después de que le hemos agregado texto
7. **Texto a agregar:** En el área blanca que se encuentra debajo escribimos el texto que deseamos ocultar.
8. **Guardar:** Al darle clic guardamos la imagen que ya contiene el texto oculto con extensión .bmp
9. **Texto extraído:** En el área blanca que se encuentra debajo se muestra el texto que se encontraba oculto en la imagen.

PRUEBAS:

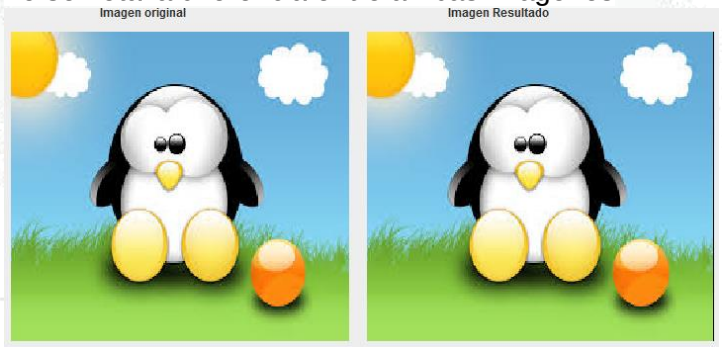
Prueba 1:
Con la imagen:



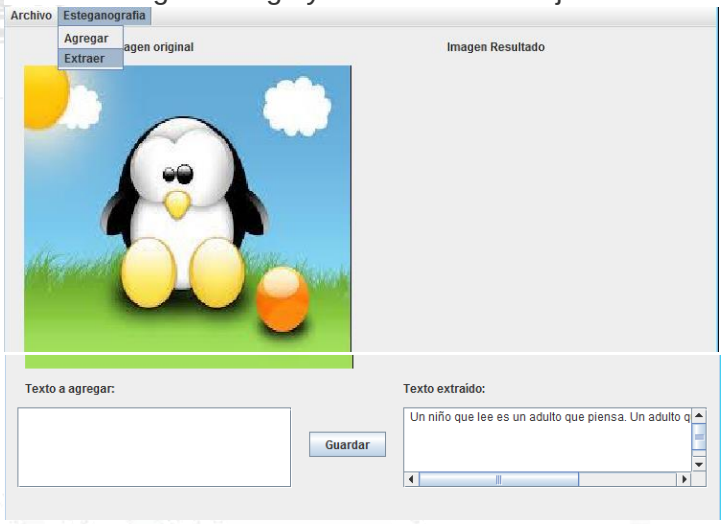
Ocultar el mensaje:
“Un niño que lee es un adulto que piensa. Un adulto que lee es un niño que imagina”.



Como se puede apreciar la imagen original y la imagen resultado que contiene el mensaje oculto no se nota la diferencia entre ambas imágenes



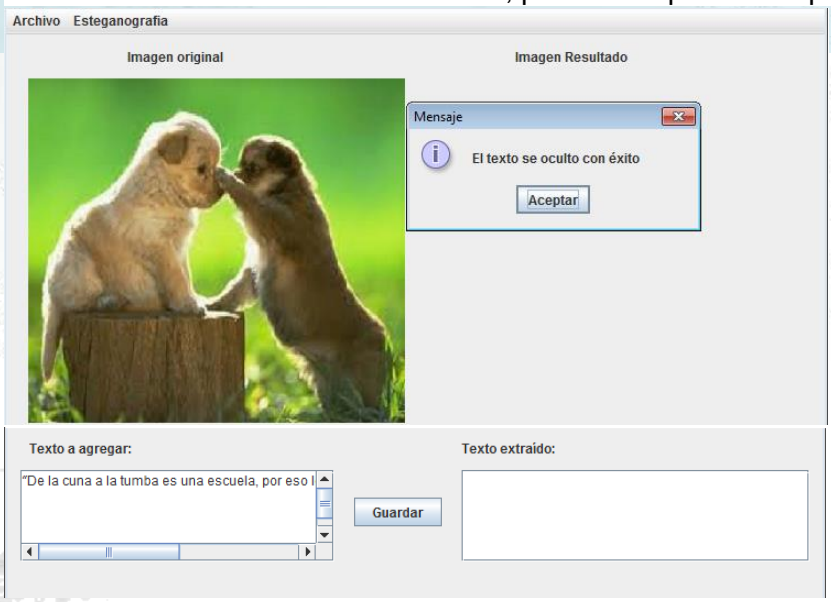
Abrir la imagen estego y mostrar el mensaje oculto:



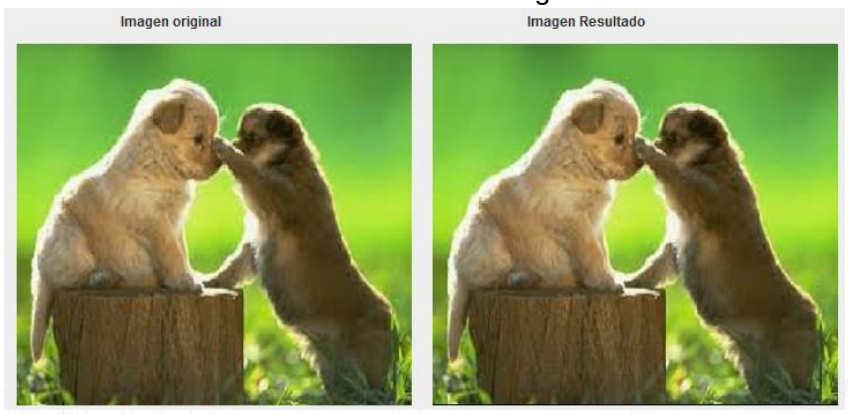
Prueba 2:

Con la imagen:

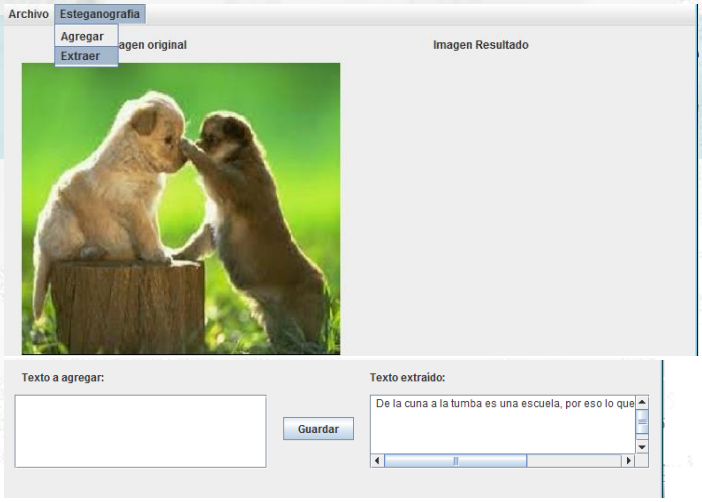
Ocultar el mensaje:
“De la cuna a la tumba es una escuela, por eso lo que llamas problemas, son lecciones.”



Como se puede apreciar la imagen original y la imagen resultado que contiene el mensaje oculto no se nota la diferencia entre ambas imágenes

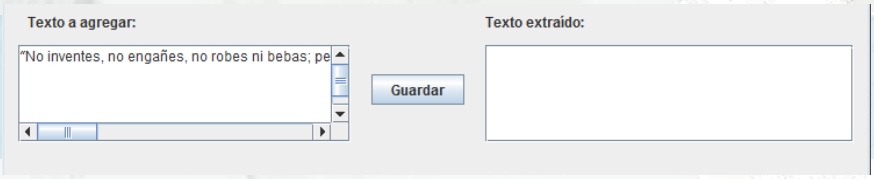


Abrir la imagen estego y mostrar el mensaje oculto:

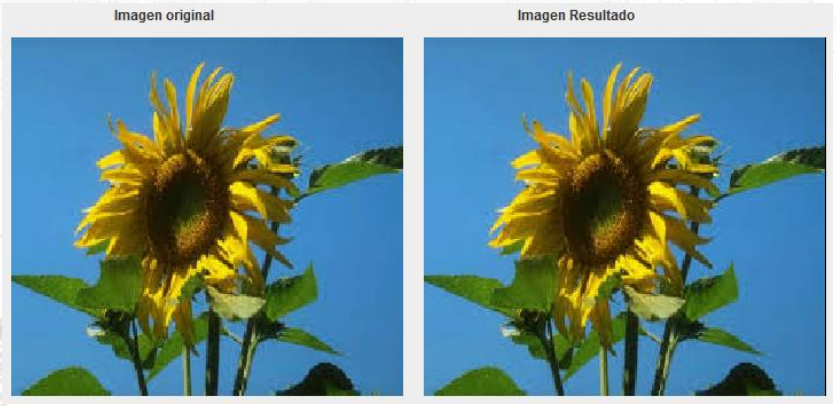


Prueba 3:
Con la imagen:

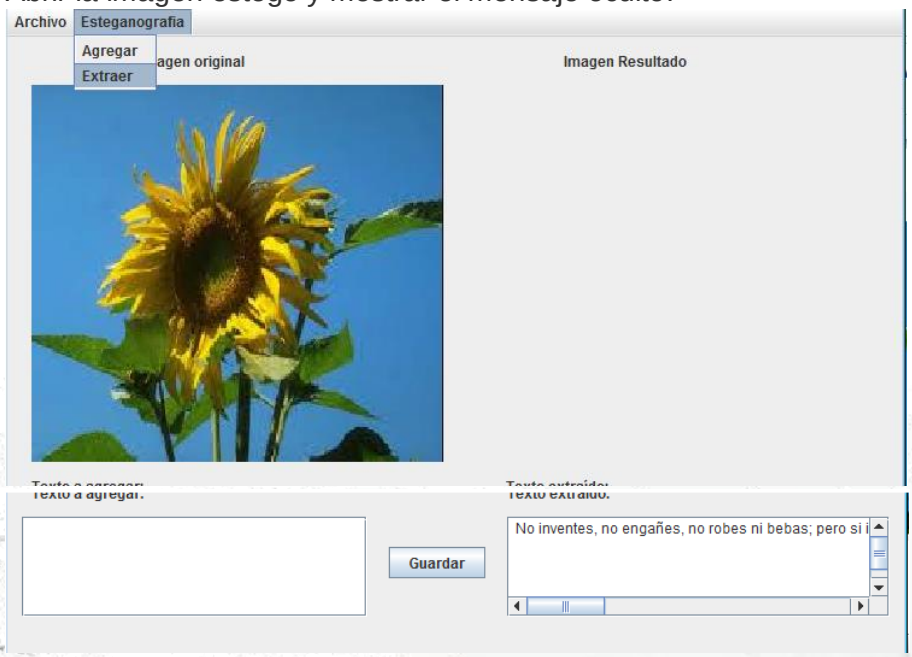
Ocultar el mensaje:
“No inventes, no engañes, no robes ni bebas; pero si inventas, invéntate un mundo mejor; si engañas, engáñale a la muerte; si robas, róbate un corazón y si bebes, bébete los mejores momentos de tu vida.” Película ‘Hitch’.



Como se puede apreciar la imagen original y la imagen resultado que contiene el mensaje oculto no se nota la diferencia entre ambas imágenes



Abrir la imagen estego y mostrar el mensaje oculto:

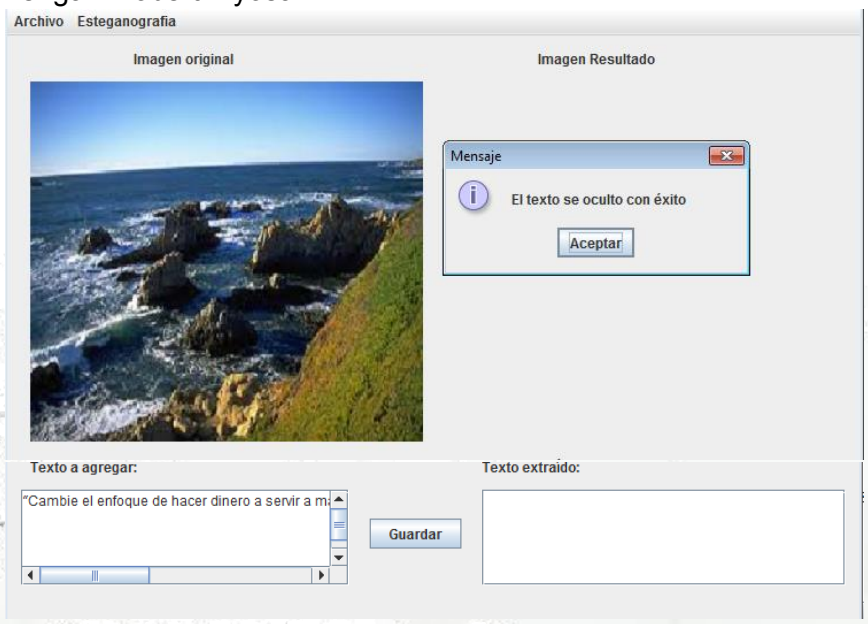


Prueba 4:

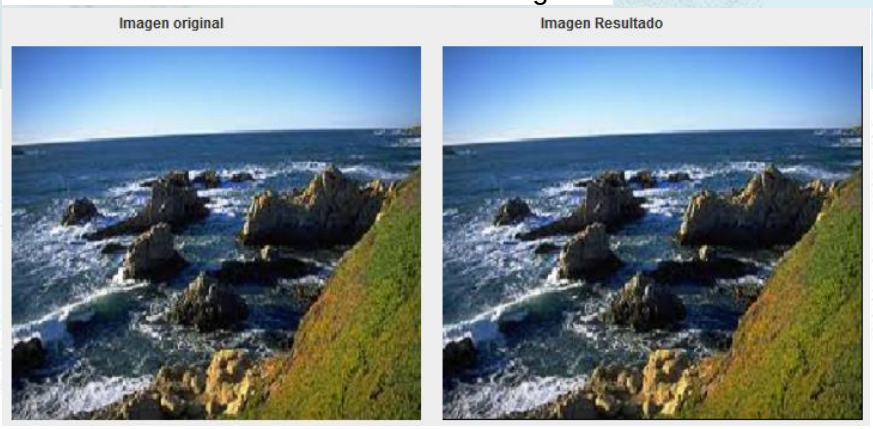
Con la imagen:

Ocultar el mensaje:

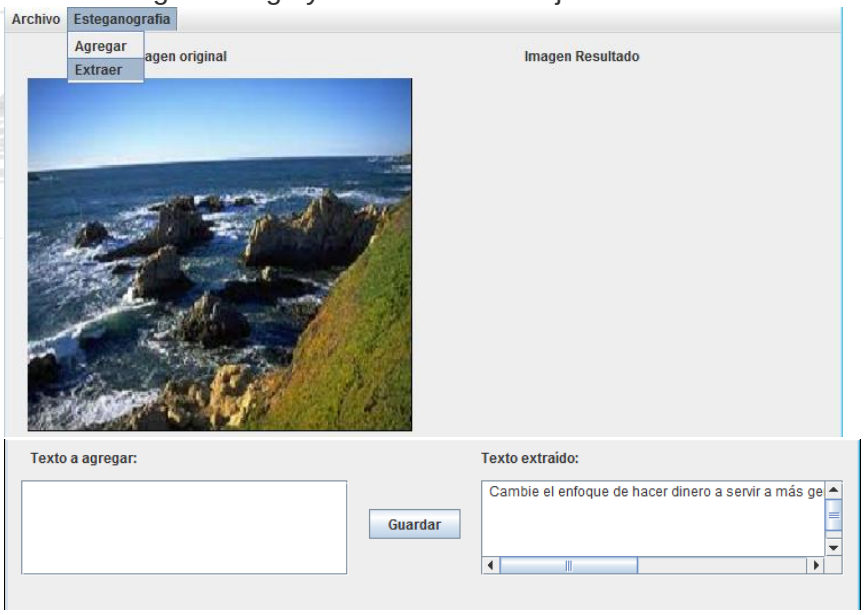
“Cambie el enfoque de hacer dinero a servir a más gente. Servir a más gente hará que el dinero venga.” Robert Kiyosaki.



Como se puede apreciar la imagen original y la imagen resultado que contiene el mensaje oculto no se nota la diferencia entre ambas imágenes



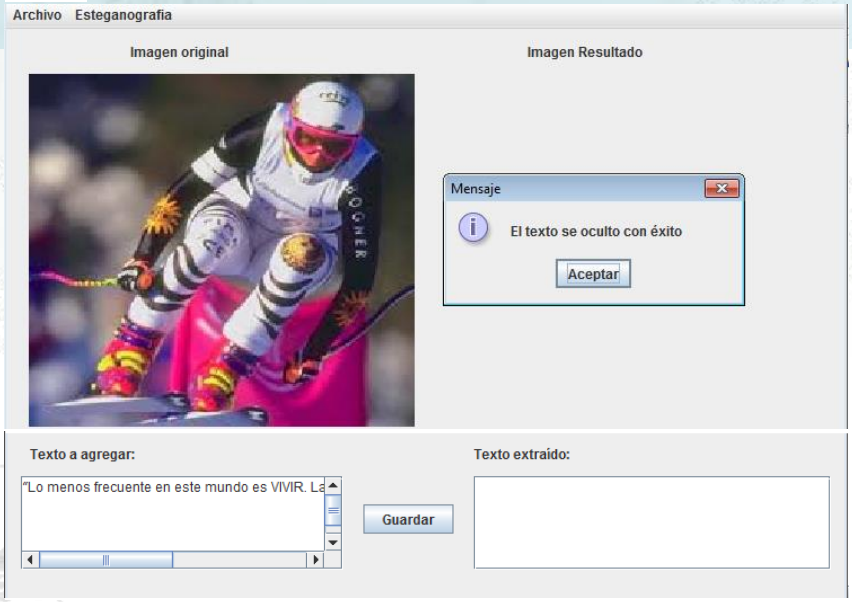
Abrir la imagen estego y mostrar el mensaje oculto:



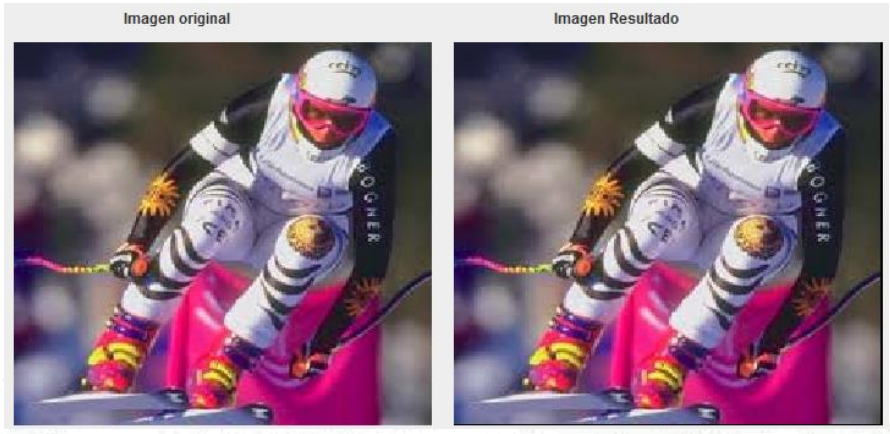
Prueba 5:

Con la imagen:

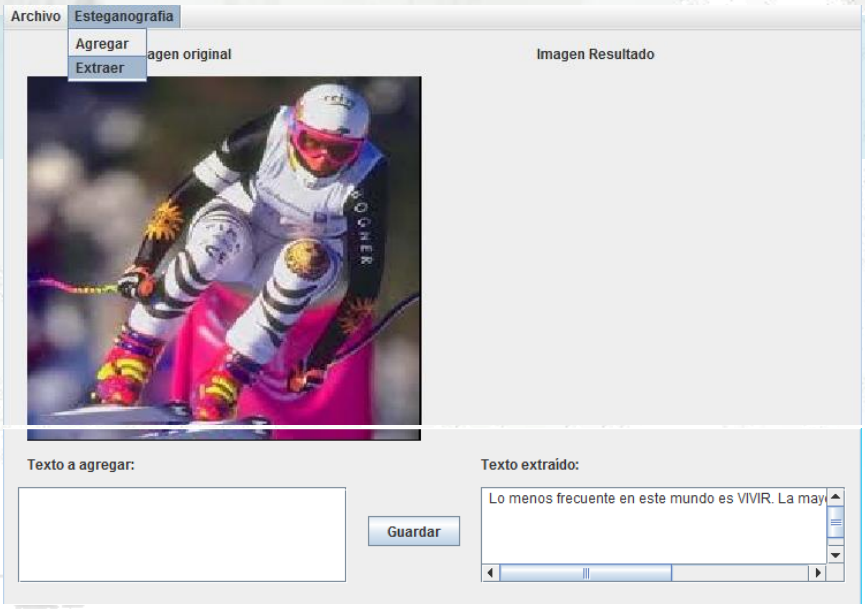
Ocultar el mensaje:
“Lo menos frecuente en este mundo es VIVIR. La mayoría de la gente solo EXISTE.” Oscar Wilde.



Como se puede apreciar la imagen original y la imagen resultado que contiene el mensaje oculto no se nota la diferencia entre ambas imágenes



Abrir la imagen estego y mostrar el mensaje oculto:



Conclusión:

La esteganografía es una técnica en constante evolución, con una larga historia y con capacidad para adaptarse a nuevas tecnologías. A medida que las herramientas de esteganografía se hacen más avanzadas, las técnicas y las herramientas empleadas en el estegoanálisis también se hacen más complejas.

Pero las aplicaciones de esta ciencia no sólo se restringen al ámbito de lo poco ético, pudiendo ayudar en campos como la medicina, protección de menores, etc.

En cualquier caso, la eficiencia de las nuevas técnicas de estegoanálisis hace necesario el uso de la esteganografía combinada con criptografía con el fin de alcanzar un nivel de seguridad razonable. La criptografía garantiza la confidencialidad de una conversación pero no esconde el hecho de que dicha conversación se está manteniendo. Por otra parte, la esteganografía en solitario puede ocultar el hecho de que una conversación se mantiene, pero una vez descubierta la interacción, es posible que un atacante conozca el contenido intercambiado. Aun cuando descubrir el contenido original fuera difícil, un atacante puede modificar el estego-objeto para impedir la comunicación (ataque activo).

Conjugando ambas técnicas se alcanza una complementariedad que multiplica la seguridad de un intercambio de mensajes.

Bibliografía:

http://www.securityartwork.es/2012/02/24/ocultando-archivos-en-otros-lsb/

http://books.google.com.mx/books?id=R_xFMGIFksoC&pg=PA101&lpg=PA101&dq=metodo+lsb&source=bl&ots=5diCc8D1sG&sig=1FJFNJ6TwuZweCosmB0X1qalzaQ&hl=es&sa=X&ei=zFFPU7uUC9SvyATmnYL4Ag&ved=0CCcQ6AEwADgK#v=onepage&q=metodo%20lsb&f=false
http://revistasuma.es/IMG/pdf/57/023-029.pdf
http://www.uv.es/eees/archivo/IE07_061.pdf
http://www.taringa.net/posts/info/2253257/Esteganografia-vs-Criptografia.html
http://www.criptored.upm.es/crypt4you/temas/privacidad-proteccion/leccion7/leccion7.html
https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRKov_V4DNx2cVBLch2c0QfYpD5sh7BDeyRlTIU9WjPmJblMmGC
https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTEjRBJRj2ZXG734QFe4t0KEZeFHi6k6unAnRACL6_fDOmXkZX5lQ
https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcQ3kkHbEcx0m6d_6u7S0iU35phvLezor7lp_-vf5-aaL3dB6LyX

Preprocesamiento de imágenes de textos antiguos

Procesamiento digital de imágenes

Facultad de Ciencias de la Computación

Introducción:

El procesamiento de textos antiguos es una tarea obligatoria para el desarrollo de muchas actividades que influyen en el desarrollo de la cultura, por lo tanto la recopilación de de documentos y su respectivo almacenamiento es una tarea que puede ser asistida por computador. Por lo tanto se propone la siguiente técnica para el preprocesamiento de los textos.

Metodología:

El gracias a los aportes de los artículos tenemos la teoría:

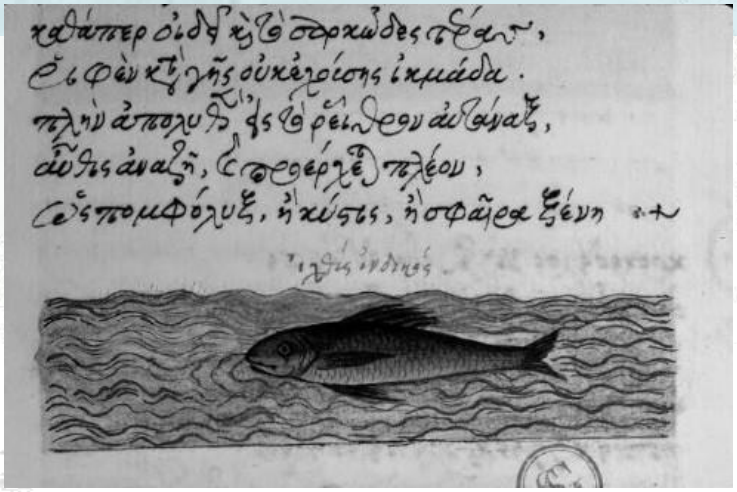
Paso 1:

Convertir la imagen a escala de grises usando la técnica de asignarle un porcentaje del valor de los canales al nuevo valor en escala de grises:

rojo al 30%

verde al 59%

azul al 11%



Paso 2:

Aplicar el filtro de wiener a la imagen usando la siguiente fórmula:

Determinar el tamaño de una ventana de pixeles, usando la fórmula:

v = (ancho/7)/(alto/10)

u = (v/2)

Dónde ancho y alto son los valores de la imagen, u limita el nuevo rango que tomara el análisis de la imagen, en el cual se revisarán en alto y ancho de la imagen, menos el valor de u tanto en ancho como en alto.

$$I_F(x,y) = \mu + \frac{(\sigma^2 - v^2) * (I_s(x,y) - \mu)}{\sigma^2}$$

Se emplea la fórmula, donde las variables son:

μ : es la media local de la ventana
σ^2 : es la varianza en la ventana

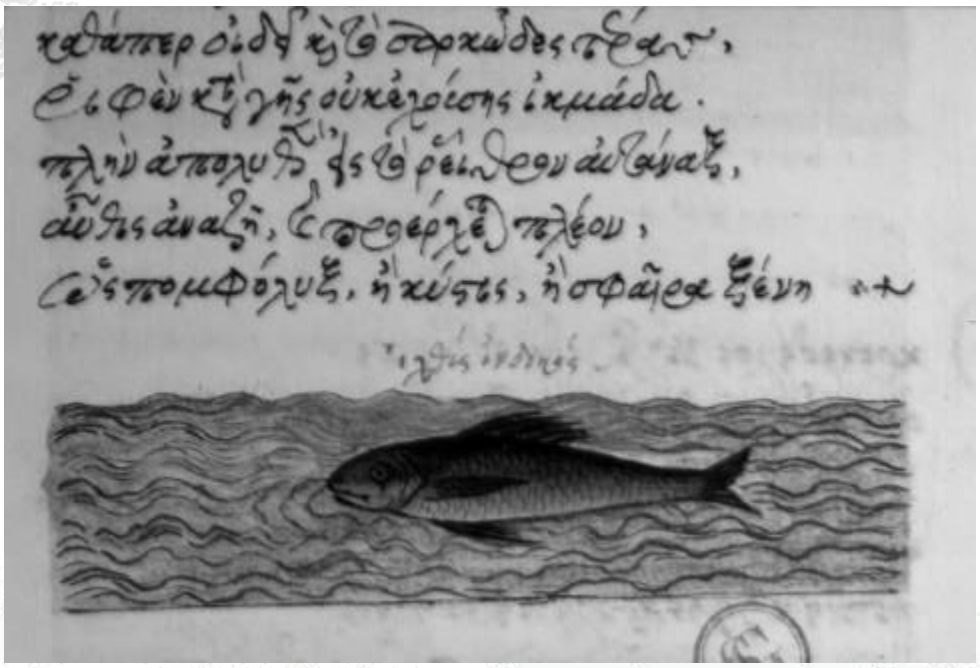
v^2: es el promedio de todas las varianzas para cada pixel en la ventana

Is : es el valor del pixel actual

Por último se le asigna al píxel el valor de:

$(I_F(x,y)*0.6+I_s(x,y)*0.4)$

Con el objetivo de preservar los valores originales en un 40% para no generar alteraciones demasiado grandes.



Paso 3:

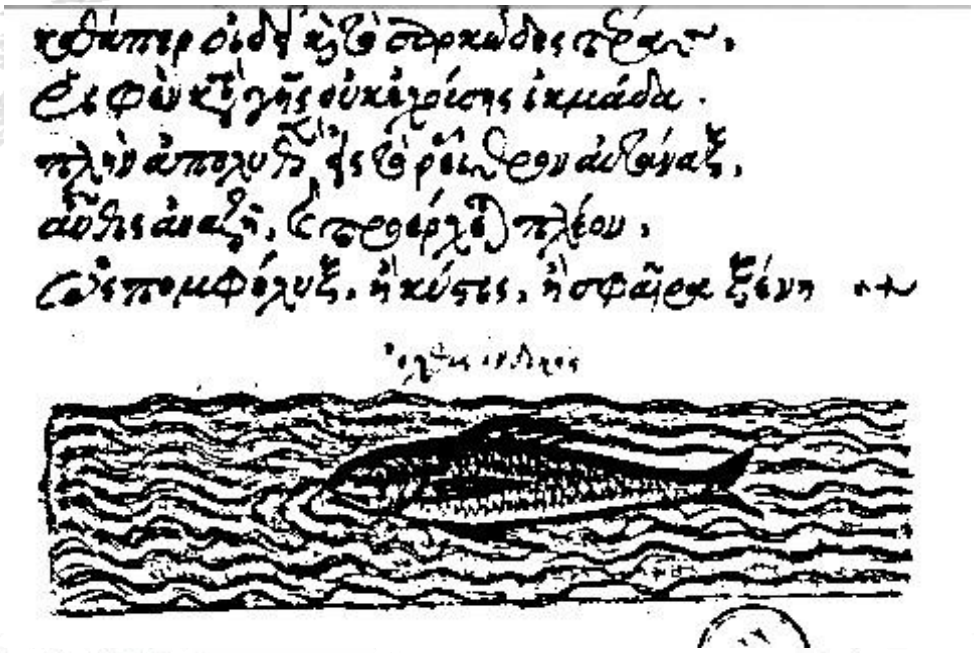
Utilizando el umbral propuesto por Sauvola y Pietikainen, con la fórmula:

$$T(x,y) = m(x,y) \cdot \left[1 + k \cdot \left(\frac{s(x,y)}{R} - 1 \right) \right]$$

Donde m(x,y) es la media y s(x,y) es la desviación estándar en una ventana de previamente calculada con la fórmula :

ventana = (alto/(ancho/10))

Con ancho y alto como valores de la imagen, k tiene un valor en el rango de 0.1 a 0.9 y R un valor de 128 en la escala de grises.



Resultados:

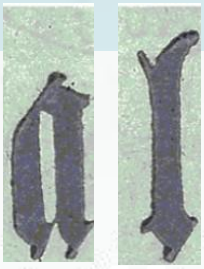
Las imágenes pueden ser clasificadas como BUENAS, REGULARES y MALAS, donde:

Buenas: Tienen un color de fondo uniforme.

Regulares: Tienen un degradado en el color de fondo y los colores de las letras son distintos

Malas: Tienen diferentes colores de fondo y las letras son de un color muy parecido al fondo.

Imágenes buenas:



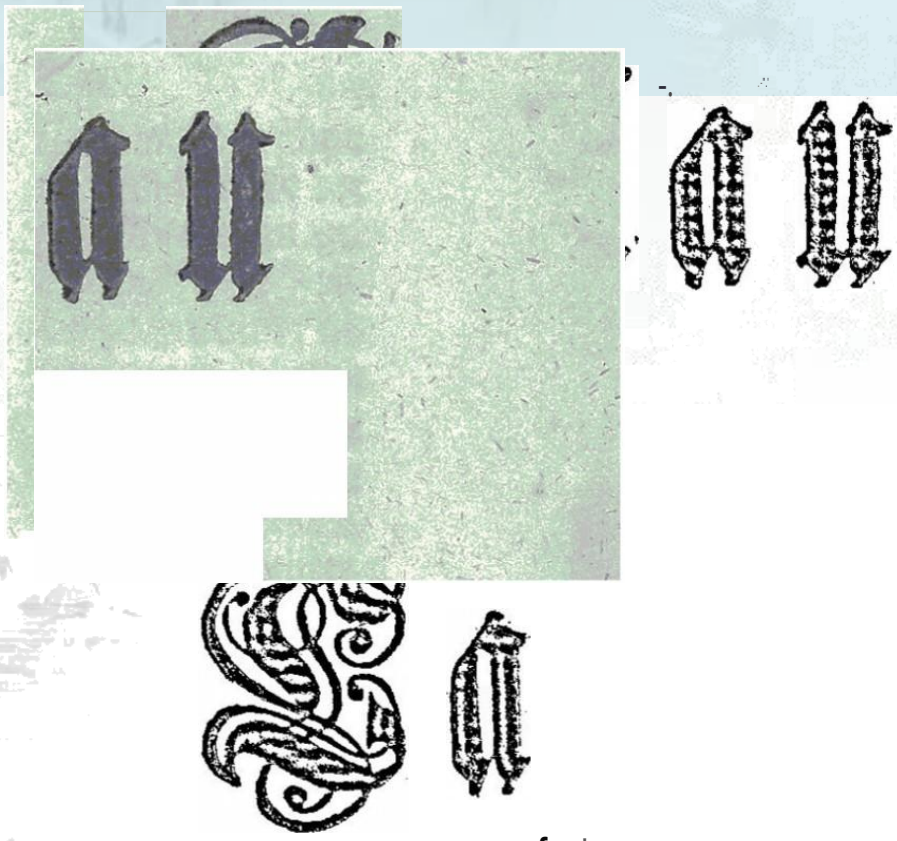
in

~)1aroora~bum *91*

benebff

~eren Statuten., Receílen

unb anbem aíren Ur

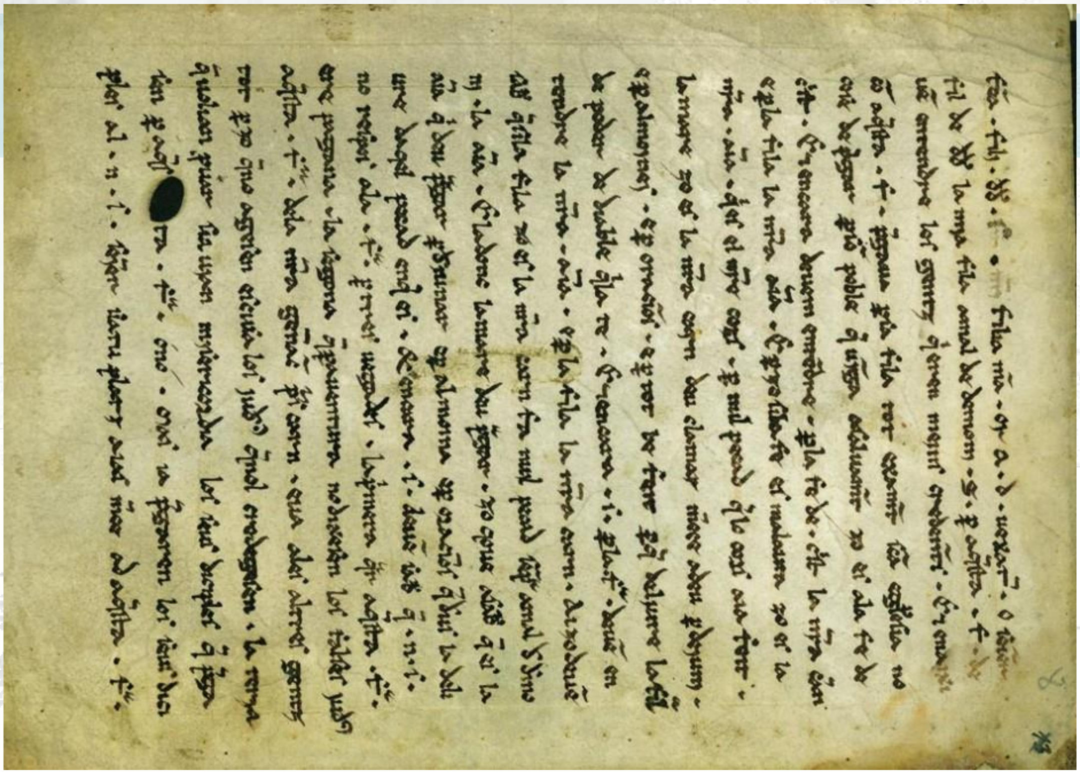


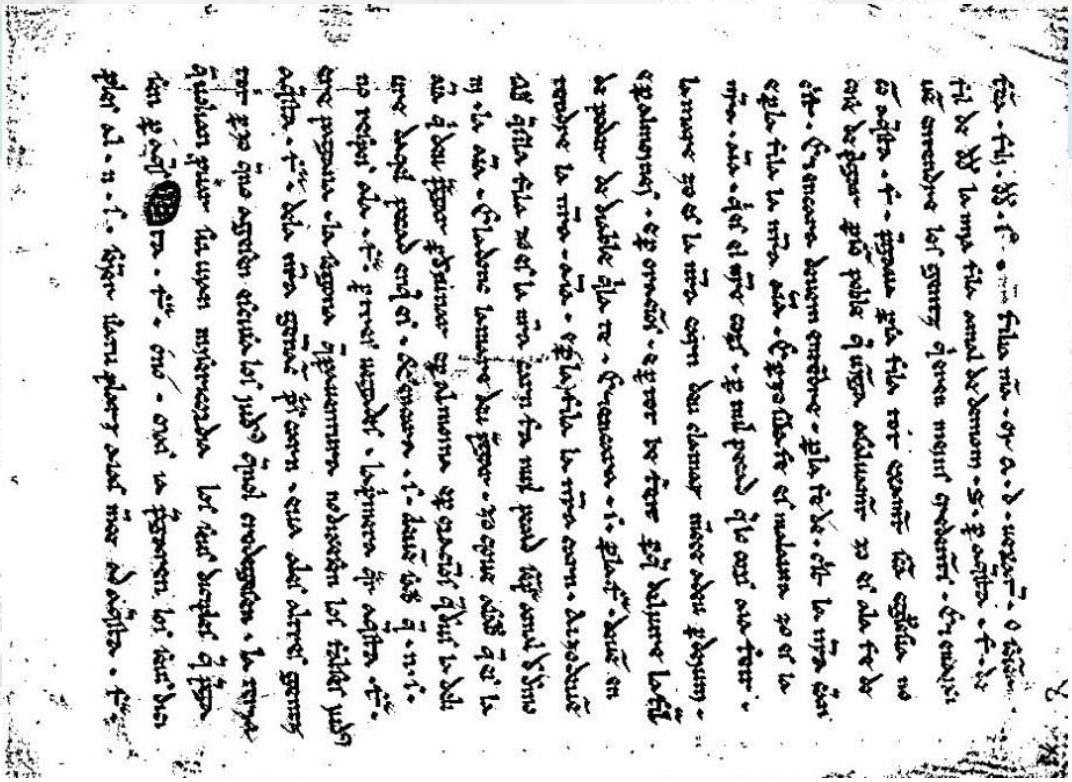
fm'

~J?arggra~bum *91itbtt,2aup~,*

btnebfl

~trtsi Statuten , Receflen, Privilegien,
unb cnbem arten Urfu~en.

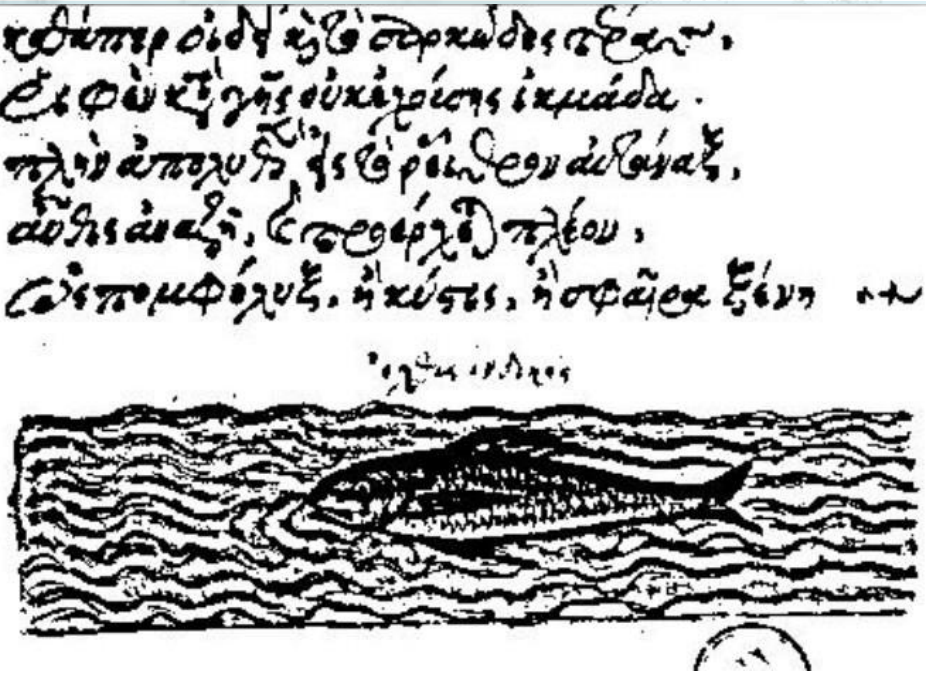




(Original arriba).

Tienen poco ruido y presentan en los caracteres, que se muestran bien definidos

Imágenes regulares:



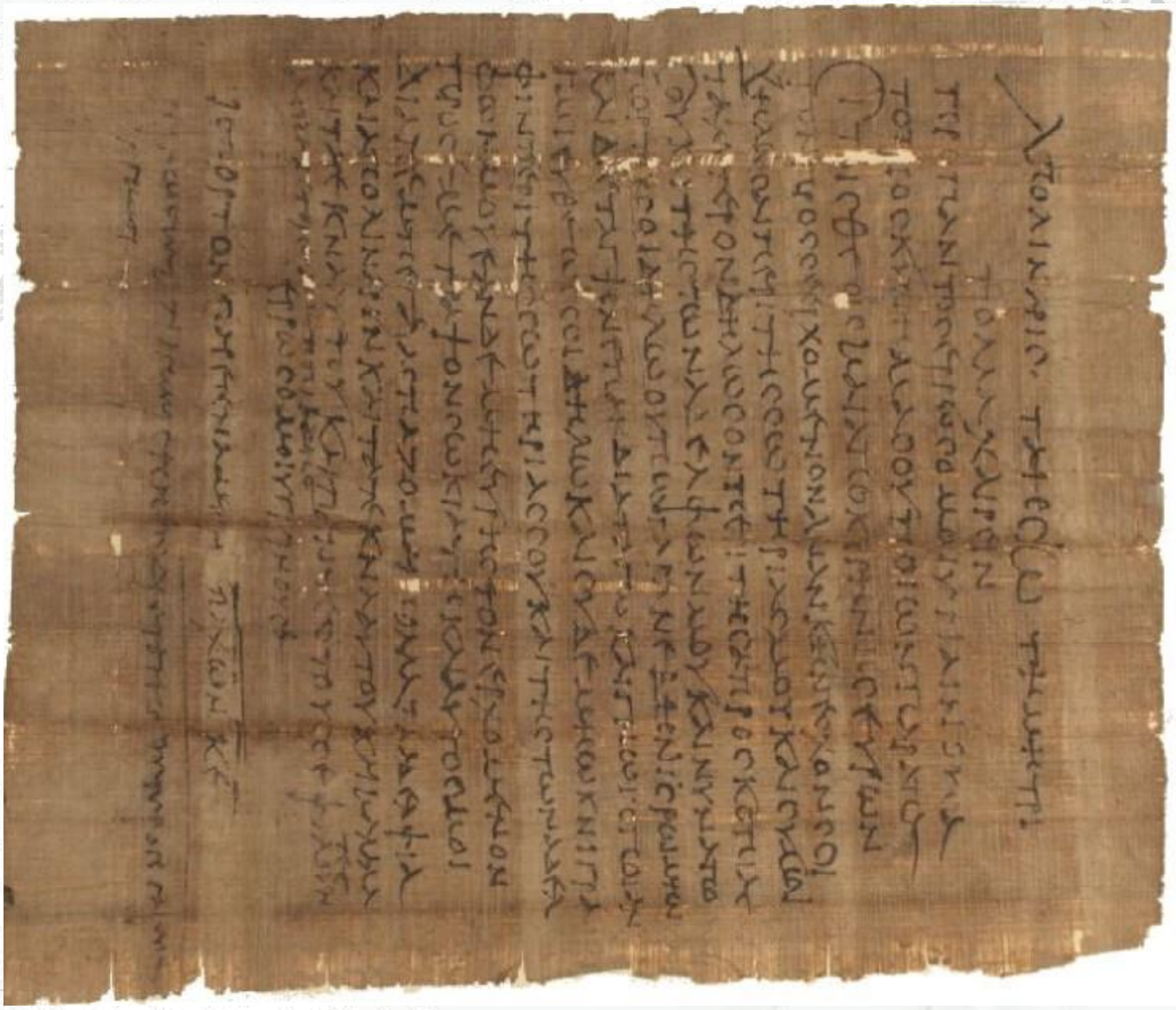
Preclarissimus liber elementorum Euclidis per
eacissimi in artem Geometrie incipit quãfoelicissi

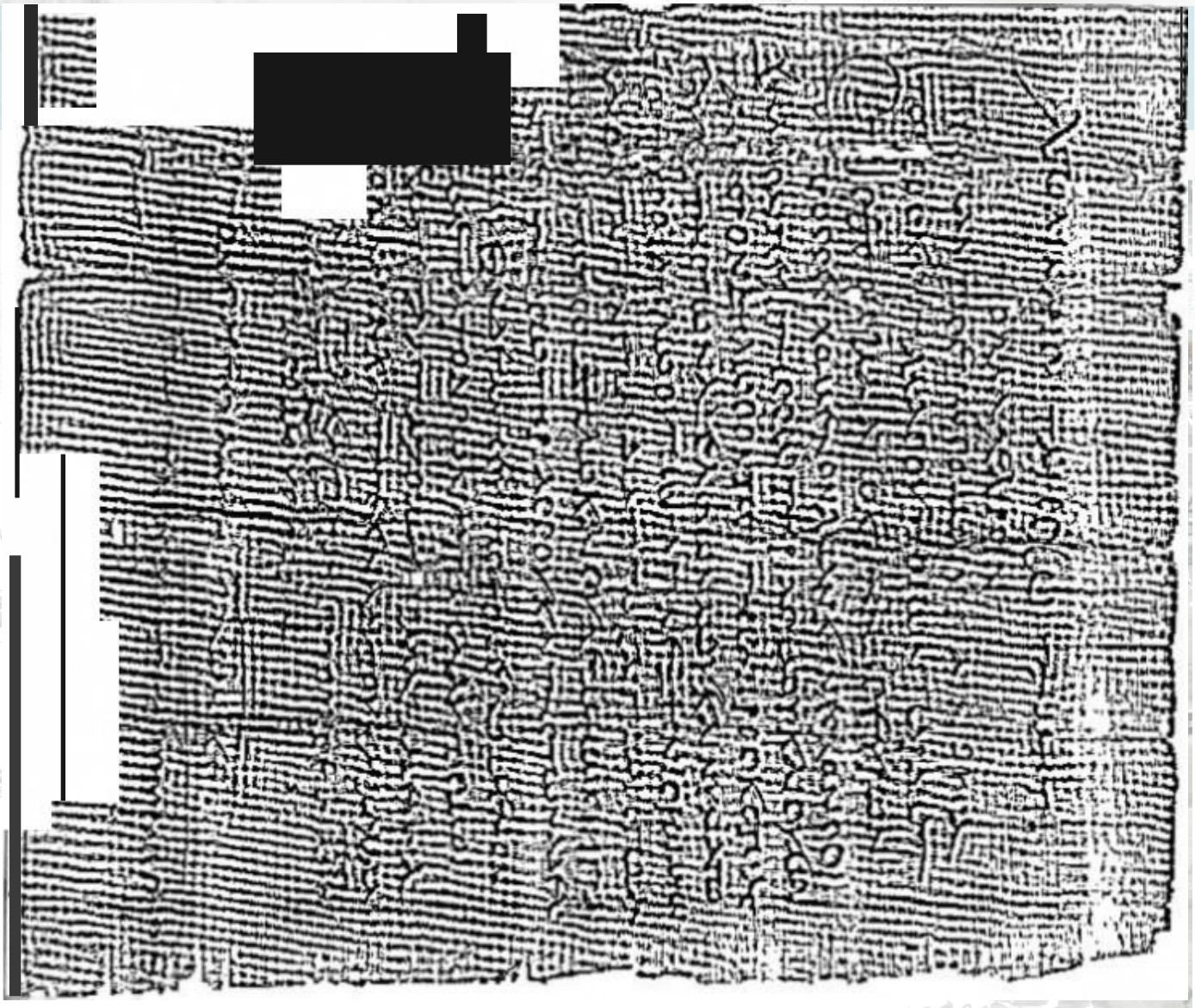
Punctus est cuius ps nõ est. C
lõgitudo siue latitudo cui?
remitates sit duo pũcta. C
ẽ ab vno pũcto ad aliũ bre
sio i extremitates suas vtrũq
pians. C Superficies ẽ q lõgitu
tudinẽ tm̃ habet. termi quidẽ
C Superficies plana ẽ ab vna
liã extẽsio i extremitates sua
C Angulus planus ẽ duarũ li
rarum tactus: quarũ expãsio
ficiẽ applicantoq; nõ directa. C Quãdo aũt angulum
linee recte rectilineũ angulus notat. C Nõ recta linea
sterit duoq; anguli vtrobiq; fuerit coles: coꝝ vter
C Linea q; linee supstãs et cui ipsam ppendicularis vo
sulus vo qui recto maior ẽ obtusus dicit. C Angul⁹ vo
to acut⁹ appellat. C Terminꝰ ẽ qd vnius cuiusq; finis ẽ
ẽ q tamen vltimus ptinet. C Circul⁹ ẽ figura plana vna
nea ptẽa: q circũferentia nõiatur cui⁹ medio pũct⁹ ẽ
linee recte ad circũferẽtiã exẽtes sibiinices sut equale
quidẽ pũct⁹ centrũ circuli dr. C Diameter circuli ẽ linea
lup ei⁹ centꝝ trãsiens extremitatesq; suas circũferẽ

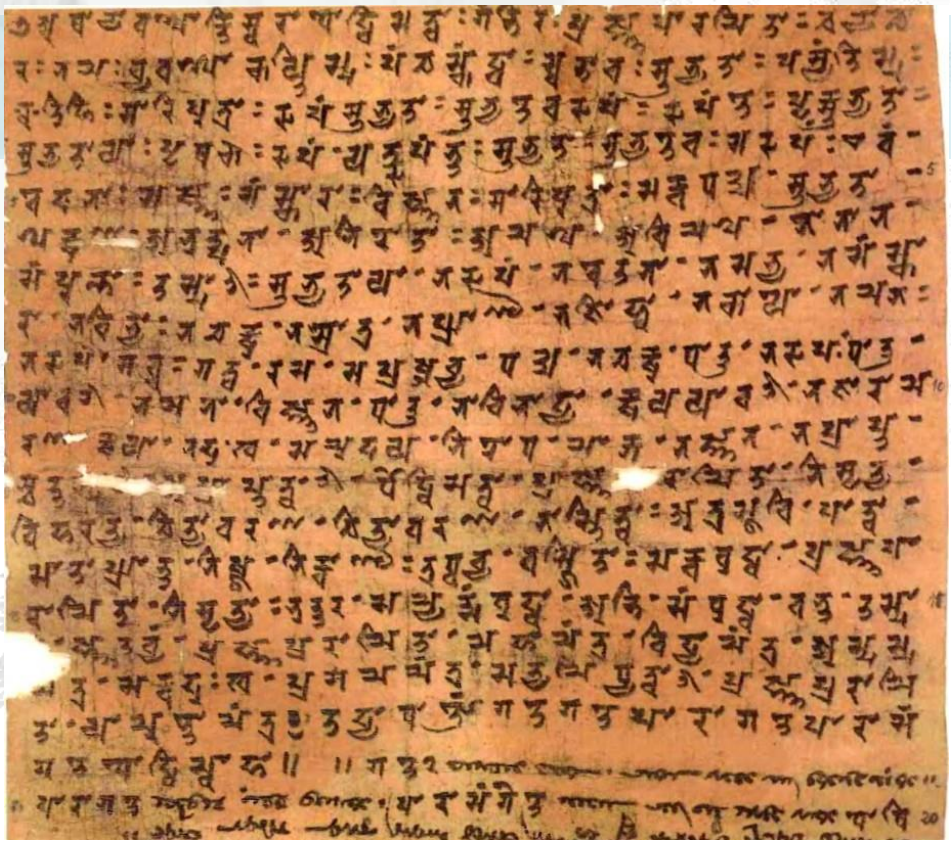
(Original arriba).

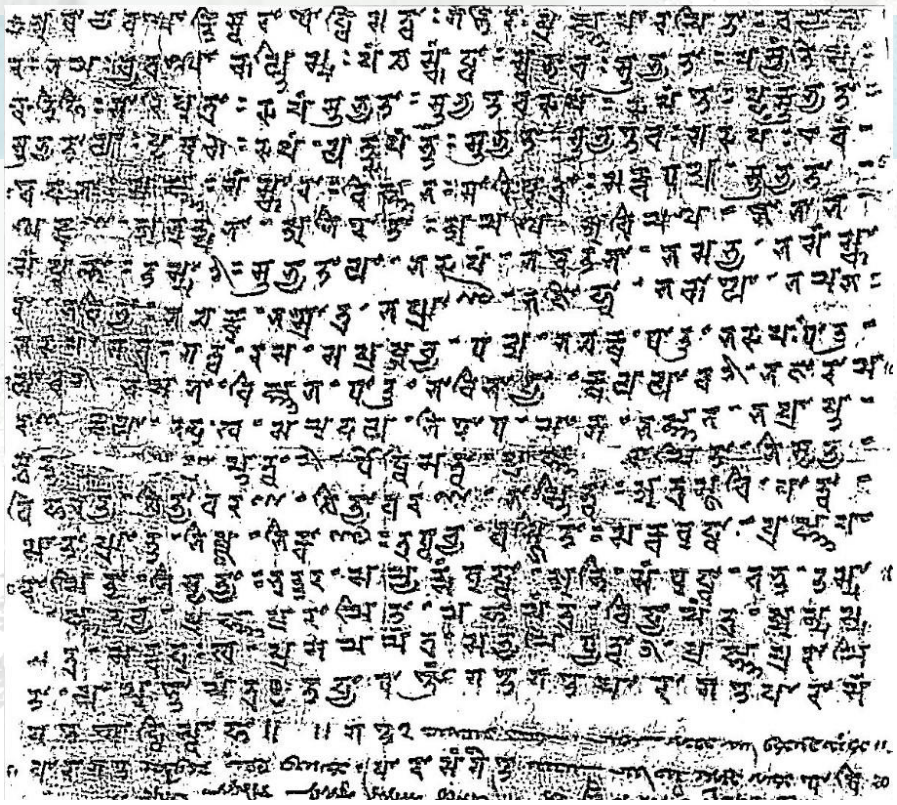
Las imágenes muestran calidad inferior a las anteriores porque los caracteres son representados con diferentes colores, existe pérdida de información en los mismos y se presentan alteraciones.

Imagenes Malas:









(Original arriba).

En este tipo de imágenes podemos observar que los cambios de color en el fondo son significativos, pues tenemos mayor pérdida de información e intrusión de ruido.

Conclusión:

De manera general se puede decir que el algoritmo utilizado es bueno en el reconocimiento de imágenes que presentan una calidad BUENA o REGULAR.

Referencias:

[1]. Hongxi Wei, Guanglai Gao, “An Efficient Binarization Method for Ancient Mongolian

Document Images”. 2010 3rd

International Conference on Advanced Computer Theory and Engineering(ICACTE).

[2]. Sabri A. Mahmoud, “Arabic Character Recognition using Modified Fourier Spectrum

(MFS)”. Arabic Character Recognition using Modified Fourier Spectrum (MFS).

[3]. J. Sauvola, “Adaptive document image binarization”. Machine Vision and Media Processing Group, Infotech Oulu, University of Oulu, P.O. BOX

4500, FIN-90401 Oulu, Finland Received 29 April 1998; accepted 21 January 1999.

[4]. Toufik Sari, “Recognition-free Retrieval of Old Arabic Document Images”. Laboratoire de Gestion Electronique de Documents (LabGED), University Badji Mokhtar, Annaba, Algeria.

[5]. Abdelkarim EL BAATI, “Recovery of Temporal Information from off-line Arabic

Handwritten”. Research Group on Intelligent Machines (REGIM), University of Sfax, ENIS,

3038 Sfax, Tunisia. Laboratory PSI , University of Rouen, 76821 Mont Saint Aignan Cedex, France

en 1

[Escribir el subtítulo del documento]

[Escriba aquí una descripción breve del documento.
Una descripción breve es un resumen corto del

contenido del documento. Escriba aquí una descripción breve del documento. Una descripció breve es un resumen corto del contenido del documento.]

user

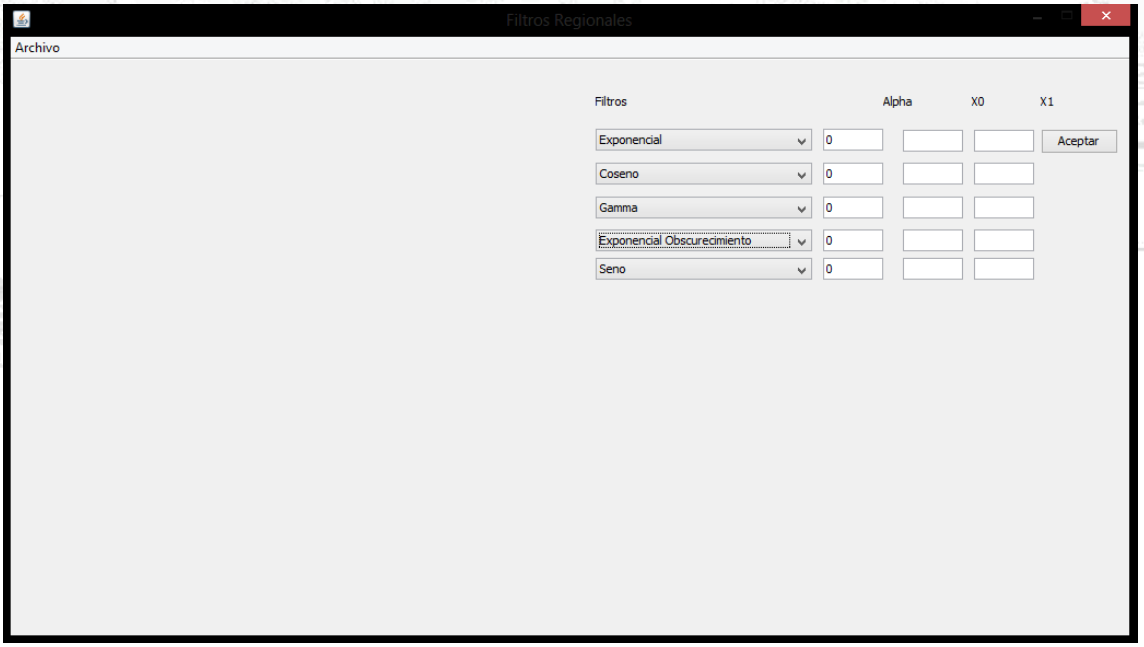
user

DESARROLLO

Pues para la realización de esta práctica fue necesario programar las formulas de los filtros regionales los cuales eran las funciones: seno, coseno, exponencial, exponencial de obscurecimiento, gamma y logaritmo.

Se les hizo un pequeño ajuste a los formulas ya antes programadas para poder realizar las operaciones de los filtros que se seleccionen de acuerdo al menú creado.

A continuación se mostrara el menú con el que se operaran las formulas en la imagen.

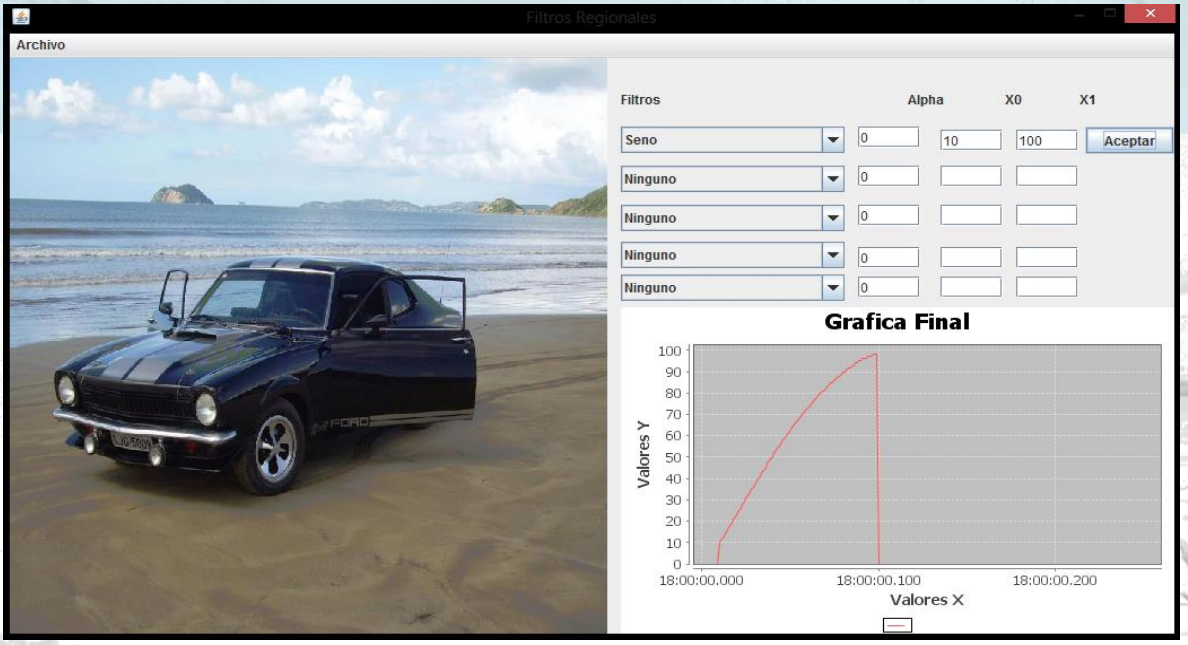


Este menú será el encargado de realizar las operaciones pertinentes a la imagen, donde cada una de ellas contiene la función programada y el usuario la asignara a la imagen, el usuario tendrá que ingresar los valores para los rangos en los que quiere que se asigne un cambio a la imagen, además que en esta ventana se presentara la imagen con las modificaciones pertinentes y una grafica en la cual se mostrara el comportamiento de los pixeles.

Función Seno Regional

Pues al aplicar la función seno se tienen que ingresar los valores de rangos x0 y x1 el valor de alpha se bloquea ya que no es necesario pedirlo en esta, para ellos se uso la siguiente función:

```
public BufferedImage SenoRegional(int x1, int x0)
    {
      int i,j,rojo,verde,azul;
      double red,green,blue,projo,pverde,pazul,xg,x1g;
      red=0;
      green=0;
      blue=0;
      projo=0;
      pverde=0;
      pazul=0;
      xg=0;
      x1g=0;
        for(i=0;i<Imagen.getWidth();i++)
        {
            for(j=0;j<Imagen.getHeight();j++)
            {
                Color color=new Color(Imagen.getRGB(i, j));
                rojo=color.getRed();
                verde=color.getGreen();
                azul=color.getBlue();
                x1g=x1-x0;
                if (rojo>=x0&&rojo<x1)
                    {
                     xg=rojo-x0;
                     projo= ((Math.PI*xg)/(2*x1g));
                     red=Math.sin(projo);
                     red= (x1g*red)+x0;
                     rojo= (int)red;
                    }
                   if(verde>=x0 && verde<x1)
                        {
                         xg=verde-x0;
                         pverde=((Math.PI*xg)/(2*x1g));
                         green=Math.sin(pverde);
                         green= (x1g*green)+x0;
                         verde= (int)green;
                        }
                    if(azul>=x0 && verde<x1)
                    {
                     xg=azul-x0;
                     pazul= ((Math.PI*xg)/(2*x1g));
                     blue=Math.sin(pazul);
                     blue= (x1g*blue)+x0;
                     azul= (int)blue;
                    }
                    Imagen.setRGB(i, j,new Color(rojo,verde,azul).getRGB());
            }
        }
        return Imagen;
    }
```



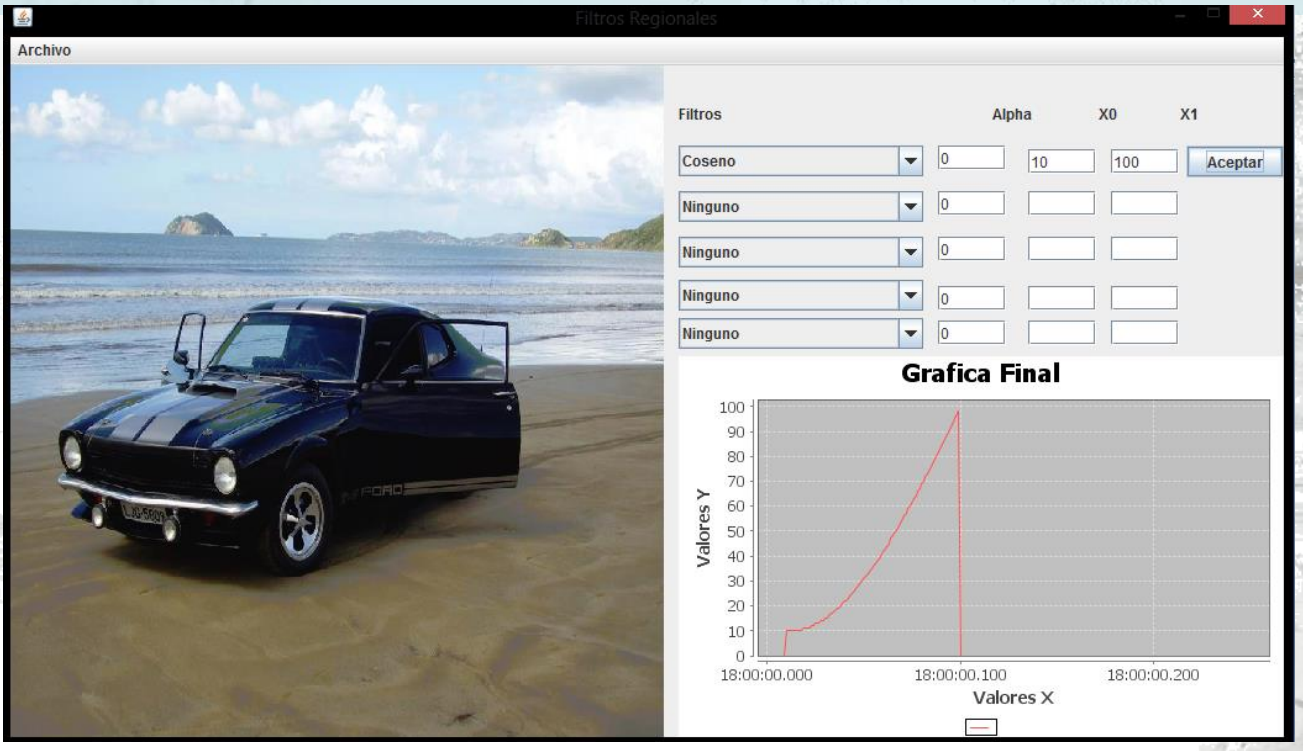
Pues los resultados arrojados por esta función son los que se presentan a continuación.

Como se puede ver la imagen se modifico en el rango 10 a 100 su valor de pixel fue modificado y se presento una grafica para ver el comportamiento de la imagen.

Función Coseno Regional

Pues al aplicar la función coseno se tienen que ingresar los valores de rangos x0 y x1 el valor de alpha se bloquea ya que no es necesario pedirlo en esta, para ellos se uso la siguiente función:

```
public BufferedImage CosenoRegional(int x1, int x0)
    {
      int rojo,verde,azul;
      double red,green,blue,Gorrito,xg;
        for(int i=0;i<Imagen.getWidth();i++)
        {
            for(int j=0;j<Imagen.getHeight();j++)
            {
                Color Colores=new Color(Imagen.getRGB(i, j));
                rojo=Colores.getRed();
                verde=Colores.getGreen();
                azul=Colores.getBlue();
                Gorrito=x1-x0;
                if(rojo>=x0 && rojo<x1)
                {
                  xg=rojo-x0;
                  red=1-Math.cos((Math.PI*xg)/(2*Gorrito));
                  red=(Gorrito*red)+x0;
                  rojo= (int)red;
                }
                if(verde>=x0 && verde<x1)
                {
                  xg=verde-x0;
                  green=1-Math.cos((Math.PI*xg)/(2*Gorrito));
                  green=(Gorrito*green)+x0;
                  verde= (int)green;
                }
                if(azul>=x0 && azul<x1)
                {
                  xg=azul-x0;
                  blue=1-Math.cos((Math.PI*xg)/(2*Gorrito));
                  blue=(Gorrito*blue)+x0;
                  azul= (int)blue;
                }
               Imagen.setRGB(i, j,new Color(rojo,verde,azul).getRGB());
            }
        }
        return Imagen;
    }
```



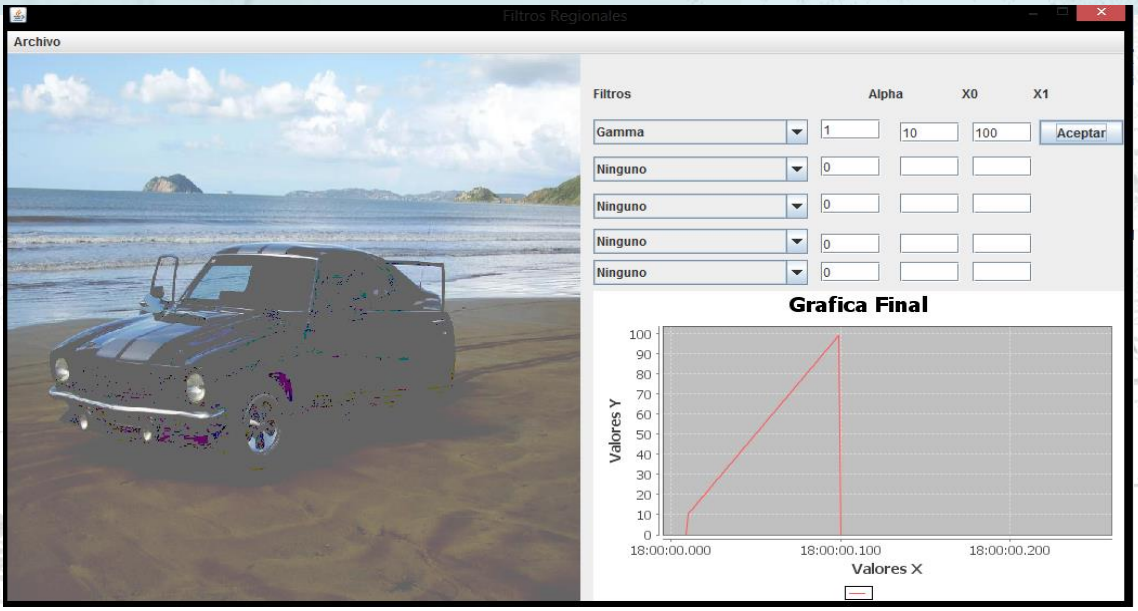
Pues los resultados arrojados por esta función son los que se presentan a continuación.

Como se puede ver la imagen se modifico en el rango 10 a 100 su valor de pixel fue modificado y se presento una grafica para ver el comportamiento de la imagen en la función coseno.

Función Gamma Regional

Pues al aplicar la función gamma se tienen que ingresar los valores de rangos x0 y x1 el valor de alpha se introducirá ya que es necesario pedirlo en esta función, para ellos se uso la siguiente función:

```
public BufferedImage GamaRegional (int x1, int x0, float gama)
    {
        int i,j,rojo,verde,azul;
        double auxr,auxv,auxb,xg,Gorrito,r,g,b,rf,gf,bf;
        for(i=0;i<Imagen.getWidth();i++)
        {
            for(j=0;j<Imagen.getHeight();j++)
            {
                Color color=new Color(Imagen.getRGB(i, j));
                rojo=color.getRed();
                verde=color.getGreen();
                azul=color.getBlue();
                Gorrito=x1-x0;
                if(rojo>=x0 && rojo<x1)
                {
                  xg=rojo-x0;
                  r=(xg/Gorrito);
                  auxr=Math.pow (r, gama);
                  rf=(auxr*Gorrito)+x0;
                  rojo=(int)rf;
                }
                if(verde>=x0 && verde<x1)
                {
                  xg=verde-x0;
                  g=(xg/Gorrito);
                  auxv=Math.pow (g, gama);
                  gf=(auxv*Gorrito)+x0;
                  verde=(int)gf;
                }
                if(azul>=x0 && azul<x1)
                {
                  xg=azul-x0;
                  b=(xg/Gorrito);
                  auxb=Math.pow (b, gama);
                  bf=(auxb*Gorrito)+x0;
                  azul=(int)bf;
                }
                 Imagen.setRGB(i, j,new Color(rojo, verde, azul).getRGB());
            }
        }

        return Imagen;
}
```



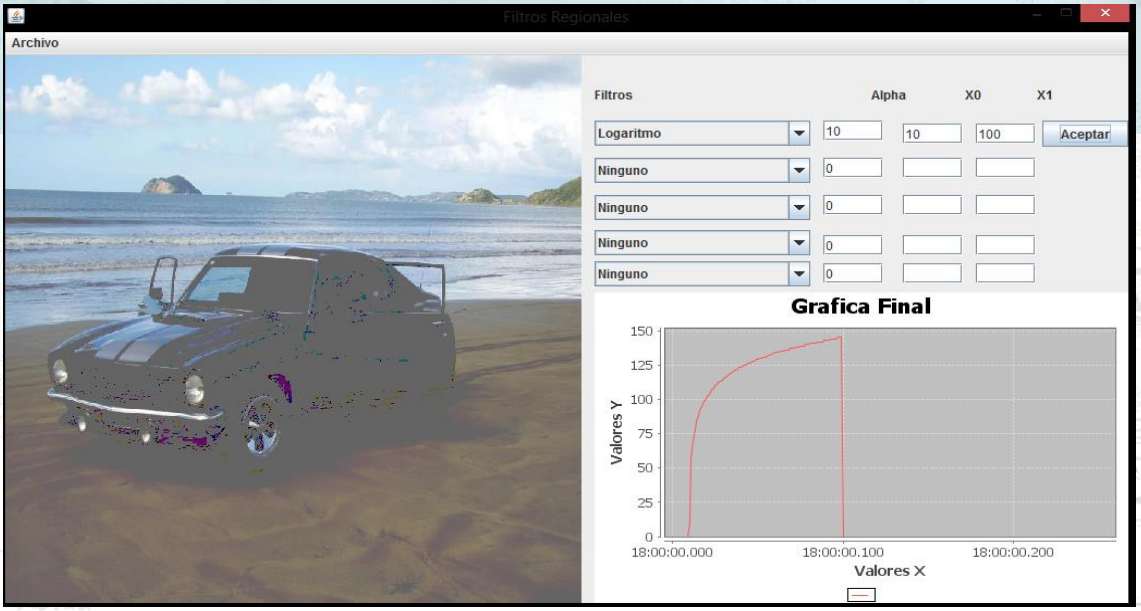
Pues los resultados arrojados por esta función son los que se presentan a continuación.

Como se puede ver la imagen se modifico en el rango 10 a 100 su valor de pixel fue modificado y se presento una grafica para ver el comportamiento de la imagen en la función gamma.

Función Logaritmo Regional

Pues al aplicar la función logaritmo se tienen que ingresar los valores de rangos x0 y x1 el valor de alpha se introducirá ya que es necesario pedirlo en esta función, para ellos se uso la siguiente función:

```
public BufferedImage LogRegional (int x1,int x0,float alf)
{
        int i,j,rojo,verde,azul,Q=x1-x0+1;
        double rojito,verdesito,azulito,A,logr,logg,logb,Gorrito,xg;
        rojito=0;
        verdesito=0;
        azulito=0;
        A=0;
        logr=0;
        logg=0;
        logb=0;
        Gorrito=0;
        xg=0;
        for(i=0;i<Imagen.getWidth();i++)
        {
            for(j=0;j<Imagen.getHeight();j++)
            {
                Color color=new Color(Imagen.getRGB(i, j));
                rojo=color.getRed();
                verde=color.getGreen();
                azul=color.getBlue();
                Gorrito=x1-x0;
                A=Gorrito/(Math.log((alf*Q)+1));

                if(rojo>=x0 && rojo<x1)
                {
                  rojito=(A*logr)+x0;
                  rojo=(int)rojito;
                  xg=rojo-x0;
                  logr=Math.log((alf*xg)+1);
                }
               if(verde>=x0 && verde<x1)
               {
                 xg=verde-x0;
                 logg=Math.log((alf*xg)+1);
                 verdesito=(A*logg)+x0;
                 verde=(int)verdesito;
               }
               if(azul>=x0 && azul<x1)
               {
                   xg=azul-x0;
                   logb=Math.log((alf*xg)+1);
                   azulito=(A*logb)+x0;
                   azul=(int)azulito;
               }
               Imagen.setRGB(i, j,new Color(rojo,verde,azul).getRGB());
            }
        }
        return Imagen;
}
```



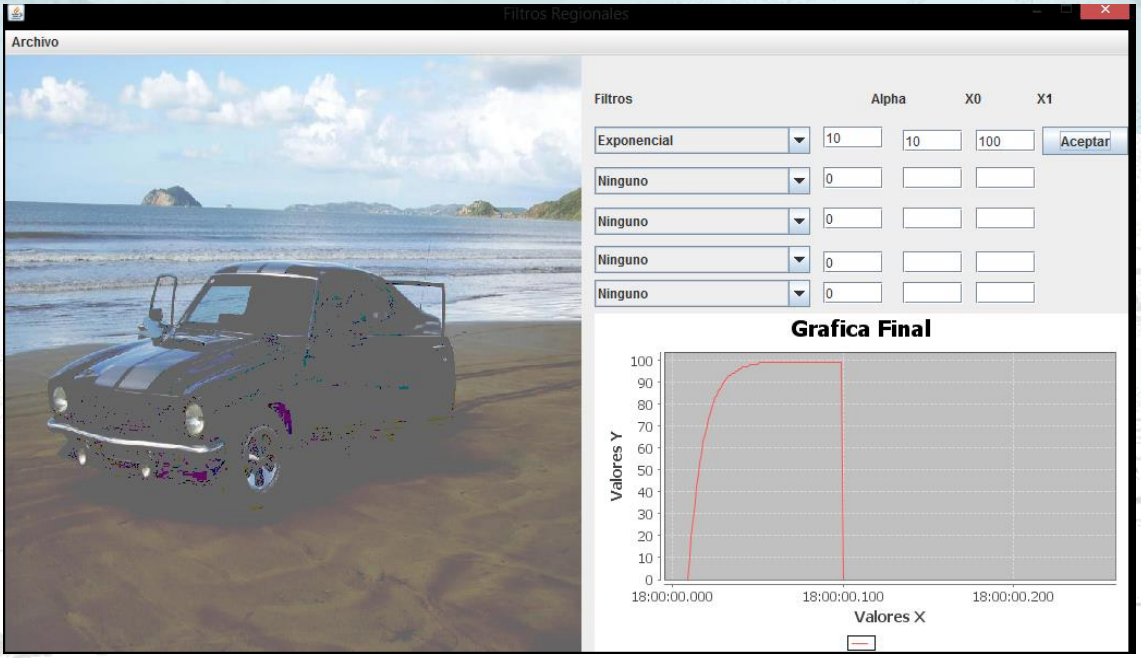
Pues los resultados arrojados por esta función son los que se presentan a continuación.

Como se puede ver la imagen se modifico en el rango 10 a 100 su valor de pixel fue modificado y se presento una grafica para ver el comportamiento de la imagen en la función Logaritmo.

Función Exponencial regional

Pues al aplicar la función exponencial se tienen que ingresar los valores de rangos x0 y x1 el valor de alpha se introducirá ya que es necesario pedirlo en esta función, para ellos se uso la siguiente función:

```
public BufferedImage ExponencialRegional (int x1, int x0, float exp)
    {
        Color Colores;
        int rojo,verde,azul;
        float auxr,auxv,auxa,A,red,green,blue,Gorrito,xg,ar,er;
        Gorrito=x1-x0;
        ar=(float) (1-(Math.pow(Math.E, (-exp))));
        A=Gorrito/ar;

        for (int i=0;i<Imagen.getWidth();i++)
        {
            for(int j=0;j<Imagen.getHeight();j++)
            {
                Colores=new Color(Imagen.getRGB(i, j));
                rojo=Colores.getRed();
                verde=Colores.getGreen();
                azul=Colores.getBlue();

                if(rojo>=x0 && rojo<x1)
                {
                    xg=rojo-x0;
                    er=(-exp*xg)/Gorrito;
                    auxr=(float)(1-(Math.pow(Math.E, er)));
                    red=(A*auxr)+x0;
                    rojo=(int)red;
                }
                 if(verde>=x0 && verde<x1)
                {
                    xg=verde-x0;
                    er=(-exp*xg)/Gorrito;
                    auxv=(float) (1-(Math.pow(Math.E, er)));
                    green=(A*auxv)+x0;
                    verde=(int)green;
                }
                  if(azul>=x0 && azul<x1)
                {
                    xg=azul-x0;
                    er=(-exp*xg)/Gorrito;
                    auxa=(float) (1-(Math.pow(Math.E, er)));
                    blue=(A*auxa)+x0;
                    azul=(int)blue;
                }
                Imagen.setRGB(i, j, new Color(rojo,verde,azul).getRGB());
            }
        }
    return Imagen;
}
```



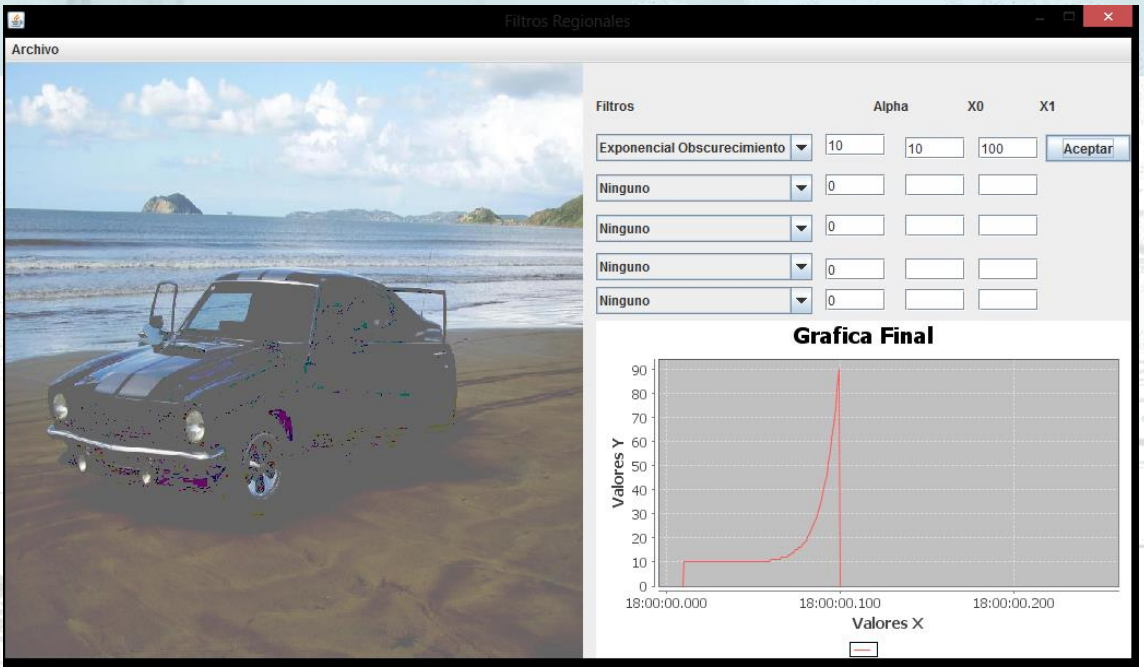
Pues los resultados arrojados por esta función son los que se presentan a continuación.

Como se puede ver la imagen se modifico en el rango 10 a 100 su valor de pixel fue modificado y se presento una grafica para ver el comportamiento de la imagen en la función exponencial.

Función Exponencial Obscurecimiento regional

Pues al aplicar la función exponencial de obscurecimiento , se tienen que ingresar los valores de rangos x0 y x1 el valor de alpha se introducirá ya que es necesario pedirlo en esta función, para ellos se uso la siguiente función:

```
public BufferedImage ExponencialObscurecimiento(int x1, int x0, double al
    {
      int i,j,rojo,verde,azul,q=x1;
      double r,g,b,A,S,Gorrito,xg,ar,auxr,auxv,auxa,er;
        for(i=0;i<Imagen.getWidth();i++)
        {
            for(j=0;j<Imagen.getHeight();j++)
            {
                Color Colores=new Color(Imagen.getRGB(i, j));
                rojo=Colores.getRed();
                verde=Colores.getGreen();
                azul=Colores.getBlue();
                Gorrito=q-x0;
                A=Gorrito/(Math.pow(Math.E, alfa))-1;
                if(rojo>=x0 && rojo<x1)
                {
                    xg=rojo-x0;
                    er=(alfa*xg)/Gorrito;
                    auxr=(Math.pow(Math.E, er))-1;
                    r=(A*auxr)+x0;
                    rojo=(int)r;
                }
                if(verde>=x0 && verde<x1)
                {
                    xg=verde-x0;
                    er=(alfa*xg)/Gorrito;
                    auxv=(Math.pow(Math.E, er))-1;
                    g=(A*auxv)+x0;
                    verde=(int)g;
                }
                if(azul>=x0 && azul<x1)
                {
                    xg=azul-x0;
                    er=(alfa*xg)/Gorrito;
                    auxa=(Math.pow(Math.E, er))-1;
                    b=(A*auxa)+x0;
                    azul=(int)b;
                }
                Imagen.setRGB(i, j,new Color(rojo,verde,azul).getRGB());
            }
        }
        return Imagen;
    }
```



Pues los resultados arrojados por esta función son los que se presentan a continuación.

Como se puede ver la imagen se modifico en el rango 10 a 100 su valor de pixel fue modificado y se presento una grafica para ver el comportamiento de la imagen en la función exponencial de obscurecimiento.

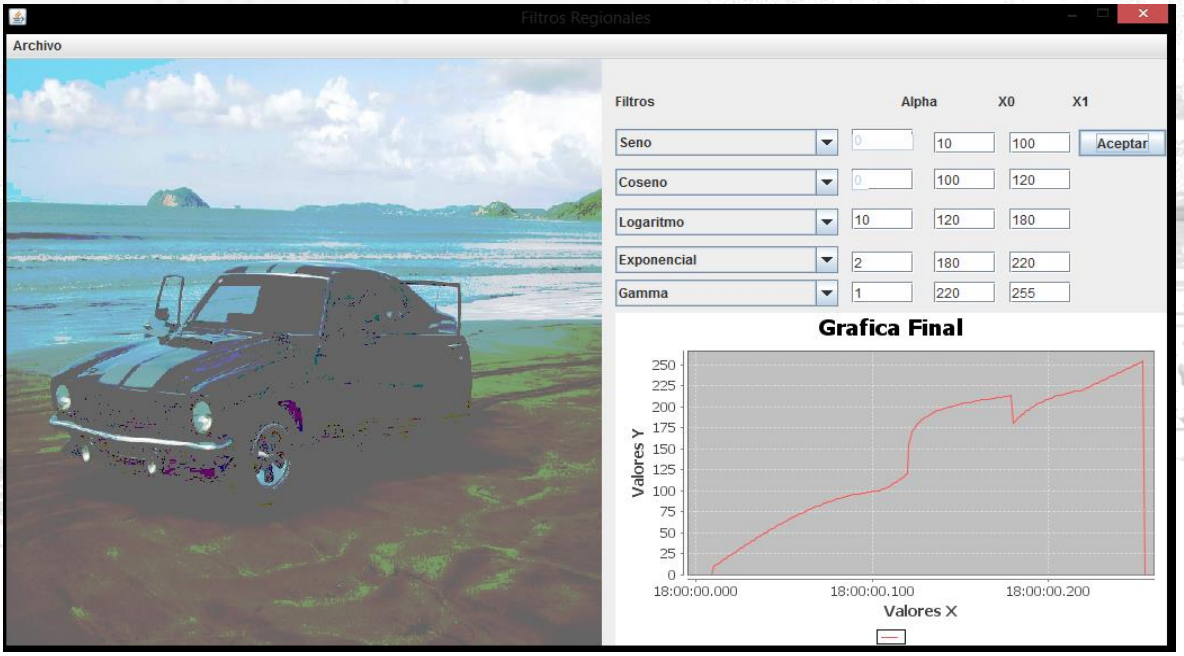
PRUEBAS

En la parte anterior se mostraron por separado cada una de las funciones, el objetivo de este parcial es poder usar 5 filtros como máximo a una imagen y que se apliquen de acuerdo al rango que se desee.

A continuación mostrare los resultados de 2 pruebas que se realizaron con los siguientes valores:

Seno	**10,100**
Coseno	100,120
Logaritmo	10,120,180
Exponencial	2,180,220

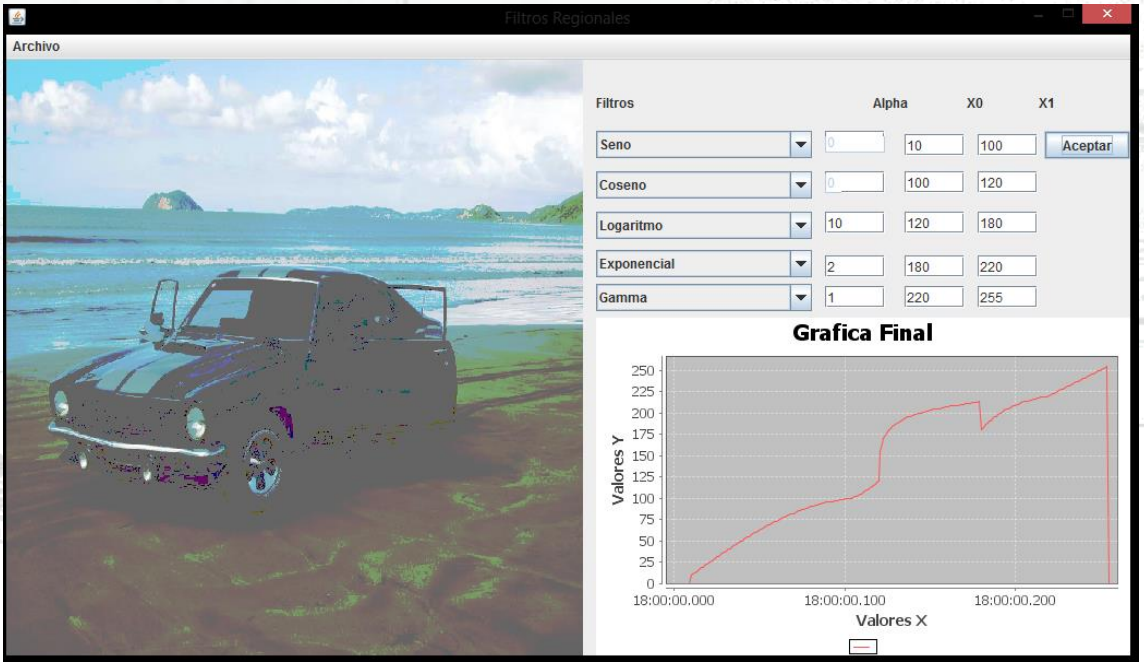
Gamma	1,220,255



Resultados arrojados por el programa dados los valores de la tabla.

SEGUNDA PRUEBA

Coseno	**0,100**
Seno	100,120
Exponencial Obscurecimiento	1,120,180
Logaritmo	2,180,220
Exponencial	1,220,255



Resultados arrojados por el programa dados los valores de la segunda tabla.

CONCLUSION

Pues como se puede ver los filtros regionales fueron programados de una manera satisfactoria de acuerdo a las formulas proporcionadas en clase.

Lo realmente complejo de este parcial fue la unión de todas las funciones aplicadas a una imagen.

Los resultados presentados fueron exitosos ya que los comportamientos presentados en la grafica, como se puede apreciar son correctos.

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
Ingeniería en Ciencias de la Computación
Procesamiento Digital de Imágenes

Práctica 7

Cesar Manuel Rodríguez Mendiola

26/02/2014

Antecedentes teóricos

En esta práctica implementaremos los operadores regionales.

Los operadores regionales permiten modi car el valor del color del pixel, tomando en cuenta el valor de los pixeles vecinos a el.

0.1. Operdor Horizontal

$$\triangle P_i' = /P[i+1, j] - P[i, j]/$$

0.2. Operador Vertical

$$\triangle P_i' = /P[i, j+1] - P[i, j]/$$

0.3. Operador Mixto

$$\triangle P' = /\triangle P'_i - \triangle P'_j/$$

$$\triangle P_i' = /P[i+1, j] - P[i, j], P[i, j+1] - P[i, j]/$$

0.4. Operador Gradiente

$$\|\nabla f(i,j)\| = min\{\sqrt{\qquad\qquad}$$

$(\Delta P')^2 + (\Delta P')^2, q\}$

i j

3

0.5. Operador Gradiente 1° Variante

$$\nabla f(i, j) = 0.5 *$$

$$\sqrt{\qquad\qquad} \quad (\Delta P'_i)^2 + (\Delta P'_j)^2$$

4

0.6. Operador Gradiente 2° Variante

$$\nabla f(i, j) = min\{(\Delta P'_i + \Delta P'_j), q$$

Desarrollo

La implementación es sencilla, solo necesitamos un doble for con los cuales recorreremos toda la imagen, teniendo en cuenta que para algunas casos no se puede recorrer todas las las o todas las columnas, esto es por la variante del método

Calculamos los delta P

Pi

```
colorAuxActual = new Color(this.imageActual.getRGB(i, j));

colorAuxVecino = new Color(this.imageActual.getRGB(i+1, j));

mediaPixelActual = (colorAuxActual.getRed()+colorAuxActual.getGreen()+colorAuxAc

mediaPixelVecino = (colorAuxVecino.getRed()+colorAuxVecino.getGreen()+colorAuxVe

Pi = (int)Math.round(Math.abs(mediaPixelVecino - mediaPixe-
lActual));
```

Pj

```
colorAuxActual = new Color(this.imageActual.getRGB(i, j));

colorAuxVecino = new Color(this.imageActual.getRGB(i, j+1));

mediaPixelActual = (colorAuxActual.getRed()+colorAuxActual.getGreen()+colorAuxAc
```

```
mediaPixelVecino = (colorAuxVecino.getRed()+colorAuxVecino.getGreen()+colorAuxVe

Pj = (int)Math.round(Math.abs(mediaPixelVecino - mediaPixe-
lActual));
```

Determinamos el borde

Gradiente

DP = Math.sqrt(Math.pow(Pi, 2) + Math.pow(Pj, 2)); colorResultante = (int)Math.round(Math.min(DP, q));

Gradiente 2° Variacion

DP = Math.sqrt(Math.pow(Pi, 2) + Math.pow(Pj, 2));

Gradiente 3° Variacion

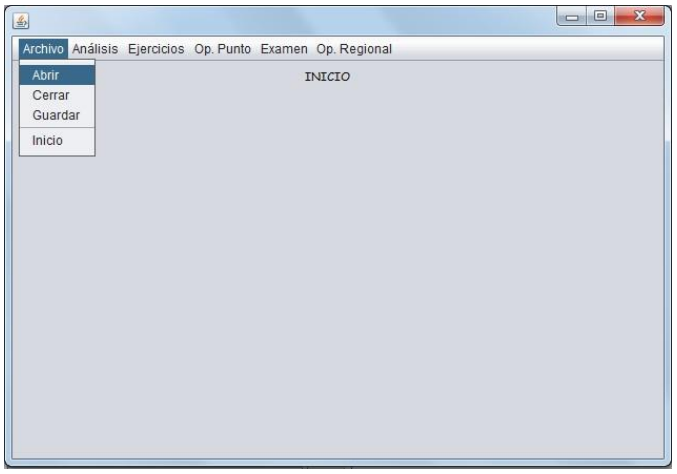
colorResultante = (int)Math.round(Math.min((Pi+Pj), q));

Cambiamos a formato sRGB

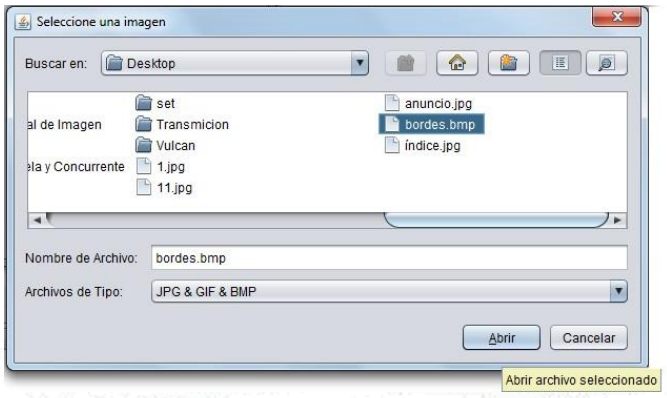
colorSRGB = (colorResultante < < 16) | (colorResultante < < 8) | colorResultante;

Resultados

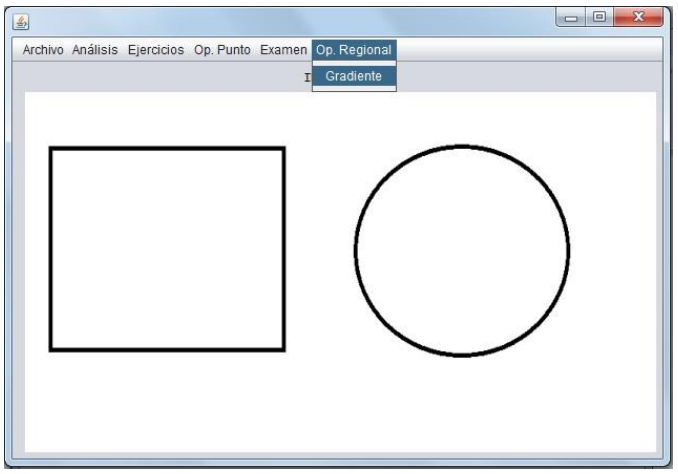
Ejecutamos la aplicación.



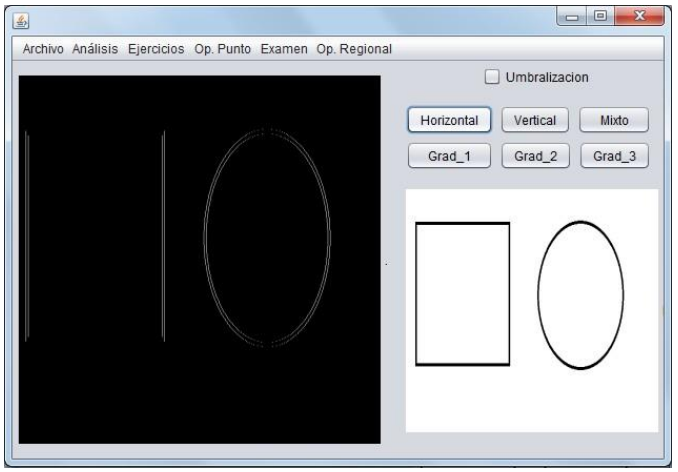
Se habre el buscador



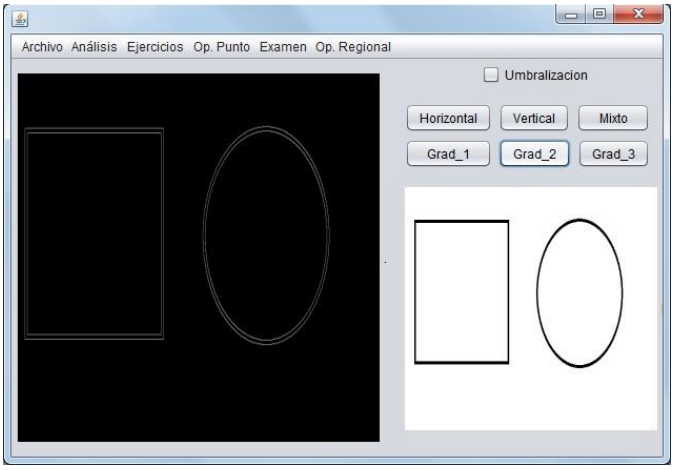
Se habre una imagen



Seleccionamos los operadores regionales



Aplicamos deccion de bordes en forma Horizontal



de bordes mediante Gradiente con la 2° Variación

Aplicamos deteccion

Concluciones

Es una aplicación que involucra diversos conocimientos, matemáticos e informáticos.

La implementación es relativamente sencilla siempre y cuando se tengan los fundamentos.

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
Ingeniería en Ciencias de la Computación
Procesamiento Digítal de Imágenes

Examen del Parcial 2

Cesar Manuel Rodríguez Mendiola
Jesús Max Melchor Jimenez

27/03/2014

Antecedentes teóricos

Con la nalidad de preservar información relevante para la humanidad o una porción de ella, es necesario transcribir documentos antiguos (legajos), para facilitar dicha tarea se hace uso de la tecnología, por ello en un primer paso, los documentos son digitalizados, posteriormente se someten a un pre-procesamiento para un futuro tratamiento digital mediante el cual se extraerá la mayor cantidad de información.

Como mencionamos anteriormente para el tratamiento digital de documentos es necesario someterlos a un pre-procesamiento, el cual será atacado a lo largo de este tema.

Para realizar el pre-procesamiento implementaremos ciertos ltros de suavizado para la eliminación de ruido (porción de documento que no aporta información: manchas, deformaciones, etc.), métodos de bianarización para realzar la tonalidad del texto y normalizar el color del fondo.

Con estos pasos esperamos obtener un documento sin ruido, la letra en un color negro y el fondo blanco. En este caso para lograr resultados aceptables aremos uso de los siguientes ltros: Escala de grises.

Negativo.

Bianarización.

Sigmoidal.

Desarrollo

Es conveniente recalcar la implementación de las librerías: jcommon y jfreechart. Esto es solo si se desea manipular el código fuente de esta aplicación.

Para el pre-procesamiento seguiremos el siguiente algoritmo:

-> Transformamos la imagen a escala de grises

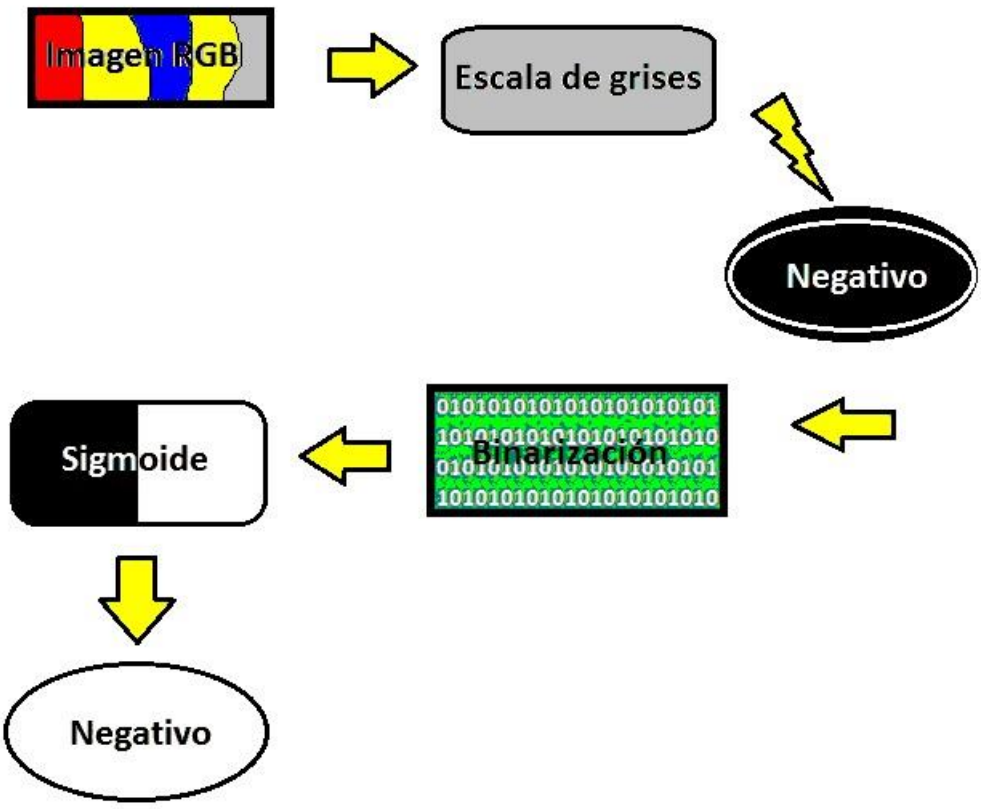
-> Negativo

-> Bianarización (iterativo, el usuario decide cuando detenerlo)

-> Sigmoide

-> Negativo

Para entender mejor el pre-procesamiento lo ilustraremos mediante el siguiente diagrama de ujo lineal.



Esta aplicación la probamos con nuestra la base de datos, conformada por 12 imágenes las cuales a criterio nuestro fueron clasi cadas en:

4 son Buenas

4 son Regulares

4 son Malas

La aplicación es iterativa, es decir, requiere que el usuario determine el instante deseado de ltrado.

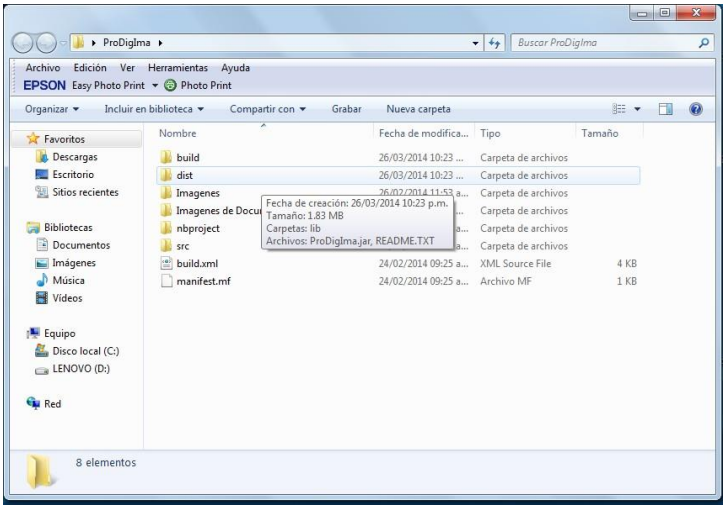
El resultado favorable es determinado por el usuario mediante las iteraciones generadas con el botón Siguiente , con el propósito de ser accesibles con el usuario, se presenta el botón Anterior con este puede regresar hasta un máximo de 5 iteraciones.

Cuando se llega a el resultado más preciso, se presionara el botón Finalizar para aplicar los últimos 2 ltros, esto es para cambiar el color del fondo y de la letra.

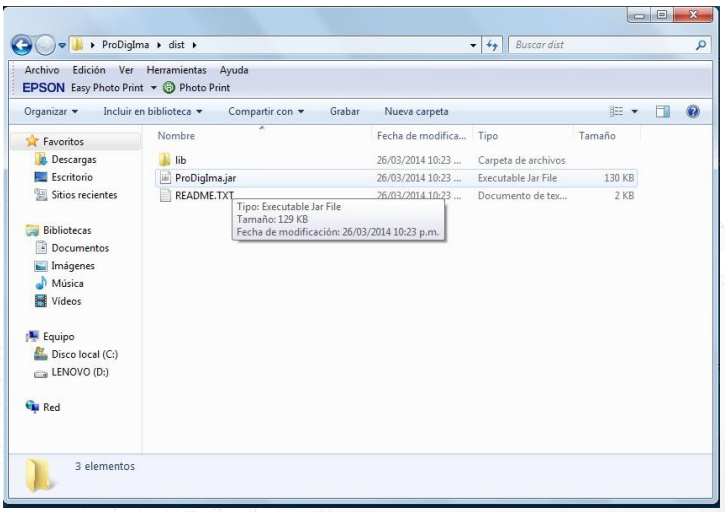
Nota: Para resolver dudas referentes a las librerías o ltros, revisar el reporte del parcial_1.

Resultados

Arimos la carpeta del proyecto ProDigMa

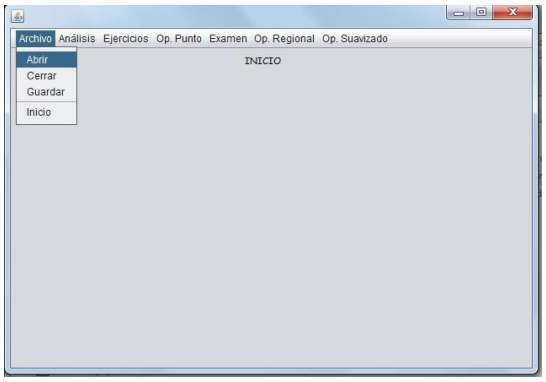


Abrimos la carpeta dist

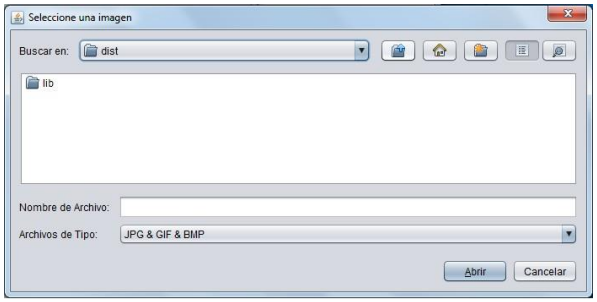


Ejecutamos la aplicación ProDigIma .

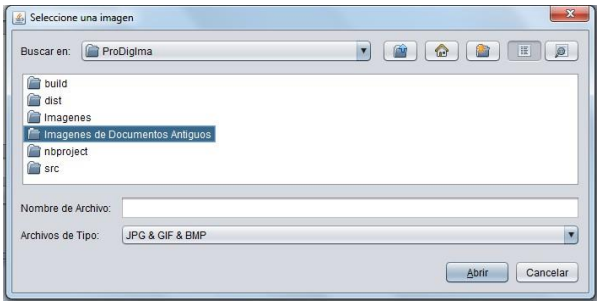
Pasos para abrir una imagen.



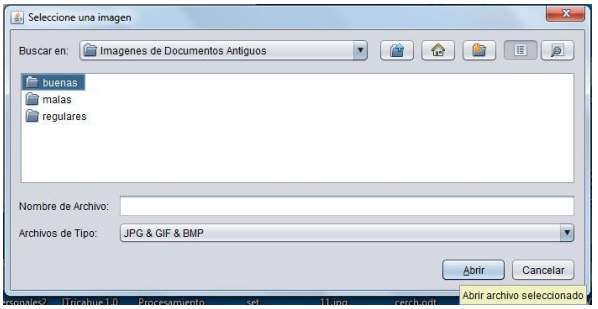
Archivo...Abrir



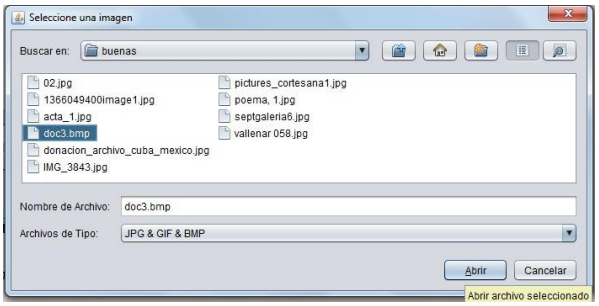
Regresamos una carpeta anterior



Seleccionamos la carpeta Imagenes de Documentos Antiguos



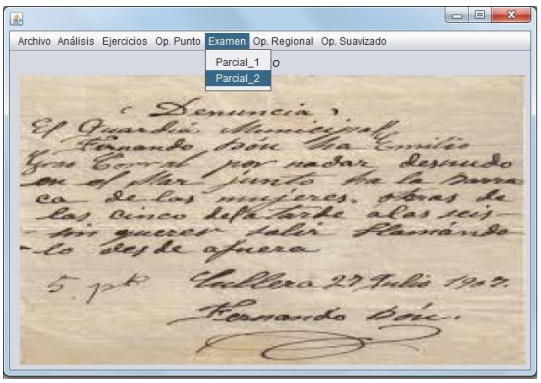
Seleccionamos la carpeta segun la clasi cacion



Abrimos la imagen a procesar.

0.1. Aplicando a imagenes denominadas BUENAS.

0.1.1. Primer ejemplo.



Del menu Examen ... seleccionamos Parcial_2

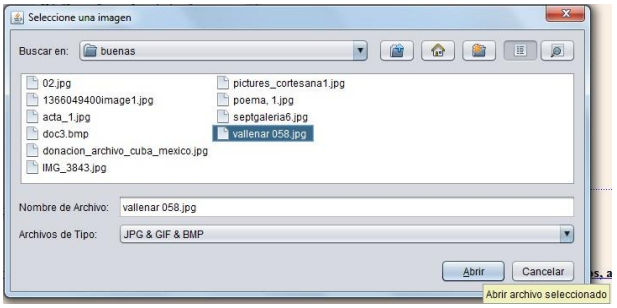
Con la primer iteración de Siguente , la imagen es transformada a escala de grises.

Con la segunda iteración de Siguente , la imagen es transformada a su negativo.

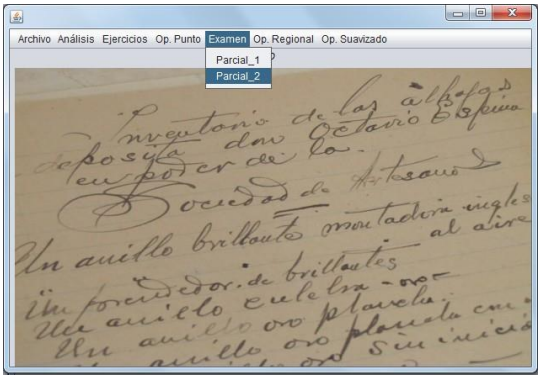
Con la vigésimo primer iteración de Siguente , se alcanza el resultado idoneo.

Finalmente se preciona el boton Finalizar para obtener este resultado.

0.1.2. Segundo ejemplo



Abrimos otra imagen

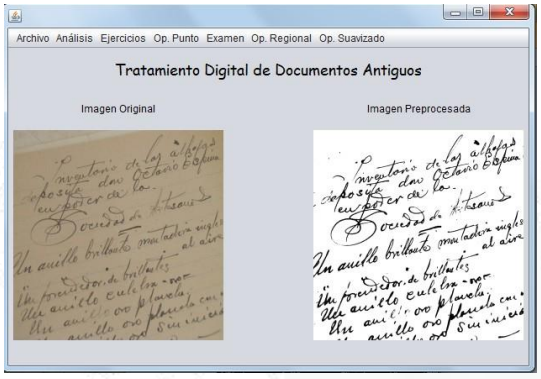


Del menu Examen ... seleccionamos Parcial_2

Con la primer iteración de Siguente , la imagen es transformada a escala de grises.

Con la segunda iteración de Siguente , la imagen es transformada a su negativo

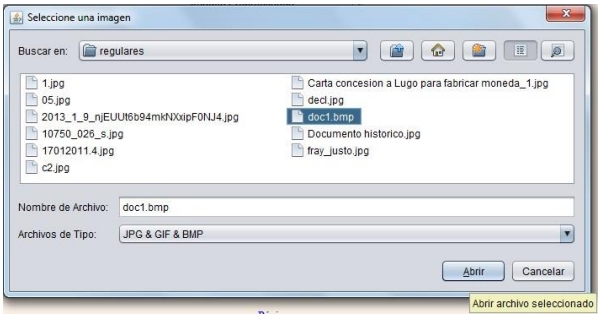
Con la decimo cuarta iteración de Siguente , se alcanza el resultado idoneo.



Finalmente se preciona el boton Finalizar para obtener este resultado.

0.2. Aplicando a imagenes denominadas REGULARES.

0.2.1. Primer ejemplo.



Abrimos una imagen Regular

Del menu Examen ... seleccionamos Parcial_2

Con la primer iteración de Siguente , la imagen es transformada a

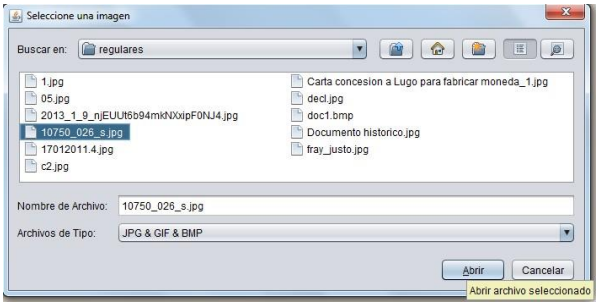
escala de grises.

Con la segunda iteración de Siguente , la imagen es transformada a su negativo.

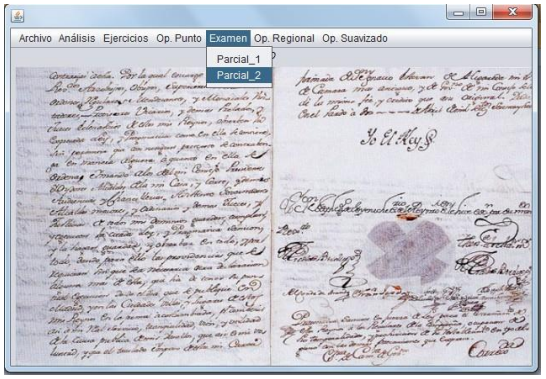
Con la vigésimo tercer iteración de Siguente , se alcanza el resultado idoneo.

Finalmente se preciona el boton Finalizar para obtener este resultado.

0.2.2. Segundo ejemplo



Abrimos otra imagen



Del menu Examen ... seleccionamos Parcial_2

Con la primer iteración de Siguente , la imagen es transformada a

escala de grises.

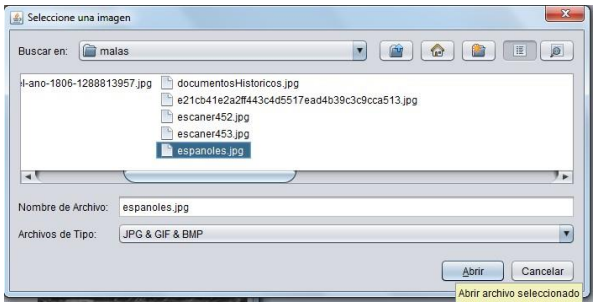
Con la segunda iteración de Siguente , la imagen es transformada a su negativo

Con la trigésimo tercera iteración de Siguente , se alcanza el resultado idoneo.

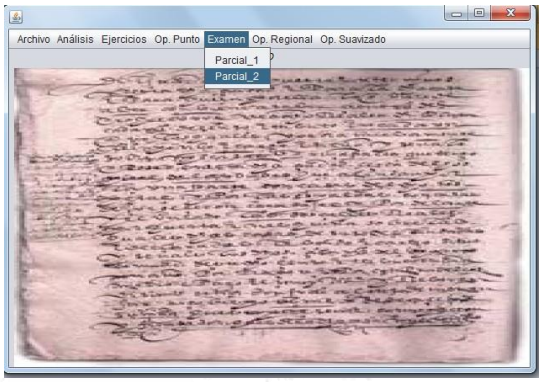
Finalmente se preciona el boton Finalizar para obtener este resultado.

0.3. Aplicando a imagenes denominadas MALAS.

0.3.1. Primer ejemplo.



Abrimos una imagen Regular



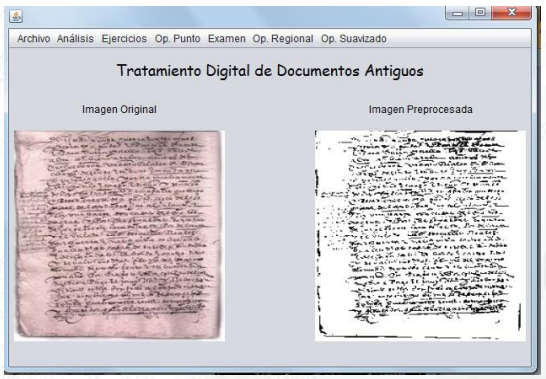
Del menu Examen ... seleccionamos Parcial_2


Archivo Análisis Ejercicios Op. Punto Examen Op. Regional Op. Suavizado
Tratamiento Digital de Documentos Antiguos
Imagen Original
Imagen Preprocesada
Siguente
Anterior
Finalizar

Con la primer iteración de Siguente , la imagen es transformada a escala de grises.

Con la segunda iteración de Siguente , la imagen es transformada a su negativo.

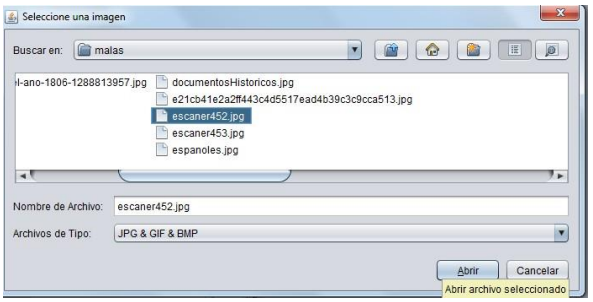
Con la vigésimo cuarta iteración de Siguente , se alcanza el resultado idoneo.



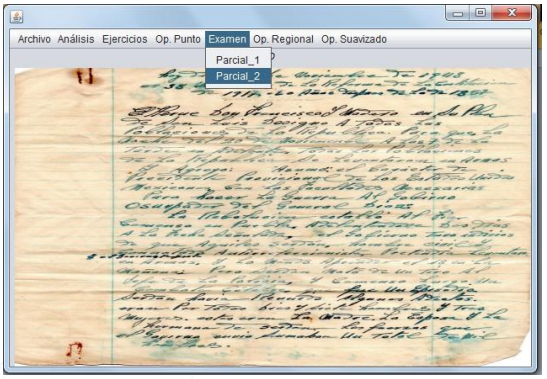
Finalmente se preciona el boton Finalizar para obtener este resul-

tado.

0.3.2. Segundo ejemplo



Abrimos otra imagen

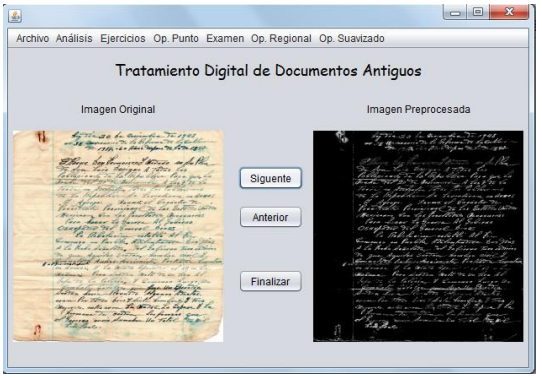


Del menu Examen ... seleccionamos Parcial_2

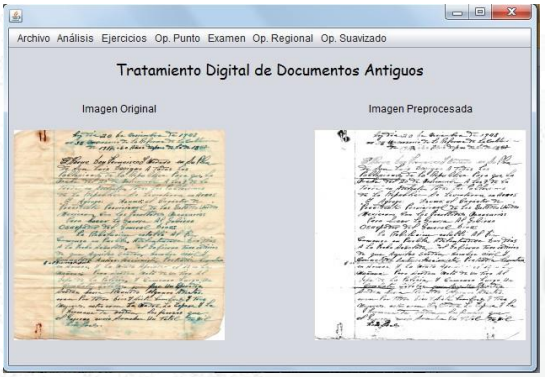
Con la primer iteración de Siguente , la imagen es transformada a

escala de grises.

Con la segunda iteración de Siguente , la imagen es transformada a su negativo



Con la vigésimo octava iteración de Siguente , se alcanza el resultado idoneo.



Finalmente se preciona el boton Finalizar para obtener este resultado.

Concluciones

Llegamos a la conclusión que la secuencia de ltros usados en nue- stro método, mostro resultados favorables para el suavizado de ruidos no tan extremos, así como los ruidos de sal y pimienta y los ruidos gaussianos.

Además de los operadores puntuales, notamos que el método de bina- rizacion implementado en la práctica, mostro buenos resultados para la eliminación de ruido regularmente notable en la imagen.

También llegamos a la conclusión de que el método implementado en esta práctica no es muy efectivo en imágenes con un exceso de ruido.

Procesamiento Digital de Imágenes
Práctica 5
"Interpolación"

Dr. Boris Escalante R.

18 de Noviembre, 2011

Reglas generales para el desarrollo de las Prácticas de Laboratorio.

El reporte de las prácticas constará de las secciones: objetivo, introducción, desarrollo (incluyendo cálculos si es el caso), resultados, conclusiones, código fuente y bibliografía.

Las prácticas deben ser originales, es decir, se sancionará a los equipos o autores de prácticas idénticas, incluyendo si fueron copiadas de prácticas de semestres anteriores.

Se recomienda trabajar en MATLAB ya que podrán obtener asesoría sobre el uso de comandos de este paquete. Esto no significa que no puedan usar otras herramientas, sin embargo, no estará garantizada la asesoría en estos casos.

El desarrollo de la práctica es trabajo de casa. El día de entrega de la práctica deberán llegar preparados, con el reporte elaborado e impreso. No se reciben reportes en formato electrónico. Durante ese día solo de revisará la práctica, se verificará el funcionamiento de los programas, sus resultados y las conclusiones que hayan obtenido con el fin de corroborar que el objetivo de la práctica se haya logrado.

1. Objetivos

Dada una imagen con un número de muestras igual a N ×M, donde N y M son el ancho y el largo de la imagen respectivamente:

Encontrar las imágenes interpoladas usando interpoladores de orden cero, lineal y cúbico.

Interpolar la imagen en el dominio de la frecuencia.

2. Introducción

El problema de construir una función continua a partir de datos discretos es inevitable cuando estos datos deben ser manipulados de cierta manera que se necesita información no incluida explícitamente. Para resolver este problema,

el esquema más utilizado es la interpolación que consiste en construir una función que aproxime de la manera más perfecta a la función original desconocida en los puntos de la medición.
La interpolación de imágenes es una operación muy importante usada en imagenología médica, procesamiento de imágenes y graficación por computadora. Existe una gran variedad de métodos de interpolación reportados en la literatura [1][2]. La interpolación de imágenes es necesaria en una gran variedad de situaciones como las que se mencionan a continuación:
1. Representar imágenes o volúmenes a un nivel deseado de discretización modificando para ello la tasa de muestreo de los pixeles o voxels (voxel, elemento de volumen equivalente a pixel en 2D).
2. Cambiar la orientación de alguna rejilla de discretización.
1

3. Combinar la información sobre un mismo objeto desde múltiples modalidades en una sola imagen (fusión de imágenes). Por ejemplo, una resonancia magnética (MRI) y una Tomografía por Emisión de Positrones (PET) del cerebro de un paciente.
4. Cambio de rejilla de discretización, por ejemplo de polares a rectangulares.
Probablemente el uso más común de la interpolación sea la mencionada en el punto número 1, donde por lo regular se desean analizar ciertos detalles de una imagen a una escala y otros detalles a una escala más fina (zoom).
Algunas transformaciones pueden involucrar un cambio de coordenadas, por ejemplo, la función de conversión de coordenadas polares, adquiridas a través de un transductor de ultrasonido, a coordenadas cartesianas necesario para la visualización de la imagen en un monitor. En general, casi cualquier transformación geométrica sobre una imagen o un volumen necesita que se efectué una interpolación.
La calidad de la imagen o volumen obtenidos dependerá del proceso utilizado para realizar la interpolación así como también del trabajo necesario para que una computadora lo ejecute en un tiempo razonable.

3. Desarrollo

Para todos los puntos siguientes y con la finalidad de poder observar el desempeño de los distintos interpoladores usar una imagen nítida de baja resolución. Se recomienda usar imágenes desde 128x128 hasta 256x256 pixeles como máximo y la imagen pentagon que se encuentra disponible en la sección de imágenes del curso.
1. Sobremuestro espacial:
Obtenga el sobremuestreo de la imagen original insertando ceros entre los pixeles de la misma con factores $T\uparrow = 2\times2$ y $T\uparrow = 4\times4$.
Obtenga la magnitud del espectro de la DFT (abs) de cada una de las imágenes sobremuestreadas y de la imagen original. Despliegue los resultados en una misma figura para efectos de comparación (en Matlab se puede usar el comando subplot). Recuerde usar fftshift para centrar los espectros y una función de escalamiento para el despliegue, ejemplo: ImFDespliegue=log(1.0 + ImF), donde ImF es la DFT de la imagen con sobremuestreo.
Explique el efecto del sobremuestreo espacial en el espectro de la DFT.
¿Qué sucedería si se intercalaran ceros entre los valores del espectro de la DFT?
2. Interpolación espacial. Interpole las imágenes con sobremuestreo obtenidas en el inciso anterior (con factores $T\uparrow = 2\times2$ y $T\uparrow = 4\times4$) usando interpoladores:
De orden cero
Lineal
Cúbico
Para cada caso y factor de interpolación despliegue con alguna función de acercamiento (zoom) una misma región seleccionada de las imágenes interpoladas. Compare y explique los resultados de los distintos interpoladores en una misma figura.
Obtenga la magnitud del espectro de la DFT (abs) de cada una de las imágenes interpoladas. Despliegue los espectros en una misma figura para compararlos. Explique el efecto de los distintos tipos de interpolación en los espectros de las DFTs.
3. Interpolación en frecuencia:
Obtenga la DFT de la imagen original y despliegue su magnitud.
Centre el espectro de la DFT (fftshift) y agregue ceros alrededor del mismo hasta completar y obtener dos espectros cuyas dimensiones correspondan a las dimensiones de la imagen original multiplicadas por los factores de interpolación $T\uparrow = 2\times2$ y $T\uparrow = 4\times4$ respectivamente.
2

Despliegue las magnitudes de las DFTs (abs) con ceros alrededor. Use una función de escalamiento para el despliegue. 23

Compare en una misma figura las magnitudes de las DFTs (abs) de las imágenes interpoladas en el punto 2 con la magnitudes de las DFTs (abs) con ceros alrededor. Explique los resultados y diferencias.

Calcule la inversa de la DFT (IDFT) de las DFTs con ceros alrededor.

4. Compare en una misma figura los resultados de las imágenes interpoladas que se obtuvieron con los cuatro métodos.

4. Resultados

Para cada inciso, desplegar las imágenes obtenidas. Explicar las diferencias encontradas entre los distintos tipos de interpoladores así como el desempeño de los mismos.

5. Código

En esta sección deberán presentar el código fuente del programa en MATLAB (o de la herramienta que se haya utilizado).

6. Conclusiones

Referencias

[1] E. Meijeringa, A Chronology of Interpolation: From Ancient Astronomy to Modern Signal and Image Processing, Proceedings of the IEEE, Vol. 90, No. 3, 2002.

[2] E. Maeland, On the Comparison of Interpolation Methods, IEEE Transactions on Medical Imaging, Vol. 1, No. 3, 1988

Benemérita Universidad Autónoma de Puebla

24

Facultad de Ciencias de la Computación

Procesamiento Digital de Imágenes

Procesamiento de imágenes de documentos antiguos para un OCR

Ignacio Acevedo Carrera

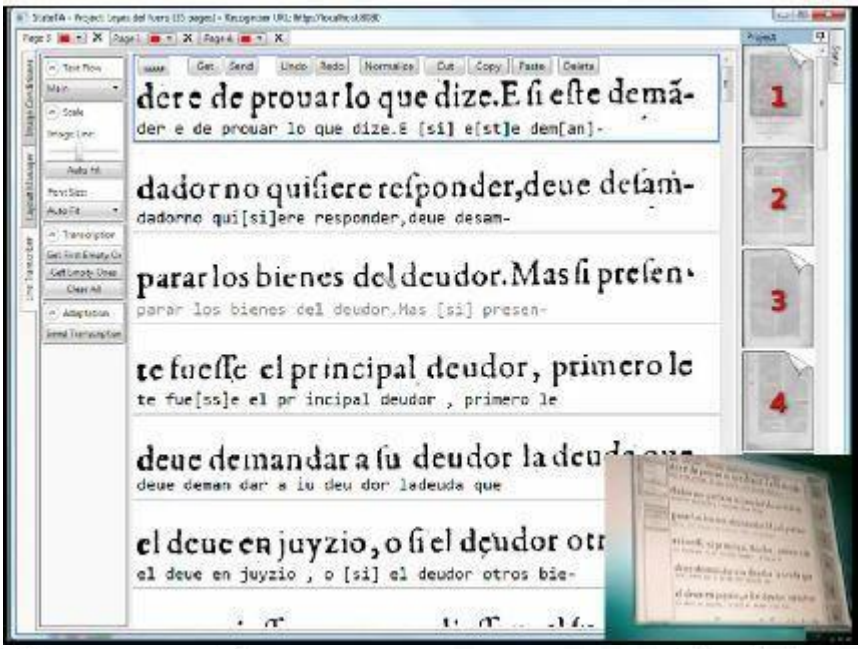
25

zourer@hotmail.com

matricula: 200923911

Resumen:

Hoy día son muchos y muy variados los programas OCR (optical character recognition) que podemos encontrar, esto se debe a que las imágenes donde pretendemos reconocer el texto pueden clasificarse en grupos que representan diferentes retos computacionales. Por una parte podemos disponer de documentos escritos a máquina o computadora y por otra a los escritos a mano; también podríamos clasificarlos como documentos antiguos o modernos; u otra clasificación propia del tipo de documentos que analice el OCR.



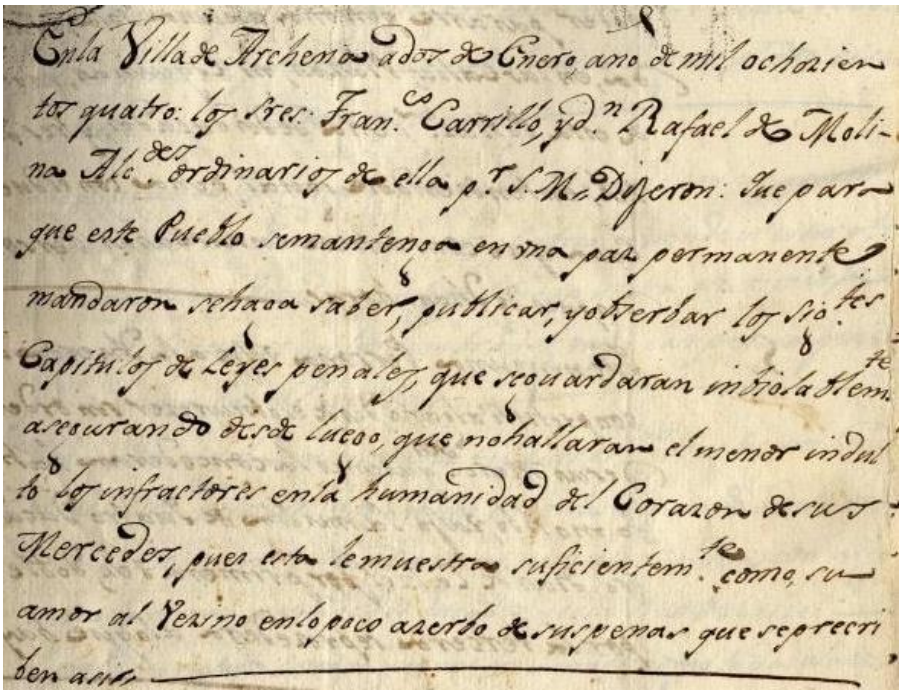
Sin importar la clasificación hay tres factores que los OCR deben solventar sin excepción:

1) La eliminación de ruido en las imágenes para quedarnos solo con las regiones de interés, en este caso las regiones que contienen a los caracteres.

2) La correcta segmentación de la imagen, es decir, separar el documento en zonas o bloques que a su vez estén separadas en renglones y luego en caracteres;

3) El reconocimiento de caracteres propiamente dicho o traducción;

En este documento solo trataremos el punto 1 que es el que se corresponde con PDI (Procesamiento Digital de Imágenes) enfocándonos a documentos antiguos, sin importar si son escritos mecánicamente o a mano.



Los documentos antiguos representan retos interesantes para el PDI, pues suelen verse afectados por muchas fuentes de ruido (humedad, manchas, agujeros, etc) que los hacen más difíciles de segmentar.

Para poder brindar una buena solución al problema, primero analizaremos las características comunes de este tipo de imágenes, luego plantearemos los objetivos y finalmente un algoritmo o propuesta de solución.

Antecedentes

El Reconocimiento Óptico de Caracteres (ROC u OCR por sus siglas en ingles), es un proceso dirigido a la digitalización de textos, los cuales identifican automáticamente a partir de una imagen símbolos o caracteres que pertenecen a un determinado alfabeto, para luego almacenarlos en forma de datos.

En los últimos años la digitalización de la información (textos, imágenes, sonido, etc) ha devenido un punto de interés para la sociedad. En el caso concreto de los textos, existen y se generan continuamente grandes cantidades de información escrita, tipográfica o manuscrita en todo tipo de soportes.

Problemas con el Reconocimiento Óptico de Caracteres

El proceso básico que se lleva a cabo en el Reconocimiento Óptico de Caracteres es convertir el texto que aparece en una imagen en un archivo de texto que podrá ser editado y utilizado como tal por cualquier otro programa o aplicación que lo necesite.

Partiendo de una imagen perfecta, es decir, una imagen con sólo dos niveles de gris, el reconocimiento de estos caracteres se realizará básicamente comparándolos con unos patrones o plantillas que contienen todos los posibles caracteres. Ahora bien, las imágenes reales no son perfectas, por lo tanto el Reconocimiento Óptico de Caracteres se encuentra con varios problemas:

- El dispositivo que obtiene la imagen puede introducir niveles de grises al fondo que no pertenecen a la imagen original.
- La resolución de estos dispositivos puede introducir ruido en la imagen, afectando los píxeles que han de ser procesados.

- La distancia que separa a unos caracteres de otros, al no ser siempre la misma, puede producir errores de reconocimiento.

- La conexión de dos o más caracteres por píxeles comunes también puede producir errores.

Esquema básico de un algoritmo de Reconocimiento Óptico de Caracteres

1) Bancarización: La mayor parte de algoritmos de ROC parten como base de una imagen binaria (dos colores) por lo tanto es conveniente convertir una imagen de escala de grises, o una de color, en una imagen en blanco y negro, de tal forma que se preserven las propiedades esenciales de la imagen. Una forma de hacerlo es mediante el histograma de la imagen donde se muestra el número de pixeles para cada nivel de grises que aparece a la imagen. Para binarizarla tenemos que escoger un umbral adecuado, a partir del cual todos los pixeles que no lo superen se convertirán en negro y el resto en blanco.

2) Fragmentación o segmentación de la imagen: Este es el proceso más costoso y necesario para el posterior reconocimiento de caracteres. La segmentación de una imagen implica la detección mediante procedimientos de etiquetado determinista o estocástico de los

contornos o regiones de la imagen, basándose en la información de intensidad o información espacial.

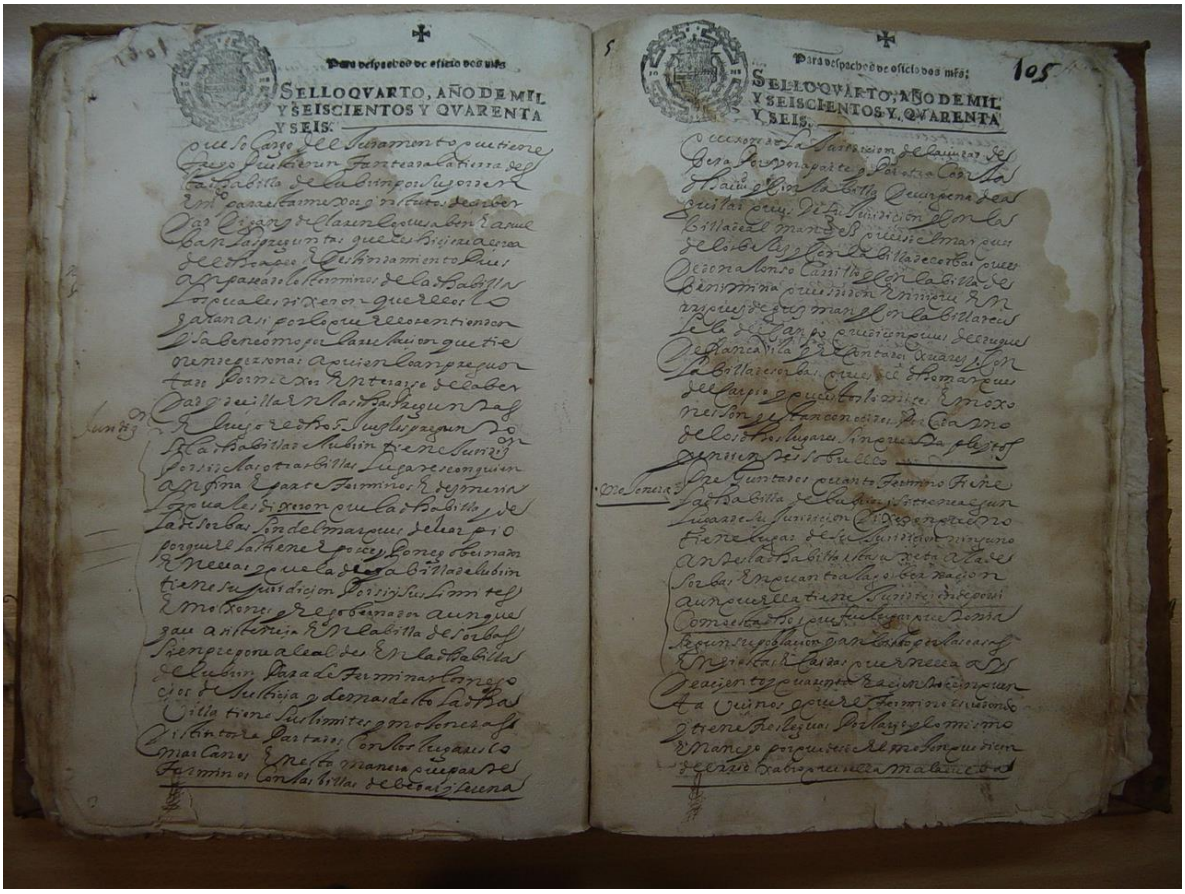
Permite la descomposición de un texto en diferentes entidades lógicas, que han de ser suficientemente invariables, para ser independientes del escritor, y suficientemente significativas para su reconocimiento.

No existe un método genérico para llevar a cabo esta segmentación de la imagen que sea lo suficientemente eficaz para el análisis de un texto. Aunque, las técnicas más utilizadas son variaciones de los métodos basados en proyecciones lineales.

3) Comparación con patrones: En esta etapa se comparan los caracteres obtenidos anteriormente con unos teóricos (patrones) almacenados en una base de datos. El buen funcionamiento del ROC se basa en gran medida a una buena definición de esta etapa. Existen diferentes métodos para llevar a cabo la comparación. Uno de ellos es el Método de Proyección, en el cual se obtienen proyecciones verticales y horizontales del carácter por reconocer y se comparan con el alfabeto de caracteres posibles hasta encontrar la máxima coincidencia.

   Existen otros métodos como por ejemplo: Métodos geométricos o estadísticos, Métodos estructurales, Métodos Neuro-miméticos, Métodos Markovianos o Métodos de Zadeh.

Reconocimiento de texto manuscrito



Un manuscrito es un documento que contiene información escrita a mano sobre un soporte flexible y manejable (por ejemplo: el papiro, el pergamino o el papel), con materias como la tinta de una pluma, de un bolígrafo o simplemente el grafito de un lápiz.

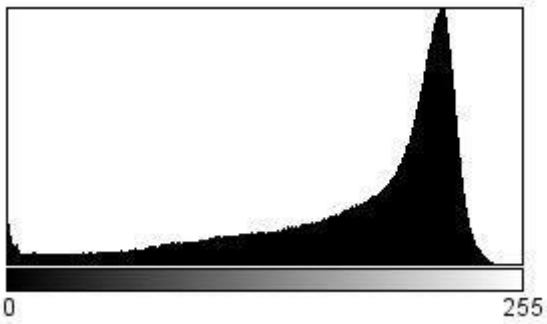
Las dificultades que podemos encontrar a la hora de reconocer un texto tipografiado, no se pueden comparar con las que aparecen cuando queremos reconocer un texto manuscrito.

El reconocimiento de un texto manuscrito continúa siendo un desafío. Aunque el texto se compone básicamente de caracteres individuales, la mayoría de algoritmos ROC no consiguen buenos resultados, ya que la segmentación de texto continuo es un procedimiento complejo, sobre todo en documentos antiguos ya que suelen estar plagados de imperfecciones, como variaciones en el color o textura del papel, manchas, hoyos o rasgaduras, incluso la difuminasion del texto.

Desarrollo del tema

Debido a que el tipo de documentos que procesaremos serán antiguos, primero analicemos que características tienen y como abordarlas.

Primero que nada, describamos desde el punto de vista computacional que representa la imagen o foto de un documento antiguo.


0
255

a) Es una matriz de pixeles generalmente en formato RGB con tonos sepia o marrones, por lo tanto en su histograma los tonos no llegaran ni a blanco ni a negro, además de ser muy parecidos los canales entre sí.

b) Las intensidades de estos pixeles normalmente tienden a claras o tienden a oscuras; las que tienden a claras tienen mayor probabilidad de representar el fondo y las oscuras de representar el texto.

c) Generalmente el 90% de la imagen es el fondo, tonos claros y muy variados por el ruido propio de la imagen y que hacen difícil localizar el umbral donde comienzan los tonos del texto.

d) Los dispositivos que digitalizaron los documentos suelen meter ruido en la imagen por lo que no se encontraran secciones uniformes de color sin importar que se trate del fondo o del texto.

Conociendo estos puntos podemos plantear metas para aproximar la mayoría de las imágenes de documentos antiguos a imágenes ideales para OCR; como por ejemplo

1) trabajar en escala de grises ya que el color no aporta un valor significativo para el análisis.

2) dispersar un poco el histograma para que los tonos claros se acerquen más al blanco y los oscuros más al negro, teniendo cuidado de no llevar más ruido al texto.

3) Fijar un límite por encima del cual ya no nos importe que tan claro sea un pixel, lo tomaremos como fondo y por tanto podemos llevarlo directamente a blanco.

4) Por debajo de este límite, aplicar técnicas de PDI para quitar el ruido del texto y poder finalmente binarizar la imagen.

Propuesta de solución

Ahora que ya conocemos el reto al que nos enfrentamos, propongo un método para alcanzar el objetivo principal en este documento, que es

binarizar una imagen de un documento antiguo y dejarla lista para la segunda etapa de un OCR.

1) Como ya hemos analizado, el color no nos aporta mucha información útil para el análisis de la imagen (excepto para fijar posiblemente el umbral de binarizacion) por lo que primero haremos una copia a escala de grises de la imagen original.

2) Observando con detalle los histogramas de las imágenes pasadas a escalas de grises de 17 documentos de prueba note que había tres posibilidades que determinarían la estrategia a seguir; primero, la imagen puede ser muy oscura, por lo que fijar un umbral muy alto significaría arrastrar el ruido hacia el texto; segundo, la imagen puede ser muy clara, por lo que fijando un umbral muy bajo, toda la imagen se iría a blanco; tercero, la imagen puede tener muy bien distribuidos los tonos claros y los oscuros, es decir, presentar dos picos muy bien marcados en el histograma, uno al principio y otro al final, por lo que un umbral bajo atraparía los tonos del texto y daría muy buenos resultados.

3) Para abordar la variación los histogramas se pueden obtener buenos resultados haciendo un uso adecuado de la función gamma; si la imagen es muy oscura un valor de

   0.7 para gamma puede aclarar de forma uniforme dejando el texto en nuestro documento prácticamente igual pues suele ser el tono más oscuro en la imagen; si la imagen es muy clara, un valor de 3.5 puede distribuir los tonos menos claros hacia un oscuro que podamos tratar más fácilmente; y si la imagen esta balanceada podemos

39

acentuar tanto los tonos oscuros como los claros aplicando un filtro por regiones de gamma con valores 40

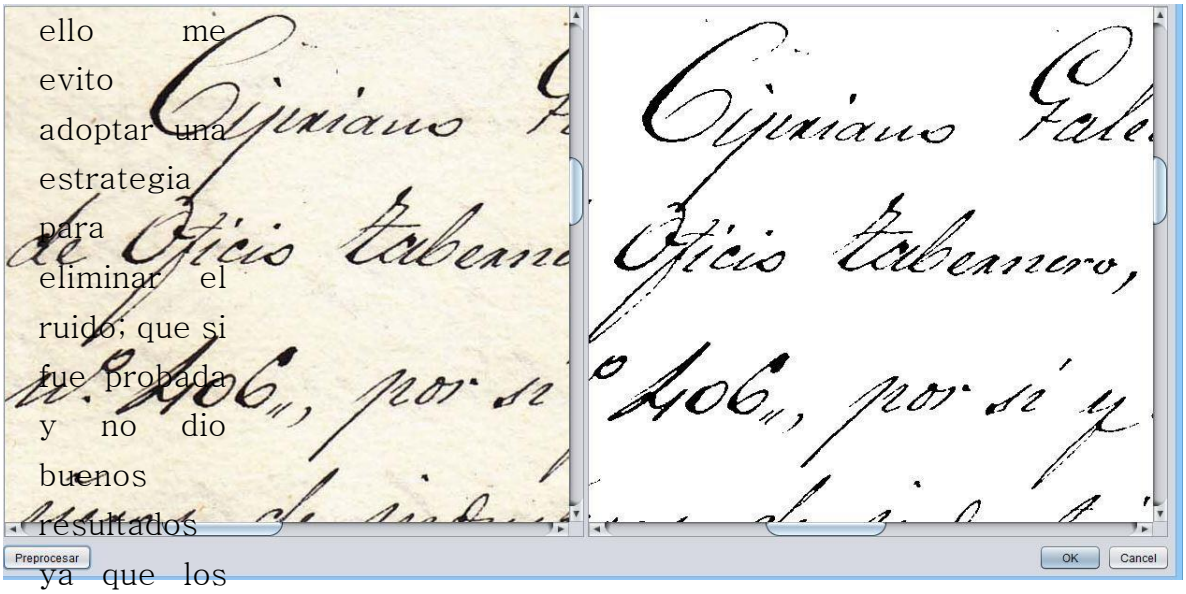
0.7 para los claros y 3.5 para los oscuros, esto facilitara sobremanera la bancarización.

4) En este punto nuestra imagen ya se encuentra casi lista, por lo que opte por buscar a través del histograma los

Conclusiones

tonos más claros en la imagen y llevarlos directamente a blanco, pues con ello me evito adoptar una estrategia para eliminar el ruido; que si fue probada y no dio buenos resultados ya que los filtro para la eliminación de ruido difuminan los bordes y hacen perder definición en los caracteres, sobre todo los escritos a mano pues suelen ser muy finos y sutiles.

5) Se probaron alguna estrategias para elegir el valor umbral, sin embargo no dieron muy buenos frutos; lo que mejor resultados dio fue aplicar el filtro sigmoide en la etapa final; esto produjo una imagen binarizada y con un texto muy bien definido en la mayoría de los casos.



Es importante hacer una análisis previo de las características de las

imágenes que pretendemos procesar, pues esto nos permitirá elegir la mejor estrategia ya que podemos hacer uso excesivo de recursos o tiempo sin ser necesario; para mi caso, invertí mucho tiempo en un filtro optimo que eliminara el ruido en la imagen y sin embargo al final no lo utilice pues me traía más problemas que soluciones.

Resumen

El objetivo del software del procesamiento digital de imágenes para realizar costuras de imágenes y crear una imagen panorámica de mejor calidad y de esa manera agrandar el ángulo de visión.

Introducción

La costura de imágenes es un proceso que tiene una gran importancia por ejemplo para la construcción de imágenes sobre el espacio e incluso para la construcción de mapas y es de suma importancia para la investigación del espacio exterior, una de sus aplicaciones más cansillas es la construcción de una imagen de 360° de visualización, es por eso que la implementación de este proyecto es muy comprometedor con la investigación.

Marco Teórico

Existe una gran diversidad de métodos para la costura de imágenes, y sus principales características es que son costosas computacionalmente, hemos tratado de desarrollar una idea de “costura global” que es eficiente y la velocidad es considerable.

Aplicación del Método de Costura de Imágenes.

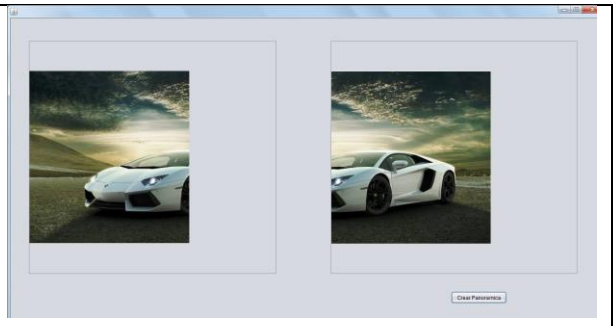
El lenguaje de programación que se utiliza en este proyecto es java y fue creado con la herramienta NetBeans 6.9 que nos ayudo con la interfaz del software.

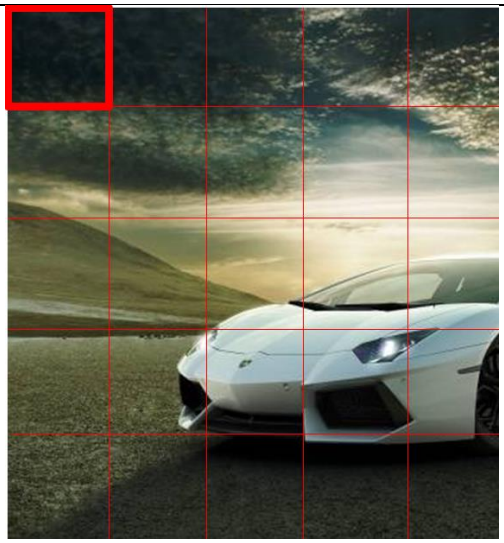
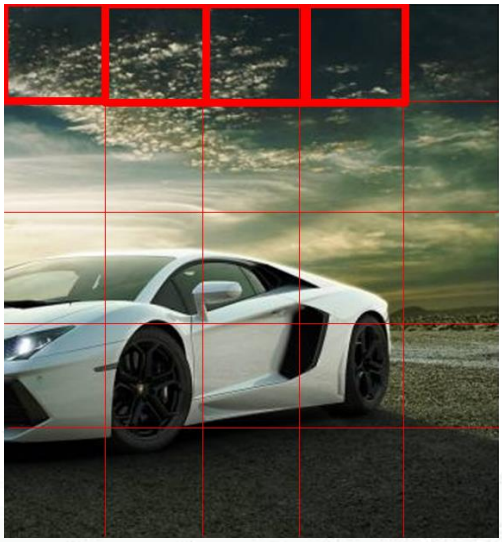
Presentaremos el algoritmo ya desde el punto de vista del software.

Y para empezar utilizamos las siguientes bibliotecas de java.

```
import java.awt.Color;
import static java.awt.color.ColorSpace.TYPE_RGB;
import java.awt.image.BufferedImage;
import java.awt.image.MemoryImageSource;
```

```
import java.io.File;
import java.io.IOException;                                    46
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.filechooser.FileNameExtensionFilter;
```

1. Cargar imagen1 e imagen2	
2. Crear un clone de imagen1 e imagen2	Este se hace internamente sobre el programa con la finalidad de no afectar las imágenes originales. Estas son las imágenes que se comparan es decir con las que se trabaja.

3. Recortar la imagen en partes iguales entre la altura y el ancho de la imagen definida en inicio por 5.	
4. Cada sub-región de imagen1 compararla con todas las sub-regiones de imagen2 y así sucesivamente.	Este método es iterativo se comparan todas las subregiones de la imagen1 con todas de las subregiones de la imagen2 
5. Seleccionar el menor error absoluto que se generó en la comparación.	Aquí el software calcula las subregiones mejor acopladas.

6. Dados los puntos de referencia de la mejor región comparada calcular la nueva altura y el ancho de la imagen.	Este es interno se calcula las nuevas dimensiones de las imágenes su ancho y altura[48]
7. Hacer la reconstrucción de la nueva imagen.	Ya sabemos la mejor subregión y las nuevas dimensiones de la imagen solo queda correr las dos imágenes para su costura. 

Referencias:

An Algorithm for Seamless Image Stitching and Its Application

Jing Xing, Zhenjiang Miao, and Jing Chen

An Efficient Multi-view Image Stitching Algorithm

Ping Zhou and Xiling Luo

BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA

FACULTAD DE COMPUTACION

PROCESAMIENTO DIGITAL DE IMÁGENES

“PARCIAL 1”

PROFESOR: IVAN OLMOS PINEDA

ALLUMNO: JORGE R. SANCHEZ CASTILLO

PRIMAVERA 2014

Introducción

Los operadores regionales es una forma de aplicar los filtros de aclaramiento y obscurecimiento pero añadiéndoles los rangos en que se quiere aplicar dicho filtro, es decir, que en una misma imagen nosotros podemos aplicar diferentes filtros a la vez, cubriendo todos los pixeles con los rangos que son ingresados por el usuario. Este proceso es llamado filtros definidos por segmentos lineales. Para determinar el valor de un píxel, se hace lo siguiente:

Se determina un punto (x, x') como valor de corrimiento

Se definen las ecuaciones de las rectas de los segmentos entre [0,x) y [x,q]

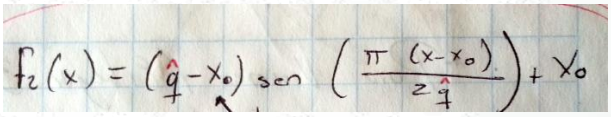
Para cada nuevo valor de un píxel z, se define si z ∈ [0,x) ó z ∈[x,q] y se calcula su nuevo valor con respecto a la ecuación de la recta

Se ocuparon los 6 filtros vistos en clase, seno, coseno, gamma, logaritmo, exponencial y exponencial obscurecimiento.

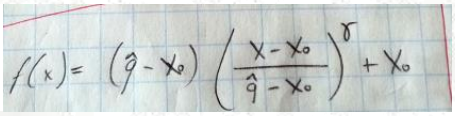
Desarrollo del tema

A nuestra interfaz se le agrego una pestaña llama "Operadores Regionales" , la cual al seleccionarlo nos abrirá una nueva ventana para que el usuario pueda ingresar el numero de filtros que desea aplicar a su imagen, dependiendo del número que ingrese el usuario (mínimo 1 , máximo 5) se abrirá una nueva ventana con los filtros disponibles para el usuario, así como los valores de alfa/gamma y por supuesto los rangos de x0 y x1.

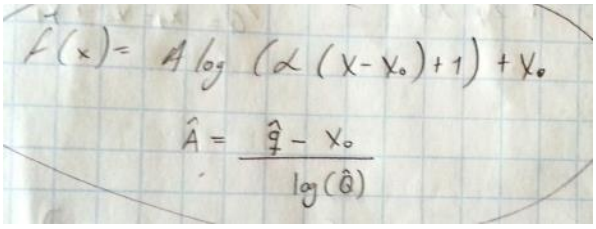
Como ya se menciono los filtros que se utilizaron fueron 6, los cuales se les modificó su función para que pudieran evaluar los puntos x0 y x1. La funciones quedaron de la siguiente manera:
Seno:



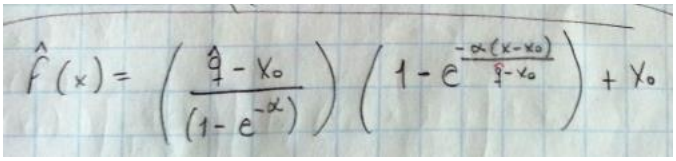
Gamma:



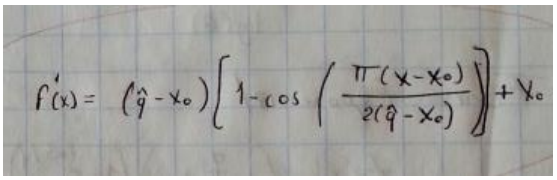
Logaritmo:



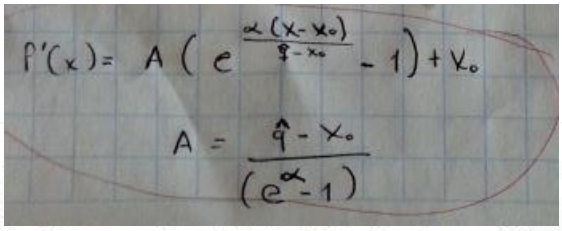
Exponencial:



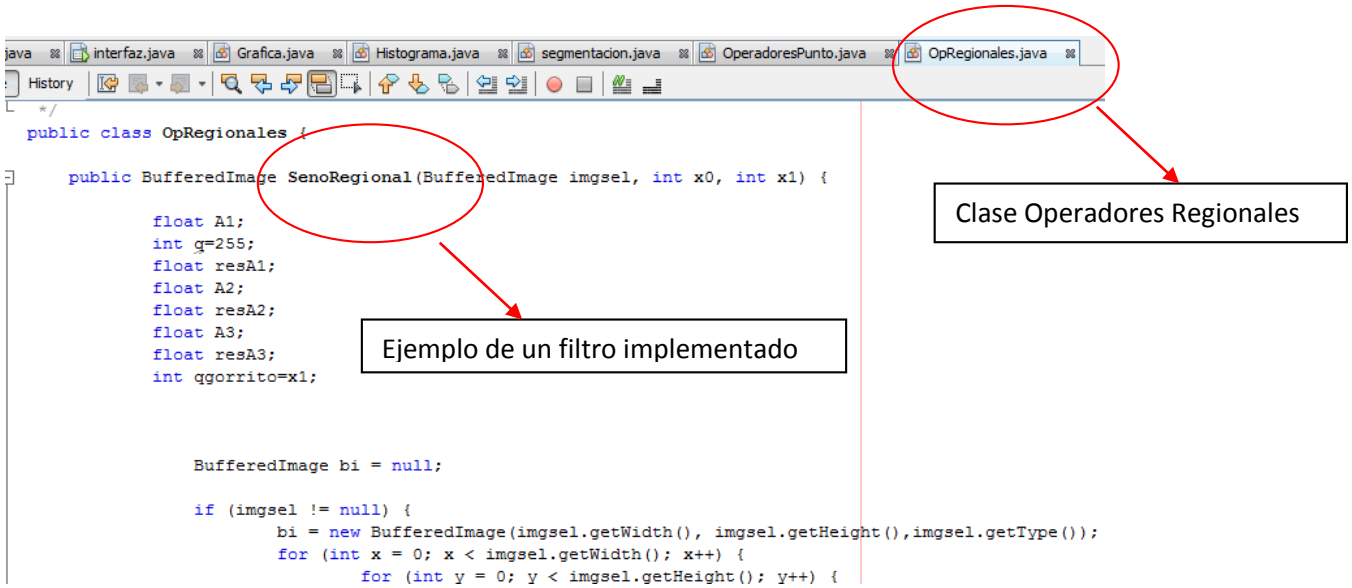
Cosenoidal:



Exponencial Obscurecimiento:



Teniendo estas funciones calculadas a mano, se definió una clase llamada OperadoresRegionales, la cual contiene estas mismas funciones ya implementadas. Agregamos nuevos métodos en la clase “Grafica” llamado para cada función, los cuales reciben como parámetro los rangos y el valor de alfa y/o gamma si son necesarios. Estos métodos se encargan de evaluar los valores de X con respecto a Y.
Por ultimo agregamos un método llamado dibujar grafica la cual se encarga de crear la grafica con el conjunto de datos X,Y y mostrarla en pantalla



Para la interfaz de la opción filtros regionales, se implementaron 5 jCombobox para poder desplegar todos los filtros implementados.
Cuando el usuario selecciona la opción de “seno” o “coseno” el jTextfield de alfa/gamma queda deshabilitado ya que estas funciones no necesitan otro parámetro más que los rangos x0 y x1.

Veamos un ejemplo para el jCombobox 1 cuando el usuario selecciona seno o gama.

```
if (jComboBox1.getSelectedItem()== "Seno")
{

    jTextField4.disable();
    x0=Integer.parseInt(jTextField5.getText());

    x1=Integer.parseInt(jTextField6.getText());
     System.out.println("x0="+x0);
    System.out.println("x1="+x1);
    imgFiltros= freg.SenoRegional(imageActual,x0,x1);
    System.out.println("Filtro 1 SENO");

    //dibujamos la grafica
   reg1.SenoPrueba(x0,x1);
     }

  if (jComboBox1.getSelectedItem()== "Gamma")
     {
    gammaregional=Float.parseFloat(jTextField4.getText());
    System.out.println("gammaRegional:="+gammaregional);
    x0=Integer.parseInt(jTextField5.getText());
    x1=Integer.parseInt(jTextField6.getText());
    System.out.println("x0="+x0);
    System.out.println("x1="+x1);


    imgFiltros= freg.GammaRegional(imageActual,gammaregional,x0,x1);
    System.out.println("Filtro 1 GAMMA");
    reg1.GammaPrueba(gammaregional,x0,x1);
```

Filtro seleccionado

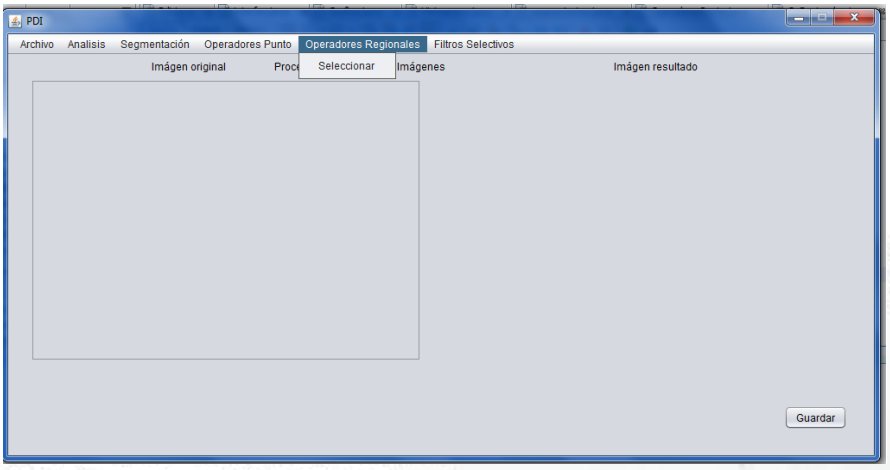
Recibimos los rangos dados por el usuario

Asignamos a la imagen los resultados

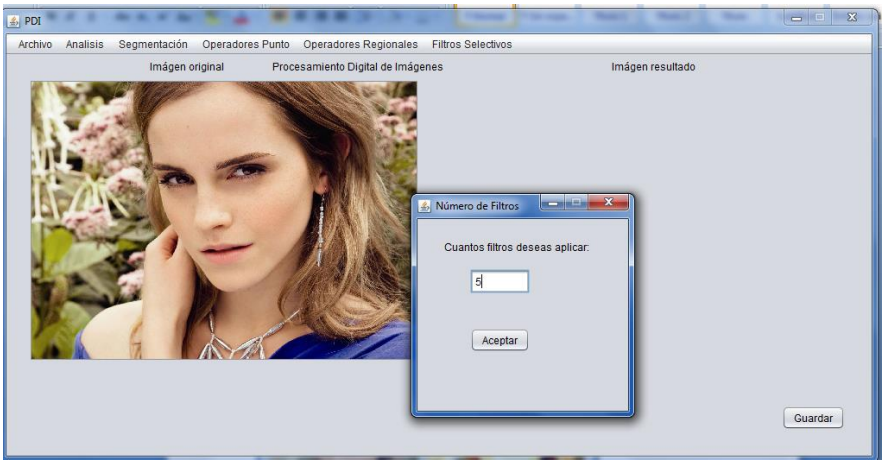
Creamos la grafica de los rangos x0 y x1

Resultados

La interfaz quedó de la siguiente manera:

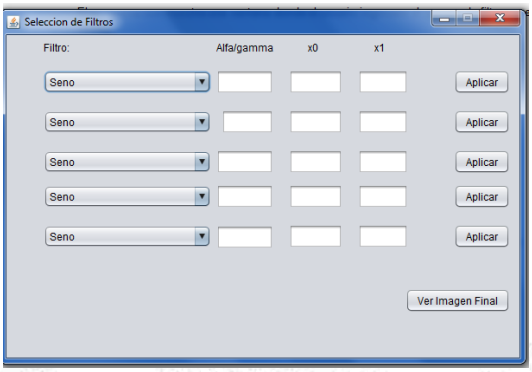


El programa nos muestra una ventana donde el usuario ingresara el numero de filtros que desea.

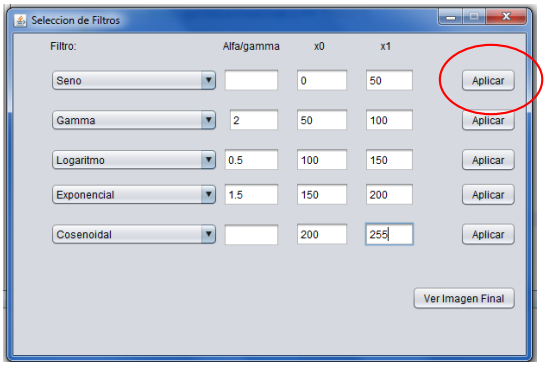


Y posteriormente el programa nos muestra una ventana con los filtros dispoinibles y los parametros de cada uno, una vez que el usuario ingresa todos los datos, el programa muestra los resultads y los filtros graficados en una sola misma grafica, es importante mencionar que el usuario debe cubrir toda la imagen RBG, desde 0 hasta 255.

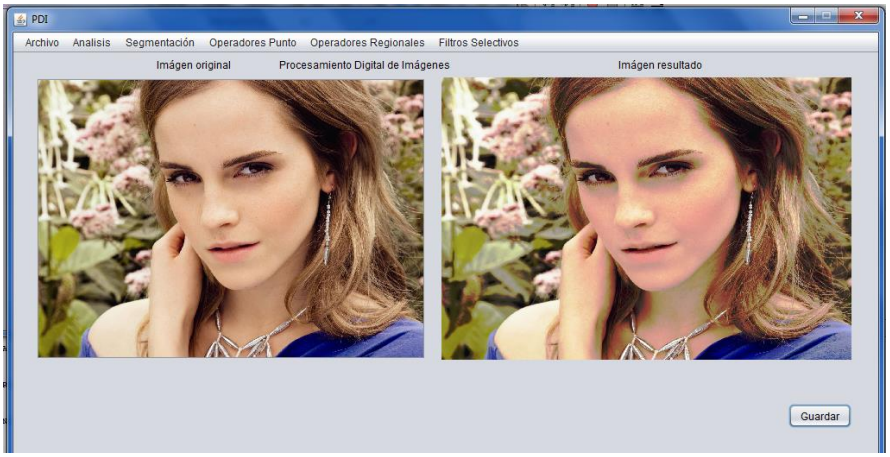
Dependiendo del número de filtros ingresado, es el número de jCombobox que se habilitarán.



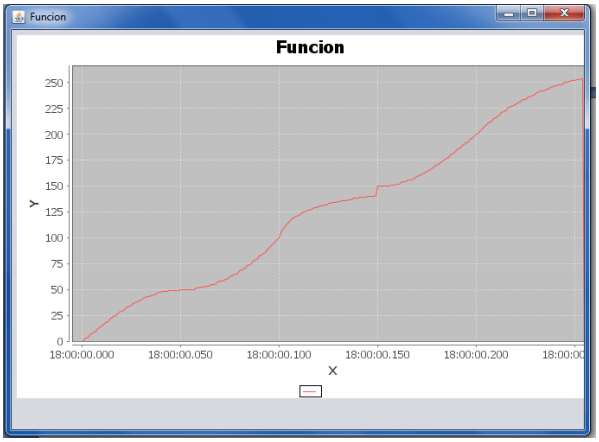
El usuario ingresará los parámetros correspondientes, es importante mencionar que el usuario debe presionar el botón “aplicar” cada que llena un filtro, una vez que ya lleno todos los filtros, seleccionará el botón “ver imagen final” para visualizar los resultados



Aparecerá la imagen resultante con todos los filtros ya aplicados.



El programa nos grafica todos los filtros en una misma grafica que se puede ver de la siguiente manera:



Conclusiones

Para concluir los filtros regionales son de gran ayuda ya que nos ayudan a mejorar notablemente una imagen, ya que podemos aplicar filtros de aclarado u obscurecimiento al mismo tiempo y así visualizar las mejoras inmediatamente sin tener que aplicar un filtro, guardar, y volver a aplicar.

Solo es importante conocer los rangos en los que queremos aplicar algún filtro para así obtener los resultados esperados.

Referencias

http://www.cs.buap.mx/~iolmos/pdi/Sesion5_FiltrosSelectivos.pdf

Notas en clase.

http://www.cs.buap.mx/~mmartin/notas/PDI-MM-Rev.2013.pdf

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

PROCESAMIENTO DIGITAL DE IMAGENES

PROF. IVÁN OLMOS PINEDA

Comparación de algoritmos para la eliminación de ruido en imágenes digitales

Elaborado por: DANIEL SORIANO GRANDE

PRIMAVERA 2014

Introducción:

Los filtros digitales constituyen uno de los principales modos de operar en el procesamiento de imágenes digitales. Pueden usarse para distintos fines, pero en todos los casos, el resultado sobre cada píxel depende de los píxeles de su entorno.

Los filtros digitales tienen distintos objetivos:

- Suavizar la imagen
- Eliminar ruido
- Realzar la imagen
- Detectar bordes

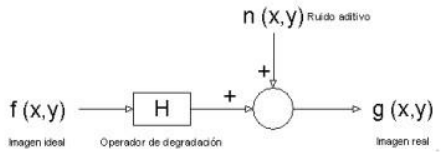
En este documento se trata de comparar los filtros Media, Mediana, Media Gaussiana Fuerte y uno propuesto por mí, que permiten eliminar ruido de una imagen digital.

Conceptos Desarrollados:

Ruido:

Es una información no deseada que contamina la imagen.

La manera en que el ruido influye en una imagen puede describirse gráficamente como:



Ruido Gaussiano

El valor final del píxel es el ideal más una cierta cantidad de error.
Produce pequeñas variaciones en la imagen.

• Suele ser debido a los componentes electrónicos (sensores, digitalizadores...)

• Espectro de energía constante para todas las frecuencias:

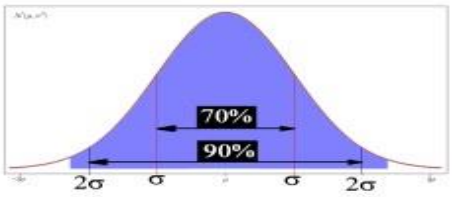
Afecta a la imagen completa.

La intensidad de todos los píxeles se ve alterada.

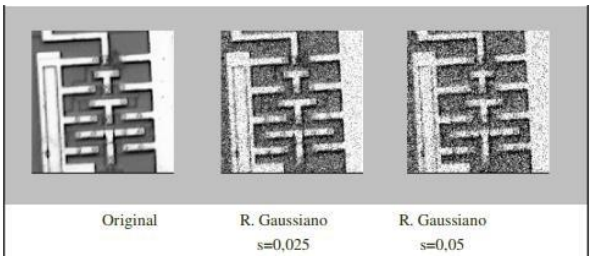
Puede describirse como una variable gaussiana que sigue una distribución normal.

$P(g(x,y)-\sigma < f(x,y) < g(x,y)+\sigma) = 70\%$

$P(g(x,y)-2\sigma < f(x,y) < g(x,y)+2\sigma) = 90\%$



Influencia del ruido gaussiano según la varianza



Filtrado de la mediana

Sustituye el valor del píxel estudiado por la mediana de los valores que engloba una ventana de selección dada.

Ventajas

+ Atenúa el ruido impulsional (Sal y pimienta)

+ Elimina efectos engañosos

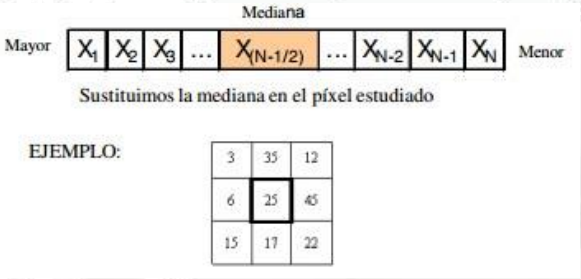
+ Preserva bordes de la imagen

Inconvenientes

- Pierde detalles (Puntos, líneas finas).

- Redondea las esquinas de los objetos

- Desplazamiento de los bordes

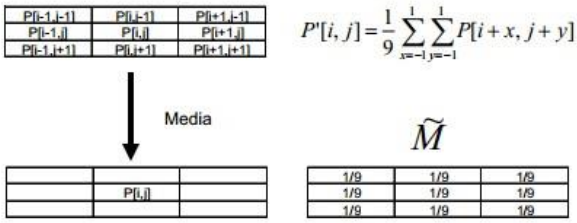


Filtro Media

El filtro de la media es el más simple, intuitivo y fácil de implementar para suavizar imágenes que el de la mediana, es decir, reducir la cantidad de variaciones de intensidad entre píxeles vecinos.

¿Cómo funciona?

Se visita cada píxel de la imagen y se reemplaza por la media de los píxeles vecinos. Se puede operar mediante convolución con una máscara determinada.



$$P'[i,j]=\frac{1}{9}\sum_{x=-1}^{1}\sum_{y=-1}^{1}P[i+x,j+y]$$

$$\widetilde{M}$$

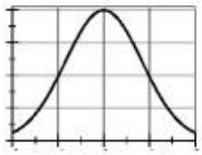
El filtro de la media ofrece ciertas desventajas:

•El filtro de la media es bastante sensible a cambios locales.

•El filtro de la media puede crear nuevas intensidades de grises que no aparecían en la imagen.

Filtro Gaussiana Fuerte

En la media gaussiana, se suele usar la función:



$$y=4e^{-x^2/\sqrt{2}}$$

$$\widetilde{M}_{Gauss}(P[i,j])=\frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}$$

$$P'[i,j]=\widetilde{M}_{MP}(P[i,j])\otimes\widetilde{P}_3[i,j]=\frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}\otimes\widetilde{P}_3[i,j]$$

Suavizado de Bordes en Imágenes + Filtro Media

Se trata de aplicar un suavizado en los bordes a la imagen basada en la minimización del gradiente y posteriormente aplicarle un filtro de Media para obtener mejores resultados.

El Algoritmo es el siguiente:

1. Para el píxel i-ésimo de la imagen (con i =1,2,…nxm) almacenamos en un vector el valor de las muestras diferenciales calculadas como la diferencia del nivel de rojo, verde, azul del píxel i-ésimo con cada uno de los niveles de rojo, verde, azul de los píxeles de su vecindad.

2. Mediante comparación, para cada uno de los valores del anterior vector construido (en valor absoluto), estudiamos si alguno de ellos excede el valor

diferencial máximo impuesto por el parámetro de entrada, donde este valor diferencial máximo se obtiene como:

Difmax=(1-parametro)*(Niveles-1)

Donde Niveles se refiere al tono máximo de color de la imagen (como es RGB será 256),y parámetro es un valor real en el intervalo [0,1], donde 0 indicaría suavizado mínimo (ya que establecería la diferencia máxima de niveles de color entre dos píxeles contiguos en 255) y 1 haría lo propio referenciando suavizado máximo (ya que no permitiría diferencia de niveles de gris ninguna en la imagen, lo que se traduce en una imagen final de un nivel de gris continuo). En este caso se utilizó parámetro=0.9

3. Si ninguno de los valores del vector, en valor absoluto, excede el diferencial máximo de niveles de color, se vuelve al paso 1 hasta completar el recorrido por la imagen. Si alguno sí supera esta cota, calculamos el nuevo valor de color del píxel a partir del píxel vecino que provoca la máxima diferencia (gradiente), de la forma:

I(i) =(i) + (Difmax - gradiente)

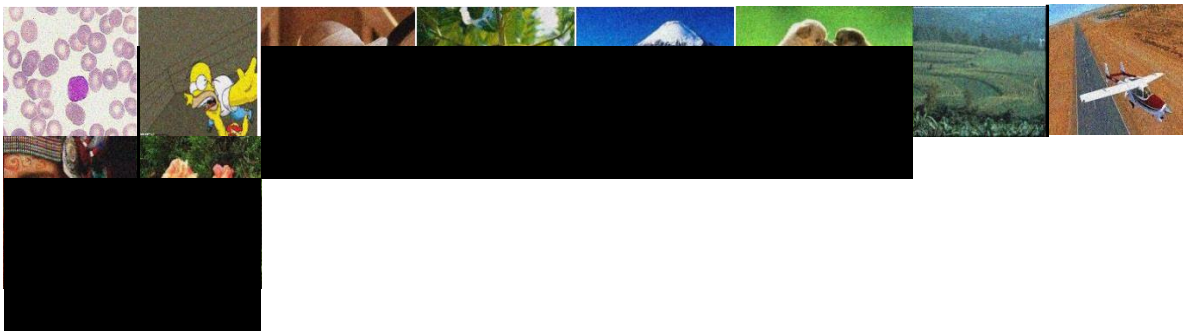
4. Una vez se hayan recorrido todos los píxeles de la imagen con este procedimiento, se aplica el filtro media a la imagen.

Experimentos:

Se tomó una muestra de 10 imágenes:

user

A estas imágenes se les aplico ruido gaussiano con ayuda del programa ImageJ.

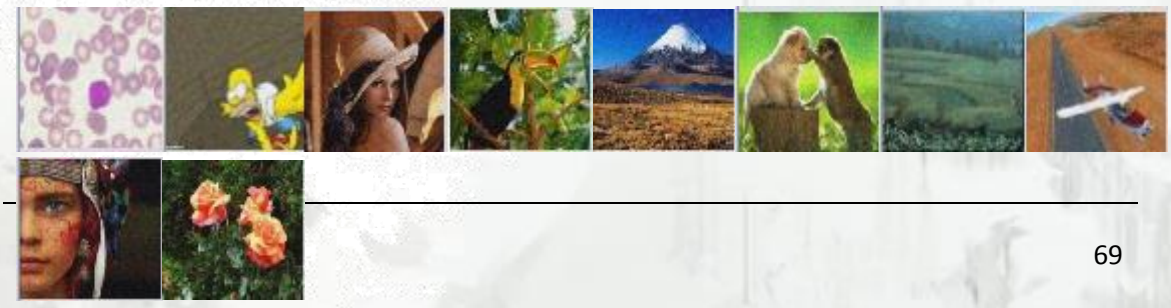


Posteriormente se les aplico cada uno de los 4 filtros explicados anteriormente

Filtro Media

Filtro Mediana

Filtro Gaussiana Fuerte



69

Suavizado de Bordes en Imágenes + Filtro Media

Para hacer una comparación a estos filtros se usó la diferencia de magnitud y la media aritmética como herramientas de apoyo.

71

Diferencia de magnitud:

Dif = | pixeloriginal – pixelfiltro |

Media Aritmética:

$$(\quad)$$

$$(\quad) \qquad (\qquad) \sum (\qquad)$$

Donde n, m son el ancho y alto de la imagen respectivamente.
Obteniendo los resultados mostrados en la tabla siguiente:
F1=Filtro Media

F2=Filtro Mediana

F3= Filtro Media Gaussiana Fuerte

F4= Suavizado de Bordes en Imágenes + Filtro Media

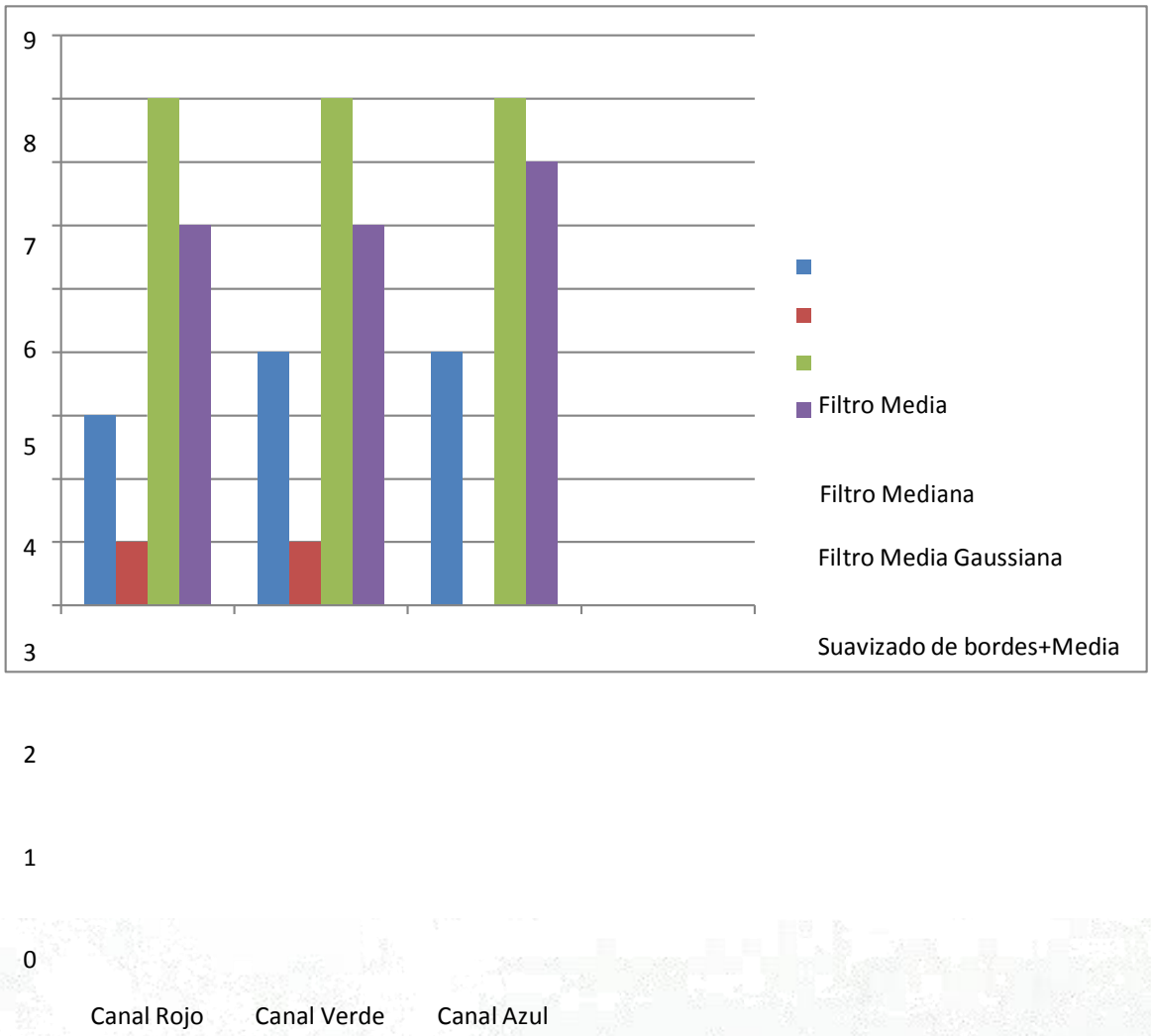
Núm. Imagen	Media por	Filtro				Filtros con media menor
		F1	F2	F3	F4	
1	Rojo	14	11	14	13	F2
	Verde	13	12	13	12	F2,F4
	Azul	9	9	9	9	NINGUNO
2	Rojo	12	14	12	12	F1,F3,F4
	Verde	12	13	11	11	F3,F4
	Azul	11	13	11	11	F1,F3,F4
3	Rojo	9	12	9	9	F1,F3,F4
	Verde	9	12	10	9	F1,F4
	Azul	10	13	10	10	F1,F3,F4
4	Rojo	10	13	10	9	F4
	Verde	10	13	10	10	F1,F3,F4
	Azul	13	15	12	11	F4
5	Rojo	17	19	16	17	F3
	Verde	13	15	12	13	F3

	Azul	13	15	12	13	F3
6	Rojo	7	9	7	7	F1,F3,F4
	Verde	7	9	7	7	F1,F3,F4
	Azul	8	10	8	8	F1,F3,F4
7	Rojo	10	12	9	10	F3
	Verde	9	11	9	10	F1,F3
	Azul	10	11	9	10	F3
8	Rojo	9	10	8	9	F3
	Verde	9	10	8	9	F3
	Azul	9	11	9	9	F1,F3,F4
9	Rojo	12	13	11	11	F3,F4
	Verde	13	14	12	12	F3,F4
	Azul	13	15	12	12	F3,F4

72

10	Rojo	15	16	14	14	F3,F4
	Verde	14	15	12	13	F3
	Azul	15	17	14	14	F3,F4

La siguiente grafica muestra en cada canal la cantidad de veces donde el filtro fue mejor o equivalente a los otros.



De lo anterior se puede concluir que el filtro para eliminar el ruido que mejor se aproximó a la imagen original fue Media Gaussiana Fuerte, de ahí le sigue Suavizado de bordes +Media, Media ,Mediana.

El método que propuse no fue el mejor pero, al quedar en 2do lugar se podría decir que es aceptable**.**

Bibliografía:

http://www.ilopez.es/proyectos/teoriadelasenal/SuavizaBordes.pdf
http://dmi.uib.es/~ygonzalez/VI/Material_del_Curso/Teoria/Tema5_Filtrado.pdf
https://opera-portal.us.es/archivos/pid/entregables/2010-

2011/Grupo9/Grupo9Documentacin15.pdf
http://www.cs.buap.mx/~iolmos/pdi/Sesion7_Suavizado.pdf
http://alojamientos.us.es/gtocoma/pid/tema3-1.pdf

74

https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSe51QMHaAw6n8nVom- peWpSExs4siBNK05BBqCTXSvLUbnwCyQ7g

https://encrypted-tbn1.gstatic.com/images?q=tbn:ANd9GcTA797LWiTRcfmtXVro_HS4wbJ4KDKP64

NDWpQrXKCgCyLtKo2PlA

https://encrypted-tbn3.gstatic.com/images?q=tbn:ANd9GcSinB3beBHZkBsvlazV6s7ZZEqw2wUwQH GKTkO5yx0WyTBfq6-4

https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTqwDk3B94GoMWpINgTc_BQfv- bCptmLxfQ960UxF_qlgurDRQeCg

https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcTAowSw5849eesVn0g7m5FCDg6D4qizFs vozvxf7TYfOdq569sKfQ

https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSSAnIwbSD4oBnfm7yCly8tvYldVoa1_k- ZT0HvQ8E_dAER_yADQA

https://encrypted-tbn1.gstatic.com/images?q=tbn:ANd9GcR32PLL0VYrtVb9Omavteus7fz3jFruJv5_g hmJyBk2dB7stGZ7OQ

https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTGj2aPXIp1RJ-ld2em80FaONp00UhiVuhyzdL4AvaN5vdt2RDe

75

°

GRAFICACIÓN OTOÑO 2014

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
Ingeniería en Cs de la Computación

Modelos de Redes
Prof. Iván Olmos Pineda

Reporte de Programa para Calcular la Latencia de un Grafo de Red

PRESENTA:
Miguel Ángel Ibarra Viveros (201016763)

Fecha de entrega:
26 de septiembre del 2014

OBJETIVO

Mediante algún programa de programación lograr calcular la LATENCIA de un Grafo conexo No dirigido, haciendo referencia a un diagrama de una red; utilizando solo como variable de entrada un archivo de texto con el numero de nodos, no de aristas, las adyacencias de los nodos, la distania entre nodo y nodo, el tamaño del paquete, y la velocidad de transferencia entre nodo y nodo.

DESARROLLO

LECTURA DEL ARCHIVO

Primeramente tenemos que leer el archivo para esto se utlizo Tokenizer para poder separar el archivo en varios tokens y poder leer correctamente cada uno de ellos.

SEPARAR ARCHIVOS

Los primero 2 tokens son el numero de aristas y el numero de nodos asi que asignamos a un arreglo los dos primero valores del archivo. Luego vienen las adyacencias, vienen acomodadas de 6 en 6 asi q separamos todas de esa misma manera. Depues vienen los TC que corresponden al numero de nodos en el grafo, también se separan de la misma manera. Casi al final se encuentran 2 variables, el nodo inicial y el nodo final, eso se guardan como variables especiales pues nos ayudaran a recorrer el grado en general. Al final se encuentra el tamaño del paquete que se enviara, este servirá para poder calcular el Tiempo de Tranmision de todo el grafo.

Para cada tipo de dato se asigno una variable e incluso se crearon una Matriz de Adyacencia y una Matriz de Pesos para poder calcular la Latencia conforme vamos recorriendo el grafo.

BENEMERITA UNVERSIDAD AUTONOMA DE PUEBLA

PROYETO 1 REDES

PROF. DR. IVAN OLMOS PINEDA

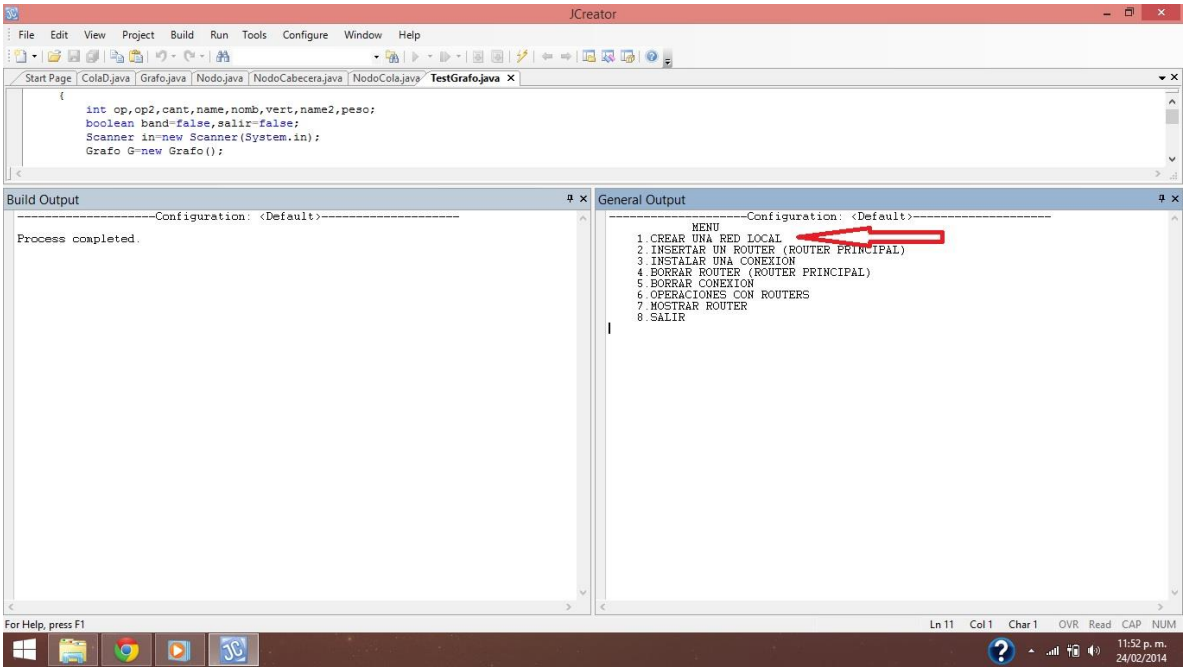
NOMBRE : Alexis Gael Tenorio Ruiz.

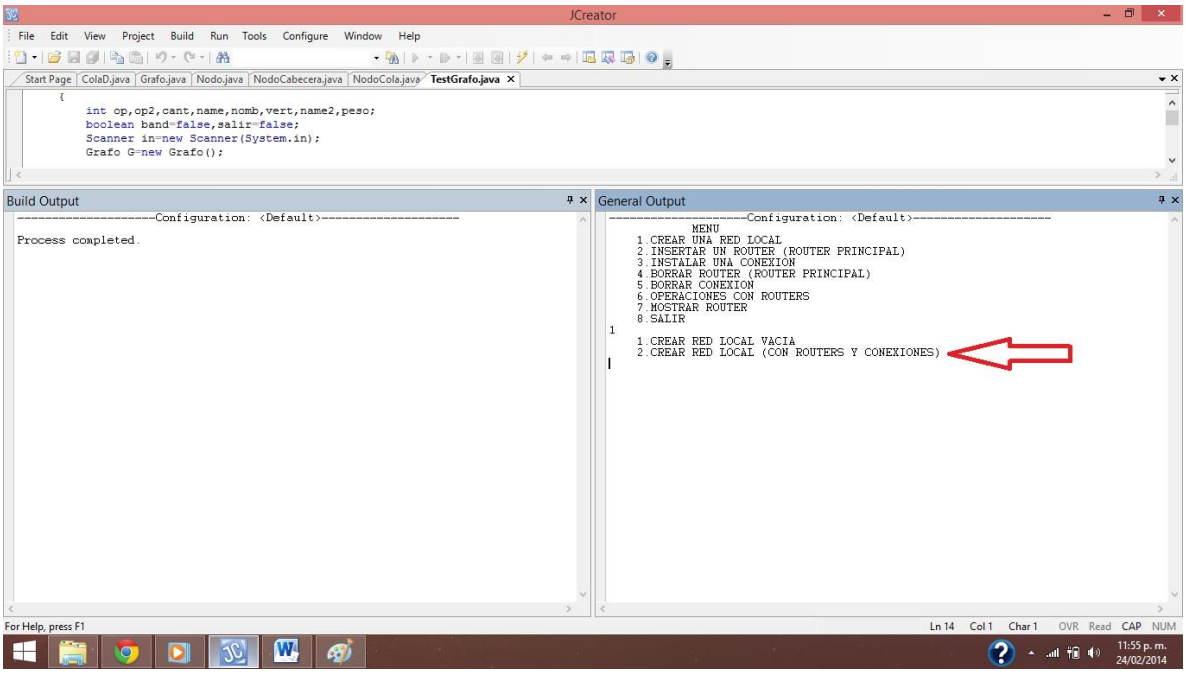
MATRICULA: 201126482.

PRIMAVERA 2014.

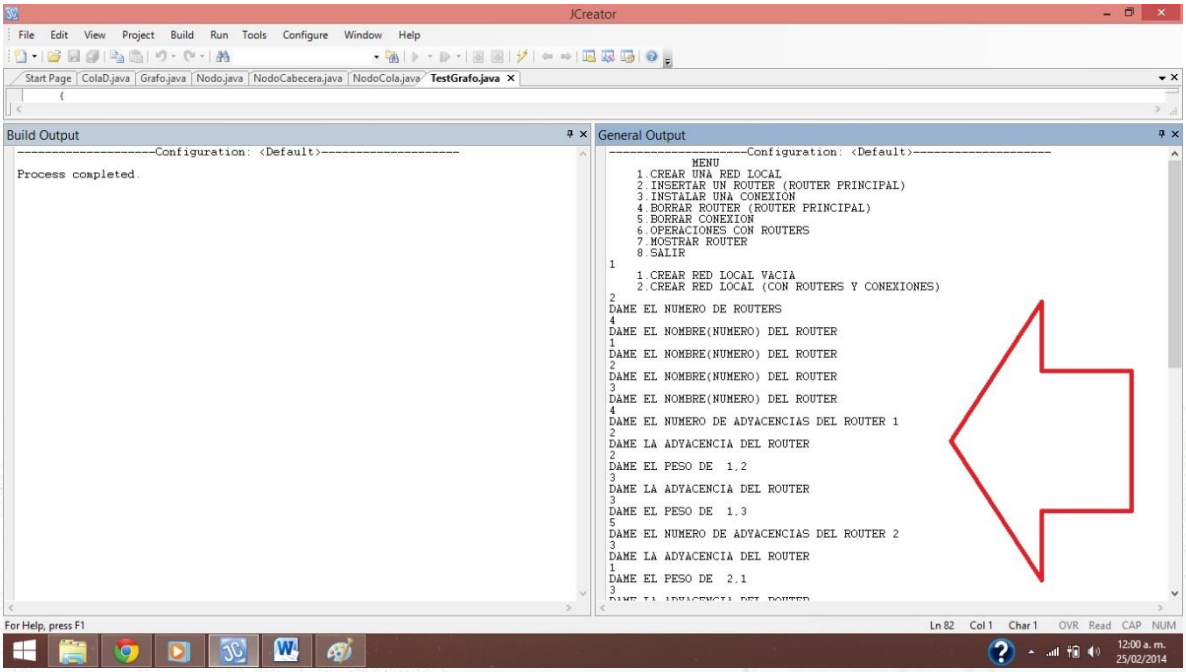
REPORTE DE PROYECTO (CREAR RED LOCAL)

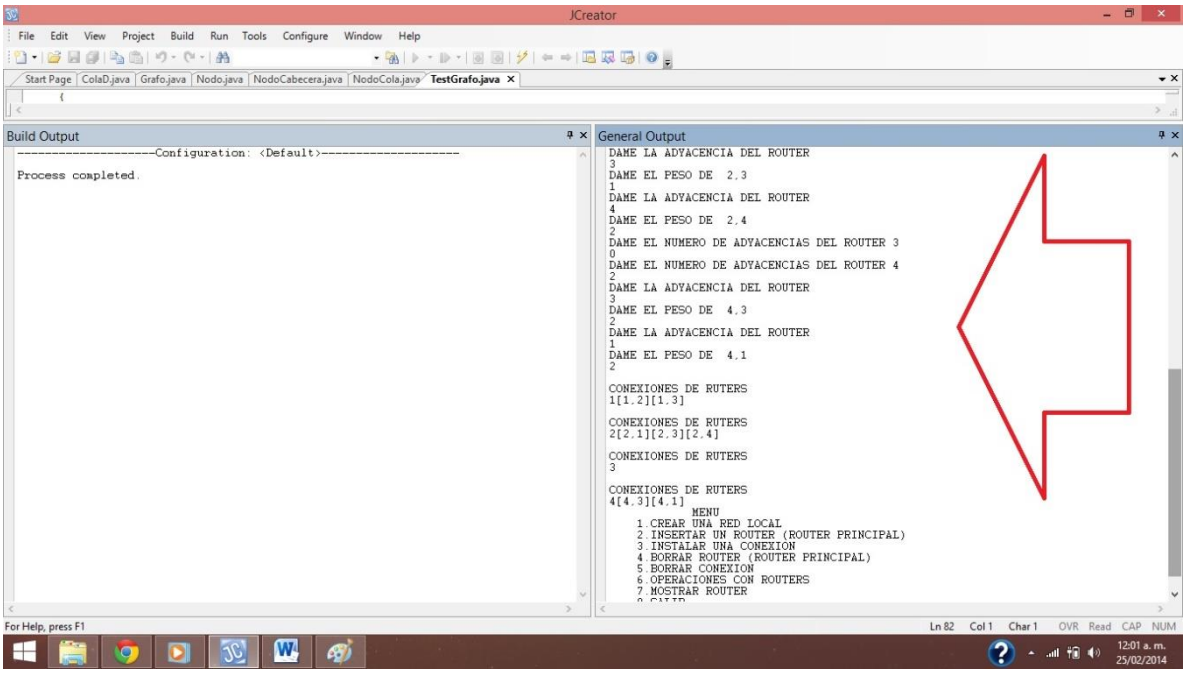
1.- Como primer paso debemos de crear la red local. Después seleccionamos la opción de crear red local (con rúters y conexiones), ya que si seleccionamos la opción crear red local vacía simplemente creara el espacio para la red.



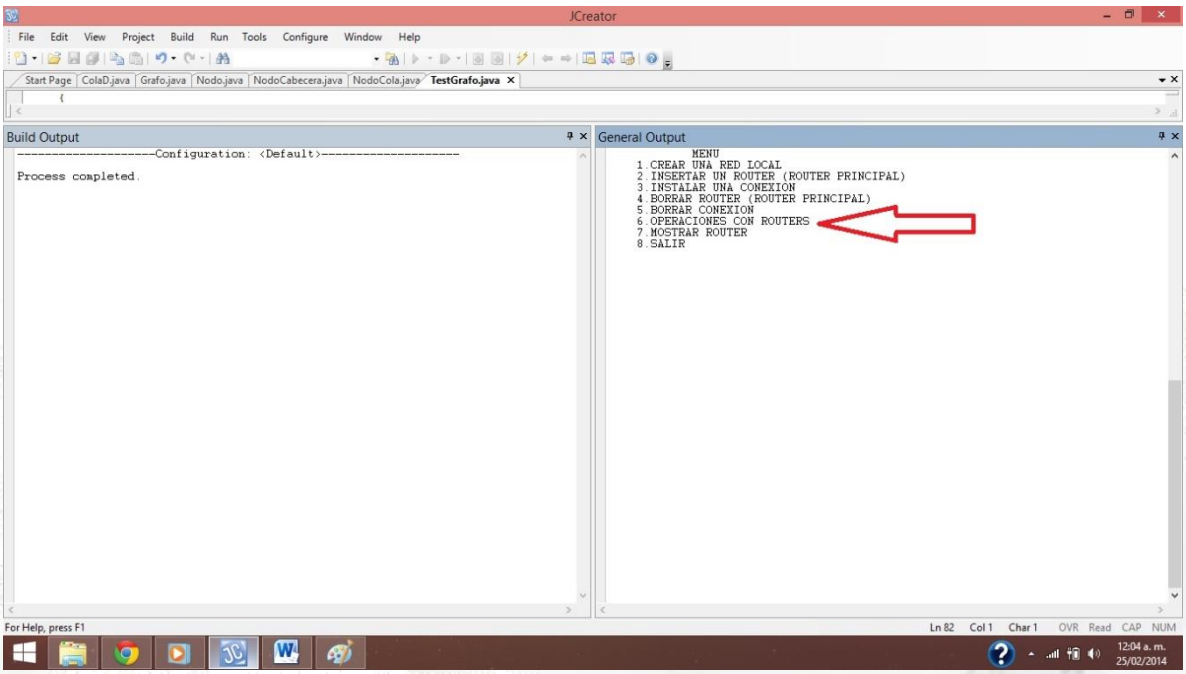


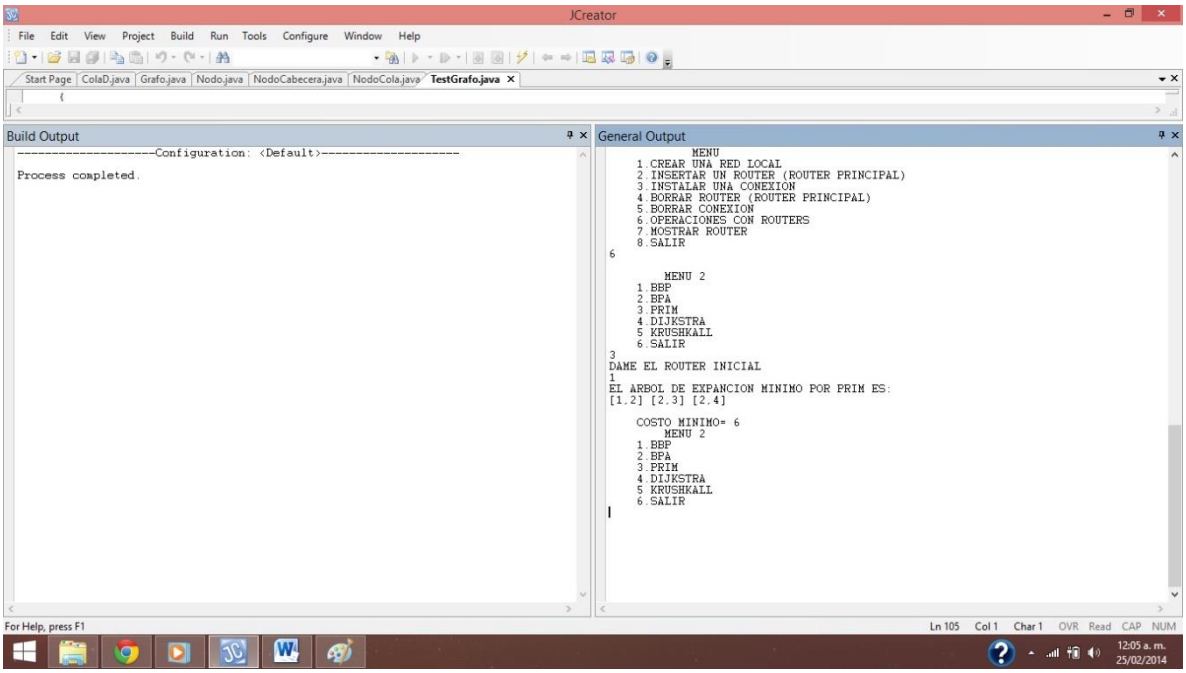
2.- ya creada la red local ingresamos los datos de rúters y conexiones como se muestran a continuación:



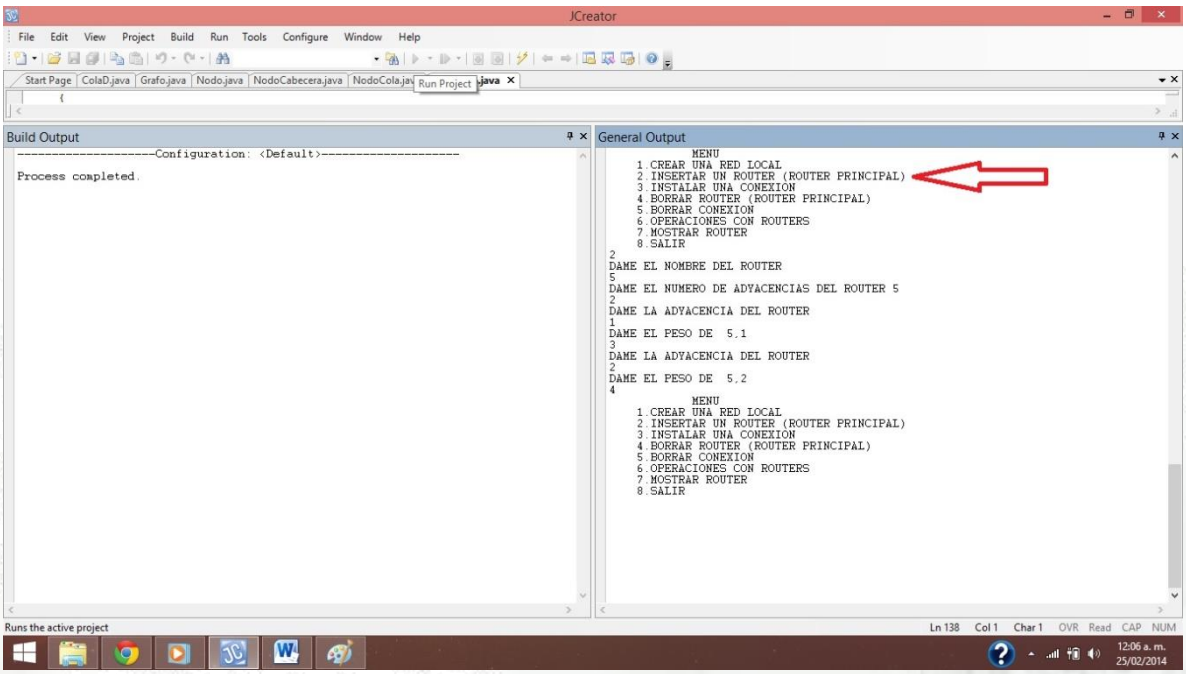


3.- Después con los datos ingresados podemos hacer operaciones para ver el camino más largo , más corto etc.

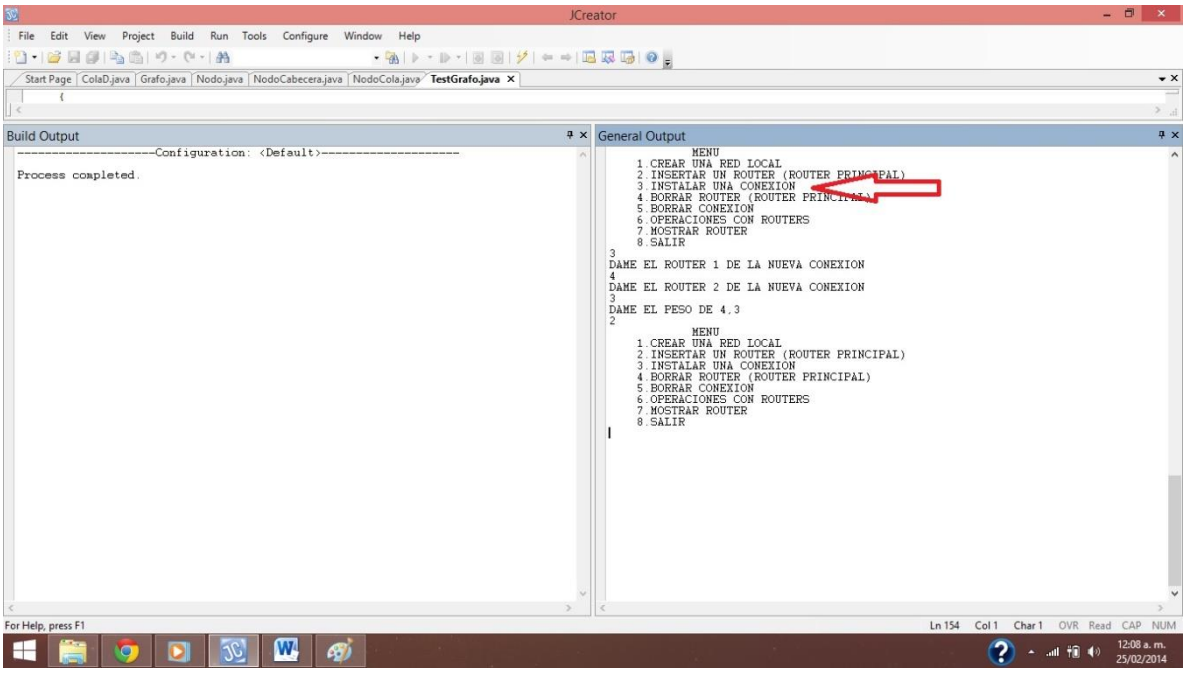




4.- Podemos agregar un nuevo rúter.



5 .- Además de que podemos agregar una nueva conexión.



Y asi es como funciona este servidor.

REDES
OTOÑO 2014

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación 88

Modelo de Redes

Segundo Parcial

Cableado Estructurado

Fecha de entrega: jueves 6 de noviembre del 2014

Equipo:

Barragán Hernández Nancy

Mellado Robles David

Meneses Casarrubias Brian

Morelos Dorantes Iván

Vázquez Juárez Jamelli

Contenido

Introducción

Las comunicaciones y la informática son algunas de las principales variables que determinan la necesidad por parte de las empresas, de contar con proveedores especializados en instalaciones complejas, capaces de determinar el tipo de topología más conveniente para cada caso, y los vínculos más eficientes en cada situación particular. Todo ello implica mucho más que el tendido de cables.

Diseñar una estructura fiable no es fácil, ya que se deben tomar en cuenta varios puntos, desde donde iniciara y hasta qué punto va a proveer a los usuarios, tomando en cuenta que existe cierto margen de error.

Si se está considerando conectar sus equipos de cómputo y celulares a un sitio central desde el cual pueda administrarlos, enlazar sus centros de comunicaciones dispersos en su área geográfica o suministrar servicios de alta velocidad a sus computadoras de escritorio o laptops, debe pensar en el diseño e implementación de infraestructuras de fibra y cableados que cumplirán con éxito todas sus demandas de voz, datos y video. Los sistemas de cableado estructurado constituyen una plataforma universal por donde se transmiten tanto voz como datos e imágenes y constituyen una herramienta imprescindible para la construcción de edificios modernos o la modernización de los ya construidos. Ofrece soluciones integrales a las necesidades en lo que respecta a la transmisión confiable de la información, por medios sólidos; de voz, datos e imagen. La instalación de cableado estructurado debe respetar las normas de construcción internacionales más exigentes para datos, voz y eléctricas tanto polarizadas como de servicios generales, para obtener así el mejor desempeño del sistema.

El siguiente trabajo consiste en proponer un Cableado Estructurado para la facultad de Ciencias de la Computación de la BUAP, en el que se tomara en cuenta los edificios A, B, C y D. dando prioridad a los cubículos de los profesores y a los laboratorios del edificio C, de manera que toda la facultad quede totalmente abastecida con red cableada o inalámbrica.

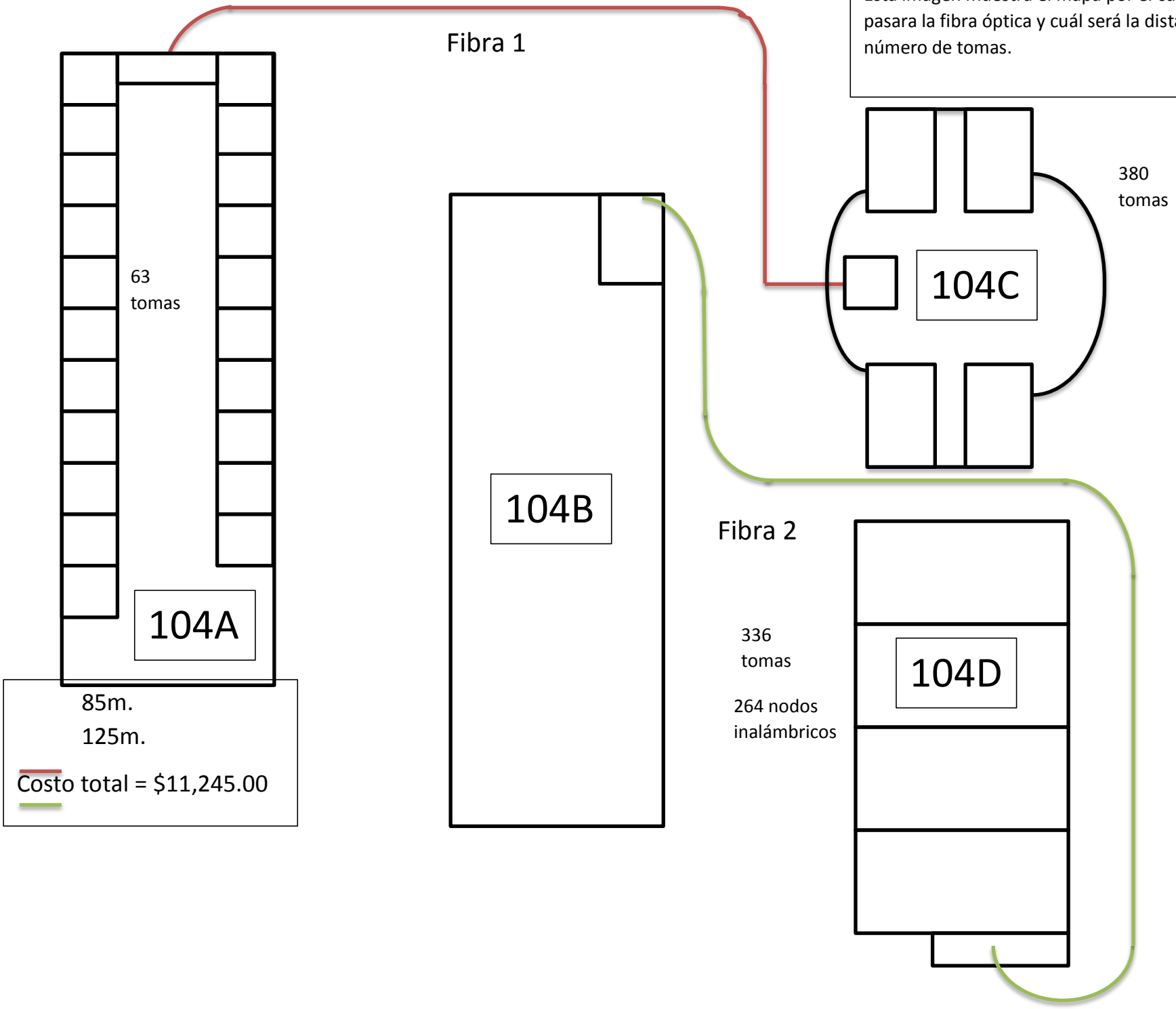
El costo depende del material que se vaya a utilizar y la mano de obra para implementar dicha estructura.

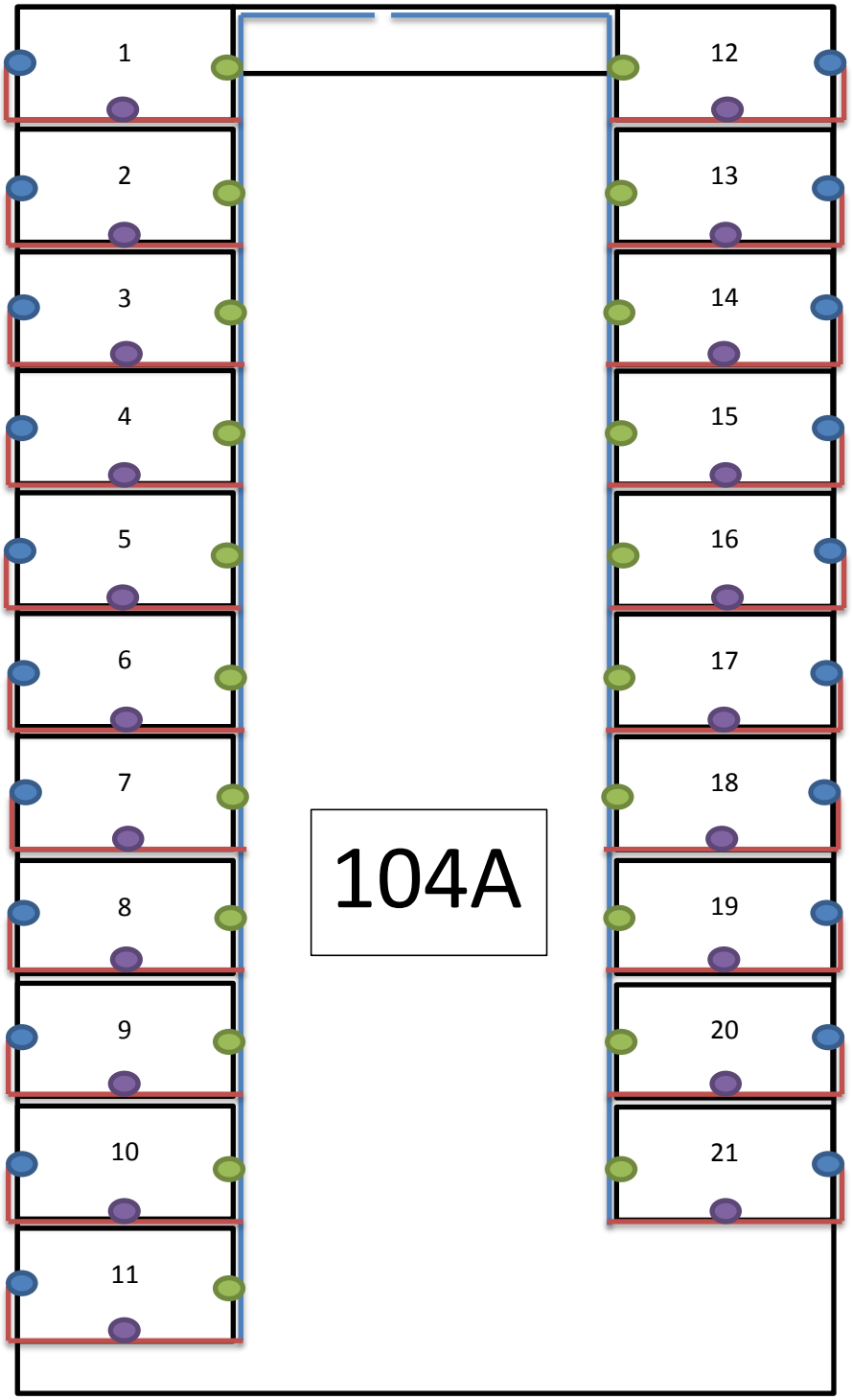
Aquí comienza nuestra propuesta.

En las siguientes imágenes está el diseño de los edificios, ya con las medidas necesarias de fibra óptica y las conexiones que tendrán, además de los costos y lugares donde se encontraran.

Desarrollo

Esta imagen muestra el mapa por el cual pasara la fibra óptica y cuál será la distancia y número de tomas.



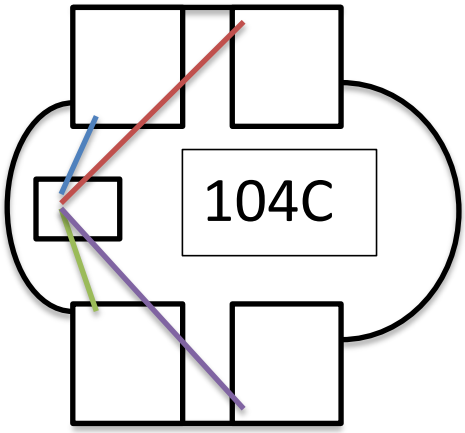


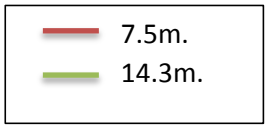
Cables Horizontales UTP Cat6			
	Distancia en metros		
Cubículo			
1	10.05	7.3	4.5
2	12.45	9.7	6.9
3	14.85	12.1	9.3
4	17.25	14.5	11.7
5	19.65	16.9	14.1
6	22.05	19.3	16.5
7	24.45	21.7	18.9
8	26.85	24.1	21.3
9	29.25	26.5	23.7
10	31.65	28.9	26.1
11	34.05	31.3	28.5
12	10.45	7.7	4.9
13	12.85	10.1	7.3
14	15.25	12.5	9.7
15	17.65	14.9	12.1
16	20.05	17.3	14.5
17	22.45	19.7	16.9
18	24.85	22.1	19.3
19	27.25	24.5	21.7
20	29.65	26.9	24.1
21	32.05	29.3	26.5

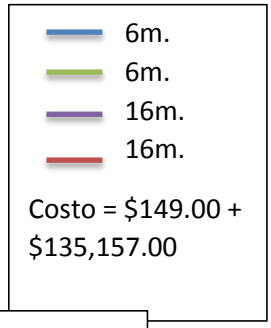
Dispositivos y cables a utilizar				
Nombre	Marca	Modelo	Cantidad	Costo
Router	Cisco	2951 Integrated Services Router	1	$135,157
Switch (80 puertos)	Cisco	Catalyst 2980G-A	1	$122,071
Patch panel (80 puertos)	Cisco	2RU	1	$91,455

Patch cord Cat6	-	-	18.9m
UTP Cat6	-	-	
Costo total		$352,765	

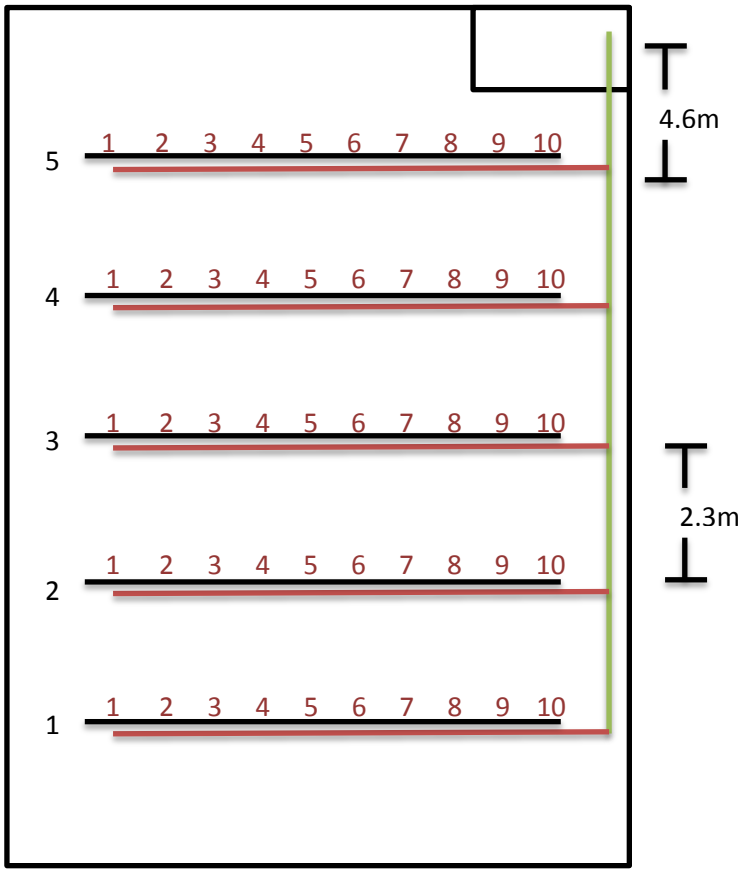
Los puntos de color verde, azul y morado muestran el número de tomas que se harán en cada cubo.







Laboratorio

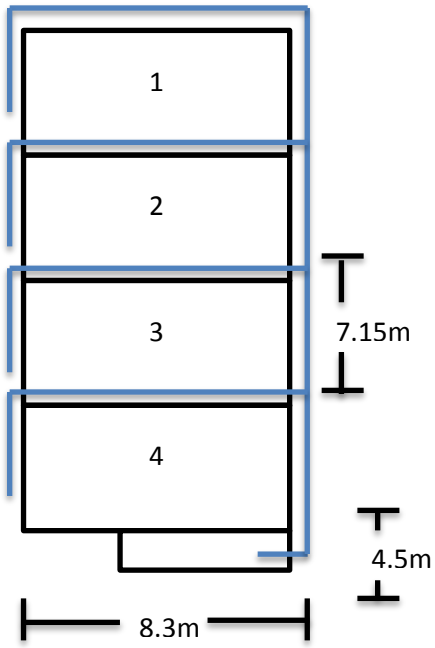


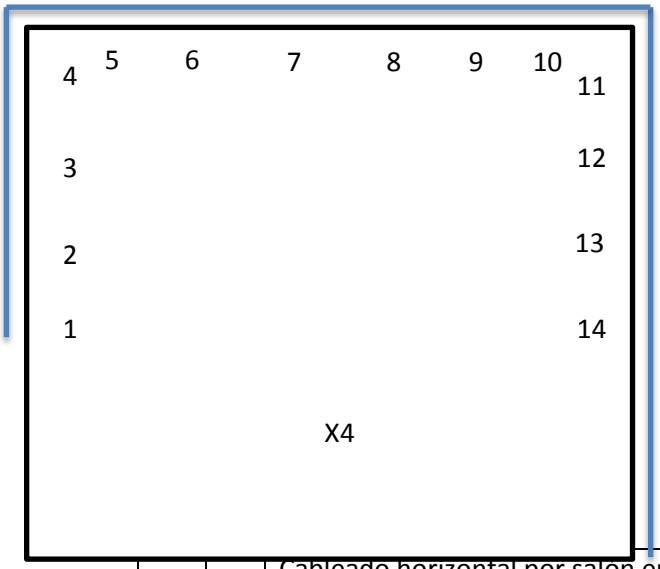
		Cableado horizontal por filas en metros				
Par		1	2	3	4	5
	1	22.5	20.2	17.9	15.6	13.3
	2	21.9	19.6	17.3	15	12.7
	3	21.3	19	16.7	14.4	12.1
	4	20.7	18.4	16.1	13.8	11.5
	5	20.1	17.8	15.5	13.2	10.9
	6	19.5	17.2	14.9	12.6	10.3
	7	18.9	16.6	14.3	12	9.7
	8	18.3	16	13.7	11.4	9.1
	9	17.7	15.4	13.1	10.8	8.5
	10	17.1	14.8	12.5	10.2	7.9

Dispositivos y cables a utilizar				
Nombre	Marca	Modelo	Cantidad	Costo
Router	Cisco	4451 Integrated Services Router	1	$135,157
Switch (80 puertos)	Cisco	Catalyst 2980G-A	2	$244,142
Patch panel (80 puertos)	Cisco	2RU	2	$182,910
Patch cord Cat6	-	-	28.5m	$5,012

UTP Cat6	-	-	1457m	
Costo total	$567,221.00 (Por 4 laboratorios = $2,268,814.00			

104D





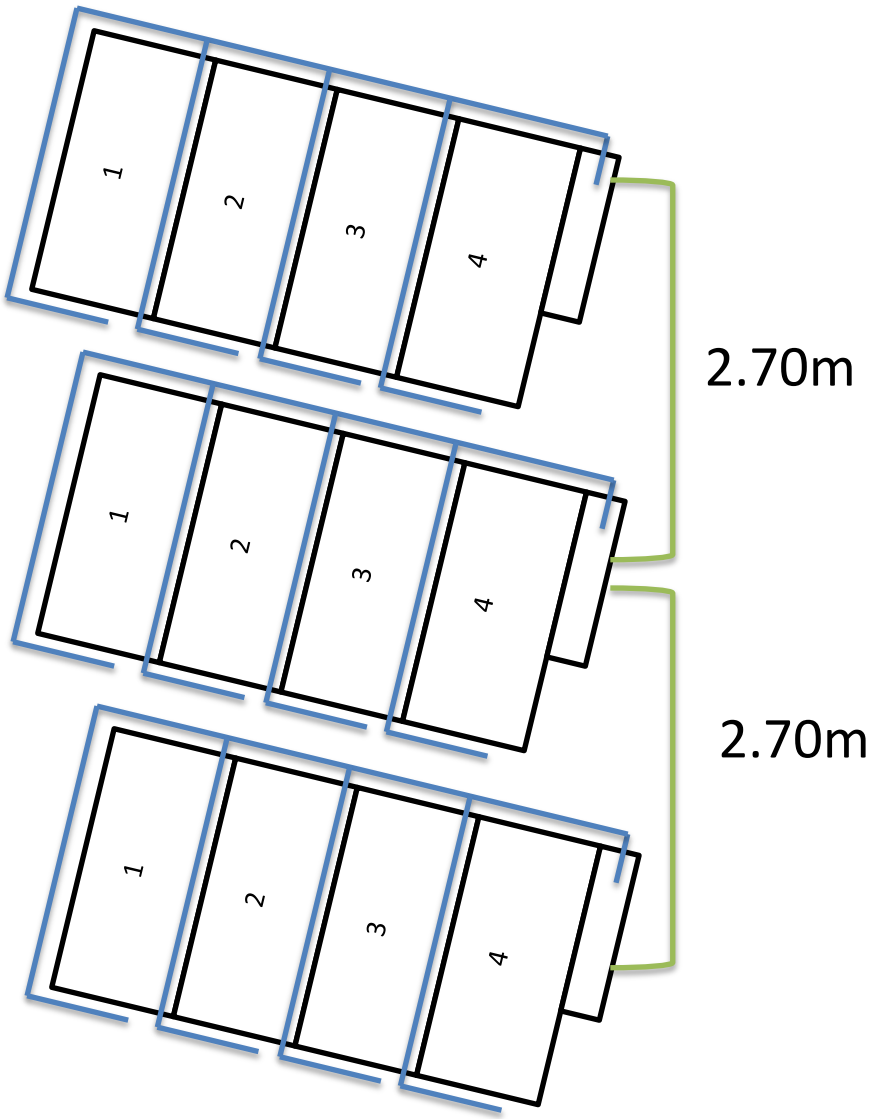
		Cableado horizontal por salón en metros			
		1	2	3	4
Par	1	46.9	39.75	32.6	25.45
	2	45.7	38.55	31.4	24.25
	3	44.5	37.35	30.2	23.05
	4	43.3	36.15	29	21.85
	5	40.8	33.65	26.5	19.35
	6	39.5	32.35	25.2	18.05
	7	38.2	31.05	23.9	16.75
	8	36.9	29.75	22.6	15.45
	9	35.6	28.45	21.3	14.15
	10	34.3	27.15	20	12.85

	11	32.6	25.45	18.3	11.15
	12	31.4	24.25	17.1	9.95
	13	30.2	23.05	15.9	8.75
	14	29	21.85	14.7	7.55

Router Cisco 4451 Integrated Services Router 1 = $135,157 + $28.00 de cables para conexiones entre pisos.

Dispositivos y cables a utilizar				
Nombre	Marca	Modelo	Cantidad	Costo
Router	Cisco	4451 Integrated Services Router	1	$135,157
Access Point	Cisco	WAP121	4	$7,732
Switch (80 puertos)	Cisco	Catalyst 2980G-A	2	$244,142
Patch panel (80 puertos)	Cisco	2RU	2	$182,910

Patch cord Cat6	-	-	33.6m	$10,910
UTP Cat6	-	-	3200.3m	
Costo total	$580,851.00 * 3 pisos = $1,742,553.00			



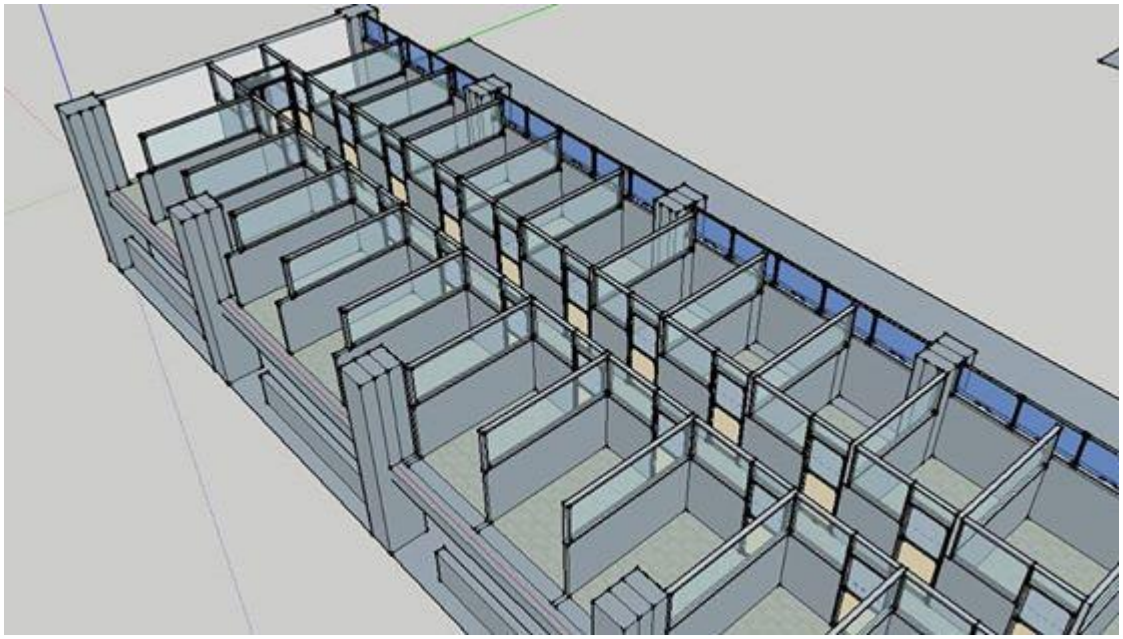
Consideraciones:

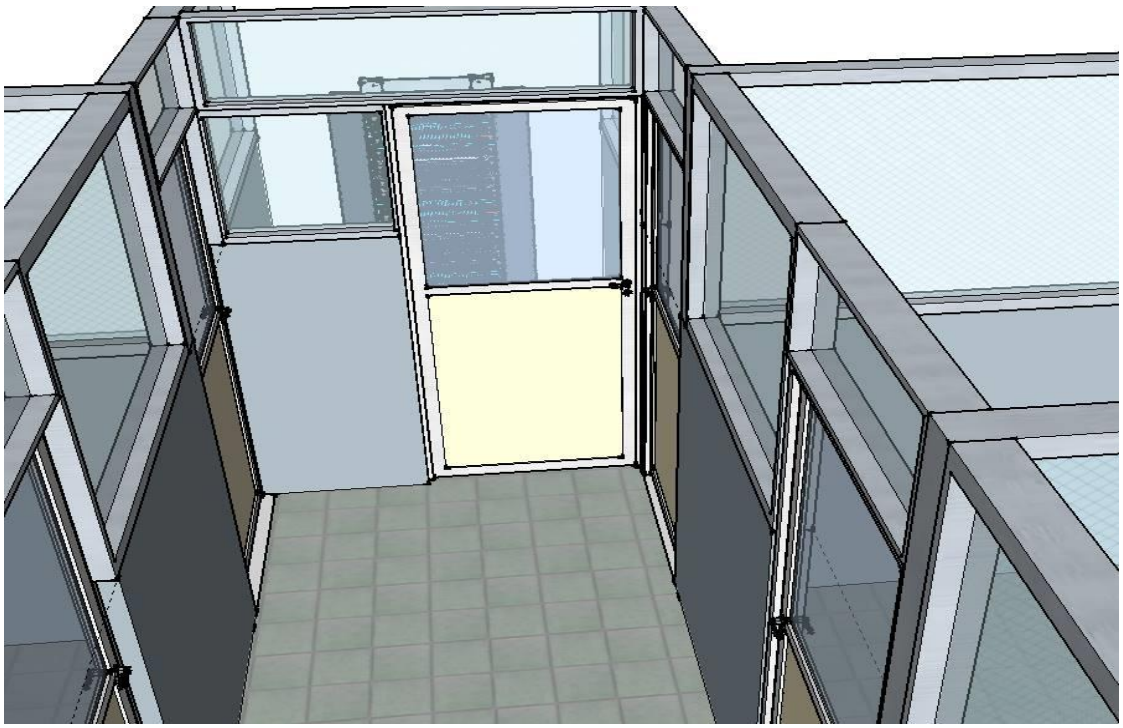
- El cable horizontal estará enterrado 15 cm en el suelo.
- Las tomas estarán 50 cm sobre el suelo.

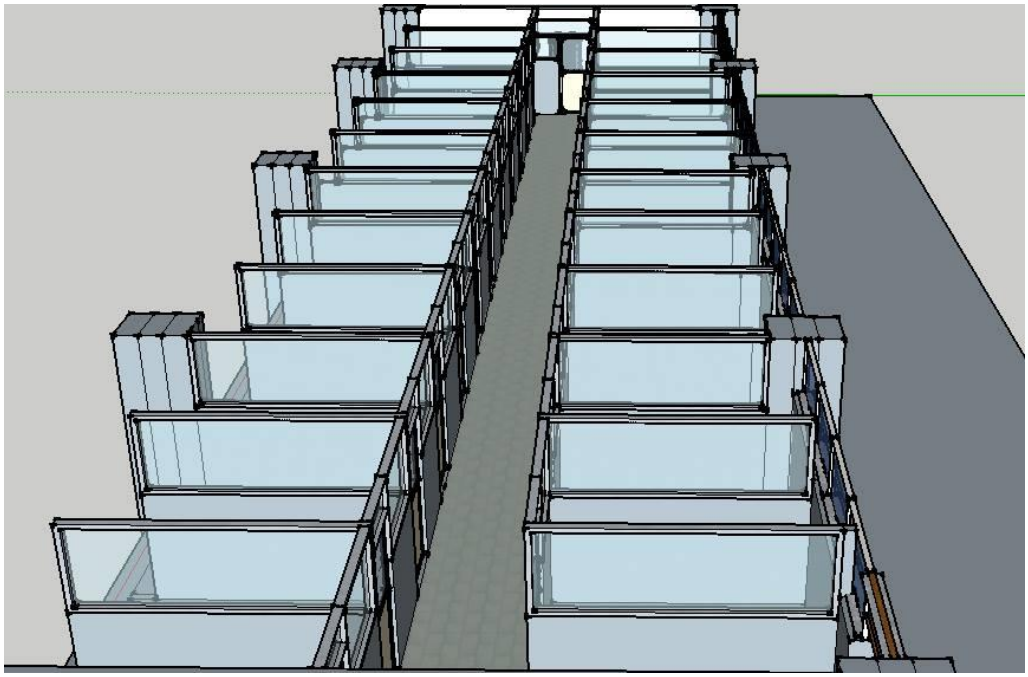
Se diseñaron los edificios de la facultad en 3D para dar una mejor idea del lugar exacto de donde se desea poner cada toma.

Este es el edificio 104A

Cubículos de profesores

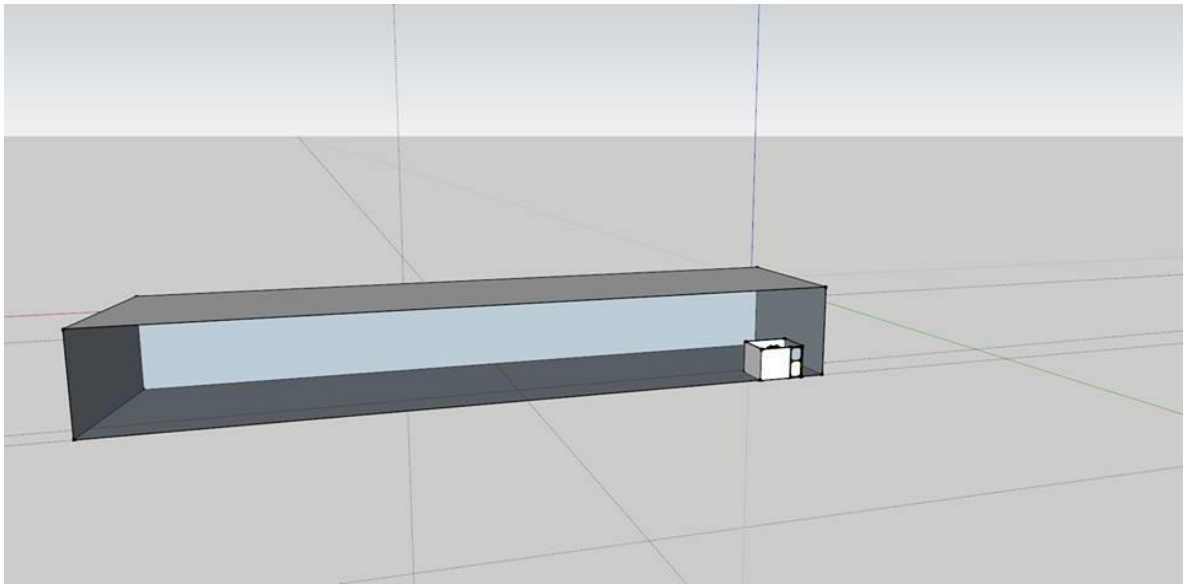


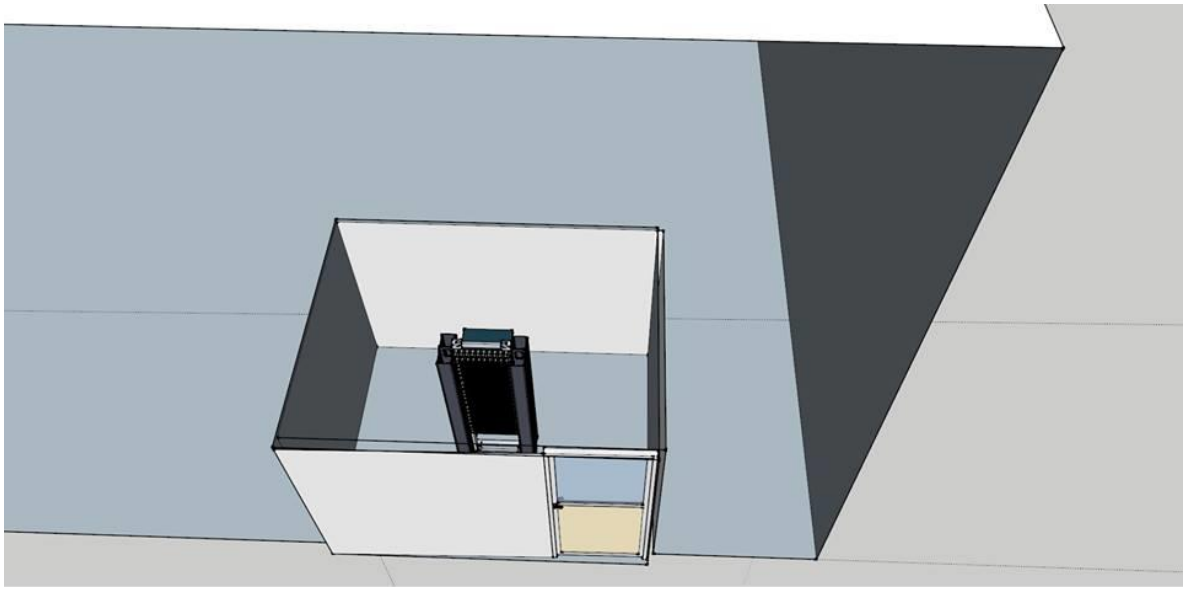


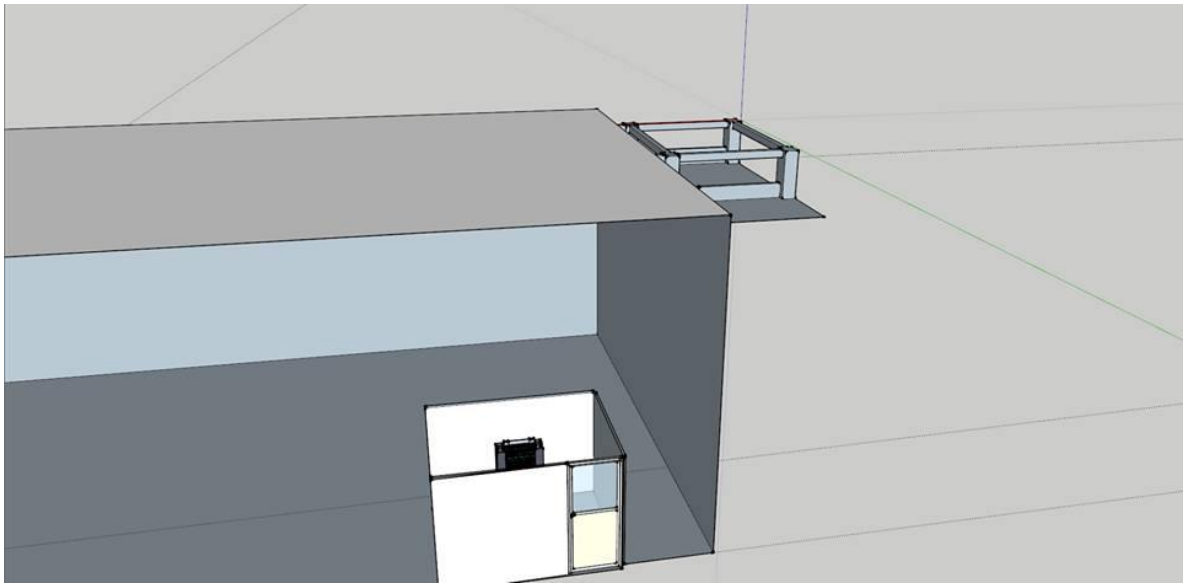


Este es el edificio 104B

En donde se presenta el cuarto de rack, cuarto destinado para el equipamiento informático y de comunicaciones.

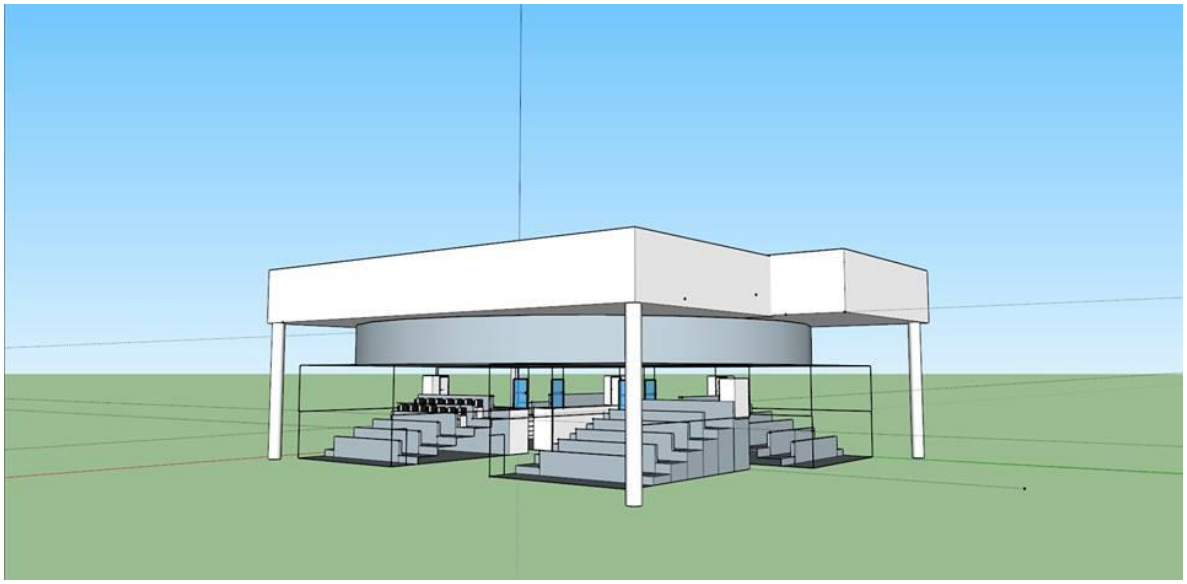




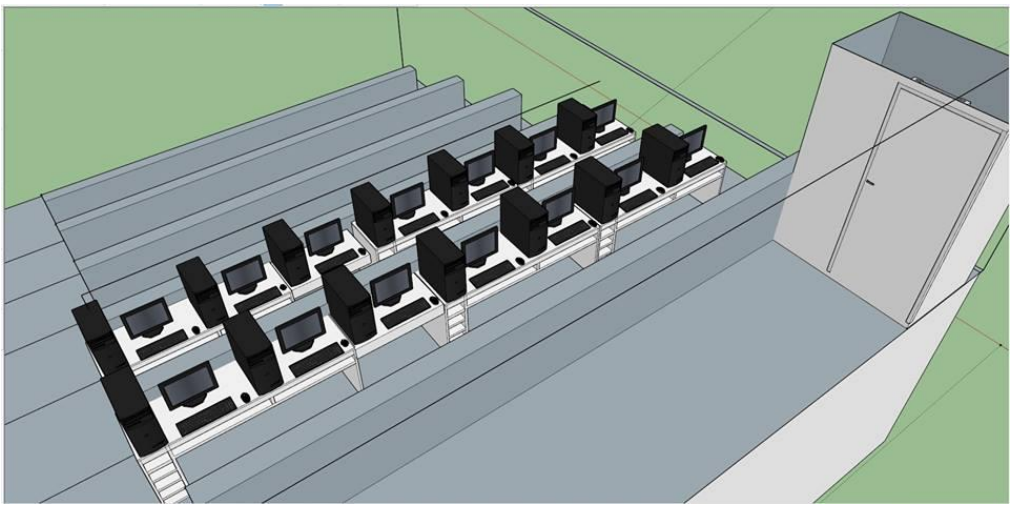


Este es el edificio 104C

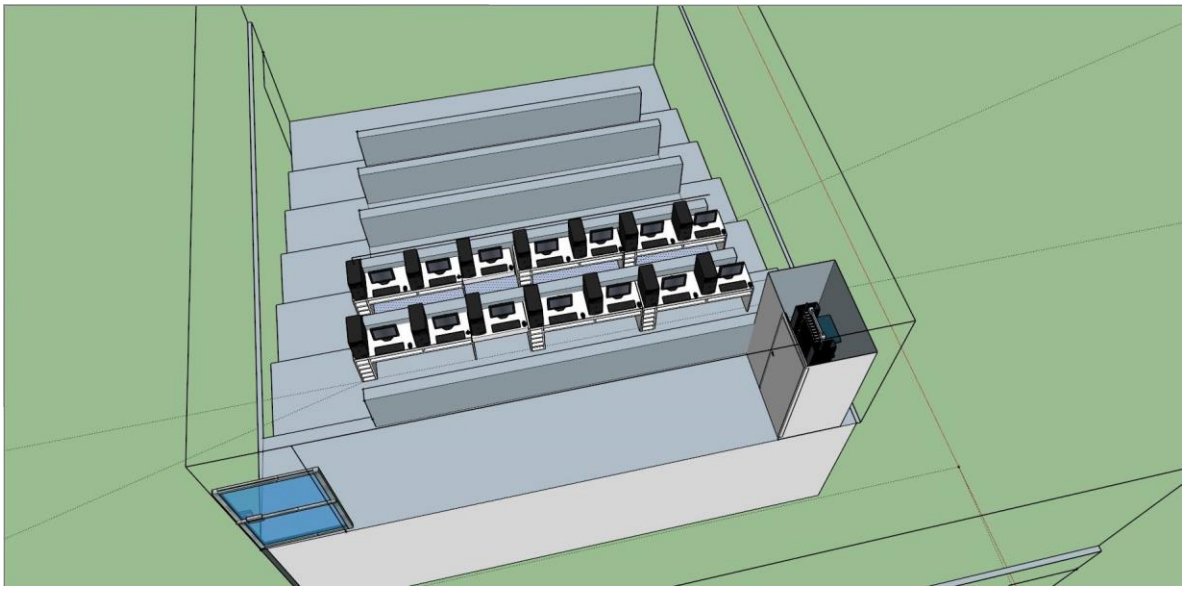
Aquí se muestran los cuatro módulos

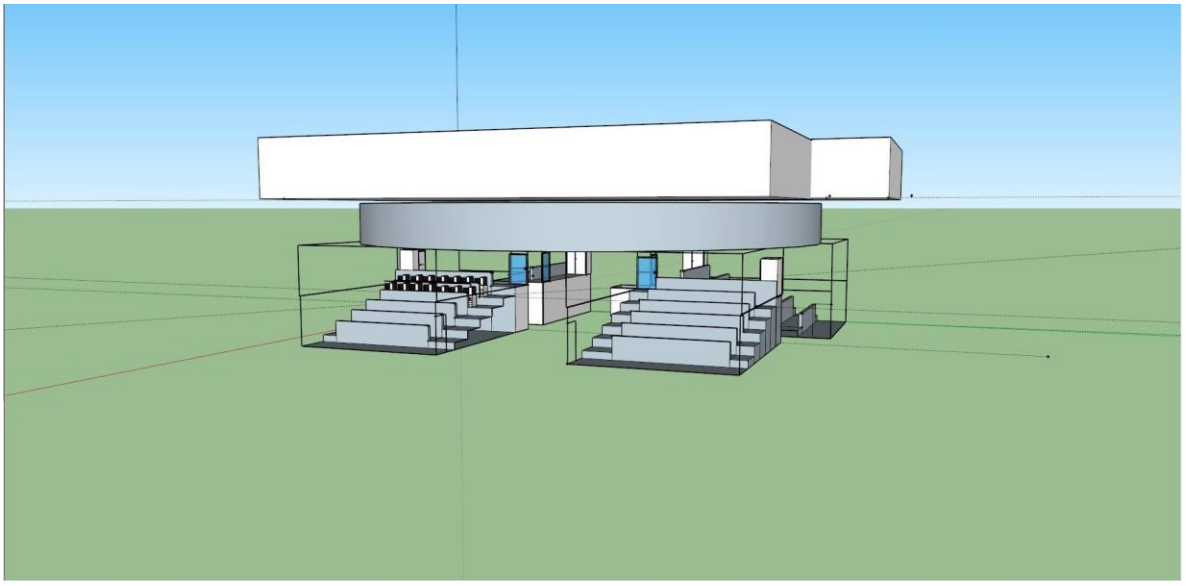


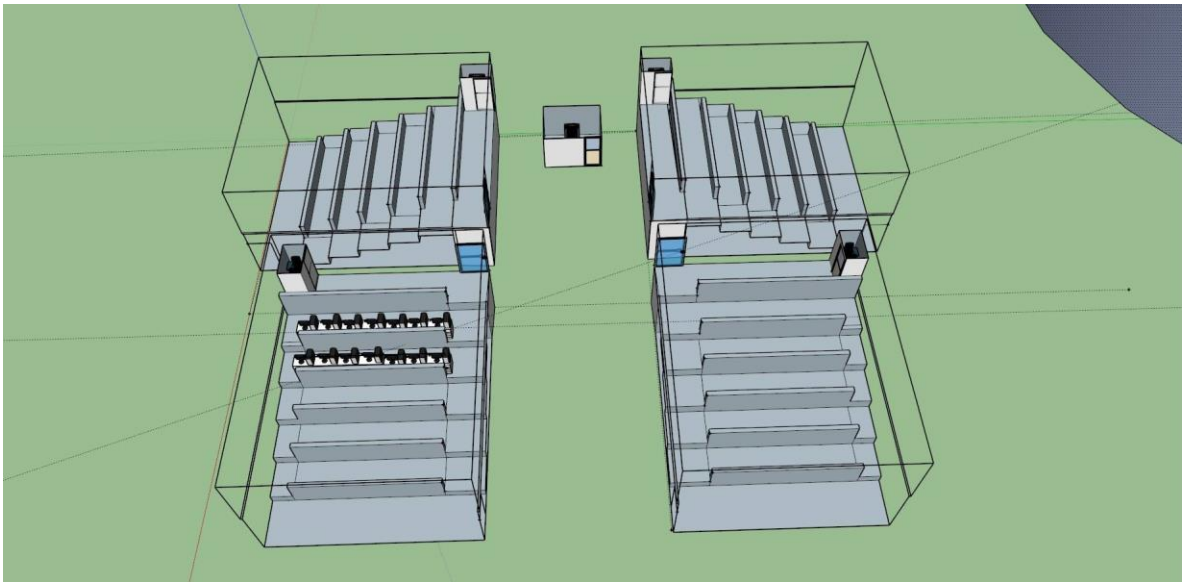
En esta imagen podemos observar el lugar donde se harán las tomas para cada una de las maquinas, tomando en cuenta que hay lugar para laptop, y que además también hay conexión inalámbrica. Son 35 máquinas y se tomaran en cuenta 50 tomas.

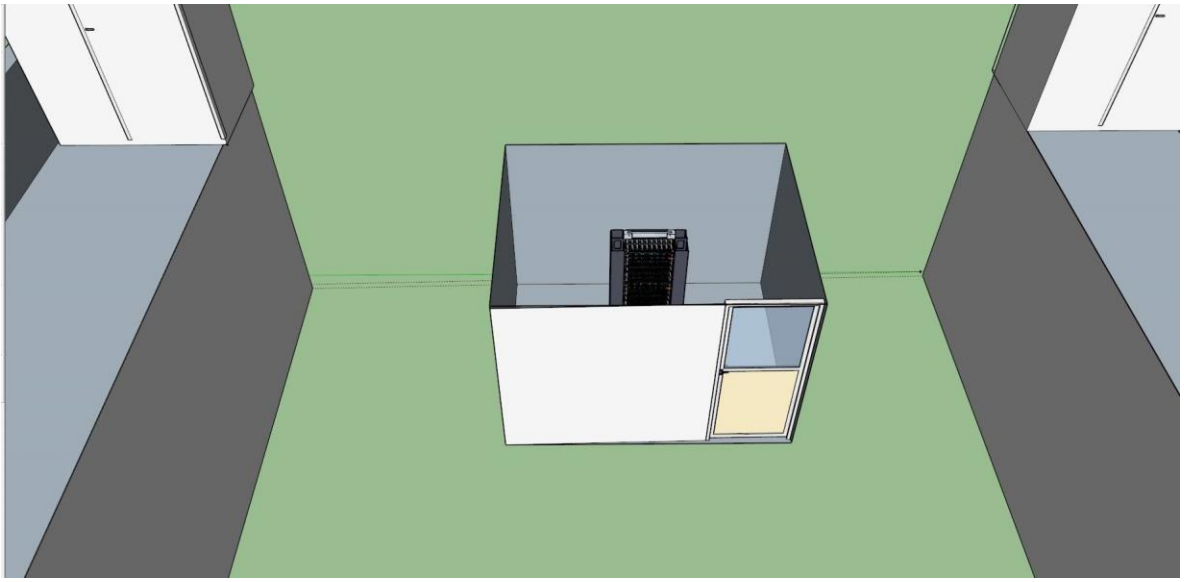


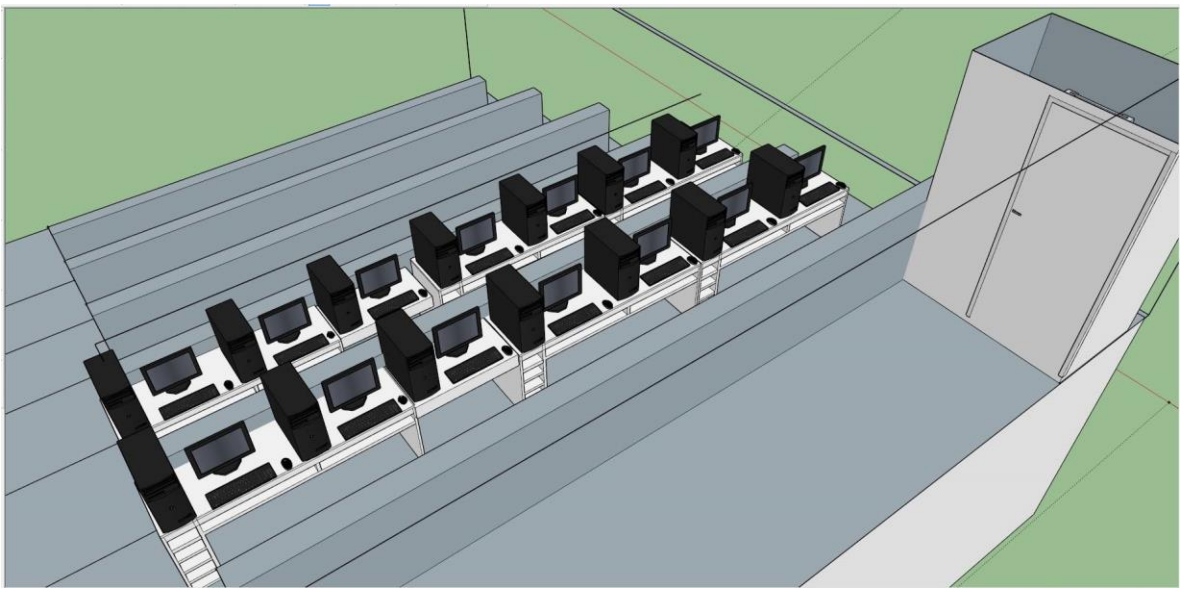
Este es el edificio 104D











Tomando en cuenta todo el material que se va a utilizar nuestra propuesta tendría un costo de $4634623000 pesos

Direcciones de Red

Red clase B 132.22.0.0/16 Para el 104A y 104C

Redes:

- Red 1: 95 hosts.
- Red 2: 95 hosts.
- Red 3: 95 hosts.
- Red 4: 95 hosts.
- Red 5: 63 hosts.
- Red 6: 2 hosts.
- Red 7: 2 hosts.
- Red 8: 2 hosts.
- Red 9: 2 hosts.
- Red 10: 2 hosts.

Red 1

2^7-2 = 126.

132.22.0.00000000 -> Red 132.22.0.0/25 Máscara de subred 255.255.255.128
132.22.0.00000001 -> Primera dirección válida 132.22.0.1
132.22.0.01111111 -> Broadcast 132.22.0.127
132.22.0.01111110 -> Gateway 132.22.0.126

Red 2

2^7-2 = 126.

Red: 132.22.0.128/25
Broadcast: 132.22.0.255
Gateway: 132.22.0.254
Primera dirección válida: 132.22.0.129
Máscara de subred: 255.255.255.128

Red 3

2^7-2 = 126.

Red: 132.22.1.0/25
Broadcast: 132.22.1.127
Gateway: 132.22.1.126
Primera dirección válida: 132.22.1.1
Máscara de subred: 255.255.255.128

Red 4

2^7-2 = 126.

Red: 132.22.1.128/25
Broadcast: 132.22.1.255
Gateway: 132.22.1.254

Primera dirección válida: 132.22.1.129
Máscara de subred: 255.255.255.128

Red 5

2^7-2 = 126.

Red: 132.22.2.0/25
Broadcast: 132.22.2.127
Gateway: 132.22.2.126
Primera dirección válida: 132.22.2.1
Máscara de subred: 255.255.255.128

Red 6

2^2-2 = 2.

Red: 132.22.2.128/30
Broadcast: 132.22.2.131
Primera dirección válida: 132.22.2.129
Máscara de subred: 255.255.255.252

Red 7

2^2-2 = 2.

Red: 132.22.2.132/30
Broadcast: 132.22.2.135
Primera dirección válida: 132.22.2.133
Máscara de subred: 255.255.255.252

Red 8

2^2-2 = 2.

Red: 132.22.2.136/30
Broadcast: 132.22.2.139
Primera dirección válida: 132.22.2.137
Máscara de subred: 255.255.255.252

Red 9
2^2-2 = 2.

Red: 132.22.2.140/30
Broadcast: 132.22.2.143
Primera dirección válida: 132.22.2.141
Máscara de subred: 255.255.255.252

Red 10

2^2-2 = 2.

Red: 132.22.2.144/30
Broadcast: 132.22.2.147
Primera dirección válida: 132.22.2.145
Máscara de subred: 255.255.255.252

Red 11*
2^2-2 = 2.

Red: 132.22.2.148/30
Broadcast: 132.22.2.151
Primera dirección válida: 132.22.2.149
Máscara de subred: 255.255.255.252

Red clase B 132.23.0.0/16 Para el 104D

Redes:

- Red 1: 184 hosts.
- Red 2: 184 hosts.
- Red 3: 184 hosts.
- Red 4: 2 hosts.
- Red 5: 2 hosts.
- Red 6: 2 hosts.
- Red 7: 2 hosts.

Red 1

2^8-2 = 254.

Red: 132.23.0.0/24
Broadcast: 132.23.0.255
Gateway: 132.23.0.254
Primera dirección válida: 132.23.0.1
Máscara de subred: 255.255.255.0

Red 2

2^8-2 = 254.

Red: 132.23.1.0/24
Broadcast: 132.23.1.255
Gateway: 132.23.1.254
Primera dirección válida: 132.23.1.1

Máscara de subred: 255.255.255.0

Red 3

2^8-2 = 254.

Red: 132.23.2.0/24
Broadcast: 132.23.2.255
Gateway: 132.23.2.254
Primera dirección válida: 132.23.2.1
Máscara de subred: 255.255.255.0

Red 4

2^2-2 = 2.

Red: 132.23.3.0/30
Broadcast: 132.23.3.3
Primera dirección válida: 132.23.3.1
Máscara de subred: 255.255.255.252

Red 5

2^2-2 = 2.

Red: 132.23.3.4/30
Broadcast: 132.23.3.7
Primera dirección válida: 132.23.3.5
Máscara de subred: 255.255.255.252

Red 6

2^2-2 = 2.

Red: 132.23.3.8/30
Broadcast: 132.23.3.11
Primera dirección válida: 132.23.3.9
Máscara de subred: 255.255.255.252

Red 7

2^2-2 = 2.

Red: 132.23.3.12/30
Broadcast: 132.23.3.15
Primera dirección válida: 132.23.3.13
Máscara de subred: 255.255.255.252

Resultados

Como se aprecia en las diferentes imágenes, el cableado queda bien escondido sin riesgos de daño por parte de personas e interferencias.

Conclusiones

Para realizar un buen cableado estructurado es necesario saber bien qué se necesita y de qué manera se quiere realizar el cableado. Se tiene que conocer bien la estructura de los edificios para tomar la mejor decisión en cuanto a los lugares por donde pasarán los cables. Se debe tener cuidado también con la distancia exacta de cada cable y en utilizar los mejores dispositivos.

Bibliografía

-A.S. Tanenbaum, "*Redes de Computadoras*", 4° Edición, Pearson Education, México, 2003.

-W. Stallings, "*Comunicaciones y Redes de Computadoras*", 7° Edición, Pearson Education, Madrid, 2004.

BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA

Message Digest (MD5)

Materia: **Modelo de Redes**

Profesor: Iván Olmos Pineda

Integrantes:

Juan Carlos Gómez Morales

Miguel Ángel Ibarra Viveros

Joselyne Meléndez Vizuet

Eduardo Francisco Pérez Rendón

Rodrigo Roaro Lack

Fecha: 27 de Noviembre del 2014

ÍNDICE

INTRODUCCIÓN

El proyecto consiste en implementar la técnica Message Digest (MD5) en un modelo Cliente-Servidor. El MD5 tiene varias aplicaciones entre ellas se encuentra el de comprobar la integridad de un cliente por medio de detección de huella, este proceso consiste en comprobar que los datos enviados no sufran modificaciones o alteraciones. Sin embargo se encuentra un defecto con el diseño de MD5, los criptógrafos comenzaron a recomendar el uso de otros algoritmos, tales como SHA-1 que desde entonces ha sido encontrado para ser vulnerables también. Los servidores de archivos proporcionan a menudo una suma de comprobación MD5 de pre-calculado para los archivos, de modo que un usuario puede comparar la suma de comprobación del archivo .La mayoría de los sistemas operativos basados en Unix incluyen utilidades suma MD5 de los paquetes de distribución,esta técnica sirve mucho para seguridad de datos.

DESARROLLO

Explicación Algoritmo

MD5 procesa un mensaje de longitud variable en una salida de longitud fija de 128 bits. El mensaje de entrada se divide en trozos de bloques de 512 bits, el mensaje se rellena de modo que su longitud es divisible por 512 - El relleno funciona de la siguiente manera: primero un solo bit, 1, se añade al final del mensaje. Esto es seguido de tantos ceros como están obligados a llevar la longitud del mensaje de hasta 64 bits menos que un múltiplo de 512 - Los bits restantes se rellenan con 64 bits que representan la longitud del mensaje original, en módulo 264.

El principal algoritmo MD5 opera en un estado de 128 bits, dividido en cuatro palabras de 32 bits, denotados A, B, C y D. Estos se inicializan a ciertas constantes fijas. El algoritmo principal, entonces funciona en cada bloque de mensaje de 512 bits a su vez, cada bloque de modificar el estado. El procesamiento de un bloque de mensaje se compone de cuatro etapas similares, rondas denominados; cada ronda se compone de 16 operaciones similares sobre la base de una función F no lineal, además de modular, y la rotación izquierda. La Figura 1 ilustra una operación dentro de una ronda. Hay cuatro posibles funciones F, una se utiliza uno diferente en cada ronda:

denotar la **XOR, AND, OR y NOT**

Pasos del algoritmo

1.-*Añadiendo Bits*

- El mensaje será extendido hasta que su longitud en bits sea congruente con 448, módulo 512. Esto es, si se le resta 448 a la longitud del mensaje tras este paso, se obtiene un múltiplo de 512.
  La extensión se realiza como sigue: un solo bit "1" se añade al mensaje, y después se añaden bits "0" hasta que la longitud en bits del mensaje extendido se haga congruente con 448, módulo 512. En todos los mensajes se añade al menos un bit y como máximo 512.

2.-Longitud de mensaje

- Un entero de 64 bits que representa la longitud 'b' del mensaje (longitud antes de añadir los bits) se concatena al resultado del paso anterior. En el supuesto no deseado de que 'b' sea mayor que 2^64, entonces sólo los 64 bits de menor peso de 'b' se usarán.
  En este punto el mensaje resultante (después de rellenar con los bits y con 'b') se tiene una longitud que es un múltiplo exacto de 512 bits. A su vez, la longitud del mensaje es

múltiplo de 16 palabras (32 bits por palabra). Con M[0 ... N-1] denotaremos las palabras del mensaje resultante, donde N es múltiplo de 16.

3.-Inicializar el buffer MD

- Un búfer de cuatro palabras (A, B, C, D) se usa para calcular el resumen del mensaje. Aquí cada una de las letras A, B, C, D representa un registro de 32 bits. Estos registros se inicializan con los siguientes valores hexadecimales:

  A: 01 23 45 67

  B: 89 ab cd ef

  C: fe dc ba 98

  D: 76 54 32 10

Paso 4. Procesado del mensaje en bloques de 16 palabras

- Primero definimos cuatro funciones auxiliares que toman como entrada tres palabras de 32 bits y su salida es una palabra de 32 bits.

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

Los operadores $\oplus, \wedge, \vee, \neg$ son las funciones XOR, AND, OR y NOT respectivamente.

En cada posición de cada bit F actúa como un condicional: si X, entonces Y sino Z. La función F podría haber sido definida usando + en lugar de v ya que XY y not(x) Z nunca tendrán unos ('1') en la misma posición de bit. Es interesante resaltar que si los bits de X, Y y Z son independientes y no sesgados, cada uno de los bits de F(X,Y,Z) será independiente y no sesgado.

Las funciones G, H e I son similares a la función F, ya que actúan "bit a bit en paralelo" para producir sus salidas de los bits de X, Y y Z, en la medida que si cada bit correspondiente de X, Y y Z son independientes y no sesgados, entonces cada bit de G(X,Y,Z), H(X,Y,Z) e I(X,Y,Z) serán independientes y no sesgados. Nótese que la función H es la comparación bit a bit

"xor" o función "paridad" de sus entradas.

Este paso usa una tabla de 64 elementos T[1 ... 64] construida con la función Seno. Denotaremos por T[i] el elemento i-ésimo de esta tabla, que será igual a la parte entera del valor absoluto del seno de 'i' 4294967296 veces, donde 'i' está en radianes.

Paso 5. Salida

- El resumen del mensaje es la salida producida por A, B, C y D. Esto es, se comienza el byte de menor peso de A y se acaba con el byte de mayor peso de D.

Las clases en las cuales consiste este proyecto son:

Cliente.class:

En esta clase se encuentra un método llamado generar mensaje el cual aplica la técnica md5,generando un mensaje de 32 caracteres máximo y lo codifica a bits,

una la contraseña codificada a bits al final a la secuencia de bits de mensaje, hace el relleno hasta llegar a 448 bits y en los 64 bits restantes coloca el tamaño del mensaje original en este clase se obtiene la comunicación con el servidor mediante sockets y se utilizan dos menús, uno que inicia sesion y registra usuarios usando la clase MD5 para generar la huell digital del “id” y “password” y así enviarla al servidor.

Flogicas. class

En esta clase se presentan las funciones lógicas a utilizar para la encriptación MD5

Load clients.class

esta clase utiliza lo que vimos en el primer parcial sobre el uso del stringtokenizer para obtener la información de los clientes por medio de un archivo txt.

MD5.class

en esta clase se aplica la encriptación MD5 la cual nos permite proteger el ID y el password del cliente al ser enviado al servidor.

Server.class

Esta clase verifica los datos que recibe del cliente y los compara con la base de datos de los clientes que se tienen almacenados en el archivo txt. Así al iniciar sesion el servidor busca que el ID que recibió del cliente se encuentra en el archivo, si lo encuentra genera una huella digital para el password y espera la huella digital del cliente para compararlas permitiendo así el inicio de sesion.

JUSTIFICACIÓN

Los resúmenes de MD5 se utiliza extensamente en el mundo del software para proporcionar la seguridad de que un archivo descargado de Internet no sea alterado. Comparando una suma de MD5 publicada con la suma de comprobación del archivo descargado, un usuario puede tener la confianza suficiente de que un archivo es igual que el publicado por los desarrolladores. Esto protege al usuario contra los 'Caballos de Troya' o 'Troyanos' y virus que algún otro usuario malicioso pudiera incluir en el software. La comprobación de un archivo descargado contra su suma MD5 no detecta solamente los archivos alterados de una manera maliciosa, también reconoce una descarga corrupta o incompleta.

Para comprobar la integridad de un archivo descargado de Internet se puede utilizar una herramienta MD5 para comparar la suma MD5 de dicho archivo con un archivo MD5SUM con el resumen MD5 del primer archivo. El MD5 también se puede usar para comprobar que los correos electrónicos no han sido alterados usando claves públicas y privadas.

Las ventajas de este tipo de algoritmos son la imposibilidad (computacional) de reconstruir la cadena original a partir del resultado, y también la imposibilidad de encontrar dos cadenas de texto que generen el mismo resultado.

Esto nos permite usar el algoritmo para transmitir contraseñas a través de un medio inseguro. simplemente se cifra la contraseña,y se envía de forma cifrada. En el punto de destino, para comprobar si el password es correcto,se cifra de la misma manera y se comparan las formas cifradas.

IMPLEMENTACIÓN

Manual de usuario:

a)Conexión con el Servidor:

b)Conexión cliente

c)Conexión segundo cliente

d)menú principal

e)menu secundario

f)Codificación Md5

g)Chat

CONCLUSIÓN

Gracias a esta implementación, pudimos constatar algunas dudas sobre la técnica MD5, además de que aprendimos la manera en que se constituye una técnica de encristalaron avanzada y la forma en que trabaja para generar la huella digital.

Ell algoritmo MD5 tiene otros usos también muy interesantes. El primero de ellos, es que, mediante un programa también podemos crear el código MD5 de un archivo propio, para que quien haga uso de él pueda comprobar su integridad.

Otra aplicación verdaderamente interesante está en las instalaciones de firmware, en las que además de proporcionarnos la información referente a la seguridad del archivo, puede servir también para comprobar que la descarga de éste se ha realizado correctamente, y dispongamos del archivo

completo y correcto. Esto, como decimos, es de gran utilidad a la hora de instalar un nuevo *firmware* o sistema operativo en nuestros dispositivos, como puede ser un router, o en el momento de flashear una ROM cocinada en un *smartphone* Android, ya que realizar una instalación de estas características con un archivo dañado o incompleto, puede dejarnos en ocasiones con un dispositivo inutilizable, o hacernos perder una buena parte de nuestro tiempo.

Por último, pero no menos interesante, otra utilidad que podemos darle al algoritmo MD5 es la de poder comprobar que un texto no haya sido modificado, y haya podido llegar de forma distinta a como era de forma original. Existe software, e incluso páginas web, en las que podemos escribir un texto y que éstas nos devuelven su *hash*; así, ofreciéndole este dato a nuestro destinatario, éste podrá comprobar si el texto que le hemos enviado no ha sido alterado antes de llegar hasta él.

BIBLIOGRAFÍA

Criptografía

http://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html#AppD

MD5

http://es.wikipedia.org/wiki/MD5

Documental sobre el algoritmo MD5

http://gabriel-sanmart.blogspot.mx/

Java Cryptography

http://books.google.com.mx/books?id=Jx-cAgAAQBAJ&pg=PA139&dq=MD5&hl=es-419&sa=X&ei=iap2VNToE4SmNu2igrgK&ved=0CE4Q6AEwBg#v=onepage&q=MD5&f=false

NO EXPLICAN CLARAMENTE:

EL PROCESO DE CONEXIÓN ENTRE CLIENTE Y SERVIDOR

QUE ALGORITMO DE ENCRIPTACION USAN

MANUAL DE USO DEL SISTEMA

7.

FACULTAD ,t~CIENCIAS
,t~1" COMPUTACIÓN

Examen 1

elo de Redes

Otoño 2014

d

Profesor:

n Olmos Pineda

'á

Alumno:

eth Salas Gómez

201121267

Marco teórico

En redes informáticas de datos se denomina latencia a la suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red.

Otros factores que influyen en la latencia de una red son:
El tamaño de los paquetes transmitidos.

El tamaño de los buffers dentro de los equipos de conectividad. Ellos pueden producir un Retardo Medio de Encolado.

La velocidad de transmisión de datos mide el tiempo que tarda un host o un servidor en poner en la línea de transmisión el paquete de datos a enviar, latencia. El tiempo de transmisión se mide desde el instante en que se pone el primer bit en la línea hasta el último bit del paquete a transmitir. La unidad de medida en el Sistema Internacional (de estar contemplado en el mismo) sería en bits/segundo (b/s o también bps), o expresado en octetos o bytes (B/s)n ya que así puede hacer la transmisión de dato.

Introducción

En el vacío, las señales entre las computadoras viajan a la velocidad de la luz, 186.000 millas (297.600 km) por segundo. En un cable de fibra óptica, se desaceleran cerca de 122.000 millas (195.200 km) por segundo. La pérdida de velocidad es de aproximadamente 8,2 microsegundos por milla, o 0,82 milisegundos por cada 100 millas (160 km). Si el paquete de datos tiene que pasar a través de un router o un conmutador, o la red utiliza NAT (NAT, del inglés Network Address Translation), o trasmisión de dirección de red, un sistema para enviar paquetes desde la red a la dirección de tu router, aumenta la latencia.

La latencia de red es más importante con paquetes pequeños que grandes trozos de datos. En paquetes de datos grandes y de lento movimiento, el arrastre adicional de un milisegundo o dos apenas se nota. Con pequeños paquetes de datos que deberían moverse rápidamente, el tiempo añadido puede ser significativo. La latencia alta es particularmente notable en comunicaciones en tiempo real de voz o de video, la pregunta no puede ser seguida de una respuesta inmediata.

Dada la importancia que tiene Internet, algunos países han definido sus propios anchos de banda en sus redes, adquiriendo sus propias fibras ópticas (*dark fibers*), lo cual les permite crear redes de alta velocidad que van desde uno a 10 gigabits o incluso superiores, usando técnicas como DWDM (*Dense Wavelength Division Multiplexing*), que permite aumentar la capacidad de transporte de las redes existentes mediante multiplexores. DWDM combina multitud de canales ópticos sobre una misma fibra, de tal modo que pueden ser amplificados y transmitidos de forma simultánea; y cada uno de estos canales, a distinta longitud de onda, puede transmitir señales de diferentes velocidades y formatos. Gracias a la existencia de estas redes de alta velocidad, es posible desarrollar proyectos en diferentes áreas científicas que involucren el procesamiento, almacenamiento, transmisión, manipulación de datos o desarrollo de aplicaciones en tiempo real.

Desarrollo

Como parte de nuestro primer examen parcial de Modelo de Redes, tenemos como fin, crear un programa que nos facilite encontrar las distintas latencias que se pueden generar a través de los diferentes caminos de una topología de red. Para obtener los resultados esperados, necesitamos saber:

- El tamaño del paquete que deseamos mandar
- Atributos entre un router y otro, que nos permitirán saber la latencia de tal paso, como:
    - Distancia (m)
    - Velocidad de Enlace (Mbps)
    - Tamaño del paquete (bits)

- Datos de Control (bits)

-Para dicho recorrido, como lo hemos visto en nuestros ejercicios de clase pasados, para llegar a dicha latencia final, debemos:

- Encontrar los todos los paths disponibles de nuetro router inicial, a nuestro router final

- Elegir un path para poder recorrer arista por arista, iremos dividiendo nuetros paquetes y así, realizar las operaciones necesarias para encontrar las latencias específicas.

$$\text{[illegible]} = \frac{\text{[illegible]}}{\text{[illegible]}}$$

$$\text{[illegible]} = \frac{\text{[illegible]}}{\text{[illegible]}}$$

$$\text{[illegible]} = T\text{[illegible]}$$

//Depende del router que estemos ocupando, será igual a 0 si es el router inicial o final

- Hacer la suma de las diferentes latencias

- Comparar las latencias de cada uno de los paths y así, elegir cuál es la latencia más pequeña.

- Encontrar el número de paquetes:

$$\#\ \text{[illegible]} = \frac{\text{[illegible]}}{\text{[illegible]}}$$

- Finalmente, encontrar el tiempo de transmisión de un paquete:

$$\text{[illegible]} = (\#\ \text{[illegible]}$$

Conclusiones

Con la ayuda de éste programa, podemos ser más exactos y precisos en las operaciones para obtener el

tiempo de transmisión de un paquete, ya que nos ayuda a evitar errores decimales, puesto que la mayoría de nuestras operaciones son de éste tipo.

Encontrar los caminos de la topología de nuestra red, es fundamental para encontrar las latencias finales, y es que la mejor latencia no se encuentra a razón del camino más corto, sino, a razón de la transmisión de paquetes, ya que ésta irá variando conforme los atributos de cada camino (arista por arista).

Referencias

[1] Wikipedia, «Latencia,» 1 Agosto 2014. [En línea]. Available: http://es.wikipedia.org/wiki/Latencia. [Último acceso: Septiembre 2014].

[2] F. Sherman. [En línea]. Available: http://www.ehowenespanol.com/latencia-red-milisegundos-

milla-info_203714/. [Último acceso: Septiembre 2014].

[3] UNAM, Dirección General de Servicios de Cómputo Académico, «Redes de alta velocidad: en el desarrollo científico y tecnológico,» 27 Noviembre 2008. [En línea]. Available: http://www.enterate.unam.mx/artic/2008/mayo/art1.html. [Último acceso: Septiembre 2014].

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Ingeniería en Ciencias de la Computación

Modelos de Redes

Trabajo: Programa en Java que calcula el tiempo de transferencia de un archivo en una red.

Autor: Iván Morelos Dorantes

26/09/2014

Índice

Introducción

Conocer el rendimiento de una red permite saber la velocidad máxima de transferencia de datos y de este modo saber cuándo hay problemas que deban repararse. Generalmente, el rendimiento de una red de computadoras es medido o cuantificado usando la velocidad de trasmisión de datos. Es una medida concreta y de fácil cálculo, que permite saber si una red está funcionando en forma óptima. Estos cálculos se hacen conociendo la latencia y la tasa de transferencia.

El objetivo de este trabajo es simular, con un programa hecho en Java, la transferencia de un archivo a través de una red virtual.

Antecedentes

Tasa de transferencia

La tasa de transferencia se refiere a la velocidad con la cual podemos transferir información y se mide en bits / seg.

- Tasa de transferencia teórica: máxima velocidad que se puede alcanzar considerando el ancho de banda del medio utilizado
- Tasa de transferencia real: velocidad de transferencia considerando las limitantes del medio empleado

Latencia

Es el tiempo que toma un bit en viajar de un extremo de un medio al otro. Depende de tres factores:

- Tiempo de propagación del bit por el medio, que depende del tiempo de propagación de la corriente o luz por el medio, además de la distancia recorrida.

- Máxima cantidad de datos que pueden ser transmitidos por la red sin segmentarse. Al tiempo de propagación del bit en un paquete se le llama tiempo de transmisión.

- Tiempos de espera para difundirse un paquete a través de un conmutador, además del tráfico de la red. A este tiempo se le denomina tiempo de cola.

Desarrollo

El programa cuenta con las siguientes clases:

- Busqueda.java
- Grafo.java
- PrimerParcial.java
- Latencias.java

Y dos archivos de entrada para escoger:

- datos.txt
- datos2.txt

datos.txt

Es el primer archivo de entrada del programa. Contiene los siguientes datos:

6,6	Número de vértices, número de enlaces
1,2,98,1000,1500,200 2,3,87,1000,1900,400 2,4,105,500,1500,300 3,5,130,250,1300,500 4,6,90,700,1100,400	Los 6 enlaces que contienen en el mismo orden: • Vértice inicial. • Vértice final. • Distancia en metros.
0.00025,0.001,0.00070,0.0093	Tiempos de cola
1,6	Vértice origen, vértice destino.
4.8	Tamaño del archivo en GB.

datos2.txt

El segundo archivo de entrada. Contiene los siguientes elementos.

9,16	Número de vértices, número de enlaces

1,2,98,1022,1500,200 2,5,87,1004,1900,400 1,5,56,1120,1400,200 2,9,105,700,1700,300 3,1,130,250,1300,500 6,2,160,250,1600,500 2,7,200,250,1300,500 3,4,90,700,1100,400 3,6,190,900,900,500 4,2,145,1100,1500,200	Los 6 enlaces que contienen en el mismo orden: • Vértice inicial. • Vértice final. • Distancia en metros. • Velocidad en Mbps. • Tamaño del paquete en bytes. • Datos de control en bytes.
0.056,0.00025,0.0024,0.001,0.00070,0.0093,0.001	Tiempos de cola
3,7	Vértice origen, vértice destino.
4.8	Tamaño del archivo en GB.

PrimerParcial.java

Es la clase principal del programa. Ya que son varios datos los que se reciben, creamos las variables que los almacenarán.

```
int[]verticesEnlacesTotales =  new int[2];
```

La variable verticesEnlacesTotales es un vector de enteros donde guardaremos la primera línea del texto. Es un vector de tamaño dos; en la primera posición guarda el número de vértices del grafo y en la segunda posición guarda el número de enlaces del grafo.

```
int[][] enlaces = null;
```

La variable enlaces es una matriz de enteros de n*7, n está dado por la segunda posición del primer vector llamado verticesEnlacesTotales.

- enlaces[n][0] – Contiene el vértice inicial del enlace.
- enlaces[n][1] – Contiene el vértice final del enlace.
- enlaces[n][2] – Contiene la distancia del enlace.
- enlaces[n][3] – Contiene los Mpbs del enlace.
- enlaces[n][4] – Contiene el tamaño del paquete que puede transportar el enlace.
- enlaces[n][5] – Contiene el tamaño de los datos de control (DC).
- enlaces[n][6] – Contiene el tamaño de los datos de usuario (DU).

```
double[] colas = null;
```

La variable colas es un vector de tipo double que almacena los tiempo de colas de los vértices.

```
int[] verticeInicialFinal = null;
```

La variable verticeInicialFinal es un vector de enteros de tamaño dos. La primera posición guarda el vertice origen y la segunda posición almacena el vértice destino.

```
double[][] calculoTpTt;
```

La matriz calculoTpTt es de tamaño n*2 y almacena los tiempo de propagación y transmisión de los enlaces.

```
double tamArchivo = 0;
```

Esta variable tamArchivo almacena el tamaño del archivo.

String[] cadena2;

La variable cadena2 almacena todas las palabras separadas por comas que contiene cadena.

String cadena;

La variable cadena se utiliza para leer el documento línea a línea.

final int velocidadLuz = 300000000;

En velocidadLuz se almacena la velocidad de la luz en m/s.

int contador = 0;

La variable contador se utiliza para leer las n líneas de los enlaces.

```
FileReader fr = new FileReader("datos.txt");
BufferedReader bf = new BufferedReader(fr);
cadena = bf.readLine();
```

Las variables fr, bf y cadena se utilizan conjuntamente para leer el archivo.

- Con fr se abre el archivo.
- Con bf se lee la línea del archivo.
- En cadena se almacena la línea leída.

La lectura y almacenamiento de la primera línea se hace con el siguiente código:

```
cadena2 = cadena.split(",");
verticesEnlacesTotales[0] = Integer.parseInt(cadena2[0]);
```

```
verticesEnlacesTotales[1] = Integer.parseInt(cadena2[1]);
```

La lectura de los enlaces se hace de la siguiente manera:

```
while (cadena != null && contador<verticesEnlacesTotales[1]) {
    cadena2 = cadena.split(",");
    calculoTpTt[contador][0] = Double.parseDouble(cadena2[2])/velocidadLuz;
    calculoTpTt[contador][1] = (Double.parseDouble(cadena2[4])*8)/(Double.parseDouble(cadena2[3])*1000000);
    for(int i=0;i<7;i++){
      if(i==6){
        enlaces[contador][i]=(Integer.parseInt(cadena2[i-2])-Integer.parseInt(cadena2[i-1]));
      }
      if(i<6){
        enlaces[contador][i]=Integer.parseInt(cadena2[i]);
      }
    }
    cadena = bf.readLine();
    contador++;
```

```
}
```

Y el resto de las lecturas se hace de forma muy similar. Al final lo que resta por hacer es enviar a la clase **Busqueda()** los enlaces, el vertice inicial y el vértice final.

Código completo de PrimerParcial.java

```
public class PrimerParcial {

    public static void main(String[] args) {

        int[]verticesEnlacesTotales =  new int[2];

        int[][] enlaces = null;

        double[] colas = null;

        int[] verticeInicialFinal = null;

        double[][]  calculoTpTt;
        double tamArchivo = 0;
        String[] cadena2;

        String cadena;

        final int velocidadLuz = 300000000;

        int contador = 0;

        try {

            FileReader fr = new FileReader("datos.txt");
            BufferedReader bf = new BufferedReader(fr);

            cadena = bf.readLine();

            //Lee y almacena el número de vértices y enlaces
            cadena2 = cadena.split(",");

            verticesEnlacesTotales[0] = Integer.parseInt(cadena2[0]);

            verticesEnlacesTotales[1] = Integer.parseInt(cadena2[1]);

            //Inicializamos la matriz que almacenará los enlaces
            enlaces = new int[verticesEnlacesTotales[1]][7];

            calculoTpTt = new double[verticesEnlacesTotales[1]][2];

            //Inicializamos el vector que guarda las colas

            colas = new double[verticesEnlacesTotales[1]-2];
```

```
            verticeInicialFinal = new int[2];
            //Lee las n líneas del documento que contienen la información de los enlaces y calcula el DU
            cadena = bf.readLine();
            while (cadena != null && contador<verticesEnlacesTotales[1]) {
               cadena2 = cadena.split(",");
               calculoTpTt[contador][0] = Double.parseDouble(cadena2[2])/velocidadLuz;
               calculoTpTt[contador][1] = (Double.parseDouble(cadena2[4])*8)/(Double.parseDouble(cadena2[3])*1000000);
               for(int i=0;i<7;i++){
                  if(i==6){
                     enlaces[contador][i]=(Integer.parseInt(cadena2[i-2])-Integer.parseInt(cadena2[i-1]));
                  }
                  if(i<6){
                     enlaces[contador][i]=Integer.parseInt(cadena2[i]);
                  }
               }
               cadena = bf.readLine();
```

```
            contador++;
        }
        Latencias l = new Latencias(enlaces, calculoTpTt);
        Busqueda b = new Busqueda(l);
        //Lee y almacena la línea correspondiente a los valores de las colas
        if(cadena != null){
            cadena2 = cadena.split(",");
            for(int i=0;i< verticesEnlacesTotales[0]-2;i++){
                colas[i] = Double.parseDouble(cadena2[i]);
            }
        }
        //Lee y almacena el vertice inicial y final
        cadena = bf.readLine();
        if(cadena != null){
            cadena2 = cadena.split(",");
            for(int i=0;i<2;i++){
                verticeInicialFinal[i] = Integer.parseInt(cadena2[i]);
            }
        }
        //Lee y almacena el tamaño del archivo
        cadena = bf.readLine();
        if(cadena != null){
            tamArchivo = Double.parseDouble(cadena);
        }
        b.buscar(enlaces, verticeInicialFinal);
    } catch (FileNotFoundException fnfe){
        fnfe.printStackTrace();
    } catch (IOException ioe){
        ioe.printStackTrace();
    }
  }
}
```

Grafo.java

La clase Grafo.java contiene las colecciones de objetos que se utilizan para manejar el grafo. Los nodos se almacenan en 3 colecciones:

- LinkedList.
- LinkedHashSet.
- HashMap.

La clase se compone de dos métodos, **agregarEnlace()** y **nodosAdyacentes()**. En la función **agregarEnlace()** se reciben los vértices inicial y final de cada enlace. La función **nodosAdyacentes()** recibe el vértice al cuál se le van a buscar los nodos adyacentes y regresa una lista ligada que contiene los nodos adyacentes.

Código completo de Grafo.java

```
package primerparcial; import
java.util.HashMap; import
java.util.LinkedHashSet; import
java.util.LinkedList; import
java.util.Map;

import java.util.Set;


public class Grafo {

   private Map<String, LinkedHashSet<String>> mapa = new HashMap();


   public void agregarEnlace(String nodo1, String nodo2) {
      LinkedHashSet<String> adyacente = mapa.get(nodo1);
      if(adyacente==null) {

         adyacente = new LinkedHashSet();

         mapa.put(nodo1, adyacente);

      }

      adyacente.add(nodo2);

   }


   public LinkedList<String> nodosAdyacentes(String ultimo) {
      LinkedHashSet<String> adyacente = mapa.get(ultimo);
      if(adyacente==null) {

         return new LinkedList();

      }

      return new LinkedList<String>(adyacente);

   }

}
```

Busqueda.java

Es la clase donde se harán las búsquedas de todos los caminos posibles en el grafo.

Lo primero que tiene la clase son 3 variables globales:

```
Latencias lat;
String INICIO;
String FIN;
```

La variable lat es un objeto de la clase Latencias que se utiliza para tener acceso a los métodos de imprimir las matrices y los caminos que tiene la clase Latencias.

INICIO es la variable donde se almacena el vértice origen. En FIN se almacena el vértice destino.

El primer método de la clase es **buscar()**, que recibe una matriz que contiene los enlaces del grafo y un vector que contiene el vértice origen y el vértice destino.

La línea lat.imprimirMatrices(); puede ser comentada, es sólo para mostrar todos los datos almacenados.

Como se había establecido, la variable INICIO guarda el vértice origen del grafo que se encuentra en la primera posición del vector inicialFinal. FIN guarda el vértice destino del grafo que está almacenado en la segunda posición de inicialFinal.

En esta clase también se crea una lista ligada llamada visitado que es una colección con todos los nodos que ya han sido visitados en cada camino. Esto se hace para evitar los ciclos ya que si un nodo se encuentra en la lista, no se vuelve a agregar.

Por último, se hace uso del método **búsquedaAmplitud()** de la misma clase.

```
public void buscar(int[][] matriz, int[] inicialFinal) {
      Grafo grafo = new Grafo();
      lat.imprimirMatrices();

      INICIO = Integer.toString(inicialFinal[0]);
      FIN = Integer.toString(inicialFinal[1]);
      for(int i=0;i<matriz.length;i++){

         grafo.agregarEnlace(Integer.toString(matriz[i][0]),Integer.toString(matriz[i][1]));

      }

      LinkedList<String> visitado = new LinkedList();

      visitado.add(INICIO);

      new Busqueda(lat).busquedaAmplitud(grafo, visitado, INICIO, FIN);

   }
```

La segunda función que contiene la clase **Busqueda** es **busquedaAmplitud()**. Esta función recibe como parámetros una instancia de la clase Grafo, la lista visitado, INICIO y FIN.

Esta es la función que nos permite encontrar todos los caminos posibles en el grafo evitando los ciclos. Cada vez que se agrega un nodo a la lista, se verifica que no se haya ingresado antes. Si es un nodo que no ha sido visitado, se compara con el fin; si es diferente, se agrega a la lista de nodos visitados, si es igual, la búsqueda del camino termina y se envía toda la lista a la función caminos.

```
private void busquedaAmplitud(Grafo grafo, LinkedList<String> visitado, String INICIO, String
FIN) {
    LinkedList<String> nodos = grafo.nodosAdyacentes(visitado.getLast());
    int p=0;
    for (String nodo : nodos) {
      if (visitado.contains(nodo)) {
        continue;
      }
      if (nodo.equals(FIN)) {
        visitado.add(nodo);
        caminos(visitado);
        visitado.removeLast();
        break;
      }
    }
    for (String nodo : nodos) {
      if (visitado.contains(nodo) || nodo.equals(FIN)) {
        continue;
      }
      visitado.addLast(nodo);
      busquedaAmplitud(grafo, visitado, INICIO, FIN);
      visitado.removeLast();
    }
  }
```

La tercera y última función de la clase **Busqueda** es **caminos()**. Este método recibe el camino encontrado y lo guarda en un vector. Al mismo tiempo convierte las cadenas en enteros.

```
  private void caminos(LinkedList<String> visitado) {
    int[] matriz2 = new int[visitado.size()];
```

```
        int i=0;
        for (String nodo : visitado) {
            matriz2[i] = Integer.parseInt(nodo);
            i++;
        }
        lat.calcularLatencias(matriz2);
    }
```

Código completo de Busqueda.java

```
package primerparcial;
import java.util.LinkedList;

public class Busqueda {
    Latencias lat;
    String INICIO;
```

```
    String FIN;


    public Busqueda(Latencias l){

        this.lat = l;

    }
public void buscar(int[][] matriz, int[] inicialFinal) {
        Grafo grafo = new Grafo();

        lat.imprimirMatrices();

        INICIO = Integer.toString(inicialFinal[0]);
        FIN = Integer.toString(inicialFinal[1]);
        for(int i=0;i<matriz.length;i++){

            grafo.agregarEnlace(Integer.toString(matriz[i][0]),Integer.toString(matriz[i][1]));

        }

        LinkedList<String> visitado = new LinkedList();

        visitado.add(INICIO);

        new Busqueda(lat).busquedaAmplitud(grafo, visitado, INICIO, FIN);

    }


    private void busquedaAmplitud(Grafo grafo, LinkedList<String> visitado, String INICIO,
String FIN) {

        LinkedList<String> nodos = grafo.nodosAdyacentes(visitado.getLast());

        int p=0;

        for (String nodo : nodos) {

            if (visitado.contains(nodo)) {

                continue;

            }

            if (nodo.equals(FIN)) {

                visitado.add(nodo);

                caminos(visitado);
                visitado.removeLast();
                break;

            }

        }

        for (String nodo : nodos) {
```

```
            if (visitado.contains(nodo) || nodo.equals(FIN)) {
                continue;
            }
            visitado.addLast(nodo);
            busquedaAmplitud(grafo, visitado, INICIO, FIN);
            visitado.removeLast();
        }
    }

    private void caminos(LinkedList<String> visitado) {
        int[] matriz2 = new int[visitado.size()];
        int i=0;
        for (String nodo : visitado) {
            matriz2[i] = Integer.parseInt(nodo);
            i++;
        }
        lat.calcularLatencias(matriz2);
```

```
    }
}
```

Latencias.java

Latencias es la última clase del programa. Aquí se calculan las latencias de cada camino y se encuentra el camino más corto.

Los métodos más importantes de la clase **Latencias()** son **calcularLatencias()** y **caminosFinales().**

Calcular Latencias

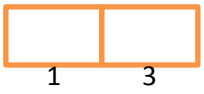
Esta función recibe el camino encontrado en forma de vector. Lo que hace primero es verificar el número de paquetes que se van a enviar en cada enlace del grafo. Por ejemplo, supongamos que el camino recibido es:

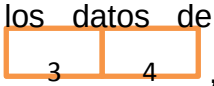
Entonces el programa toma dos pares de números que forman dos enlaces contiguos. En el camino de ejemplos estos dos pares serían 

y 

.

Una vez que los encuentra, revisa en la matriz de enlaces que efectivamente existan esos enlaces. Los únicos datos que utiliza el programa son la primera y segunda columna de cada fila, ya que esos datos son el vértice inicial y el vértice final, respectivamente. Una vez que se verifica la existencia de los enlaces, el programa toma el valor de la columna 6 de cada fila de la matriz de enlaces, es decir, la tamaño máximo de datos de usuario que puede pasar por cada enlace.

Volviendo al ejemplo, si el tamaño de los datos de usuario del enlace 

es mayor que el del enlace 

, supongamos 1000 y 300 respectivamente, se guarda el valor de 1000 en una variable, y a esa variable se le restan los 300. Ya se dividió una vez, así que el contador de paquetes se aumenta en uno. El proceso se repite hasta que el tamaño inicialmente mayor sea igual o menor al tamaño del segundo enlace.

El ejemplo se ilustra en el siguiente pseudocódigo:

```
contadorPaquetes=0;

paquete= paqueteMayor;

Mientras que(paquete mayor a paqueteMenor){
        vectorQueGuardaLosTamaños[algunaPosicion] = paqueteMenor;
        paquete = paquete – paqueteMenor;
        contadorPaquetes = contadorPaquetes+1;

        si(paquete es igual a paqueteMenor ó paquete es menor a paqueteMenor){
                vectorQueGuardaLosTamaños[algunaPosicion] = paqueteMenor;
                contadorPaquetes = contadorPaquetes+1;
        }
}
```

Después de que ya se han almacenado los paquetes para cada enlace, resta hacer las multiplicaciones del número de paquetes por el tiempo de propagación (TP) y tiempo de transmisión (TT). Para hacer esto se buscan los pares adyacentes de vértices del vector que tiene los caminos en la matriz que contiene los enlaces. Si coinciden, se multiplican los números de paquetes por los tiempos de propagación y transmisión correspondientes. El resultado se guarda en una matriz de 1000x1000 justo delante del último vértice del camino. Como esta matriz guarda todos los caminos que se generan, un ejemplo de una fila es el siguiente:

1	3	4	7	8	9	0.0038454

Donde se aprecia el camino guardado y enseguida el tiempo de latencia del camino.

Caminos Finales

La segunda función importante de esta clase es **caminosFinales()**. Lo que hace es imprimir todos los caminos y verificar cuál es la menor latencia. Por último muestra el número de paquetes que se enviarán dependiendo del tamaño del

archivo y del tamaño de datos de usuario del primer enlace. También calcula el tiempo total de transferencia en segundos.

La forma de verificar la menor latencia es la siguiente:

Al mostrar en pantalla todos los caminos, al mismo tiempo se compara una variable que contiene inicialmente un número muy grande. Si el tiempo de latencia del camino que se está imprimiendo es menor a la variable, se guarda el camino y la variable toma el valor de la latencia. Este proceso se sigue hasta terminar todos los caminos. Como resultado tendremos la menor latencia y el camino que la genera.

Ejemplo en pseudocódigo:

Variable = Infinito;

1	3	4	7	8	9	0.0038454

Ciclo(recorrer todo la fila del camino) {

    Si(tiempo de latencia del camino es menor que Variable){

        Guardar camino;

        Variable = tiempo de latencia del camino;

    }

}

Lo último que hace la función es mostrar los resultados.

Primero muestra los caminos seguidos del tiempo de latencia de cada uno de ellos. Después muestra el camino con la latencia menor, el número de paquetes que se tienen que crear para enviar todo el archivo y el tiempo total de transferencia en segundos.

Código completo de Latencias.java

```
package primerparcial;


public class Latencias {
    int[][] matriz;
    double[][] colas;
    double[][] valores;

    double[][] todosCaminos;

    int[] paquetes;
```

```
int nCamino;
double tam;
int[] temp2;
int[] temp3;


public Latencias(int[][] matriz, double[][] valores, double[][] colas, double tam){
    this.matriz = matriz;
    this.valores = valores;
    this.todosCaminos = new double[1000][1000];
    this.nCamino = 0;
    this.temp2 = new int[1000];
    this.temp3 = new int[1000];
    this.colas = colas;
    this.tam = tam;
    this.paquetes = new int[1000];
    for(int i=0;i<todosCaminos.length;i++){
        for(int j=0;j<todosCaminos.length;j++){
            todosCaminos[i][j] = -1;
            paquetes[i] = -1;
        }
    }
}

void reset(int[] mat){
    for(int i=0;i<1000;i++){
        mat[i]=-1;
    }
}

void imprimirMatrices(){
    for(int i=0;i<matriz.length;i++){
        for(int j=0;j<7;j++){
            System.out.print(matriz[i][j]+" ");
```

```
            }
            System.out.println("");

        }

        System.out.println("");
        System.out.println("");

        for(int i=0;i<valores.length;i++){

            for(int j=0;j<2;j++){
                System.out.printf("%.10f     ",valores[i][j]);

            }

            System.out.println("");

        }

        System.out.println("");
        System.out.println("");

        for(int i=0;i<colas.length;i++){

            System.out.printf("%d     +      %.10f\n",(int)colas[i][0],colas[i][1]);

        }
        System.out.println("\n\n\n\n\n\n");

    }
```

```
void calcularLatencias(int[] caminos){
   reset(temp2);
   reset(temp3);
   int con5=0;
   paquetes[0]=1;
   con5++;
   int pos=0;
   for(pos=0;pos<caminos.length;pos++){
      todosCaminos[nCamino][pos] = caminos[pos];
   }
   int cont4=0;
   for(int i=0;i<caminos.length-2;i++){
      for(int j=0;j<matriz.length;j++){
         if(caminos[i]==matriz[j][0] && caminos[i+1]==matriz[j][1]){
            for(int k=0;k<matriz.length;k++){
               if(caminos[i+1]==matriz[k][0] && caminos[i+2]==matriz[k][1]){
                  int contPaq=0;
                  int paq=0;
                  if(i==0){
                     paq = matriz[j][6];
                     if(paq<matriz[k][6] || paq==matriz[k][6]){
                        temp2[contPaq]=matriz[j][6];
                        contPaq++;
                     }
                     while(paq>matriz[k][6]){
                        paq=paq-matriz[k][6];
                        temp2[contPaq]=matriz[k][6];
                        contPaq++;
                        if(paq<matriz[k][6] || paq==matriz[k][6]){
                           temp2[contPaq]=paq;
                           contPaq++;
                        }
```

```
    }
}else{
    int cont3=0;
    while(temp2[cont3]!=-1){
        paq=temp2[cont3];
        if(paq<matriz[k][6] || paq==matriz[k][6]){
            temp3[cont4]=paq;
            contPaq++;
            cont4++;
        }
        while(paq>matriz[k][6]){
            paq=paq-matriz[k][6];
            temp3[cont4]=matriz[k][6];
            contPaq++;
            cont4++;
            if(paq<matriz[k][6] || paq==matriz[k][6]){
                temp3[cont4]=paq;
                contPaq++;
```

```
                    }
                }
                cont3++;
                cont4++;
            }
            reset(temp2); int p=0;
            while(temp3[p]!=-1){
                temp2[p]=temp3[p];
                p++;
            }
        }
        reset(temp3); cont4=0;
        paquetes[con5]=contPaq;
        con5++;
    }
  }
  break;
 }
}
}
for(int i=0;i<caminos.length-1;i++){
  for(int j=0;j<matriz.length;j++){
    if(caminos[i]==matriz[j][0] && caminos[i+1]==matriz[j][1]){
      if(i==0){
        todosCaminos[nCamino][pos] = paquetes[i]*valores[j][0] +
paquetes[i]*valores[j][1];
      }else{
        todosCaminos[nCamino][pos] += paquetes[i]*valores[j][0] +
paquetes[i]*valores[j][1];
      }
      for(int l=0;l<colas.length;l++){
        if(matriz[j][1]==colas[l][0]){
```

```
                    todosCaminos[nCamino][pos]+= colas[l][1];
                }
            }
        }
    }
    nCamino++;
    reset(paquetes);
}

void caminosFinales(){
    int h=0;
    double min=Double.POSITIVE_INFINITY;
    System.out.println("Caminos : Latencias\n");
    for(int i=0;i<todosCaminos.length;i++){
        for(int j=0;j<todosCaminos.length;j++){
            if(todosCaminos[i][j] == -1){
```

```
                break;
            }
            if (todosCaminos[i][j] - Math.floor(todosCaminos[i][j]) == 0) {
                if(j>0){
                    System.out.printf("- %d ",(int)todosCaminos[i][j]);
                }else{
                    System.out.printf("%d ",(int)todosCaminos[i][j]);
                }
            }else{
                if(todosCaminos[i][j]<min){
                    h=i;
                    min=todosCaminos[i][j];
                }
                System.out.printf(":%.10f",todosCaminos[i][j]);
            }
        }
        System.out.println("");
        if(todosCaminos[i][0] == -1){
            break;
        }
    }

    System.out.println("Camino con menor latencia : número de paquetes que deberán enviarse :
tiempo total de transferencia\n");
    for(int j=0;j<todosCaminos.length;j++){
        if(todosCaminos[h][j] == -1){
            break;
        }
        if (todosCaminos[h][j] - Math.floor(todosCaminos[h][j]) == 0) {
            if(j>0){
                System.out.printf("- %d ",(int)todosCaminos[h][j]);
            }else{
```

```
                System.out.printf("%d ",(int)todosCaminos[h][j]);
            }
        }else{
            for(int i=0;i<matriz.length;i++){
                if(todosCaminos[h][0]==matriz[i][0] && todosCaminos[h][1]==matriz[i][1]){
                    System.out.println(": "+(int)Math.ceil((tam*1024*1024*1024)/matriz[i][6])+":
"+(Math.ceil(tam*1024*1024*1024)/matriz[i][6])*min);
                }
            }
        }
    }
    System.out.println("");
  }
}
```

Resultados

Con el archivo datos.txt como entrada, se obtuvieron los siguientes resultados:

```
Caminos : Latencias

1 - 2 - 3 - 5 - 6 :0.0106948833
1 - 2 - 4 - 6 :0.0010496410

Camino con menor latencia : número de paquetes que deberán enviarse : tiempo total de transferencia

1 - 2 - 4 - 6 : 3964586: 4161.390981893765
```

Con el archivo datos2.txt como entrada los resultados fueron los siguientes:

```
Caminos : Latencias

3 - 1 - 2 - 7 :0.0563463683
3 - 1 - 2 - 5 - 7 :0.0573639278
3 - 1 - 2 - 9 - 7 :0.0573352769
3 - 1 - 5 - 7 :0.0570966167
3 - 4 - 2 - 7 :0.0027165305
3 - 4 - 2 - 5 - 7 :0.0037340900
3 - 4 - 2 - 9 - 7 :0.0037054391
3 - 4 - 8 - 5 - 7 :0.0127893014
3 - 4 - 8 - 9 - 7 :0.0127566914
3 - 6 - 2 - 7 :0.0010526333
3 - 6 - 2 - 5 - 7 :0.0020701928
3 - 6 - 2 - 9 - 7 :0.0020415419
3 - 6 - 8 - 5 - 7 :0.0110796133
3 - 6 - 8 - 9 - 7 :0.0110470033

Camino con menor latencia : número de paquetes que deberán enviarse : tiempo total de transferencia

3 - 6 - 2 - 7 : 12884902: 13563.077226143669
```

Conclusión

Los resultados que arrojó el programa indican que los tiempos de cola de los dispositivos intermedios influyen mucho en el tiempo final de transferencia, algunos caminos cortos tuvieron un tiempo mayor que caminos más largos.

En cuanto al código, fue un poco complicado implementar el sistema de búsqueda de caminos en el grafo ya que es una estructura de la que no tenía muchos conocimientos. Por todo lo demás fue tardado pero no complicado.

Hacer todo el trabajo y estar revisando una y otra vez el código para corregir errores es una buena forma de estudiar porque de tanto repasar se queda muy bien en la memoria cómo se determinan las latencias en una red, cómo se puede saber cuántos paquetes recorrerán la red para que un archivo sea enviado completamente y lo más importante, cuánto tiempo se llevará dicho envío.

Bibliografía

OLMOS, Iván. Curso Redes de Computadoras [en línea]. [Fecha de consulta: 29 de agosto del 2014]. Disponible en: < http://www.cs.buap.mx/~iolmos/ >.

De la Fuente, R (2010). Inteligencia Artificial, Introducción y tareas de búsqueda. [en línea]. Disponible en http://www.aconute.es/iartificial/documentos/ia_intro_busqueda.pdf. [Fecha de consulta: 20 de septiembre del 2014].

Molina, J., Torres, C., Restrepo, C. (2008). Técnicas de inteligencia artificial para la solución de laberintos de estructura desconocida. Scientia et Technica, (39),

137
,13
8.

FACULTAD DE CIENCIAS DE LA COMPUTACION

MODELO DE REDES

Reporte tecnico Programa de Modelo de Redes

Erika Leonor Basurto Munguia

Otoño 2014

Olmo Pineda Ivan

Reporte tecnico:

Este programa calcula la tasa de tranferencias Real,

la cual calcula los bits por segundo que trabaja y el tiempo que tarda de un emisor a un receptor.

El programa obtiene todos lo camino de un punto de partida a un punto de llegada .

En cada camino se obtine su respectiva latencia y el numero de paquete.

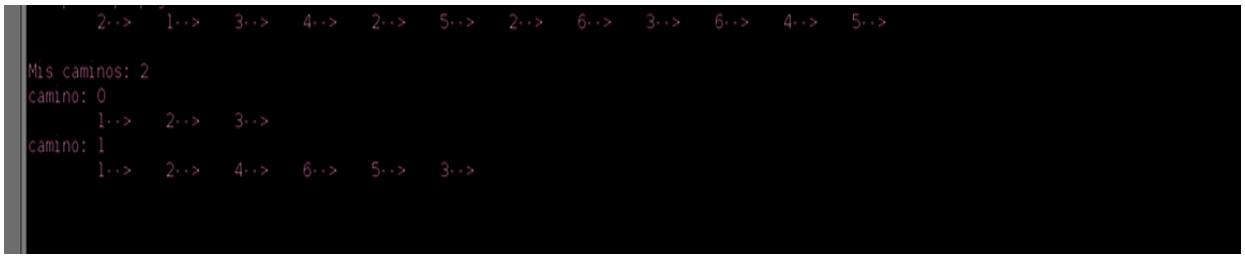
El programa como inicio requiere un archivo.txt que contenga

formato_de_los_grafos.txt

```
#Vertices,#Arcos
Vertice Inicial,Vertice Final,Distancia(metros),Velocidad(Mbps),Tamaño de paquete(B),Datos de control(B)
.
.
.
Tiempo de cola 1, Tiempo de cola2...
Vertice Origen, Vertice Destino
Tamaño del archivo
```

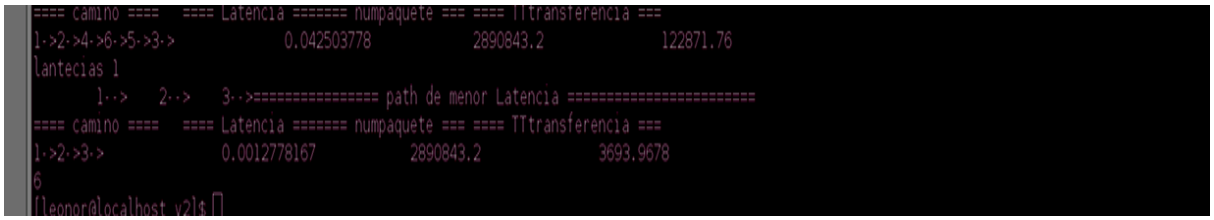
Al ejecutar el programa tiene que ingresar el nombre de archivo

Obtienes todos los caminos quepuede seguir:



Ahora obtendra los caminos y su respectiva informacion:

despues obtendra el o los caminos de menor latencia:



Nota: EL achivo que contiene todala informacion debe de estar en el mismo fichero ademas de que debe ingresar el nombre del archivo correctamente si no el programa no funcionara.

Benemérita universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Puebla, Pue.

Alumna:

Elizeth Escudero Álvarez

Materia:

Modelo de Redes

Reporte Técnico

Índice

Introducción

En el grupo de la materia de modelos de redes de la facultad de Ciencias de la Computación de la Benemérita Universidad Autónoma de Puebla, se encuentra creando programas de cómputo para resolver cálculos sobre los tiempos totales de transferencia en una red.

Aunque ya existen programas que realizan estos cálculos, se realizan pruebas para estos en esta materia.

Para poder realizar estos programas se debe tener conocimientos de programación, las técnicas y herramientas normalmente usadas en la ingeniería de Software.

Este reporte tiene como objetivo presentar los conceptos fundamentales aplicados de la Ingeniería de Software para este programa.

Este reporte se basa en los libros Expermentation in Software Engineering: An Introduction. Basics of Software Engineering Experimentation y Software Metrics- A Rigorous And Practical Approach

2. Enfoques y Estrategias

Este programa está basado en algoritmos de búsqueda en grafos

Ya que estos tienen la necesidad de crear un mecanismo de navegación autónoma. Un grafo, representa un conjunto de nodos unidos en una red. Si dos nodos están unidos, al viajar de uno a otro se considerara sucesor el nodo al que nos movemos, y predecesor el nodo del que venimos. Además, normalmente existirá un coste vinculado al desplazamiento entre nodos. Un algoritmo de búsqueda tratará, de encontrar un camino óptimo entre dos nodos como por ejemplo un camino que minimice el coste de desplazamiento, o el número de pasos a realizar. La principal diferencia entre los algoritmos es la información que guardan a cerca del grafo.

Existen varios tipos de algoritmos de enrutamiento basados en los de búsqueda de grafos como por ejemplo Bellman-Ford está basado en Djikstra.

Para este software de igual forma se basó en el algoritmo de Djistra, ya que este recorre todos los caminos posibles, y nuestro programa requiere calcular todas las latencias de los caminos.

3. Experimentos formales

Cabe mencionar que este programa primero se estaba desarrollando en lenguaje de programación C, pero resultaba un tanto más complejo que el lenguaje en java, así que se optó por dejarlo en java.

También se intentó empezar bajo el algoritmo de Ford-Fulkerson, ya que es el primer algoritmo propuesto para resolver el problema de flujo máximo. Se basa en el concepto de grafo residual (Capacidad residual: es la capacidad adicional de flujo que un arco puede llevar: $c_f(u,v) = c(u,v) - f(u,v)$.

Pero después se decidió por el de Djisktra ya que estos algoritmos se basan en el algoritmo mencionado.

4. Terminología

Bit: Un bit es un dígito del sistema de numeración binario. Byte: un byte debe ser considerado como una secuencia de bits contiguos, cuyo tamaño depende del código de información o código de caracteres en que sea definido.
Kbyte: (KB) es una unidad de almacenamiento de información y equivale a 103 bytes.
Mbyte: (MB) es una unidad de medida de cantidad de datos informáticos que equivale a 106 bytes.

Unidad	Signo	Equivalencia
bit	bit	0 or 1
byte	B	8 bits
kibibit	Kibit	1024 bits
kilobit	kbit	1000 bits
kibibyte (binario)	KiB	1024 bytes
kilobyte (decimal)	kB	1000 bytes
megabit	Mbit	1000 kilobits
mebibyte (binario)	MiB	1024 kibibytes
megabyte (decimal)	MB	1000 kilobytes
gigabit	Gbit	1000 megabits
gibibyte (binario)	GiB	1024 mebibytes
gigabyte (decimal)	GB	1000 megabytes
terabit	Tbit	1000 gigabits
tebibyte (binario)	TiB	1024 gibibytes
terabyte (decimal)	TB	1000 gigabytes
petabit	Pbit	1000 terabits
pebibyte (binario)	PiB	1024 tebibytes
petabyte (decimal)	PB	1000 terabytes
exabit	Ebit	1000 petabits
exbibyte (binario)	EiB	1024 pebibytes
exabyte (decimal)	EB	1000 petabytes

Términos y formulas usadas para la elaboración de código.

La Latencia
Relacionado con el tiempo que toma un bit de viajar de un extremo de un medio al otro.

Depende de tres factores:
Tiempo de propagación del bit por el medio, que depende del tiempo de propagación de la corriente o luz por el medio, además de la distancia recorrida

Máxima cantidad de datos que pueden ser transmitidos por la red sin segmentarse. Al tiempo de propagación del bit en un paquete se le llama tiempo de transmisión.

Tiempos de espera para difundirse un paquete a través de un conmutador, además del tráfico de la red. A este tiempo se le denomina tiempo de cola.

El producto de la latencia por la tasa de transferencia sirve para determinar el tamaño de los buffers para almacenar datos en los sistemas conectados a una red.

Latencia = Tiempo de propagación + Tiempo de transmisión + Tiempo de cola
Tiempo de propagación = distancia a recorrer/ velocidad de la luz
Tiempo de transmisión = tamaño del paquete/ tasa de transferencia teórica

6. Planificación

Tiempo Total de Transferencia

Algoritmo de grafos

Se crea grafo a partir de los datos proporcionados.

Los datos de obtienen desde un archivo de texto

No. De paquetes * Latencia

Algoritmo de Dijsktra que se ha utilizado como base para crear el recorrido

```
DIJKSTRA (Grafo G, nodo_fuente s)
       para u ∈ V[G] hacer
           distancia[u] = INFINITO
           padre[u] = NULL
       distancia[s] = 0
       adicionar (cola, (s, distancia[s]))
       mientras que cola no es vacía hacer
           u = extraer_mínimo(cola)
           para todos v ∈ adyacencia[u] hacer
               si distancia[v] > distancia[u] + peso (u, v) hacer
                   distancia[v] = distancia[u] + peso (u, v)
                   padre[v] = u
                   adicionar(cola,(v, distancia[v]))
```

Lectura de archivo

En esta clase se encarga de abrir y leer el archivo y va obteniendo una cadena con el nombre de todos los nodos.
Crea un grafo vacío, luego los nodos vacíos y los agrega al grafo, ingresa los arcos, y después mandara a llamar a Djikstra

```
public class Grafos {
      public static void main(String[] args) {
      int Cant_Nod=0,Cant_Arc=0;
     int i;

     ArrayList<String> Body,Extra;
     ArrayList<Integer> Nodos;
     int numNod,numArc;
     Procesamiento Data = new Procesamiento("C:\\Users\\Marcus\\datos.txt");
     Nodos = Data.getNodos()

     Grafo Grf = new Grafo();
       for(i=0;i<Cant_Nod;i++){
        Grf.ingresarNodo(""+Nodos.get(i));
     }
```

Clase para ingresar los datos, una vez abierto el archivo de texto esta clase comenzara a leerlo por línea, almacenara los datos, y estos serán acomodados o tomados en cuenta por separado, las líneas están separadas por “,” (comas)

```
public class Datos {
     public  double Distancia;
     public  double Velocidad;
     public  double DC_DU;
     public  double DC;
     public double TP,TT;
     public double LatParcial;


     public Datos(String s){
        String[] datos = s.split(",");
        double a,b;
        a = Double.parseDouble(datos[4]);
        b = Double.parseDouble(datos[5]);

        this.Distancia = Double.parseDouble(datos[2]);
        this.Velocidad = Double.parseDouble(datos[3]);
        this.DC_DU = a-b;
```

```
        this.DC = Double.parseDouble(datos[5]);

        this.TP=0;
        this.TT=0;
        this.LatParcial=0;
    }

}
```

Estructura del Grafo

Representa la estructura grafo, compuesta por nodos y arcos.
Luego se crea un método para las operaciones sobre el grafo, Ingresa un nuevo nodo al grafo, adiciona enlaces entre los nodos, determina si el enlace será dirigido o no dirigido. Se marcara si la arista es dirigida o no.

```
public class Grafo {
    private ArrayList <String>nombres;
    private ArrayList <Arista>aristas;
    private Hashtable <String,Nodo> nodos;
    //Estras para el manejo visual (No es necesario para el funcionamiento)

      public Grafo(){
      nombres=new ArrayList<String>();
         nodos=new Hashtable <String,Nodo>();
         aristas=new ArrayList <Arista>();

    }
  public void ingresarNodo(String nombre){
         nombres.add(nombre);
         nodos.put(nombre,new Nodo(nombre));
    }

  public void addEnlace(String nodoInicial,String nodoTerminal,Datos dat,int opt){
         if(opt == 0){
           nodos.get(nodoInicial).agregarEnlace(nodoTerminal,dat);
         }else{
           nodos.get(nodoInicial).agregarEnlace(nodoTerminal,dat);
           nodos.get(nodoTerminal).agregarEnlace(nodoInicial,dat);
         }
         }
```

Modificación de Djisktra

Referencias

Ingeniería De Software II
Pagina web: ingenieriasoftware12.blogspot.com

Curso Redes de Computadoras, Material Audiovisual del Curso Modelos de Redes.
Profesor Ivan Olmos
Pagina web: www.cs.buap.mx/~iolmos/redes/3_Rendimiento.pdf

Tesis. Algoritmo Ford y Fulkerson Mejorado
Autor, Juan González Oviedo

Gestión del riesgo en la fase de ingeniería de requisitos de un proyecto software
Pagina web: www.monografias.com

Conceptos de Ingeniería de Software Empírica
Cecilia Apa, Rosana Robaina, Stephanie de León, Diego Vallespir.
Pagina web: www.fing.edu.uy/inco/pedeciba/bibliote/reptec/TR1002.pdf

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

Modelo de Redes

Reporte Técnico

Gaona Bernabé Graciela 201114932

Fecha de entrega: 26/Septiembre/2014

Otoño 2014

Reporte Técnico

Este proyecto tiene como finalidad dar a conocer el menor tiempo de transmisión de un paquete en una red, desde un punto A, a un punto B. La red está representada por un grafo, conteniendo aristas y vértices.

Para llegar a nuestro objetivo, es necesario conocer todas las rutas posibles de A hasta B y conocer la latencia para cada camino posible.

La latencia se define como:

Latencia= Tiempo de propagación + Tiempo de transferencia + Tiempo de cola

Tiempo Total de Transferencia= # paquetes *latencia

A continuación se describe el esquema de solución.

El programa está planteado en cinco clases.

1. Nodo: esta clase tiene como atributos los datos de las aristas, como: vértice inicial, vértice final, distancia, velocidad, tamaño del paquete en bytes y datos de control.

Los métodos de esta clase son definidos para darle valor a cada una de las variables antes mencionadas.

También se establecen métodos para calcular los tiempos de propagación y de transmisión, así como para calcular los datos de usuario.

2. Lista: esta clase tiene el comportamiento de una lista ligada, teniendo como atributo una raíz, de tipo Nodo. Tiene los métodos básicos de una lista ligada, como insertar y buscar un elemento x, llamada Regresa_Nodo; y una función adicional para guardar los datos en una Matriz de adyacencia.

3. Caminos: Aquí se encuentra el algoritmo para encontrar todos los caminos posibles de un nodo origen a un nodo destino, recibiendo como parámetro en su constructor: la matriz de adyacencia del grafo, número de vértices, vértice origen y vértice destino.

Todos_Caminos es el método principal para encontrar todas las vías posibles. Está basado en el algoritmo Búsqueda a lo profundo recursivo.

Recibe como parámetros una cadena, origen y destino. Se lleva un registro de los nodos visitados, marcando 1 como visitado y 0 no visitado. En un arreglo de cadenas se guardaran los caminos encontrados. En cada iteración se marca el vértice origen como visitado, y a la cadena se le agrega el origen; si el vértice origen es igual al destino, entonces se agrega la cadena al arreglo T_Caminos; en caso contrario para todos los vértices, al cumplir con la condición de no estar visitados y ser adyacente con el vértice origen, invoca de nuevo a la función Todos_Caminos. Cuando termina de realizar esta condición para todos los vértices, se marca el nodo origen como no visitado.

Al final de este método, se tienen todos los caminos almacenados en un arreglo de cadenas, posteriormente se definen las rutinas para guardar cada camino en una LinkedList, para no preocuparnos por darle dimensión, como lo haríamos con un arreglo normal.

Al finalizar todas las rutinas nuestros caminos estarán almacenados en un arreglo de LinkedList, llamado Caminos_T [ ].

4. Grafo: La clase en la cual se instancian cada una de las clases antes descritas.

Como primer paso, se procede a la lectura del archivo, con el formato solicitado; se almacenan el número de vértices y aristas, en las variables vértices y arcos; se lee los atributos de cada arco con la función datos, y como parámetro de envía una cadena de texto; con la clase StringTokenizer se hace el procedimiento de separación de cadenas, el método datos () regresa un objeto de tipo Nodo, donde se almacenan todos los datos leídos en una línea. Ese nuevo Nodo, se agrega a una Lista Ligada llamada lista.

Se leen los tiempos de cola, los vértices origen y destino, y tamaño del archivo, todos almacenados en variables del mismo nombre, las adyacencias se guardan en una matriz M [ ] [ ].

Para encontrar todos los caminos del grafo leído, en un método llamado Caminos_B, se crea una instancia de la clase Caminos, enviando los parámetros requeridos, aclarando que a los vértices origen y destino se les resta uno, ya que en la clase Caminos se indexa desde cero, sin embargo el resultado no se ve afectado, porque los valores de regreso están indexados desde 1, como en el grafo original.

La variable de la clase grafo, Caminos_T, contendrá todos los caminos encontrados, guardados en un arreglo de LinkedList.

Hasta este punto, solo tenemos los datos completos y los todos los caminos posibles desde un origen a un destino, lo que procede es hacer el cálculo de latencias para cada camino.

En cada arco, almacenado en la variable lista, se realiza el cálculo de los tiempos de transferencia y propagación, de forma individual.

Dentro del procedimiento Latencia_Ind () se hace el cálculo de división de paquetes, que se almacenan en un arreglo de listas llamada Division_Paquetes [ ]. Para llevar un mejor control de dicha división, se define una nueva rutina llamada Divide_Paq, en la cual se recorre la última lista agregada, se comparan los tamaños de paquetes con el actual, y se hacen las subdivisiones necesarias. Ese método regresa una variable entera m, que significa multiplica por, es decir, la variable tiempo de cola, será multiplicada por este valor, lo que supone el tiempo de espera de cada sub paquete.

Ahora, se recorre cada camino de la variable Camino_T [ ], se busca en la lista, cada arco de los caminos, por ejemplo: Camino=1, 2, 4, 5, 6 arcos= 1,2 2,4 4,5 5,6

ás el

licho

encia

chivo

en 1

user

Reporte técnico

Introducción

En la actualidad, una gran cantidad de datos es transmitida mediante una red. La capacidad de transmisión puede ser calculada mediante la *latencia*. Esta está relacionada con el tiempo que toma un bit en viajar de un extremo de un medio a otro.

Existen 3 factores importantes para calcular la latencia:

- Tiempo de propagación del bit por el medio, que depende del tiempo de propagación de la corriente o luz por el medio, además de la distancia recorrida.
- Máxima cantidad de datos que pueden ser transmitidos por la red sin segmentarse. Al tiempo de propagación del bit en un paquete se le llama tiempo de transmisión.
- Tiempos de espera para difundirse un paquete a través de un conmutador, a demás del tráfico de la red. A este tiempo se le denomina tiempo de cola

El presente es el reporte del examen del primer parcial, el cual fue un programa capaz de calcular las latencias de una red.

Desarrollo

Para poder calcular las latencias de una red es necesario utilizar la siguiente formula:

$$Latencia = Tiempo\ de\ propagación + Tiempo\ de\ transmisión + Tiempo\ de\ cola$$

De la cual:

$$Tiempo\ de\ propagación = \frac{distancia\ a\ recorrer}{velocidad\ de\ la\ luz}$$

$$Tiempo\ de\ transmisión = \frac{tamaño\ del\ paquete}{tasa\ de\ transferencia\ teorica}$$

Tiempo de cola= Tiempo que un paquete espera en la cola de salida de un router a ser retransmitido.

El producto de la latencia por la tasa de transferencia sirve para determinar el tamaño de los buffers para almacenar datos en los sistemas conectados a una red.

El programa fue desarrollado en el lenguaje java, en netbeans.

Lo primero que hice fue crear un algoritmo que fuera capaz de obtener todos los paths de un grafo cargado desde archivo. Para ello mi implementación fue basada en el algoritmo BPP(Busqueda Primero en Profundidad) en el cual se hizo un analisis para saber en que momento generar un camino. Este algoritmo de decribe a continuacion: Primero se carga el grafo, despues se establecen el nodo inicial y el nodo final, se empieza a correr el algoritmo el cual va tomando las cabeceras co sus respectivas adyacencias y, si la adyacencia es el nodo final se genera un path el cual es una lista de listas, enseguida se duplica el path y se borra el ultimo elemento para poder continuar con la ejecución del algoritmo recursivo. Aprovechando la recursion, al salir de esta, de la misma manera se borra el nodo final. De esta manera al final tengo todos los caminos posibles del grafo.

Para calcular las latencias todoas las sentencias son secuencuiales ya que no logre implementar recursion para ahorrarme lineas de codigo.

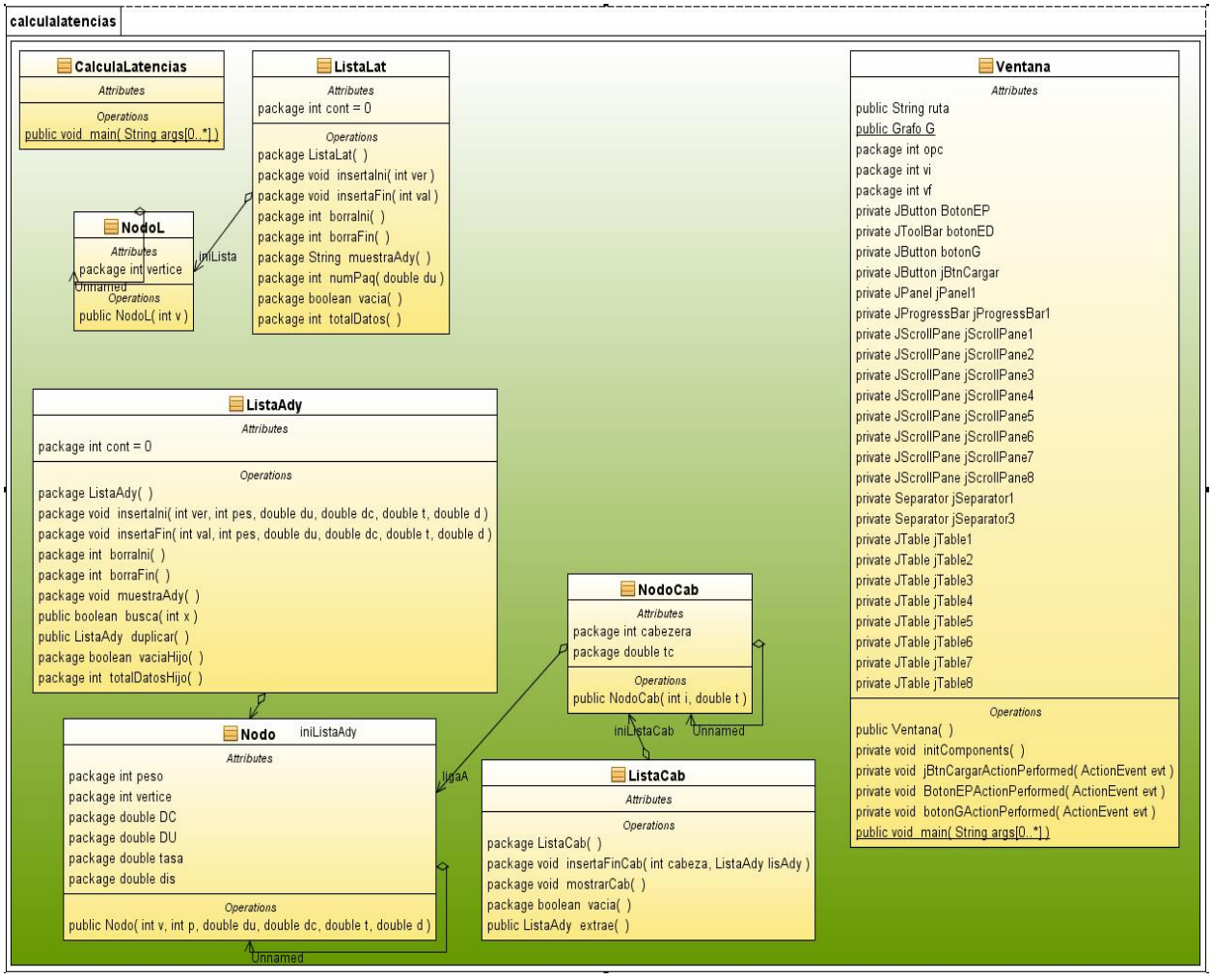
Lo primero que se hace es obtener cada camino y guardarlo en una nueva lista, esto con el fin de no internevir con todos los atributos del nodo original. Enseguida se manda a la funcion que calcula y genera los paquetes de dicho camino.

Una vez que tenemos el numero de paquetes por camino, se procede a efectuar las operaciones para calcular el tiempo de propagacion, de transmision y de cola.

Estos se imprimen en consola, asi como tambien el mejor camino.

Por ultimo se calcula el tiempo total de transferencia de un archivo de tamaño x el cual es asignado por el usuario en megabytes.

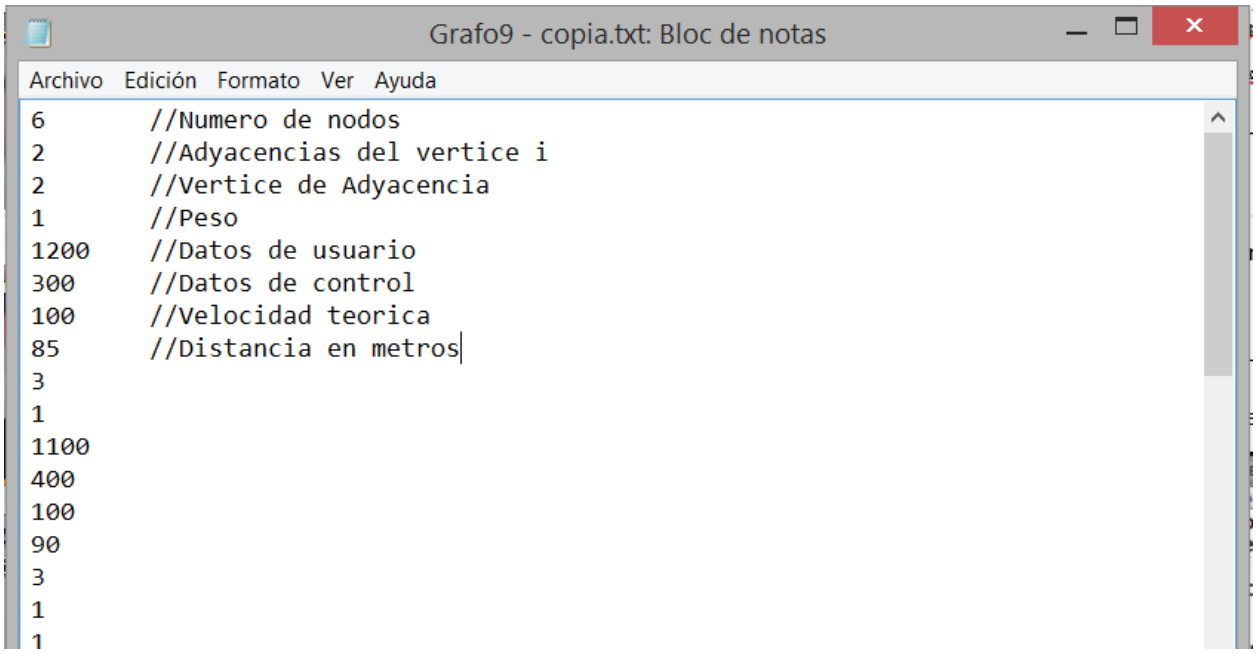
A continuacion se presenta el diagrama UML del programa:



La clase ventana es la encargada de mostrar la interfaz grafica, las clases nodoL, nodo y nodoCab son utilizadas para poder crear mis listas de adyacencias, ya que es la estructura fundamental del programa.

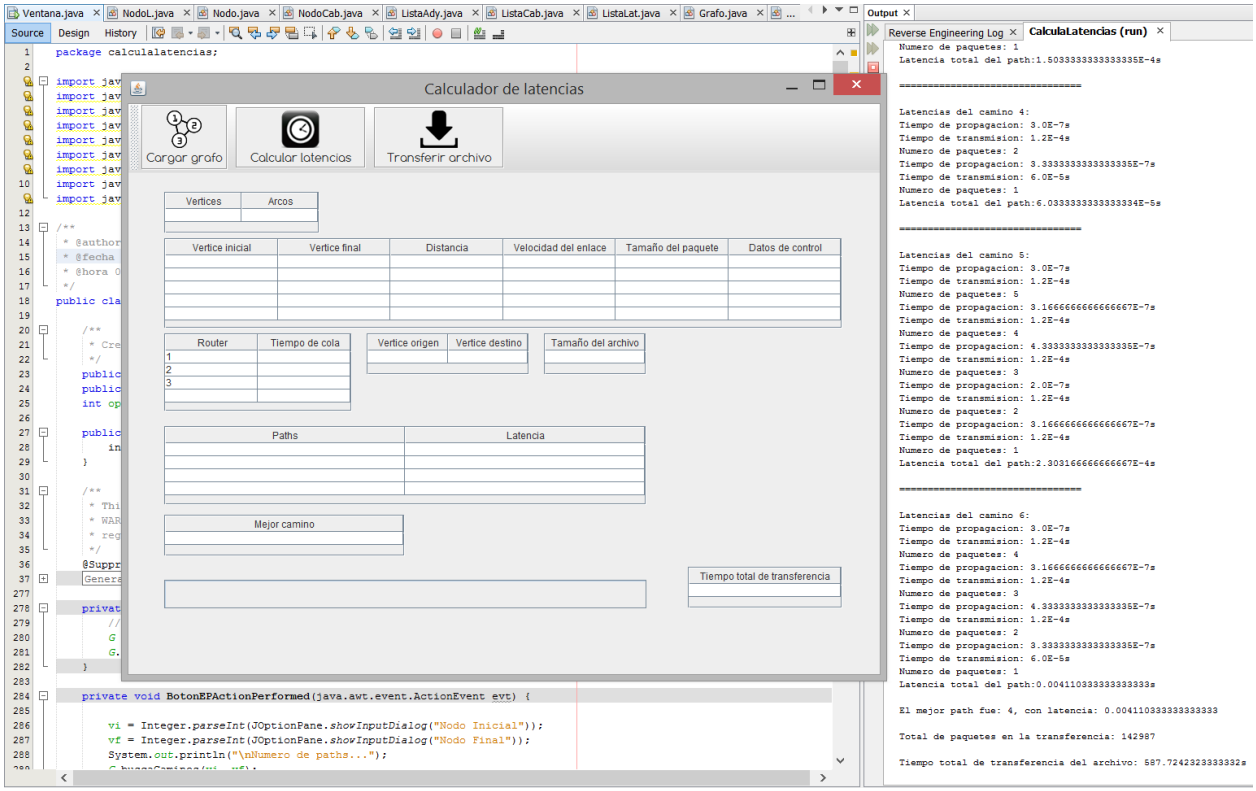
La clase Grafo es la clase principal ya que contiene todos los metodos para generar los paths y las latencias.

Para la creacion del grafo se lee de un archivo de texto el cual contiene los parametros del nodo.

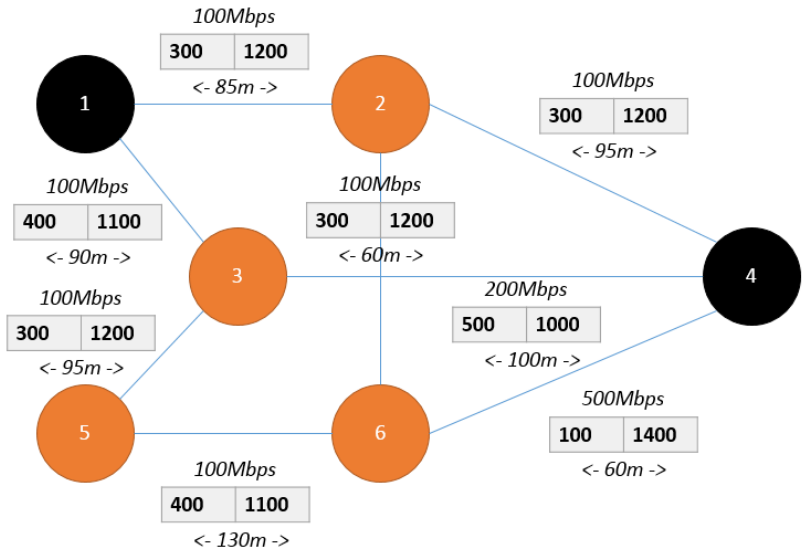


Al final del archivo dice “colas” la cual sirve como delimitador para que apartir de ahí generar una lista de tiempos de colas.

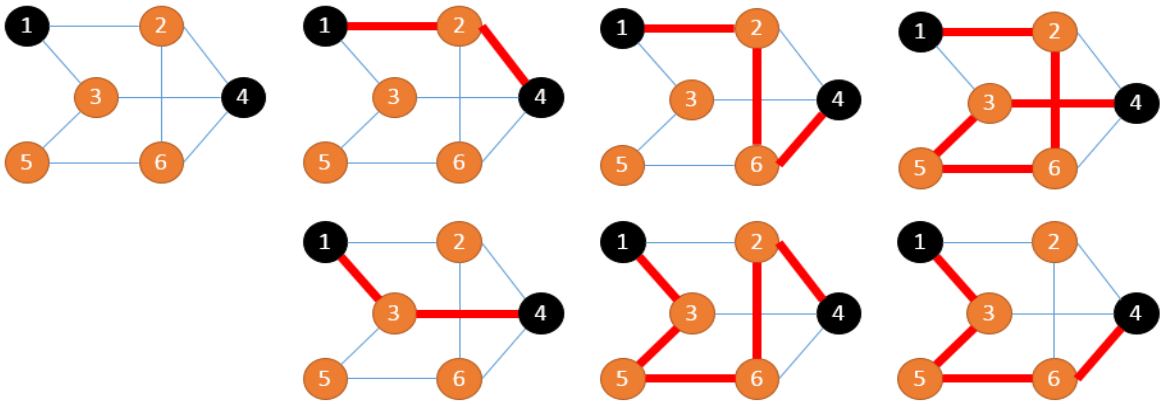
Programa ejecutandose:



Las pruebas se realizaron con este grafo:



Caminos:



Nota: ya no me dio tiempo de implementar todo en mod grafico, asi que las latencias estan en consola.

Conclusiones

Lo aprendido en esta practica fue que los datos que se transmiten en la red, viajan en paquetes y estos paquetes son subdividos en màs paquetes dependiendo del tamaño del paquete en cada router.Un paquete no siempre puede viajar por el mismo camino, sin embargo siempre buscara el mejor camino en eficiencia.

Bibliografía

[1] A. S. Tanembaum, Redes de computadoras, Prentice Hall.

Diapositivas del profesor.

Benemérita universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Puebla, Pue.

Alumnos:

Israel Cruz Guzmán

Elizeth Escudero Álvarez

Materia:

Modelo de Redes

Reporte Técnico

Índice

Introducción

En el grupo de la materia de modelos de redes de la facultad de Ciencias de la Computación de la Benemérita Universidad Autónoma de Puebla, se encuentra creando programas de cómputo para mejorar la seguridad en la comunicación.

Aunque ya existen programas que realizan protegen la comunicacione, se realizan pruebas para estos en esta materia.

Para poder realizar estos programas se debe tener conocimientos de programación, las técnicas y herramientas normalmente usadas en la ingeniería de Software.

Este reporte tiene como objetivo presentar los conceptos fundamentales aplicados de la Ingeniería de Software para este programa.

Este reporte se basa en los libros Expermentation in Software Engineering: An Introduction. Basics of Software Engineering Experimentation y Software Metrics- A Rigorous And Practical Approach

2. Enfoques y Estrategias

Este programa está basado en algoritmos como la comuncicacion mediante sockets y sistema de encriptación MD5.

Los conectores o mejor conocidos como sockets son un mecanismo de comunicación entre procesos que permiten la comunicación bidireccional tanto entre procesos que se ejecutan en una misma máquina como entre procesos lanzados en diferentes máquinas.
Cuando los procesos están en máquinas distintas la comunicación se lleva a cabo a través de redes de ordenadores.

El sistema de encriptación de MD5 utiliza un algoritmo criptográfico de tipo hash que genera como salida 128 bits. Este algoritmo tiene como principal utilidad la de generar 128 bits como resumen de un conjunto de caracteres, independientemente del número de caracteres de entrada que se pasen al algoritmo.

En realidad el algoritmo MD5 no es un algoritmo de encriptación si no que su principal cualidad es la de generar huellas de documentos, una cadena que identificad de forma única a un documento. Pero como vamos a ver, se puede utilizar también para evitar enviar datos en claro.

3. Experimentos formales

La realización de este programa había empezado a desarrollarse en lenguaje de programación C++, pero resultaba un tanto más complejo que el lenguaje en java, así que se optó por dejarlo en java, ya que llega haber problemas que son habituales cuando conectamos con sockets programas que corren en plataformas distintas.

En el mercado hay montones de microprocesadores y cada ordenador decide cual usa. El problema es que cada microprocesador de estos define los enteros, los char, etc, etc como quiere. Lo normal es que un entero, por ejemplo, sean cuatro bytes (32 bits), aunque algunos micros antiguos eran de 2 bytes (16 bits) y los más modernos empiezan a ser de 8 bytes (64 bits).

Si mandamos un entero a través de un socket, estamos enviando estos cuatro bytes. Si los micros a ambos lados del socket tienen el mismo orden para los bytes, no hay problema, pero si tienen distinto, el entero se interpretará incorrectamente.

La máquina virtual Java, por ejemplo, define los char como de 2 bytes (para poder utilizar caracteres UNICODE), mientras que en el resto de los micros habituales suele ser de un byte. Si enviamos desde Java un carácter a través de un socket, enviamos 2 bytes, mientras que si lo leemos del socket desde un programa en C, sólo leeremos un byte, dejando el otro "pendiente" de lectura.

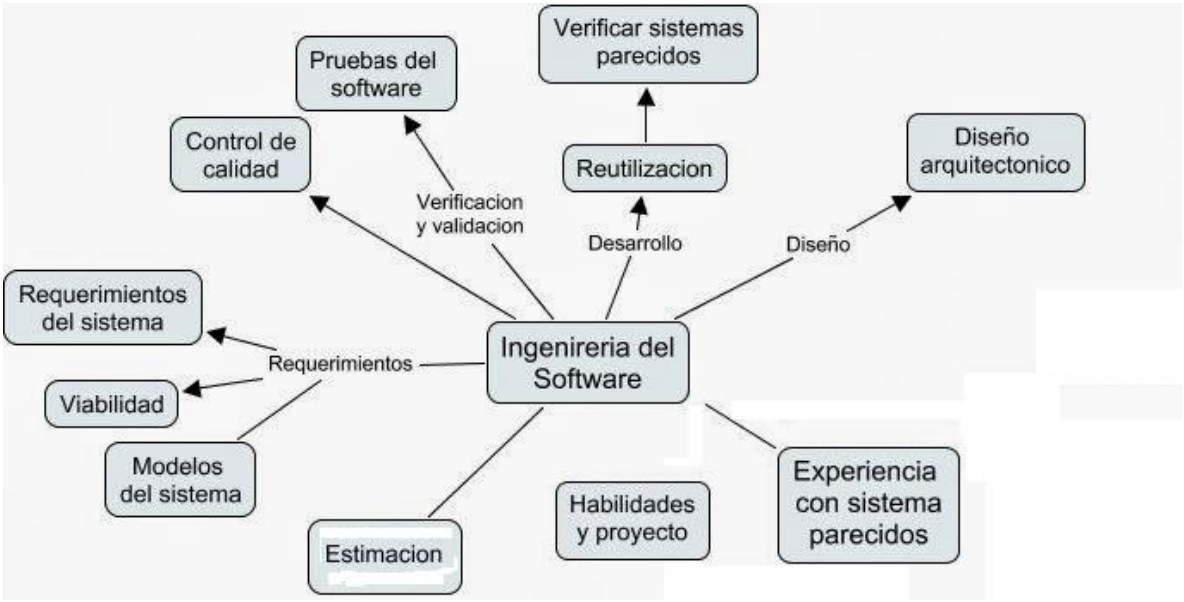
4. Terminología

Sockets:
Designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.
El término socket es también usado como el nombre de una interfaz de programación de aplicaciones (API) para la familia de protocolos de Internet TCP/IP, provista usualmente por el sistema operativo.

Encriptacion:

Es el proceso mediante el cual cierta información o texto sin formato es cifrado de forma que el resultado sea ilegible a menos que se conozcan los datos necesarios para su interpretación. Es una medida de seguridad utilizada para que al momento de almacenar o transmitir información sensible ésta no pueda ser obtenida con facilidad por terceros. Opcionalmente puede existir además un proceso de desencriptación a través del cuál la información puede ser interpretada de nuevo a su estado original, aunque existen métodos de encriptación que no pueden ser revertidos. El término encriptación es traducción literal del inglés y no existe en el idioma español. La forma más correcta de utilizar este término sería cifrado.

6. Planificación



Codigo

PrincipalChat

Se declaran variables a utilizar por parte del servidor y la ip se cambia al cliente que se va a mandar.

```
public class PrincipalChat extends JFrame{
   public JTextField campoTexto; //Para mostrar mensajes de los usuarios
   public JTextArea areaTexto; //Para ingresar mensaje a enviar
   private static ServerSocket servidor; //
   private static Socket conexion; //Socket para conectarse con el cliente
   private static String ip = "127.0.0.1"; //ip a la cual se conecta
```

Se establece el puerto en el cual se va a trabajar y se manda a llamar elhilo para escribir o leer mensajes.

```
    try {
       //main.mostrarMensaje("No se encuentra Servidor");
       servidor = new ServerSocket(11111, 100);
       main.mostrarMensaje("Esperando Cliente ...");

       //Bucle infinito para esperar conexiones de los clientes
       while (true){
          try {
             conexion = servidor.accept(); //Permite al servidor aceptar conexiones

             //main.mostrarMensaje("Conexion Establecida");
             main.mostrarMensaje("Conectado a : " + conexion.getInetAddress().getHostName());

             main.habilitarTexto(true); //permite escribir texto para enviar

             //Ejecucion de los threads
             executor.execute(new ThreadEnvia(conexion, main));
          } catch (IOException ex) {
             Logger.getLogger(PrincipalChat.class.getName()).log(Level.SEVERE, null, ex);
          }
       }
    } catch (IOException ex) {
       Logger.getLogger(PrincipalChat.class.getName()).log(Level.SEVERE, null, ex);
    } //Fin del catch
    finally {
    }
```

ThreadClase envía

Se declaran las variables del servidor, los usuarios y contraseñas que se utilizaran para loguearse, el algoritmo de encriptación.

```
public class ThreadEnvia implements Runnable {
   private final PrincipalChat main;
   private ObjectOutputStream salida;
   private String mensaje;
   private Socket conexion;
   private ObjectInputStream entrada;
   private Socket cliente;
   public String auxc;
   public String aleatorio = "2 ";
   public String[] Usuario = {"Isra","Ale"};
   public String[] Contraseña = {"121289","505050"};
   private static final char[] CONSTS_HEX = { '0','1','2','3','4','5','6','7','8','9','a','b','c','d','e','f' };
```

Una vez logueados Estara pendiente enviar

```
//Evento que ocurre al escribir en el areaTexto
      main.campoTexto.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent event) {
            mensaje = event.getActionCommand();
            enviarDatos(mensaje); //se envia el mensaje
            main.campoTexto.setText(""); //borra el texto del enterfield
         } //Fin metodo actionPerformed
      }
      );//Fin llamada a addActionListener
   }
```

Algoritmo MD5

```
    public static String encriptaEnMD5(String stringAEncriptar)
   {
      try
      {
         MessageDigest msgd = MessageDigest.getInstance("MD5");
         byte[] bytes = msgd.digest(stringAEncriptar.getBytes());
         StringBuilder strbCadenaMD5 = new StringBuilder(2 * bytes.length);
         for (int i = 0; i < bytes.length; i++)
         {
            int bajo = (int)(bytes[i] & 0x0f);
            int alto = (int)((bytes[i] & 0xf0) >> 4);
```

```
            strbCadenaMD5.append(CONSTS_HEX[alto]);
            strbCadenaMD5.append(CONSTS_HEX[bajo]);
        }
        return strbCadenaMD5.toString();
    } catch (NoSuchAlgorithmException e) {
        return null;
    }
}
```

Se leen los mensajes que llegan por parte del usuario, se identifica onn un 1 si se esta intentando loguear, solamente se enviara el nombre de usuario, después se revisa la base datos si existe el nombre de usuario. Si el usuario no existe ”usuario incorrecto”, si el usuario existe se genera un archivo aleatoerio, y se envía al usuario. Se identifica con un 3 la firma digital que envía el usuario. El archivo aleatorio generado por el servidor se le añade la contraseña que esta en la base de datos, el archivo resultante se encripta con el md5 y se genera la firma digital. Una vez generada se compara con la firma enviada por el usuario y si las dos corresponden manda un mensaje de “contraseña correcta”.
Si todo procede en correcccion se da el intercambio de mensajes.

```
public void run() {
    try {
        salida = new ObjectOutputStream(conexion.getOutputStream());
        salida.flush();
    } catch (SocketException ex) {
    } catch (IOException ioException) {
     ioException.printStackTrace();
    } catch (NullPointerException ex) {
    }

    try {
        entrada = new ObjectInputStream(cliente.getInputStream());
    } catch (IOException ex) {
        Logger.getLogger(ThreadRecibe.class.getName()).log(Level.SEVERE, null, ex);
    }
    do { //procesa los mensajes enviados dsd el servidor
        try {//leer el mensaje y mostrarlo
            mensaje = (String) entrada.readObject(); //leer nuevo mensaje
            StringTokenizer tokens=new StringTokenizer(mensaje);
            String str=tokens.nextToken();
            if(str.equals("1")){
                str=tokens.nextToken();
                int encontrado = 0;
                for(int i = 0; i < Usuario.length; i++){
                    if(str.equals(Usuario[i])){
                        encontrado = 1;
                        Random aux;
                        for(int j = 0; j < 13; j++){
                            aux = new Random ();
```

```
                aleatorio = aleatorio + aux;
            }
            salida.writeObject(aleatorio);
            salida.flush(); //flush salida a cliente
            StringTokenizer tokens2=new StringTokenizer(aleatorio);
            String str2=tokens2.nextToken();
            str2=tokens2.nextToken();
            str=tokens.nextToken();
            String md5 = encriptaEnMD5(str2+Contraseña[i]);
            auxc = md5;

          }
        }
        if(encontrado == 0){
          salida.writeObject("Usuario incorrecto");
          salida.flush(); //flush salida a cliente
        }

      }
      else{
        if(str.equals("3")){
          str=tokens.nextToken();

          if(auxc.equals(str)){
            salida.writeObject("Contraseña correcta");
            salida.flush(); //flush salida a cliente
          }
          else{
            salida.writeObject("Contraseña incorrecta");
            salida.flush(); //flush salida a cliente
          }
        }
        else{
          main.mostrarMensaje(mensaje);
        }
      }


    } //fin try
```

Clase : Cliente

Se inicia la interfaz grafica para logueo, y se ejecuta la interfaz para iniciar el chat, se pasa como parámetro el usuario, la contraseña y la ip del servidor.

```
        PrincipalChat a = new
PrincipalChat(jTextField1.getText(),jTextField2.getText(),"127.0.0.1");
```

```
        a.setVisible(true);
        this.dispose();
```

se establece la ip del servidor y se inicializa el hilo con el nombre de usuario, la contraseña y el indicador 1, con esto indicamos al servidor el nombre de usuario.

```
//Ejecucion de los Threads
        executor.execute(new ThreadEnvia(cliente, this,"1"+" "+Usuario+" "+Contraseña));
```

Clase: ThreadEnvia

Con este método solo dejamos el indicar 1, el usuario, y se envían estos datos al servidor, la contraseña que se paso de la clase anterior no sale del equipo.

```
  public void enviarLogin(String mensa){
    StringTokenizer tokens=new StringTokenizer(mensa);
    String str=tokens.nextToken();
    String mensaj;
    System.out.println(mensa);
    mensaj = str;
    str=tokens.nextToken();
    mensaj = mensaj+" "+str;
    str=tokens.nextToken();
    String md5 = encriptaEnMD5(str);
    mensaj = mensaj + " " + md5;

      try {
       salida.writeObject(mensaj);
       System.out.println("mensaje: "+mensaj);
       salida.flush(); //flush salida a cliente
       //main.mostrarMensaje(main.nick+">>> " + mensaje);
    } //Fin try
    catch (IOException ioException){
       main.mostrarMensaje("Error escribiendo Mensaje");
    } //Fin catch

  }
```

Se recibe el mensaje del servidor revisando si el usuario existe, si el usuario existe se recibe con el indicador 2, el archivo aleatorio generado por el servidor. Este archivo aleatorio se le agrega la constraseña que esta guardada en el equipo del cliente y se encripta con md5 para obtener la firma digital.
Con el indicador 3 se envía la firma digital al servidor. Si las firmas digitales coinciden, se recibe mensaje de contraseña correcta y se procede al envio de mensajes , en caso contrario de haber un error, nos regresa a la ventana donde nos pide usuario y contraseña.

```
public void run() {
    try {
      salida = new ObjectOutputStream(conexion.getOutputStream());
      salida.flush();
      enviarLogin(login);
   } catch (SocketException ex) {
   } catch (IOException ioException) {
    ioException.printStackTrace();
   } catch (NullPointerException ex) {
   }
    try {
      entrada = new ObjectInputStream(cliente.getInputStream());
   } catch (IOException ex) {
      Logger.getLogger(ThreadRecibe.class.getName()).log(Level.SEVERE, null, ex);
   }
   do { //procesa los mensajes enviados desde el servidor
      try {//leer el mensaje y mostrarlo
         mensaje = (String) entrada.readObject(); //leer nuevo mensaje
         StringTokenizer tokens=new StringTokenizer(mensaje);
         String str=tokens.nextToken();

         if(mensaje.equals("Contraseña incorrecta")){
            cliente b = new cliente("Contraseña incorrecta");
            b.setVisible(true);

            //b.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
            main.dispose();
         }
         if(mensaje.equals("Usuario incorrecto")){
            cliente b = new cliente("Usuario incorrecto");
            b.setVisible(true);

            //b.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
            main.dispose();
         }
         if(str.equals("2")){
            str=tokens.nextToken();
            String auxc = "3 "+str+main.Contraseña;
            String md5 = encriptaEnMD5(str+main.Contraseña);
```

```
            salida.writeObject("3 "+md5);
            salida.flush();
         }
         //System.out.println(mensaje);
         else{
            main.mostrarMensaje(mensaje);
         }
      } //fin try
```

Referencias

Ingeniería De Software II
Pagina web: ingenieriasoftware12.blogspot.com

Curso Redes de Computadoras, Material Audiovisual del Curso Modelos de Redes.
Profesor Ivan Olmos
Pagina web: www.cs.buap.mx/~iolmos/redes/3_Rendimiento.pdf

Encriptacion
http://encripdedatos.blogspot.mx/Autor, Juan González Oviedo

Wikipedia
https://www.google.com.mx/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#

EN 1

MODELO DE REDES

PROFESOR: Dr. Ivan Olmos Pineda

ALUMNOS:

Jordy Joaquín Cuan Robledo

Roberto Jiménez Coronado

Iván Lozano Sánchez

Rafael Pérez Aguirre

Alejandro Quiroz Flores

Introducción

A partir de conocimientos aprendidos (en mayor o menor medida) durante el ultimo periodo de la clase, se nos pidió programaramos una aplicación que los pusiera en práctica resolviendo un problema de la vida real, tradicional para la enseñanza de intercomunicación de equipos de computo en redes: un chat mediante el uso de sockets.

No obstante, los requerimientos de dicho programa fueron elevados a un mayor grado de dificultad intentando aplicar conceptos como encriptación de mensajes y aplicación de huellas digitales, lo cual proporciona una mayor seguridad para los clientes de nuestro programa en la red. Además, como último reto para nuestro equipo, se agregaron los requerimientos del uso del gestor de bases de datos de gran escalabilidad, como PostgreSQL, y la transferencia de archivos dentro del mismo chat.

No está de más mencionar la importancia que hoy en día tiene la seguridad en las comunicaciones mediante redes de computadoras. Debido a posibles robos de información importante y/o de carácter privado, el tema de seguridad, al igual que el correcto manejo de datos y atractivas funciones son ya esenciales para la gran parte de programas en la actualidad,

Problema

Programar una aplicación de Chat, la cual implemente algunos de los conceptos aprendidos durante la materia de Modelos de Redes.

En nuestro caso, la aplicación es capaz de realizar tareas como:

- Uso de encriptación (AES) y huellas digitales (MD5) para la transmisión segura de datos
- Uso de bases de datos mediante un gestor robusto (PostgreSQL).
- Comunicaciones mediante sockets, establecidas entre pares específicos de clientes.
- Envío de mensajes de texto y archivos de audio/video/documentos/etc…

Marco teórico

- **MD5** es un algoritmo de reducción criptográfico, desarrollado en 1991 y diseñado por el profesor Ronald Rivest del MIT (Massachusetts Institute of Technology). Permite obtener una huella digital de una serie de datos de entrada. Con la huella se puede, por ejemplo, verificar la integridad de los datos, es decir, comprobar que no hayan sufrido alguna alteración.
    - Genera una huella de salida hexadecimal de tamaño determinado para cualquier tipo de entrada, sin importar tipo de dato, longitud o tamaño.

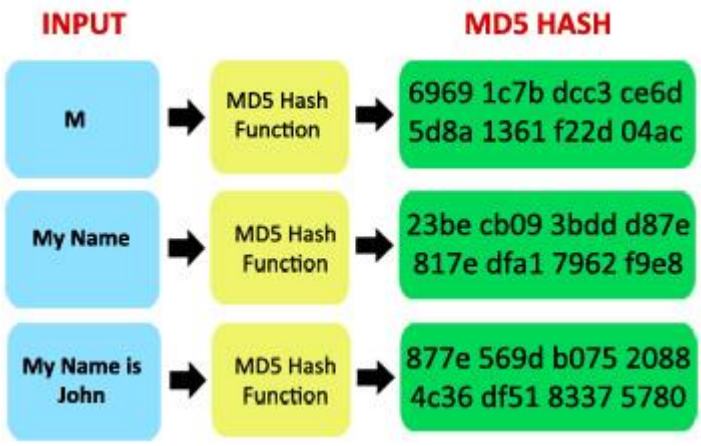


Ilustración 1 - Pequeño ejemplo de uso de huella MD5

- **AES** (Advanced Encryption Standard)es un algoritmo de cifrado por bloques simétricos de datos. Se desarrolló en 1997, por el Instituto Nacional de Normas y Tecnología (NIST) con el propósito de proteger información sensible.
    - Es capaz de usar llaves criptográficas de 128,192 y 256 bits para encriptar y desencriptar datos en bloques de 128 bits.

- **PostGreSQL** es un sistema de gestión de bases de datos objeto-relacional basado en el proyecto POSTGRES, de la universidad de Berkeley. PostGreSQL es una derivación libre (OpenSource) de este proyecto, y utiliza el lenguaje SQL92/SQL99.
    - Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. PostGreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostGreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos. Permite la declaración de funciones propias, así como la definición de disparadores.

- **Socket** es un método para la comunicación entre dos programas en una red. Un socket se define como el punto final en una conexión. Los sockets se crean y se utilizan con un sistema de peticiones o de llamadas de función a veces llamados interfaz de programación de aplicación de sockets (API, application programming interface).

- Es una dirección de Internet, determinada al combinar una dirección IP y un número de puerto. Esto permite que un enlace determine no solo la dirección de dispositivo, sino a qué aplicación de ese dispositivo va dirigida la conexión.

Especificaciones de la Implementación

El proyecto fue aplicado y programado en el lenguaje Java e conjunto con SQL debido a las facilidades que proporciona el primero para el uso de sockets y el segundo por ser el utilizado por el manejador de bases de datos.

El funcionamiento general es el siguiente: A partir de dos componentes esenciales en el programa, el servidor y los clientes, se busca establecer comunicaciones seguras entre un par específico de clientes.

Se programó una intuitiva y sencilla interfaz gráfica para los clientes que permite realizar tareas sencillas como seleccionar un contacto para chatear, escribir y ver los mensajes enviados con un contacto específico.

Se utiliza una pantalla de “login y registro” en el cliente que verifique su identidad comparando los datos enviados con la base de datos del servidor. En caso de no haer sido registrado aún, se agrega al cliente a la base de datos del servidor. La contraseña se envía y procesa mediante la implementación de una huella MD5.

Se utilizan bases de datos para llevar una tabla de usuarios registrados y de usuarios “contacto” de otro cliente, los cuales seran los únios con los que dicho cliente puede comunicarse.

El tipo de encriptamiento utilizado para los mensajes enviados entre clientes es AES (Advanced Encryption Standard).

Capturas de Pantalla de la Ejecución

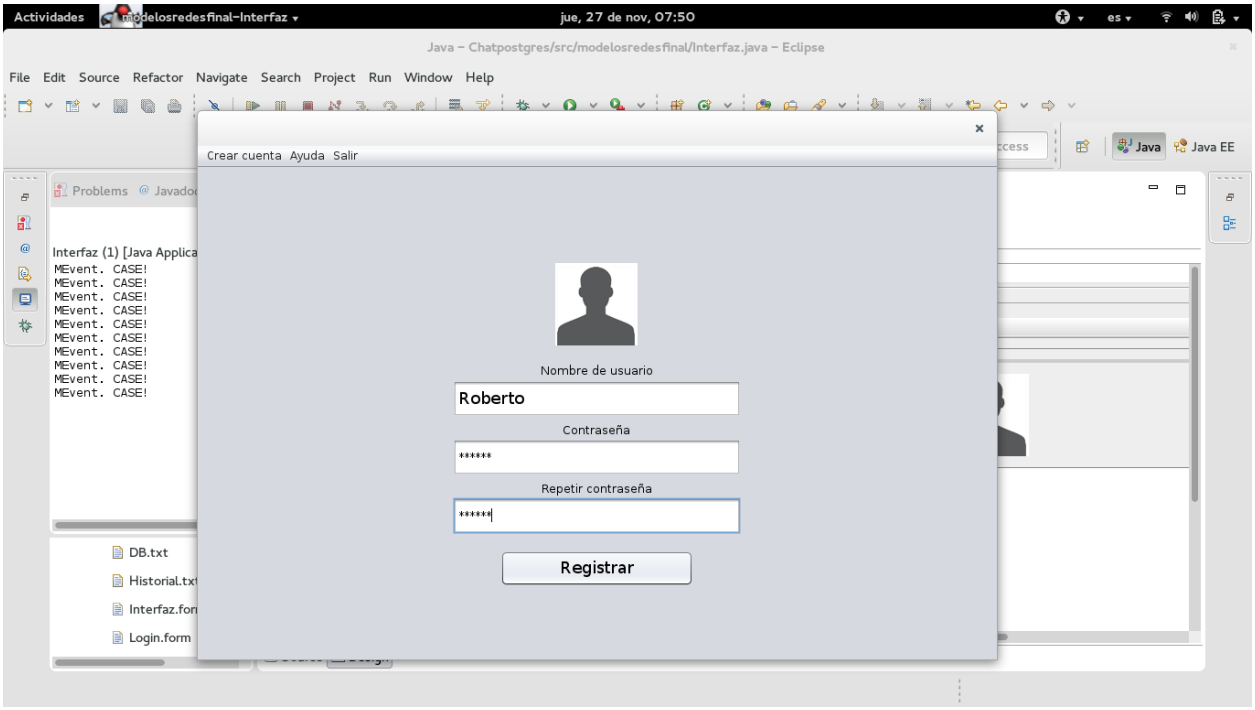


Ilustración 2 – Creación de cuenta

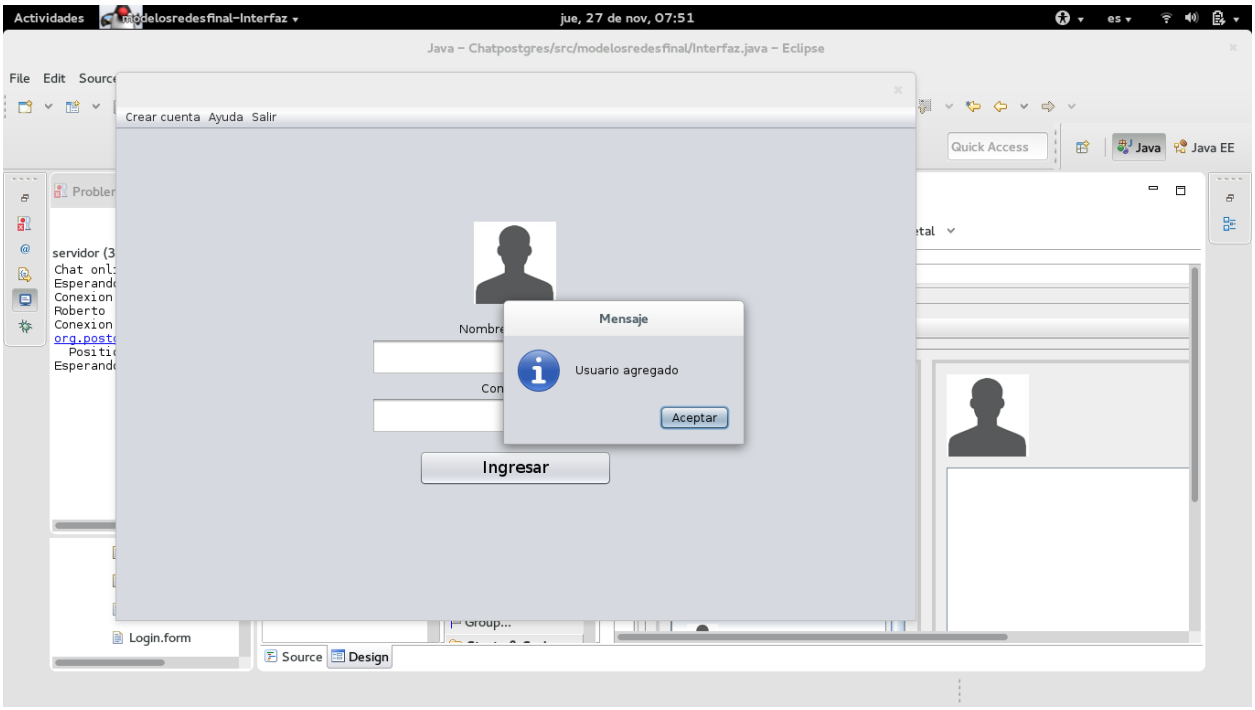


Ilustración 3 - Confirmación de creación de cuenta. Muestra buen manejo de la base de datos de usuarios

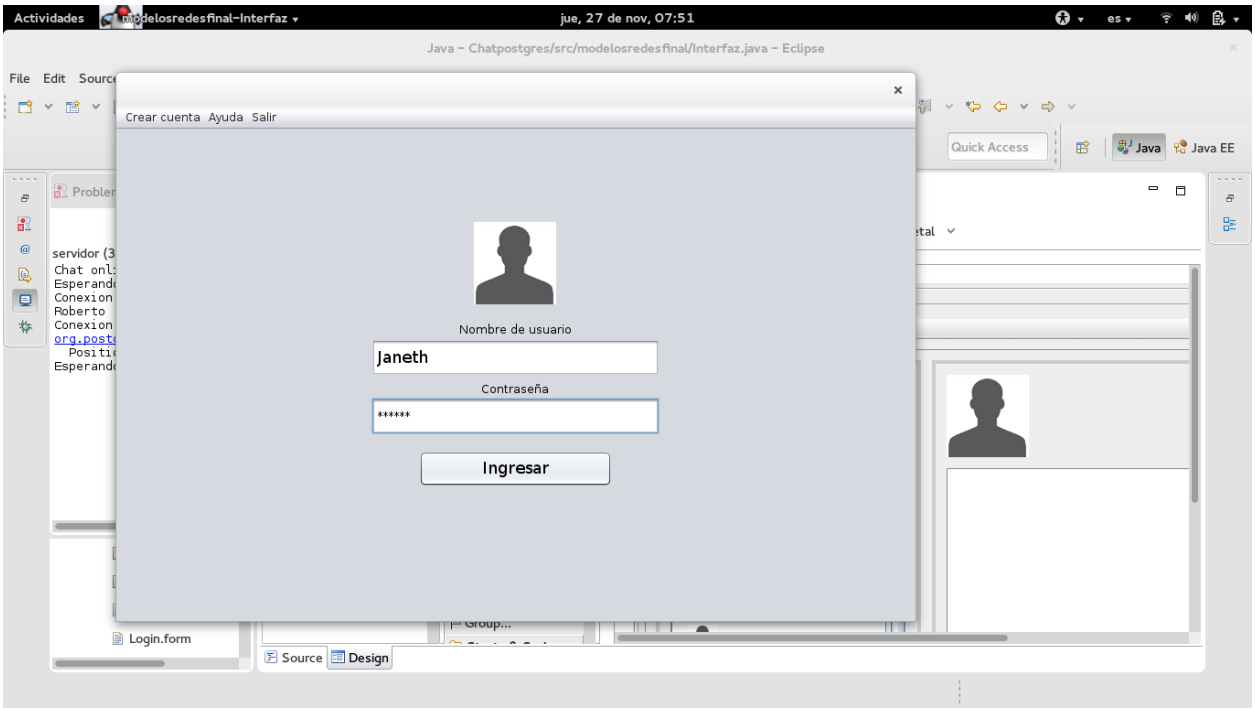


Ilustración 4 – Pantalla de logeo simple.

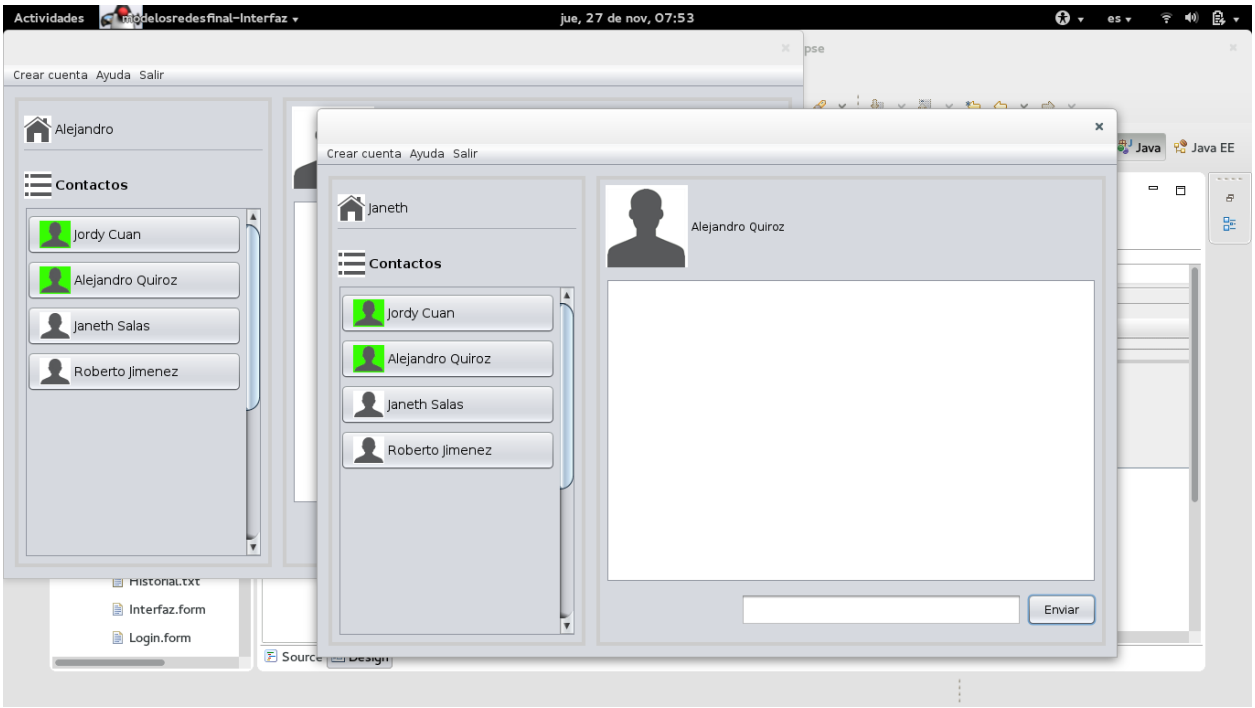


Ilustración 5 – Interfaz de clientes logueados al mismo tiempo.

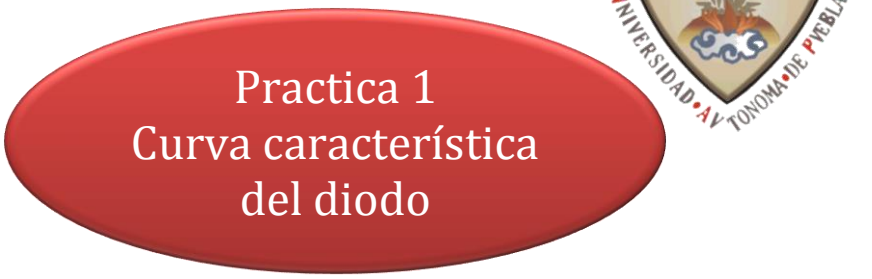
Conclusión

La implementación de herramientas de programación como las que utilizamos en éste proyecto (encriptamiento, uso de huella digital MD5, etc.) representaron un duro reto para su finalización, a pesar de haber utilizado y creado con anterioridad programas con el mismo objetivo principal (Chat con comunicación por sockets).

Por fortuna, pudimos resolver dichas problemáticas individualmente, aunque no pudieron verse todas aplicadas en conjunto por cuestiones de tiempo. Para muestra de ello, se encuentran nuestros códigos auxiliares adjuntos a este proyecto.

Otro punto a destacar es la importancia de entender que éstos requerimientos tan mencionados no son simplemente aprendidos y requeridos en un único proyecto de Modelos de Redes (incluso hay herramientas que nunca aprendimos en éste curso), sino que son exigencias de todo programa formal de actualidad, por lo que nos fue de muy gran ayuda su desarrollo e implementación.

en 1



Jordy Joaquin Cuan Robledo

Modelos de Redes

Iván Olmos Pineda

Puebla, Pue. a 26 de septiembre de 2014

Objetivo

Implementar el algoritmo de búsqueda de la mejor latencia para una red finita dada, encontrando todos los caminos (paths) entre nodos y calculando su respectiva latencia, esto para determinar el tiempo que se tomaría un archivo al ser enviado de una computadora a otra atreves de la red proporcionada, con el uso de un algoritmo visto en clase que calcula latencias.

Introducción

Se nos ha requerido como primer examen parcial implementar el algoritmo de latencias que calcula el tiempo de latencia que tiene un paquete al ser enviado a través de una computadora a otra computadora sobre una red dada mediante un fichero. ¿Cómo conseguiremos estas latencias?

Bueno, para el cálculo de las latencias por camino de red existe una formula, descrita a continuación:

$$Latencia = Tiempo\ Propagación + Tiempo\ Transmisión + Tiempo\ de\ Cola$$

Más clara, la formula vendría siendo la siguiente:

$$Latencia = \frac{Distancia\ a\ recorrer}{Velocidad\ de\ la\ Luz} + \frac{Tamaño\ del\ paquete}{Tasa\ de\ transferencia} + Tiempo\ de\ Cola$$

Donde:

- *Distancia a recorrer* es la longitud del cable.
- *Velocidad de la luz* es la constante definida por el IEEE que es $300{,}000\ Kms/s$
- *Tamaño del paquete* es la suma de los datos de control y los datos de usuario a ser transmitidos en el paquete (en Bytes).

- *Tasa de Transferencia* es la velocidad ideal de red.
- *Tiempo de Cola* es el tiempo de retraso que tiene un router que consta del instante en que este recibe un paquete, lo analiza y lo manda hacía afuera.

Toda esta información será proporcionada en un fichero dado por el profesor. Además este fichero tendrá información para la ejecución, como el número de vértices y enlaces, la relación de los nodos mediante enlaces, tamaño de archivo y otros (a detalle más adelante).

Y con estas latencias que hemos calculado, ¿Cómo conseguiríamos el tiempo de transferencia? Bien, ya que tenemos la latencia del camino lo único que debemos hacer es conseguir el número total de paquetes a enviar a través del primer enlace y de ahí conseguimos la proporción de estos con la latencia (que se tiene por cada paquete individual). Para que quede más claro:

$$Tiempo\ de\ Transferencia = Total\ de\ paquetes\ \ x\ \ Latencia\ del\ camino$$

Con eso bastará, fácil ¿no? Eso es para un camino, vamos a complicarnos la vida.

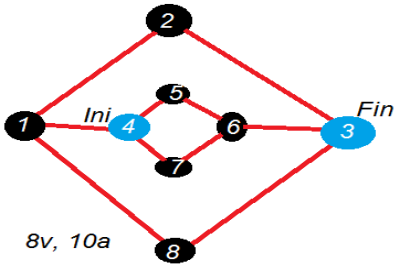
Algoritmo

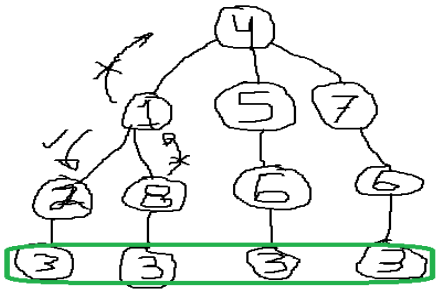
Para este problema, se expuso en la introducción la solución para un camino, pero por cada enlace que existe entre el nodo de inicio y el final hay que hacer una sumatoria volviéndolo un poco más complejo:

$$\left(\sum_{Primer\ Enlace}^{Total\ Enlaces-1} \#Paquetes\ a\ Enviar\ (Tprop + Ttrans) + TC_{enlaceActual}\right)$$

$$+Tprop_{ultimoEnlace} + Ttrans_{ultimo\ Enlace} = LATENCIA\ total\ del\ camino$$

Ahora bien, para poder encontrar la latencia total de un camino, necesitamos determinar el camino, o más claro, encontrar todos los caminos existentes entre el nodo inicial y el nodo final.





Superficialmente, lo que hace el algoritmo es:

1. Tomamos el primer nodo y vemos todos sus enlaces.

2. Para cada enlace, creamos una lista a partir de la lista de la anterior llamada del algoritmo, verificamos que no se haya recorrido con anterioridad, si no es así, añadimos a una lista y comparamos si es el nodo final, de ser el final se añade esta lista temporal de caminos a la lista principal y se hace el regreso de la llama. De no ser así, el algoritmo se vuelve a llamar.
3. Se retorna la lista de caminos.

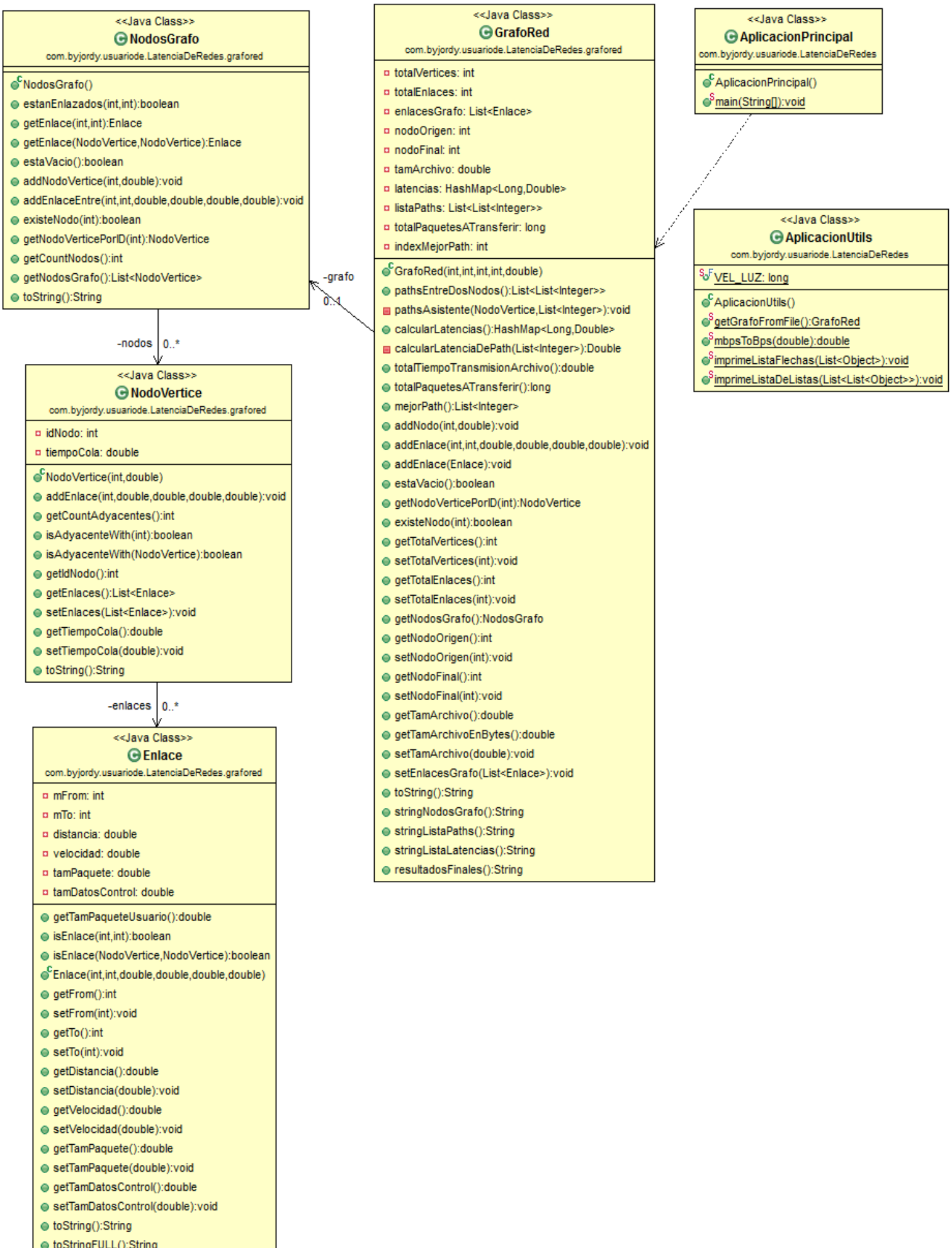
De esto podemos observar que se ha usado una lista de listas del índice del nodo. Así estamos guardado todos los posibles caminos. También se puede observar por la recursividad del algoritmo que estamos usando una implementación de *DFS (Deep-First Search, Búsqueda primero en profundidad)* modificado, pues no estamos tomando en cuenta si ya ha sido visitado en un recorrido anterior. Solo verificamos que no se encuentre dos veces en nuestro mismo camino.

Implementación

Para la implementación se ha utilizado el IDE de trabajo Eclipse, donde se ha desarrollado todo el código y el diagrama UML, mediante su plugin llamado ObjectAID. Además, se han empleado pequeñas clases propiedad de Oracle que he utilizado para implementar la lista de listas y la apertura del fichero. Estas son:

- Clase *List<E>* – Una lista de listas se vería así List<List<Integer>>
- Clase *JFileChooser* – Que nos permite abrir el archivo de modo gráfico

¿Mencioné un diagrama UML? Sí, es el diagrama de la implementación para el código. En él se muestra como están estructuradas todas nuestras clases, su relación, los atributos, y los métodos de cada objeto.


<<Java Class>>
NodosGrafo
com.byjordy.usuariode.LatenciaDeRedes.grafored
NodosGrafo()
estanEnlazados(int,int):boolean
getEnlace(int,int):Enlace
getEnlace(NodoVertice,NodoVertice):Enlace
estaVacio():boolean
addNodoVertice(int,double):void
addEnlaceEntre(int,int,double,double,double,double):void
existeNodo(int):boolean
getNodoVerticePorID(int):NodoVertice
getCountNodos():int
getNodosGrafo():List<NodoVertice>
toString():String
-nodos
0..*
<<Java Class>>
NodoVertice
com.byjordy.usuariode.LatenciaDeRedes.grafored
idNodo: int
tiempoCola: double
NodoVertice(int,double)
addEnlace(int,double,double,double,double):void
getCountAdyacentes():int
isAdyacenteWith(int):boolean
isAdyacenteWith(NodoVertice):boolean
getIdNodo():int
getEnlaces():List<Enlace>
setEnlaces(List<Enlace>):void
getTiempoCola():double
setTiempoCola(double):void
toString():String
-enlaces
0..*
<<Java Class>>
Enlace
com.byjordy.usuariode.LatenciaDeRedes.grafored
mFrom: int
mTo: int
distancia: double
velocidad: double
tamPaquete: double
tamDatosControl: double
getTamPaqueteUsuario():double
isEnlace(int,int):boolean
isEnlace(NodoVertice,NodoVertice):boolean
Enlace(int,int,double,double,double,double)
getFrom():int
setFrom(int):void
getTo():int
setTo(int):void
getDistancia():double
setDistancia(double):void
getVelocidad():double
setVelocidad(double):void
getTamPaquete():double
setTamPaquete(double):void
getTamDatosControl():double
setTamDatosControl(double):void
toString():String
toStringFULL():String
-grafo
0..1
<<Java Class>>
GrafoRed
com.byjordy.usuariode.LatenciaDeRedes.grafored
totalVertices: int
totalEnlaces: int
enlacesGrafo: List<Enlace>
nodoOrigen: int
nodoFinal: int
tamArchivo: double
latencias: HashMap<Long,Double>
listaPaths: List<List<Integer>>
totalPaquetesATransferir: long
indexMejorPath: int
GrafoRed(int,int,int,int,double)
pathsEntreDosNodos():List<List<Integer>>
pathsAsistente(NodoVertice,List<Integer>):void
calcularLatencias():HashMap<Long,Double>
calcularLatenciaDePath(List<Integer>):Double
totalTiempoTransmisionArchivo():double
totalPaquetesATransferir():long
mejorPath():List<Integer>
addNodo(int,double):void
addEnlace(int,int,double,double,double,double):void
addEnlace(Enlace):void
estaVacio():boolean
getNodoVerticePorID(int):NodoVertice
existeNodo(int):boolean
getTotalVertices():int
setTotalVertices(int):void
getTotalEnlaces():int
setTotalEnlaces(int):void
getNodosGrafo():NodosGrafo
getNodoOrigen():int
setNodoOrigen(int):void
getNodoFinal():int
setNodoFinal(int):void
getTamArchivo():double
getTamArchivoEnBytes():double
setTamArchivo(double):void
setEnlacesGrafo(List<Enlace>):void
toString():String
stringNodosGrafo():String
stringListaPaths():String
stringListaLatencias():String
resultadosFinales():String
<<Java Class>>
AplicacionPrincipal
com.byjordy.usuariode.LatenciaDeRedes
AplicacionPrincipal()
main(String[]):void
<<Java Class>>
AplicacionUtils
com.byjordy.usuariode.LatenciaDeRedes
VEL_LUZ: long
AplicacionUtils()
getGrafoFromFile():GrafoRed
mbpsToBps(double):double
imprimeListaFlechas(List<Object>):void
imprimeListaDeListas(List<List<Object>>):void

Ejecución y Pruebas

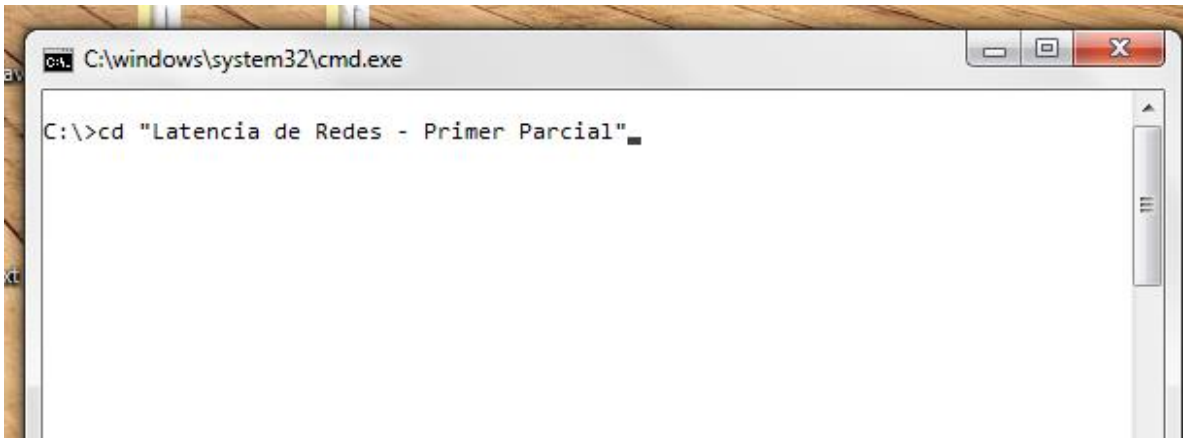
Para poder ejecutar el programa, empezaré diciendo que **solamente trabaja en consola**, cualquier intento de ejecución en otro lado no permitirá visualizar los resultados de modo funcional.

Habiendo aclarado cómo funciona el programa, explico que para la ejecución del jar de nuestro proyecto, se hace de la siguiente forma:

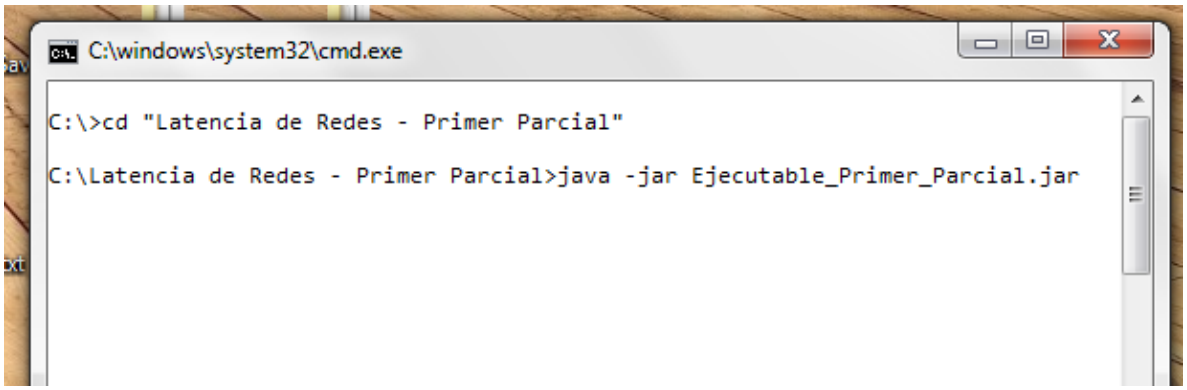
$$\boldsymbol{Dir:>\ \ java\ \ \ -jar\ \ \ <NombreDelEjecutable>.jar}$$

Bien, ahora solo nos falta ejecutarlo y ver los resultados que proporciona. Vamos:

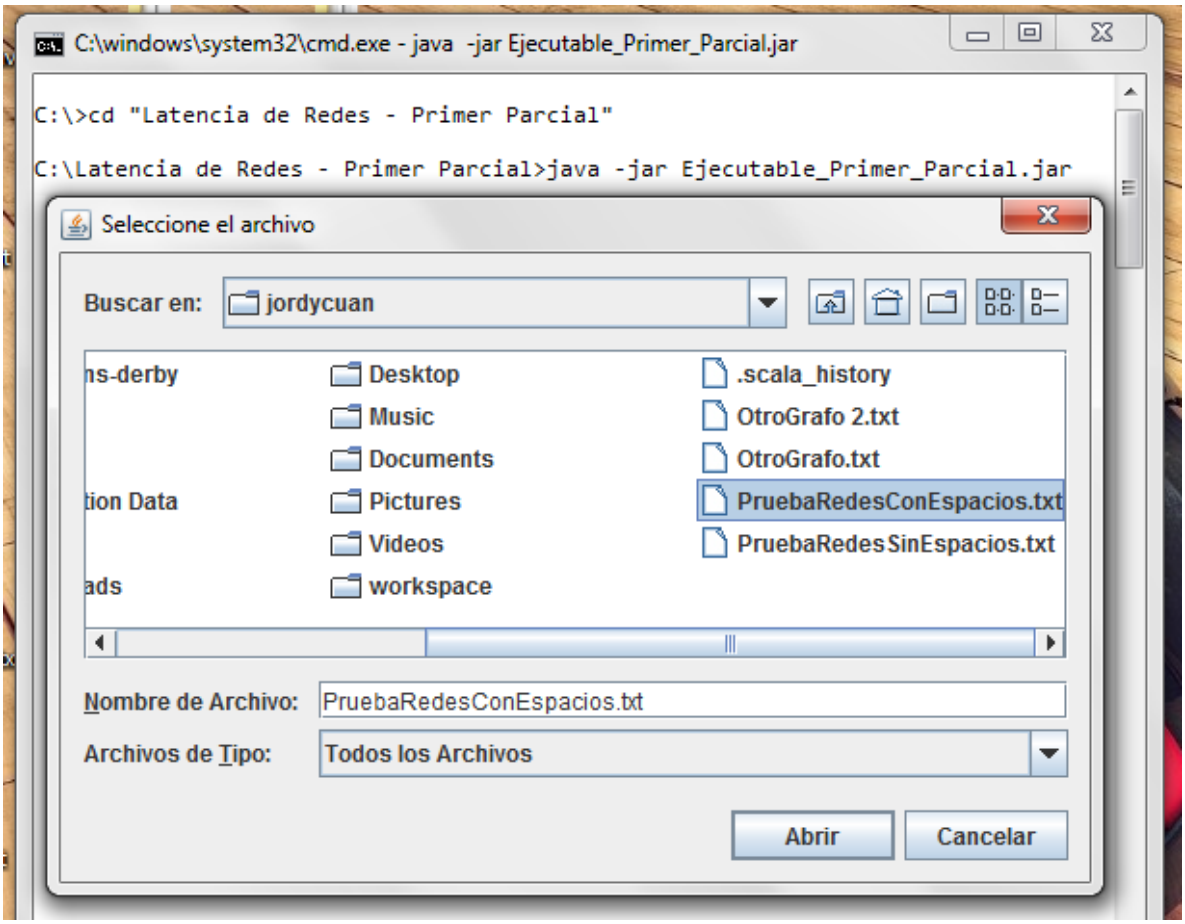
1. En la consola de nuestro SO, nos dirigimos a la carpeta donde está nuestro ejecutable jar.



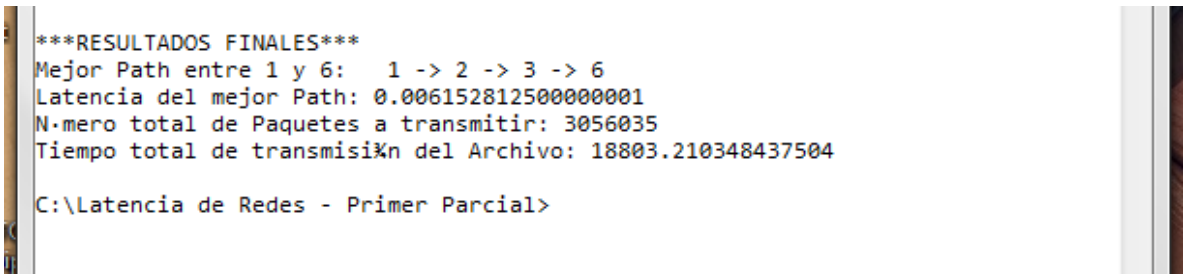
2. Ya dentro de la carpeta, pasamos a ejecutar nuestro proyecto.



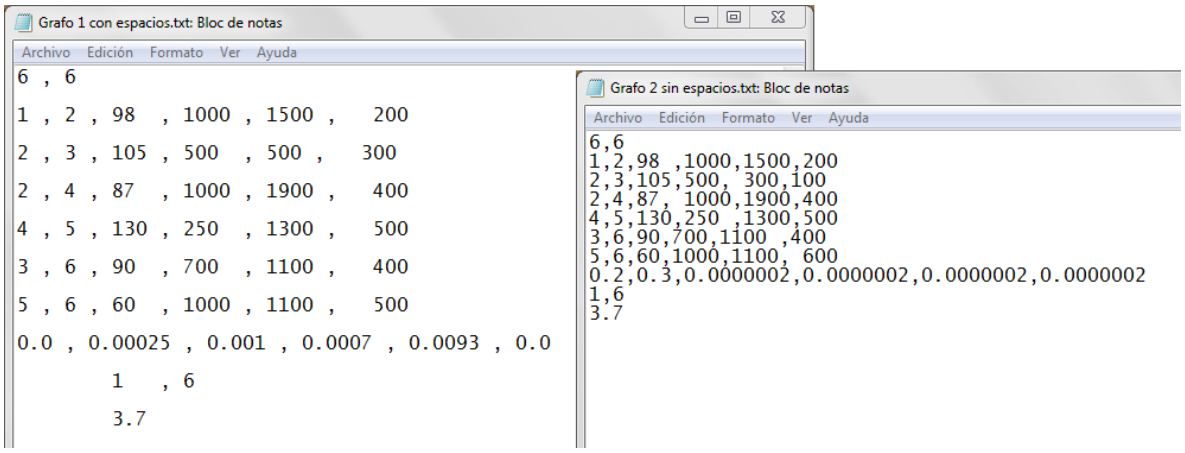
3. Al principio nos pedirá especifiquemos un fichero válido para nuestro programa. Busquemos uno y lo abrimos.



4. El programa hará los cálculos necesarios y presentará en pantalla los resultados conseguidos en su ejecución.

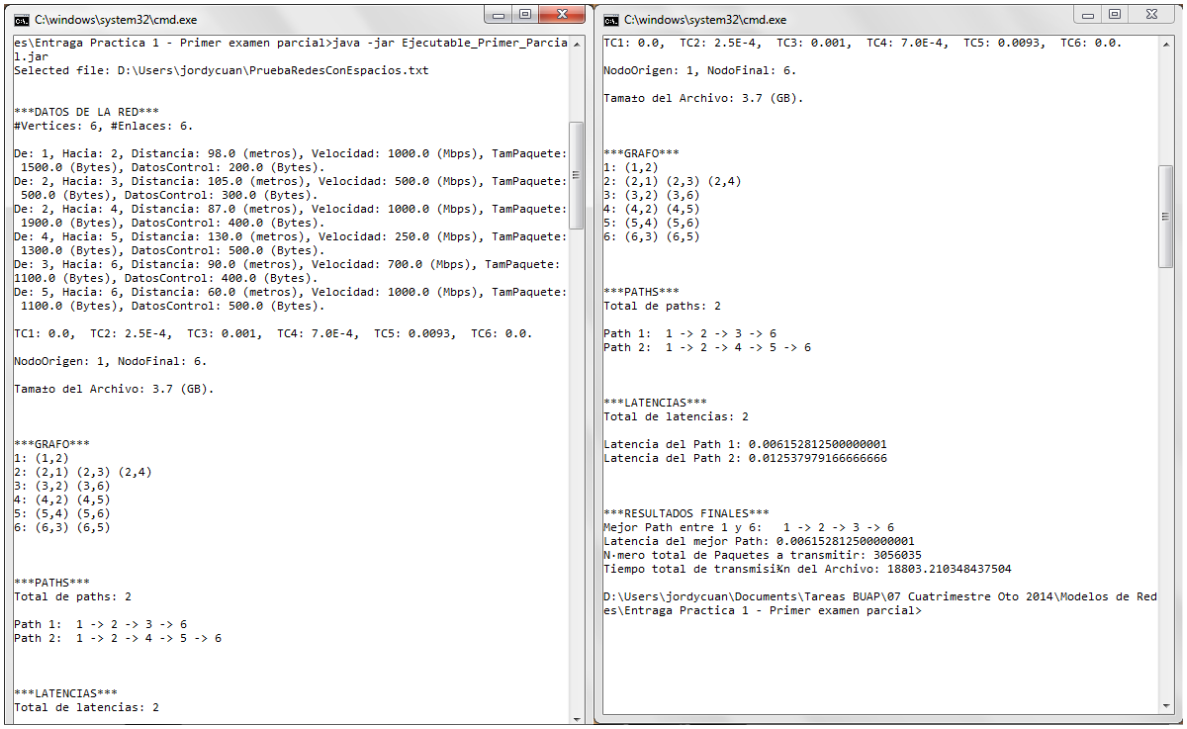


- Adicionalmente, este programa soporta la apertura de ficheros con infinidad de espacios entre datos y separada la información por un renglón, o bien, sien espacios y no habrá ningún problema. Más claro en la siguiente imagen.



Resultados

Finalmente solo puedo mostrar como resultados lo que tengo en consola al finalizar la ejecución de mi programa. No hubo errores.



Conclusiones

Como conclusión principal puedo decir que aprendí completamente a resolver este cálculo de latencias. Ahora, si me pusieran un problema de estos a resolverlo con papel y lápiz, podría hacer los cálculos y resolver el problema con una calculadora, pues más que aprendido este algoritmo no lo puedo tener.

Como otros puntos, también aprendí a diseñar una aplicación más compleja que será muy fácil encontrarme con algo similar en el trabajo en el mundo exterior. Y obvió, repasar mis conocimientos sobre programación.

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

MATERIA: MODELOS DE REDES

PROFESOR: IVÁN OLMOS PINEDA ALUMNO:

RODRIGO ROARO LACK (201035388) REPORTE

DEL 1ER PARCIAL

FECHA DE ENTREGA: 26 DE SEPTIEMBRE DEL 2014

OBJETIVO

Desarrollar un programa que permita calcular el tiempo de latencia, dado un grafo como entrada, de cada uno de sus caminos y obtener el camino con menor latencia y el TTT de un archivo que va a transitar por ese camino.

INTRODUCCIÓN

El parcial consiste en la realización de un programa que trabaja con un grafo, el cual va a ser dado como entrada en un archivo. En la ejecución del programa, este grafo va a ser almacenado en ciertas estructuras de datos y sus datos de igual manera.

Los datos almacenados que se obtuvieron grafo le va a permitir al programa obtener cada uno de los caminos que existen en el mismo, dado un origen y un destino. Además, por cada camino se va a obtener un tiempo de latencia, que se calcula con las fórmulas ya vistas en clase.

El resultado final de este programa es la obtención del camino con el menor tiempo de latencia. Una vez obtenido este camino, se va a calcular el número de paquetes de un cierto archivo que se van a enviar por este camino y el TTT en segundos que tarda este archivo en llegar al destino especificado.

MARCO TEÓRICO

Grafo

Un grafo es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto. Son objeto de estudio de la teoría de grafos.

Típicamente, un grafo se representa gráficamente como un conjunto de puntos (vértices o nodos) unidos por líneas (aristas o arcos).

Prácticamente cualquier problema puede representarse mediante un grafo, y su estudio trasciende a las diversas áreas de las ciencias exactas y las ciencias sociales.

Latencia

Se denomina latencia a la suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red.

Otros factores que influyen en la latencia de una red son:

- El tamaño de los paquetes transmitidos.
- El tamaño de los buffers dentro de los equipos de conectividad. Ellos pueden producir un Retardo Medio de Encolado.

Búsqueda en profundidad

Una Búsqueda en profundidad (en inglés DFS o Depth First Search) es un algoritmo que permite recorrer todos los nodos de un grafo o árbol (teoría de grafos) de manera ordenada, pero no uniforme. Su funcionamiento consiste en ir expandiendo todos y cada uno de los nodos que va localizando, de forma recurrente, en un camino concreto. Cuando ya no quedan más nodos que visitar en dicho camino, regresa (Backtracking), de modo que repite el mismo proceso con cada uno de los hermanos del nodo ya procesado.

DESARROLLO

Para el desarrollo de este programa, opté por realizarlo primero en modo consola mediante JCreator y cuando comprobé su funcionamiento realicé el programa en una interfaz gráfica mediante NetBeans.

El programa fue desarrollado en el lenguaje de programación Java.

A continuación, se muestra el proceso que seguí para poder llegar al resultado final de este programa:

- Lectura y almacenado del archivo
- Construcción del grafo
- Obtención de cada uno de los caminos del grafo
- Cálculo del tiempo de latencia de cada camino
- Obtención del camino con menor tiempo de latencia, número de paquetes del archivo y cálculo del TTT
- Implementación gráfica

Lectura y almacén del archivo

Para poder hacer funcionar el programa, se requiere obtener un grafo mediante un archivo de entrada, el cual cuenta con la siguiente estructura que fue especificada por el profesor:

1. #Vértices, #Arcos (#V, #E)
2. #Vértice inicial, #Vértice final, distancia (metros), velocidad (Mbps), tamaño del paquete (bytes), DC (bytes) (Cada una de las líneas siguientes va a poseer el mismo formato, el número de líneas con este formato está dado por el #Arcos).
3. TC_1, TC_2, TC_3, …, $TC_{\#V}$ (cada uno en segundos)
4. VO, VD (origen – destino)
5. Tamaño del archivo (GB)

El archivo se lee en el método leerArchivo de la clase Lectura.java y a la vez se almacena cada uno de los datos en una estructura de datos de tipo Vector. Opté

por almacenar las líneas en vectores para poder crearlos de manera dinámica y pudieran funcionar para grafos de distintos tamaños y datos.

Como cada una de las clases de este programa está relacionadas entre sí, se pueden acceder a los datos necesarios de las clases si lo requiere una clase específica. (Algunas variables no son accesibles desde otra clase porque son internas de las clases y no requieren ser referenciadas por las demás clases del programa).

A continuación se muestra el método donde se lee el archivo y la referencia a un método donde se almacena la primera línea del archivo:

```
public void leerArchivo(JTextArea texto) throws IOException{
    String str;
    StringTokenizer tokens;
    int c = 0;

    entrada = new BufferedReader(new FileReader(f));

    //# de nodos y arcos
    str = entrada.readLine();
    texto.append(str + "\n");
    tokens = new StringTokenizer(str, "\\s \\,");
    almacenarGrafo(tokens);
    c += 1;
```

Figura 1. Método leerArchivo de la clase Lectura.java

El método almacenarGrafo almacena la primera línea del archivo. Cada uno de los métodos que almacenan las líneas del archivo están hechos de forma similar a este método. A continuación se muestra el contenido de este método:

```
public void almacenarGrafo(StringTokenizer tokens){
    while (tokens.hasMoreElements()){
        String temp = tokens.nextToken();
        grafo.add(Integer.parseInt(temp));
    }
}
```

El método leerArchivo sigue ejecutándose hasta que lea por completo el contenido del archivo y de igual manera se almacena cada una de las líneas en los Vectores.

Figura 2. Método almacenarGrafo de la clase Lectura.java

Construcción del grafo

El grafo se crea mediante la clase Grafo.java. De esta forma, se pueden crear los nodos y los arcos del grafo dado el archivo de entrada.

El grafo se almacena en una estructura de tipo Mapa Hash, el cual permite acceder a cada uno de sus elementos mediante una referencia que sirve de llave. Opté por

El método leerArchivo sigue ejecutándose hasta que lea por completo el contenido del archivo y de igual manera se almacena cada una de las líneas en los Vectores.

Construcción del grafo

El grafo se crea mediante la clase Grafo.java. De esta forma, se pueden crear los nodos y los arcos del grafo dado el archivo de entrada.

El grafo se almacena en una estructura de tipo Mapa Hash, el cual permite acceder a cada uno de sus elementos mediante una referencia que sirve de llave. Opté por

realizarlo de esta forma porque resultaba sencillo hacer referencia a los Nodos mediante el uso de llaves y de igual manera almacenan datos de forma dinámica. También, los arcos del grafo se almacenan en un Mapa Hash para poder acceder a sus elementos mediante llaves (Un ejemplo de una llave para acceder a un arco es "1<-->2").

A continuación se muestra el método que almacena un arco del grafo (El Mapa Hash del grafo posee un conjunto de objetos de tipo Nodo, el cual se define en la clase Nodo.java. Esta clase almacena los vértices del grafo. Además, dentro del Mapa Hash de los arcos se almacena un objeto de tipo Arco, el cual se define en la clase Arco.java. Esta clase almacena los datos obtenidos del archivo correspondientes a un arco específico):

```
//Añade un arco
public void añadirArco(Nodo v1, Nodo v2, Arco e){
    if(!grafo.containsKey(v1) || !grafo.containsKey(v2)) //Si no existe alguno de los dos vértices
        throw new NoSuchElementException("No existen los vertices en el grafo");
    //Añade en amabas direcciones
    grafo.get(v1).add(v2);
    grafo.get(v2).add(v1);
    arcos.put(v1.getName() + "<-->" + v2.getName(), e);
}
```

Figura 3. Método añadirArco de la clase Grafo.java

A continuación se muestra el método que imprime el grafo (El grafo se imprime mostrando los vértices y los nodos adyacentes a cada vértice):

Figura 3. Método añadirArco de la clase Grafo.java

A continuación se muestra el método que imprime el grafo (El grafo se imprime mostrando los vértices y los nodos adyacentes a cada vértice):

```java
//Imprimir el grafo
@Override
public String toString(){
    String s = "";
    for (Nodo v: grafo.keySet()){
        s += v.getName() + ": ";
        for (Nodo tmp: grafo.get(v)) {
            s += tmp.getName() + " ";
        }
        s += "\n";
    }
    return s;
}
```

Figura 4. Método que imprime el grafo dentro de la clase Grafo.java

A continuación, se muestran los atributos de las clases donde se almacenan los Nodos y los Arcos del grafo (a la clase Arco.java le asigne un atributo llamado DU que almacena los datos de usuario del paquete contenido en ese arco, de esta forma puede acceder a este dato de forma sencilla en el programa):

```
public class Nodo implements Comparable<Nodo>{
    public String nombre;
    public int distancia;
    public Nodo pre;
```

Figura 5. Atributos de la clase Nodo.java

Obtención de los caminos del grafo

Para poder realizar esta operación, tuve que realizar una investigación acerca de los algoritmos que se utilizan para poder recorrer un grafo y obtener cada uno de los caminos en el grafo. Los resultados de mi investigación arrojaron que el algoritmo DFS (Depth-First Search o Búsqueda en profundidad) era el más óptimo para esta tarea debido a que realiza el recorrido del grafo mediante un camino y mediante el backtracking se puede regresar a cierto nodo para volver a buscar otro caminos. Esto lo realiza la clase DFS.java.

```
public class Arco {
    public Double distancia;
    public int velocidad;
    public int tam;
    public int DC;
    public int DU;
```

Figura 6. Atributos de la clase Arco.java

El algoritmo lo modifiqué de tal manera que recorriera el grafo dado un vértice de origen y un vértice destino y al principio inserta el nodo inicial en una pila y en un conjunto. El conjunto sirve para obtener los nodos que son recorridos en el camino. El algoritmo realiza el backtracking cada que ya no existan más nodos en el camino y vuelve a generar otro camino. Cuando finaliza el recorrido del camino, saca los

Obtención de los caminos del grafo

Para poder realizar esta operación, tuve que realizar una investigación acerca de los algoritmos que se utilizan para poder recorrer un grafo y obtener cada uno de los caminos en el grafo. Los resultados de mi investigación arrojaron que el algoritmo DFS (Depth-First Search o búsqueda en profundidad) era el más óptimo para esta tarea debido a que realiza el recorrido del grafo mediante un camino y mediante el backtracking se puede regresar a cierto nodo para volver a buscar otro caminos. Esto lo realiza la clase DFS.java.

El algoritmo lo modifiqué de tal manera que recorriera el grafo dado un vértice de origen y un vértice destino y al principio inserta el nodo inicial en una pila y en un conjunto. El conjunto sirve para obtener los nodos que son recorridos en el camino. El algoritmo realiza el backtracking cada que ya no existan más nodos en el camino y vuelve a generar otro camino. Cuando finaliza el recorrido del camino, saca los

elementos de la pila hasta encontrar alguna que permita realizar el recorrido de otro camino. El algoritmo funciona de manera recursiva para que fuera optimizada la obtención de los caminos del grafo y para cada vez que se llame el método mande el vértice adyacente como si fuera el vértice inicial (esto se utiliza para la pila y el conjunto).

A continuación se muestra el código del algoritmo:

A continuación se muestra el código del algoritmo:

```
//Algoritmo DFS modicado para encontrar todos los caminos
public void caminos(Grafo G, Nodo v, Nodo f){
    caminoActual.push(v.getName()); //Añade el nodo actual (v) al camino
    camino.add(v.getName());
    ((Vector) c.get(i)).add(v.getName());

    if (v.getName().equals(f.getName())){
        //System.out.println(camino); //Camino encontrado desde el nodo inicial hasta el final
        c.add(new Vector());
        Enumeration e = ((Vector) c.get(i)).elements();
        i++;
        while (e.hasMoreElements()){
            ((Vector) c.get(i)).add(e.nextElement());
        }
    }
    else{
        for (Nodo aux: G.arcosNodo(v)){ //Obtiene los nodos adyacentes al nodo actual
            if (!camino.contains(aux.getName())){ //Si no está el nodo, hace la llamada recursiva
                caminos(G, aux, f);
            }
        }
    }

    caminoActual.pop(); //Se quita el nodo actual del camino al acabar de explorarlo
    camino.remove(v.getName());
    ((Vector) c.get(i)).remove(v.getName());
}
```

Figura 7. Algoritmo utilizado para obtener los caminos del grafo, modificado a partir del DFS original, dentro de la clase DFS.java

Calculo del tiempo de latencia de cada camino

Una vez que el algoritmo anterior haya realizado la búsqueda de cada uno de los caminos del grafo, se procedió a mandar cada uno de estos caminos a la clase Latencia.java. Esta clase permite calcular cada uno de los parámetros de la fórmula general de latencia (latencia = TP + TT + TC) y al final se obtienen cada uno de los datos de esta fórmula. Esto se realiza en la clase Latencia.java.

Figura 7. Algoritmo utilizado para obtener los caminos del grafo, modificado a partir del DFS

original, dentro de la clase DFS.java

Calculo del tiempo de latencia de cada camino

Una vez que el algoritmo anterior haya realizado la búsqueda de cada uno de los caminos del grafo, se procedió a mandar cada uno de estos caminos a la clase Latencia.java. Esta clase permite calcular cada uno de los parámetros de la fórmula general de latencia (latencia = TP + TT + TC) y al final se obtienen cada uno de los datos de esta fórmula. Esto se realiza en la clase Latencia.java.

En esta clase, existen los métodos que permiten obtener el tiempo de propagación, el tiempo de transmisión y los tiempos de cola adecuados. De igual manera existe un método que se encarga de realizar la división de paquetes, la cual es necesaria para asignar un parámetro de la fórmula de latencia.

A continuación se muestra el método encargado de realizar la división de los paquetes:

```
public int divPaquetes(String e, String pre, int du, int band){
    int n = 0;
    int tam;
    paq.put(e, new ArrayList());

    if (band == 0){ //Para el primer paquete del camino
        paq.get(e).add(du);
        n = 1;
    }
    else{ //Considera los demás paquetes del camino
        for (int tmp: paq.get(pre)) { //Recorre el arco anterior
            if (tmp > du){ //Si el paquete del arco anterior es mayor al del actual
                int ndu = tmp;
                n += ndu / du;
                tam = ndu / du;
                for (int i = 0; i < tam; i++)
                    paq.get(e).add(du);
                ndu = ndu % du;
                paq.get(e).add(ndu);
                n++;
            }
            else{ //En caso contrario
                if (paq.get(e).contains(du))
                    paq.get(e).add(tmp);
                else
                    paq.get(e).add(du);
                n++;
            }
        }
    }
```

Figura 8. Método que se encarga de realizar la división de paquetes entre cada salto encontrado dentro del camino, de la clase Latencia.java

A continuación, se muestra el método que realiza el cálculo de la latencia del camino que se haya asignado:

Figura 8. Método que se encarga de realizar la división de paquetes entre cada salto encontrado dentro del camino, de la clase Latencia.java

A continuación, se muestra el método que realiza el cálculo de la latencia del camino que se haya asignado:

```java
public void calculoLatencia(){
    //latencia = Tiempo de propagación + Tiempo de transmisión + Tiempo de cola
    //latencia = TP + TT + TC;
    //La latencia es una fórmula adaptativa. Depende del número de arcos que existan por camino.
    //En cada salto, se pueden dar divisiones del paquete que indican los números de las constantes de la fórmula
    //La división del paquete esta asociada a la capacidad de datos de usuario del arco que le sigue
    int num;
    Arco tmp;
    String aux = "";

    for (int i = 0; i < cam.size() - 1; i++){
        String str = cam.elementAt(i) + "<-->" + cam.elementAt(i + 1);
        if (arc.get(str) == null){
            String str2 = cam.elementAt(i + 1) + "<-->" + cam.elementAt(i);
            tmp = arc.get(str2);
        }
        else
            tmp = arc.get(str);
        calculoTP(str, tmp.getDistancia());
        calculoTT(str, tmp.getTam(), tmp.getVelocidad());
        tiemposCola(str);
        if (i == 0)
            num = divPaquetes(str, aux, tmp.getDU(), 0);
        else
            num = divPaquetes(str, aux, tmp.getDU(), 1);
        lat += (num * (TP.get(str) + TT.get(str))) + (num * TC.get(str));
        num++;
        aux = str;
    }
}
```

Figura 9. Método encargado de realizar el cálculo de la latencia del camino asignado dentro de la clase Latencia.java

Obtención de los resultados finales

Los resultados finales de este programa son obtener el camino con el menor tiempo de latencia, el número de paquetes del archivo que va a transitar por el camino y el TTT en segundo.

Para que se obtengan estos datos, el programa tiene que haber realizado todo el procedimiento anteriormente especificado. Una vez obtenidos los caminos y el tiempo de latencia de cada uno de ellos, se procede a realizar este último paso.

Figura 9. Método encargado de realizar el cálculo de la latencia del camino asignado dentro de la clase Latencia.java

Esto se realiza dentro de la clase principal del programa, llamada GUI.java, que de igual manera ejecuta el programa de forma gráfica.

A continuación se muestra el método que procesa los resultados finales esperados:

Obtención de los resultados finales

Los resultados finales de este programa son obtener el camino con el menor tiempo de latencia, el número de paquetes del archivo que va a transitar por el camino y el

TTT en segundo.

Para que se obtengan estos datos, el programa tiene que haber realizado todo el procedimiento anteriormente especificado. Una vez obtenidos los caminos y el tiempo de latencia de cada uno de ellos, se procede a realizar este último paso.

Esto se realiza dentro de la clase principal del programa, llamada GUI.java, que de igual manera ejecuta el programa de forma gráfica.

A continuación se muestra el método que procesa los resultados finales esperados:

```
public void resultadoFinal(){
    double min = Double.MAX_VALUE;
    int pos = 0;
    camino = null;

    for (int i = 0; i < busqueda.c.size() - 1; i++){
        if ((double) latencia.elementAt(i) < min){
            min = (double) latencia.elementAt(i);
            pos = i;
            camino = (Vector) busqueda.c.get(i);
        }
    }

    //Número de paquetes y TTT
    String v1 = camino.firstElement().toString();
    String v2 = camino.elementAt(1).toString();
    String str = v1 + "<-->" + v2;
    if (G.arcos.get(str) == null){
        String str2 = v2 + "<-->" + v1;
        e = G.arcos.get(str2);
    }
    else
        e = G.arcos.get(str);

    double archivo = L.tam * 1024 * 1024 * 1024;
    int du = e.getDU();
    int np = (int) Math.ceil(archivo / du);
    int ttt = (int) (np * min);
```

Figura 10. Método encargado de presentar los resultados finales y requeridos del programa, de la clase GUI.java

Implementación gráfica

Como algo extra al programa, realicé la implementación gráfica del mismo para que tuviera una mejor presentación y mostrara ciertas características de los datos que se obtienen del grafo. Esto se realiza en la clase GUI.java

Los métodos que se encargan de realizar todo el procedimiento anteriormente especificado están contenidos dentro de esta clase. Estos métodos se encargan de hacer las diferentes llamadas y procesos que son requeridos para realizar la ejecución completa del programa.

A continuación se muestra el método que se encarga de realizar la ejecución de todo el procedimiento anterior. Este método realiza la referencia a los demás métodos de la clase una vez que se haya abierto el archivo a ser procesado:

Figura 10. Método encargado de presentar los resultados finales y requeridos del programa, de la clase GUI.java

Implementación gráfica

Como algo extra al programa, realicé la implementación gráfica del mismo para que tuviera una mejor presentación y mostrara ciertas características de los datos que se obtienen del grafo. Esto se realiza en la clase GUI.java

Los métodos que se encargan de realizar todo el procedimiento anteriormente especificado están contenidos dentro de esta clase. Estos métodos se encargan de hacer las diferentes llamadas y procesos que son requeridos para realizar la ejecución completa del programa.

A continuación se muestra el método que se encarga de realizar la ejecución de todo el procedimiento anterior. Este método realiza la referencia a los demás métodos de la clase una vez que se haya abierto el archivo a ser procesado:

```
public void abrir() throws IOException{
    JFileChooser open = new JFileChooser();
    open.setCurrentDirectory(new File(System.getProperty("user.home")));
    FileNameExtensionFilter filter = new FileNameExtensionFilter("Archivos de texto (.txt)", "txt");
    open.setFileFilter(filter);
    int res = open.showOpenDialog(this);
    if (res == JFileChooser.APPROVE_OPTION){
        File fichero = open.getSelectedFile();
        String archivo = fichero.getName();
        String ext = archivo.substring(archivo.lastIndexOf("."),archivo.length());
        init();
        if (ext.equals(".txt")){
            L = new Lectura(fichero);
            L.leerArchivo(areaArchivo);
            G = new Grafo((int)L.grafo.firstElement(), (int)L.grafo.lastElement());
            crearNodos();
            crearArcos();
            obtenerArcos();
            imprimirGrafo();
            obtenerCaminos();
            calcularLatencia();
            resultadoFinal();
        }
        else
            JOptionPane.showMessageDialog(null,"La extensión del archivo no es la correcta","Error de validacion",JOptionPan
    }
}
```

Figura 11. Método encargado de realizar todo el proceso requerido en el programa de la clase GUI.java

A continuación se muestran unos métodos que manejan una tabla que se presenta en la ejecución del programa. Dicha tabla presenta cada uno de los caminos del grafo y sus tiempos de latencia. En el programa se pueden ver los datos que se obtuvieron en el cálculo de los tiempos de latencia del camino, mediante la selección de una fila de la misma. Esta selección activa un desplegable que permite seleccionar un arco del camino para que se muestren los resultados de dicho arco:

Figura 11. Método encargado de realizar todo el proceso requerido en el programa de la clase GUI.java

A continuación se muestran unos métodos que manejan una tabla que se presenta en la ejecución del programa. Dicha tabla presenta cada uno de los caminos del grafo y sus tiempos de latencia. En el programa se pueden ver los datos que se obtuvieron en el cálculo de los tiempos de latencia del camino, mediante la selección de una fila de la misma. Esta selección activa un desplegable que permite seleccionar un arco del camino para que se muestren los resultados de dicho arco:

```
public void llenarTabla(int pos){
    Object nuevo[] = {temp.getRowCount() + 1, "", ""};
    temp.addRow(nuevo);
    tabla.setValueAt(camino, pos, 0);
    tabla.setValueAt(latencia.elementAt(pos) + " seg", pos, 1);
}

public void generarLista(int pos){
    Latencia tmp;
```

Figura 12. Métodos que manejan a la tabla y al desplegable que se presentan en la interfaz del programa, dentro de la clase GUI.java

RESULTADOS

Para poder comprobar que el programa funcionara de la manera correcta, se hicieron pruebas con 3 diferentes grafos, los cuales se incluyen sus archivos en una carpeta del proyecto del programa. Al final, se ejecutaron cada uno de estos archivos en el programa de forma exitosa y se obtuvieron los resultados de cada uno de ellos.

A continuación se muestra la interfaz del programa de este 1er parcial:

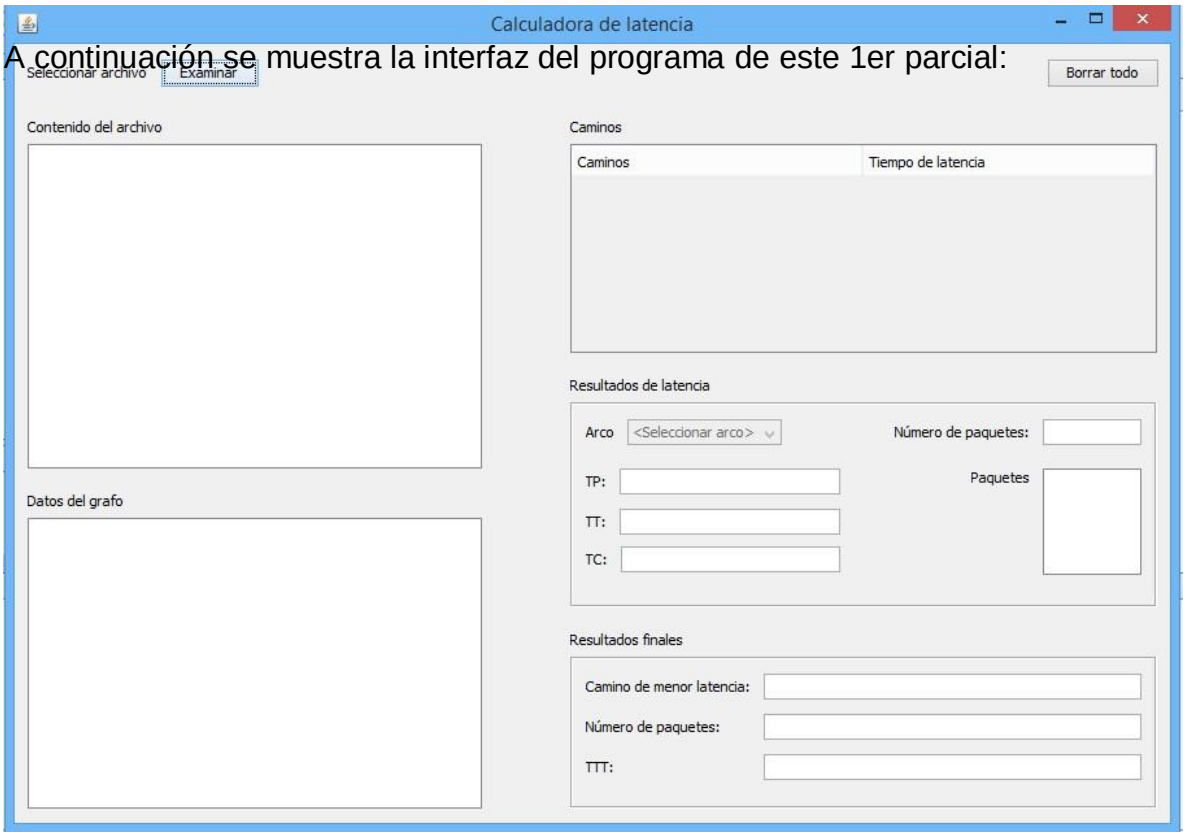


Figura 12. Métodos que manejan a la tabla y al desplegable que se presentan en la interfaz del programa, dentro de la clase GUI.java

Figura 13. Interfaz del programa del 1er parcial

A continuación se muestra la ejecución de este programa con un archivo dado y los resultados obtenidos dentro del mismo:

```
8,12
1,2,90,100,1500,100
1,3,95,1000,1500,100
2,4,80,100,1400,200
2,5,82,100,1500,100
3,4,82,1000,1400,200
3,5,80,100,1300,200
4,6,90,100,1000,500
4,7,90,1000,1000,500
5,6,90,100,1000,500
5,7,90,100,1000,500
6,8,70,100,1000,500
7,8,70,100,1000,500
0,0.000025,0.000025,0.000025,0.000025,0.000025,0.000025,0
1,8
4.7
```

Figura 14. Archivo proporcionado para la ejecución del programa

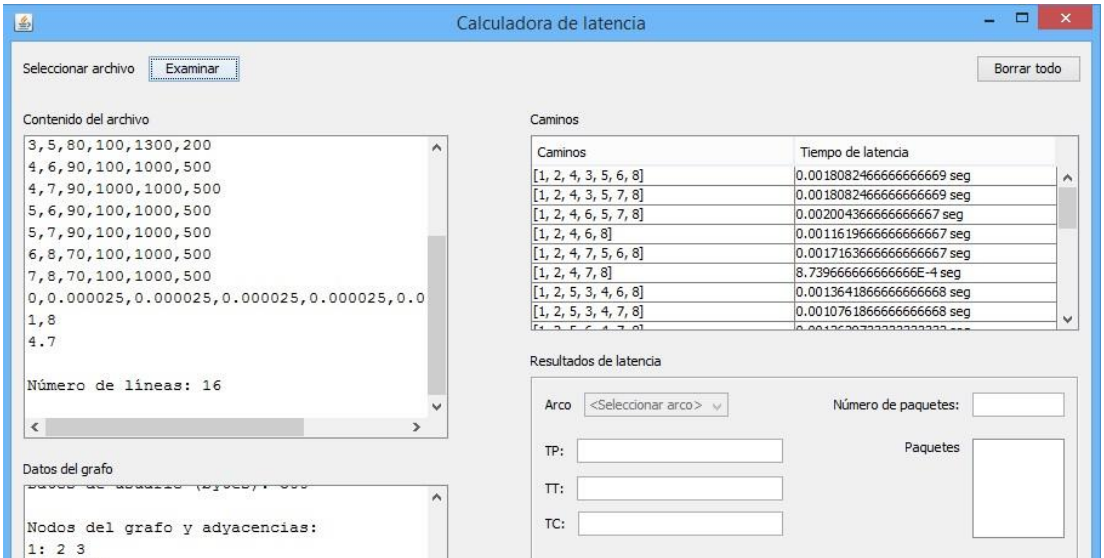


Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

A continuación se muestra la parte de la interfaz donde se pueden consultar los datos obtenidos de cada uno de los arcos de un camino, después de haber calculado la latencia de dicho camino:

Figura 15 cionado y

Figura 16. Sección del programa donde se pueden consultar datos respecto a los cálculos de latencia a través de cada uno de los arcos de los caminos

CONCLUSIONES

La realización de este programa me permitió practicar con la programación orientada a objetos en el lenguaje Java, de manera que pudiera resolver el problema propuesto por el profesor. Mediante el uso de este programa, se pueden obtener los resultados del tiempo de latencia de cada uno de los caminos de un cierto grafo, y con ello obtener el camino con el menor tiempo de latencia, el número de paquetes

necesidad de estarlos realizando de forma escrita y con posibilidad de errores.

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Modelos de Redes
Primer Parcial:
Latencia

Mellado Robles Davis
200827022

1. Antecedente

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Latencia

En redes informáticas de datos se denomina latencia a la suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red.

Depende de tres factores:

1. Tiempo de propagación del bit por el medio, que depende del tiempo de propagación de la corriente o luz por el medio, además de la distancia recorrida
2. Máxima cantidad de datos que pueden ser transmitidos por la red sin segmentarse. Al tiempo de propagación del bit en un paquete se le llama tiempo de transmisión
3. Tiempos de espera para difundirse un paquete a través de un conmutador, además del tráfico de la red. A este tiempo se le denomina tiempo de cola

Ecuaciones relacionadas a la latencia.

$$Latencia = Tiempo\ de\ propagacion(TP) + Tiempo\ de\ transmisión(TT) + Tiempo\ de\ cola(TC)$$

$$TP = \frac{Distancia\ a\ recorrer}{Velocidad\ de\ la\ luz}$$

$$TT = \frac{Tamaña\ del\ paquete}{Tasa\ de\ trasnferencia\ teorica}$$

Ejemplo 1:

Considere la siguiente red, calcular latencia y tiempo en trasferir 3.5 GBytes de E al nodo Rec

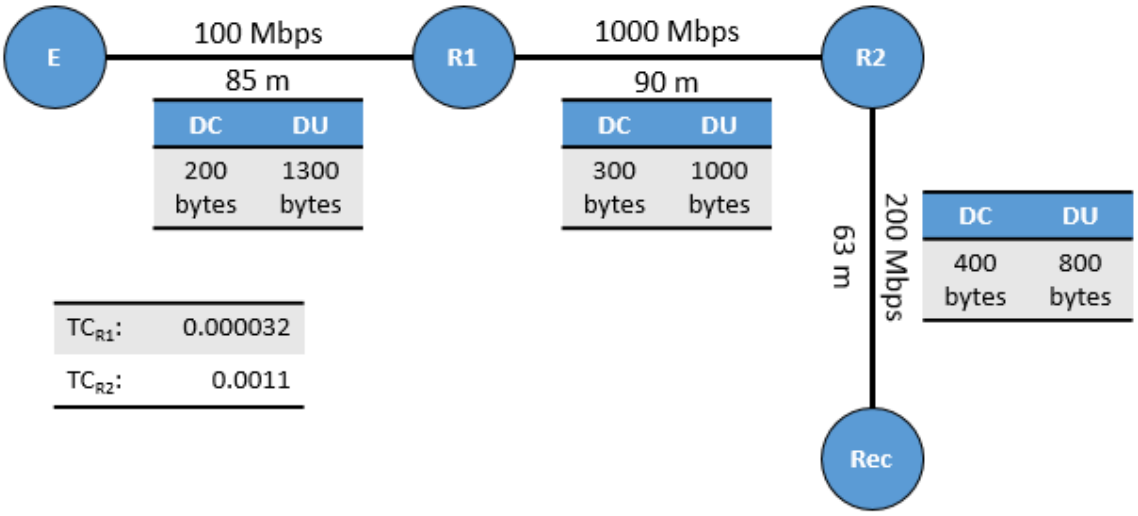


Figura 1: Red de nodos ejemplo 1

Solución:

$$Lat_{E-Rec} = TP + TT + TC$$

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

$$Lat_{E-Rec} = (TP_{E-R1} + TT_{E-R1}) + TC_{R1} + 2 * (TP_{R1-R2} + TT_{R1-R2}) + 2 * TC_{R2} + 3 * (TP_{R2-Rec} + TT_{R2-Rec})$$

$TP_{E-R1} = \frac{85\ m}{300\ 000\ kms} =$	$\mathbf{2.83333333333333 \cdot 10^{-7}}$
$TT_{E-R1} = \frac{1500\ bytes}{100\ Mbps} =$	$\mathbf{1.4305114746094 \cdot 10^{-5}}$
$TP_{R1-R2} = \frac{90\ m}{300\ 000\ kms} =$	$\mathbf{3 \cdot 10^{-7}}$
$TT_{R1-R2}\ \frac{1300\ bytes}{1000\ Mbps} =$	$\mathbf{1.2397766113281 \cdot 10^{-6}}$
$TP_{R2-Rec} = \frac{63\ m}{300\ 000\ kms}$	$\mathbf{2.1 \cdot 10^{-7}}$
$TT_{R2-Rec}\ \frac{1200\ bytes}{200\ Mbps} =$	$\mathbf{5.7220458984375 \cdot 10^{-6}}$

$$Lat_{E-Rec} = \mathbf{2.83333333333333 \cdot 10^{-7} + 1.4305114746094 \cdot 10^{-5} + 0.000032} + \mathbf{2\ (3 \cdot 10^{-7} + 1.2397766113281 \cdot 10^{-6}) + 2 \cdot (0.0011)} + \mathbf{3\ (2.1 \cdot 10^{-7} + 5.7220458984375 \cdot 10^{-6})}$$

$$Lat_{E-Rec} = \mathbf{4.65888448079427 \cdot 10^{-5} + 0.0022030795532 + 1.7796137695312 \cdot 10^{-5}}$$

$$\boldsymbol{Lat_{E-Rec} = \quad 0.002267464139}$$

$$TTTransf\ = \#\ Paq * Latencia$$

$$\#Paq = \frac{TamArch}{DU\ Paq_I}$$

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

$$\#Paq = \frac{3.5 GBytes}{1300\ Bytes}$$

$$\#Paq = \frac{3758096384}{1300}$$

$$\boldsymbol{\#Paq = 2890843.3723076922}$$

$$TTTransf = 2890843.3723076922 * 0.002267464139$$

$$TTTransf = 6554.8836781735181\ Segundos$$

$$\boldsymbol{TTTransf = 109.248061302892\ Minutos}$$

2. Desarrollo

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Desarrollar un programa que lea de un archivo una red de nodos de la siguiente manera:

- # 1: No. vértices, No. arcos
- # 2: Vi, Vf, distancia(mts), Velocidad(Mbps), TP, DC(bytes)
- # ...
- # ...
- # ...
- # 3: TC1, TC2,...,TCnv(seg)
- # 4: V-origen, V-destino
- # 5: Tamaño de archivo (GB)

El programa deberá calcular los caminos posibles, determinar la ruta con menor latencia y calcular el tiempo que tardaría en trasferir un archivo del tamaño indicado en el archivo del origen al destino.

Ejemplo de archivo de nodos:

```
3,3
E,   R1,  85,  100, 1500,200
R1,  R2,  90,  1000,1300,300
R2,  Rec, 63,  200, 1200,400
0.000032, 0.0011
E,Rec
3.5
```

Línea 1: Una red de 3 vértices con 3 aristas.

Línea 2: Nodo E conectado a Nodo R1 con una distancia de 85 metros, 100 Mbps de velocidad, Tamaño de paquete de 1500 y datos de control de 200.

Línea 3: Nodo R1 conectado a Nodo R2 con una distancia de 90 metros, 1000 Mbps de velocidad, Tamaño de paquete de 1300 y datos de control de 300.

Línea 4: Nodo R2 conectado a Nodo Rec con una distancia de 63 metros, 200 Mbps de velocidad, Tamaño de paquete de 1200 y datos de control de 400.

Línea 5: Tiempos de Cola de 0.000032 para R1 y 0.0011 para R2

Línea 6: Nodo origen y nodo destino (E,Rec).

Línea 7: Tamaño del archivo a trasferir de 3.5 GBytes

Algoritmo empleado para calcular los caminos posibles de una red de grafos:

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Algoritmo en pseudocódigo:

```
1  método DFS( origen,final):
2      marcamos origen como visitado
3      recuperar el path si se llego a final
4          para cada vertice v adyacente a origen en el Grafo:
5              si v no ah sido visitado:
6                 marcamos como visitado v
7                 llamamos recursivamente DFS( v )
8      marcamos origen como no visitado
```

Clases implementadas:

LeeFichero.java	Lee de un fichero la configuración de la red de nodos
Arco.java	Implementación de métodos para manejar las operaciones de un arco entre dos nodos.
Arista.java	Implementación grafica de Arco
Caminos.java	Implementación de un algoritmo para calcular los posibles caminos de una red de nodos a partir de un nodo origen y un nodo destino.
Enlace.java	Implementación de métodos para el control de las aristas desde el nodo donde se originan las aristas.
Grafo.java	Implementación de métodos para el control de la red de grafos.
Lienzo.java	Clase encargada del pintado del grafo en pantalla.
Nodo.java	Implementación de los nodos de la red.
Principal.java	Clase principal donde controla todo el programa, las llamadas a las demás clases y el cálculo de las respuestas
Punto.java	Implementación de nodo gráfico.

3. Resultado

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

La siguiente ejecución se realizó con los siguientes archivos de entrada:

Archivo 1:

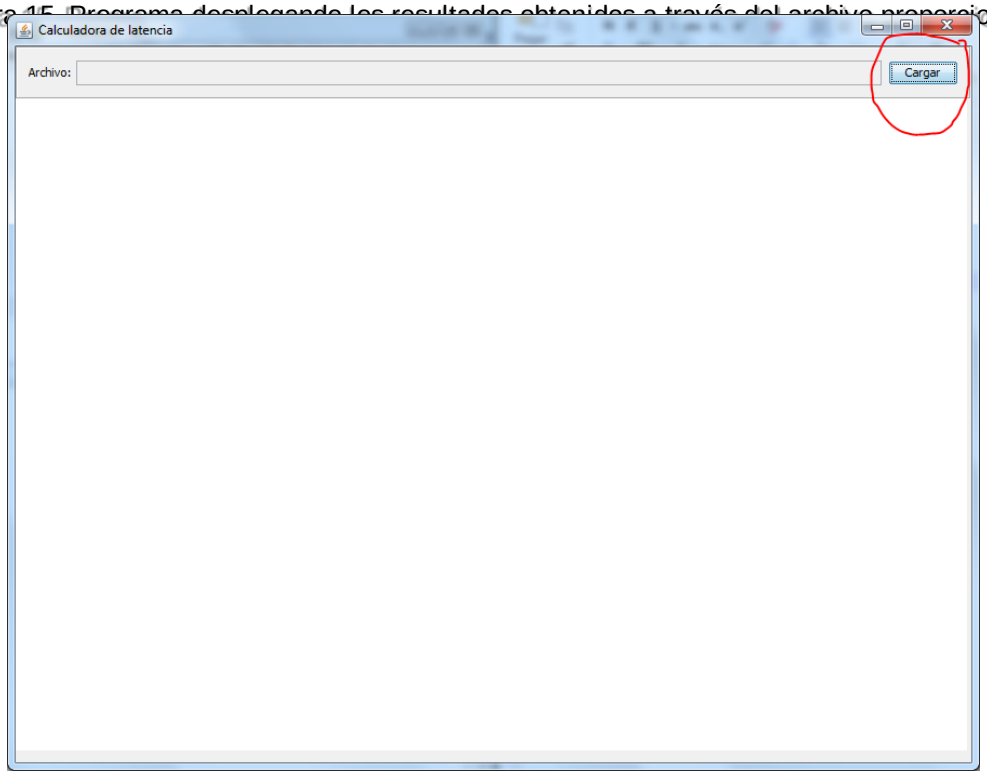
```
# 1: No. vertices, No. arcos
# 2: Vi, Vf, distancia(mts), Velocidad(Mbps), TP, DC(bytes)
# ...
# ...
# ...
# 3: TC1, TC2,...,TCnv(seg)
# 4: V-origen, V-destino
# 5: Tamaño de archivo (GB)

3,3
E,  R1, 85, 100,    1500,   200
R1, R2, 90, 1000,   1300,   300
R2, Rec,    63, 200,    1200,   400
0.000032, 0.0011
E,Rec
3.5
```

Archivo 2:

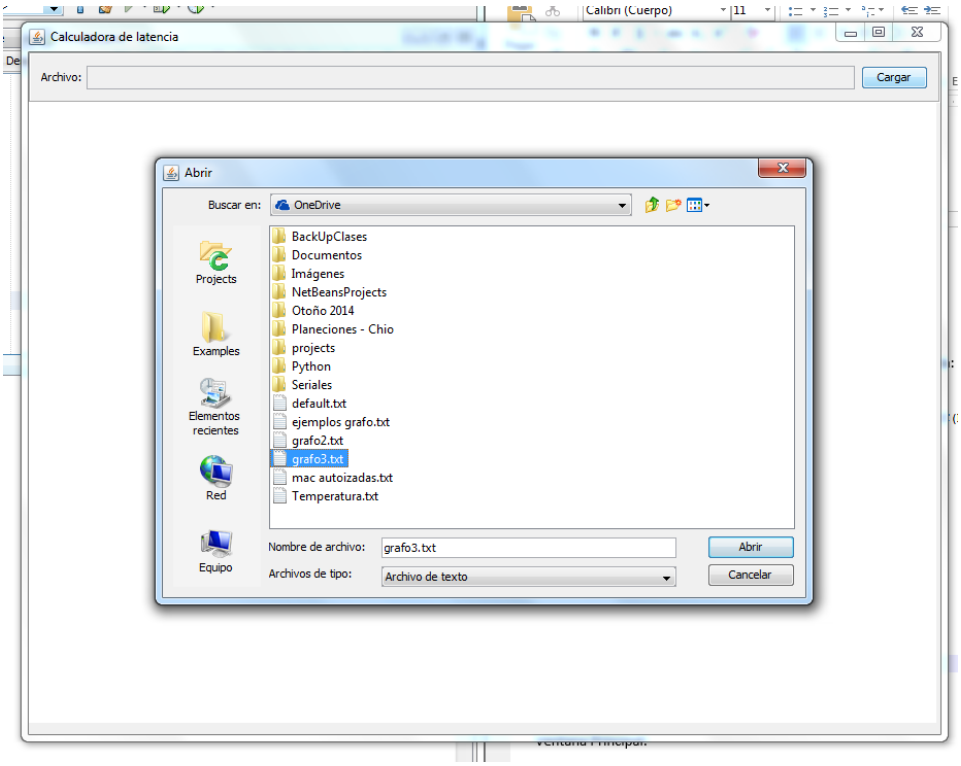
```
# 1: No. vertices, No. arcos
# 2: Vi, Vf, distancia(mts), Velocidad(Mbps), TP, DC(bytes)
# ...
# ...
# ...
# 3: TC1, TC2,...,TCnv(seg)
# 4: V-origen, V-destino
# 5: Tamaño de archivo (GB)

6,6
A,  R1, 98, 1000,   1500,   200
R1, R2, 87, 1000,   1900,   400
R1, R3, 105,500,    1500,   300
R2, R4, 130,250,    1300,   500
R3, B,  90, 700,    1100,   400
R4, B,  60, 1000,   1100,   500
0.00025, 0.001, 0.00070, 0.0093
A,B
4.8
```

Ventana Principal:

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y

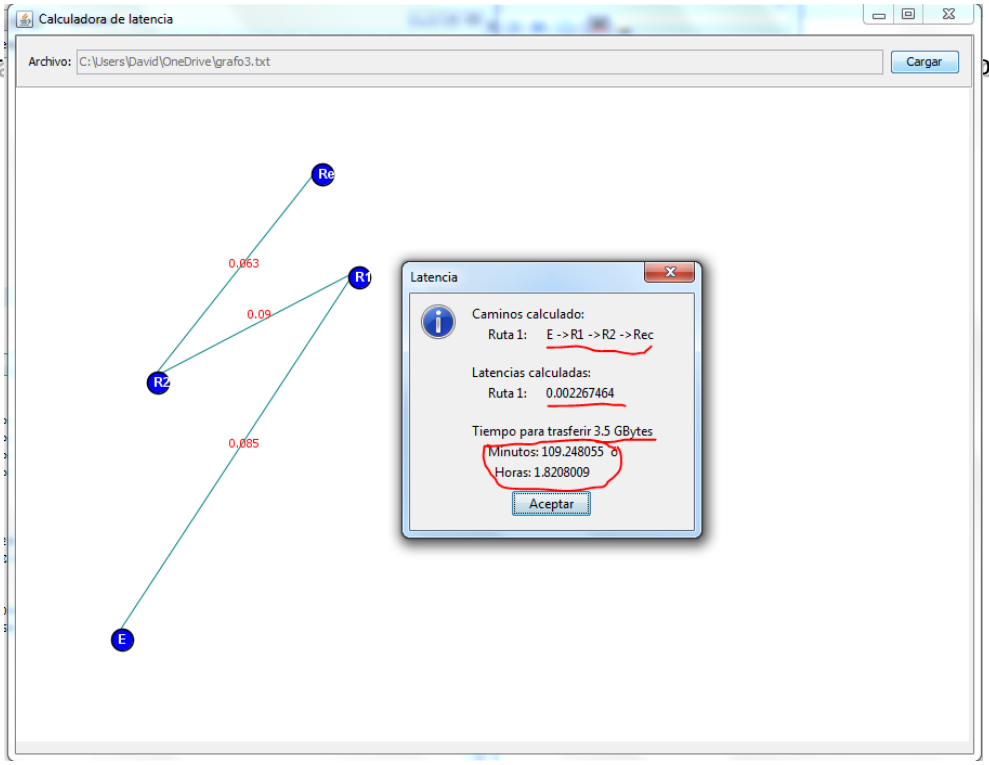


Seleccionando archivo de entrada.

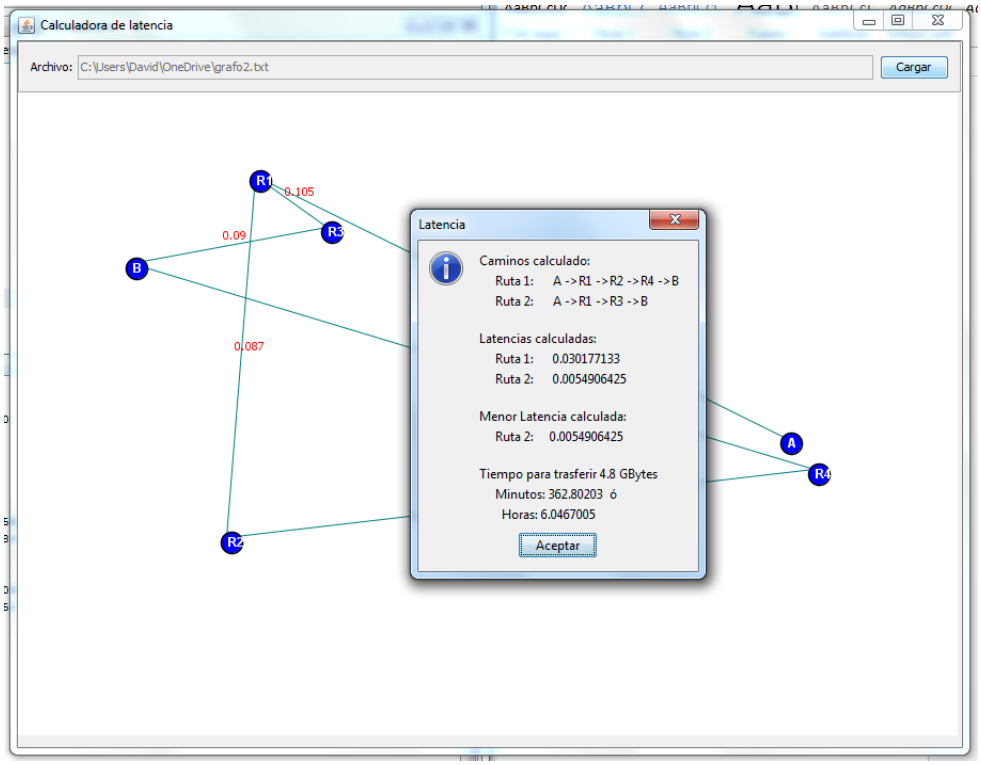


Grafo dibujado y ventana de resultados del archivo 1.

Figur onado y



Grafo dibujado y ventana de resultados del archivo 2.



4. Conclusiones
   - La latencia de una red depende principalmente de tres factores descritos por la formula al inicio de este documento mencionado:

$$Latencia = Tiempo\ de\ propagacion(TP) + Tiempo\ de\ transmisión(TT) + Tiempo\ de\ cola(TC)$$

TP, TT y TC.

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

- La velocidad en una red no es uniforme en todos sus segmentos, cada segmento varía su velocidad según sus características.
- En una red hay varios posibles caminos para un origen y destino.

FACULTAD de CIENCIAS de la COMPUTACIÓN

BENEMÉRITA

NIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de computación.

Ingeniería en ciencias de la computación.

Modelo de Redes.

PROYECTO FINAL: Sistema de validación de acceso a usuarios basado en MD5 para servicio chat.

Equipo:

López López Daniel

Delgadillo Arroyo Diana

Sánchez Ramírez Carlos Alberto

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y

Hernandez Ramírez Walberth

Alvarado Pineda Esperanza

Dr.: Olmos Pineda Ivan.

Fecha: 27-11-14

INTRODUCCION.

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

El chat o también conocido como cibercharla, designa una comunicación escrita realizada de manera instantánea entre dos o más personas. El chat sirve para comunicarse con grupos de personas las cuales opinan de diferentes temas y se entretienen incluso con herramientas como el video chat y enviándose enlaces para ver otras páginas. Un chat permite una interacción fluida mediante texto síncrono.

Es común que los usuarios del chat utilicen seudónimos o alias llamados Nick, el cual determinan cuando realizan su registro para poder tener acceso a una cuenta en ese chat.

Para poder llevar a cabo este registro es necesario llevar a cabo un procedimiento de validación tanto del usuario para poder comprobar si el usuario existe, en caso de que si exista ahora será necesario solicitar su passsword y realizar la validación de esta contraseña.

La seguridad de la información en este caso, del nombre de usuario y la contraseña, es el conjunto de medidas preventivas y reactivas de las organizaciones y de los sistemas tecnológicos que permiten resguardar y proteger la información buscando mantener la confidencialidad, la disponibilidad e integridad de la misma.

OBJETIVO

Nuestro objetivo es implementar un sistema de validación para que los usuarios puedan tener acceso a su cuenta de chat. Realizando el proceso de validación basado en MD5.

MARCO TEORICO.

La seguridad de las redes electrónicas y de los sistemas de información suscita cada vez más preocupación, en paralelo al rápido aumento del número de usuarios y del valor de sus transacciones. La seguridad ha cobrado ahora una importancia crítica, hasta el punto de que constituye un requisito previo para el crecimiento del comercio electrónico y el funcionamiento de la economía en su conjunto.

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos.

En la seguridad de la información es importante señalar que su manejo está basado en la tecnología y debemos de saber que puede ser confidencial. La información está centralizada y puede tener un alto valor. Puede ser divulgada, mal utilizada, ser robada, borrada o saboteada. Esto afecta su disponibilidad y la pone en riesgo. La información es poder, y según las posibilidades estratégicas que ofrece tener acceso a cierta información, esta se clasifica como:

Critica: es indispensable para la operación de la empresa.

Valiosa: es un activo de la empresa y muy valioso.

Sensible: debe de ser conocida por las personas autorizadas.

Existen dos palabras muy importantes que son riesgo y seguridad:

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Riesgo: es la materialización de vulnerabilidades identificadas, asociadas con su probabilidad de ocurrencia, amenazas expuestas, así como el impacto negativo que ocasione a las operaciones de negocio.

Seguridad: es una forma de protección contra los riesgos.

La seguridad de la información comprende diversos aspectos entre ellos la disponibilidad, comunicación, identificación de problemas, análisis de riesgos, la integridad, confidencialidad, recuperación de los riesgos.

Precisamente la reducción o eliminación de riesgos asociado a una cierta información es el objeto de la seguridad de la información y la seguridad informática. Más concretamente, la seguridad de la información tiene como objeto los sistemas el acceso, uso divulgación, interrupción o destrucción no autorizada de información. Los términos seguridad de la información, seguridad informática y garantía de la información son usados frecuentemente como sinónimos porque todos ellos persiguen una misma finalidad al proteger la confidencialidad, integridad y disponibilidad de la información. Sin embargo, no son exactamente lo mismo existiendo algunas diferencias sutiles. Estas diferencias radican principalmente en el enfoque, las metodologías utilizadas, y las zonas de concentración. Además, la seguridad de la información involucra la implementación de estrategias que cubran los procesos en donde la información es el activo primordial. Estas estrategias deben tener como punto primordial el establecimiento de políticas, controles de seguridad, tecnologías y procedimientos para detectar amenazas que puedan explotar vulnerabilidades y que pongan en riesgo dicho activo, es decir, que ayuden a proteger y salvaguardar tanto información como los sistemas que la almacenan y administran.

CONFIDENCIALIDAD.

La confidencialidad es la propiedad que impide la divulgación de información a personas o sistemas no autorizados. A grandes rasgos, asegura el acceso a la información únicamente a aquellas personas que cuenten con la debida autorización. La pérdida de la confidencialidad de la información puede adoptar muchas formas.

INTEGRIDAD.

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Es la propiedad que busca mantener los datos libres de modificaciones no autorizadas. A groso modo, la integración es el mantener con exactitud la información tal cual fue generada, sin ser manipulada o alterada por personas o procesos no autorizados.

La violación de la integridad se presenta cuando un empleado, programa o proceso modifica o borra los datos importantes que son parte de la información, asi mismo hace que su contenido permanezca inalterado a menos que sea modificado por personal autorizado, y esta modificación sea registrada, asegurando su precisión y confiabilidad. La integridad de un mensaje se obtiene adjuntándole otro conjunto de datos de

comprobación de la integridad: la firma digital es uno de los pilares fundamentales de la seguridad de la información.

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

DISPONIBILIDAD.

La disponibilidad es la característica, cualidad o condición de la información de encontrarse a disposición de quienes deben acceder a ella, ya sean personas, procesos o aplicaciones. Groso modo, la disponibilidad es el acceso a la información y a los sistemas por personas autorizadas en el momento que así lo requieran.

La disponibilidad además de ser importante en el proceso de seguridad de la información, es además variada en el sentido de que existen varios mecanismos para cumplir con los niveles de servicio que se requiera. Tales mecanismo se implementan en infraestructura tecnológica, servidores de correo electrónico, de bases de datos, de web, etc. Mediante el uso de clusters o arreglos de discos, equipos en alta disponibilidad a nivel de red, servidores espejo, replicación de datos redes de almacenamiento, enlaces redundantes.

AUTENTICACION O AUTENTIFICACION.

Es la propiedad que permite identificar el generador de la información. Por ejemplo al recibir un mensaje de alguien, estar seguro que es de ese alguien el que lo ha mandado, y no una tercera persona haciéndose pasar por la otra (suplantación de identidad). Esta propiedad se puede considerar como un aspecto de la integridad, si está firmado por alguien, está realmente enviado por el mismo.

SERVICIOS DE SEGURIDAD.

El objetivo de un servicio de seguridad es mejorar la seguridad de los sistemas de procesamiento de datos y la transferencia de información en las organizaciones. Los servicios de seguridad están diseñados para contrarrestar los ataques a la seguridad y hacen uso de uno o más mecanismos de seguridad para proporcionar el servicio.

Los protocolos de seguridad son un conjunto de reglas que gobiernan dentro de la transmisión de datos entre la comunicación de dispositivos para ejercer una confidencialidad, la integridad, autenticación y el no repudio de la información. Se componen de:

Criptografía: (cifrado de datos). Se ocupa de transposicionar u ocultar el mensaje enviado por el emisor hasta que llegará su destino y pueda ser descifrado por el receptor.

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Lógica: (estructura y secuencia). Llevar un orden en el cual se agrupan los datos del mensaje el significado del mensaje y saber cuándo se va a enviar el mensaje.

Identificación: (autenticación). Es una validación de identificación es la técnica mediante la cual un proceso comprueba que el compañero de comunicación es quien se supone que es y no se trate de un impostor.

MD5

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Para nuestra identificación usaremos MD5, que es una técnica que permite obtener una huella digital de una serie de datos de entrada. Con la huella se puede verificar la integridad de los datos, es decir, comprobar que no hayan sufrido alteración. Además, puede emplearse para la validación de usuarios o procesos, aunque no es la más usada, a comparación de otras técnicas como RSA.

Es uno de los algoritmos de reducción criptográficos, proporciona un código asociado a un archivo o un texto concretos. De esta forma, a la hora de descargar un determinado archivo, como puede ser un instalador, el código generado por el algoritmo, viene unido al archivo.

Ha sido utilizado en una amplia variedad de aplicaciones de seguridad, y también se utiliza comúnmente para comprobar la integridad de los datos. MD5 fue diseñado por Ron Rivest en 1991 para reemplazar una función.

MD5 procesa mensajes de cualquier longitud y procesa bloques uniformes de 512 bits a la vez, hasta concluir con el mensaje total a fin de entregar a la salida un bloque "resumen" de 128 bits (longitud fija).

El estándar de cifrado (encriptación) avanzado AES, Advanced Encryption Standard (AES), es uno de los algoritmos más seguros y más utilizados hoy en día - disponible para uso público. Está clasificado por la Agencia de Seguridad Nacional, National Security Agency (NSA), de los Estados Unidos para la seguridad más alta de información secreta "Top Secret". Su historia de éxito comenzó 1997, cuando el Instituto Nacional de Estándares y Tecnología, National Institute of Standards and Technology (NIST), anunció la búsqueda de un sucesor para el estándar de cifrado DES. Un algoritmo llamado "Rijndael", desarrollado por los criptólogos belgas Joan Daemen y Vincent Rijmen, fue destacado en seguridad, así como en el rendimiento y la flexibilidad. Este algoritmo le gano a varios competidores, y fue oficialmente presentado como el nuevo estándar de cifrado AES en el 2001 y se transformó en estándar efectivo en el 2002. El algoritmo se basa en varias sustituciones, permutaciones y transformaciones lineales, ejecutadas en bloques de datos de 16 bytes - por lo que se le llama blockcipher. Estas operaciones se repiten varias veces, llamadas "rondas". En cada ronda, un único "roundkey" se calcula de la clave de encriptación, y es incorporado en los cálculos. Basado en esta estructura de bloque de AES, el cambio de un solo bit, ya sea en la clave, o en los bloques de texto simple y claro, resulta en un bloque de texto cifrado/encriptado completamente diferente - una clara ventaja sobre cifrados de flujo tradicionales. La diferencia entre AES-128, AES-192 y AES-256, es la longitud de la clave: 128,

192 o 256 bits - todos drásticamente mejorados en comparación con la clave DES de 56 bits. A modo de ejemplo: Descifrar una clave de 128 bits AES con una supercomputadora estándar del momento, llevaría más tiempo que la edad del universo. ¡Boxcryptor utiliza incluso claves de 256! Hasta el día de hoy, no existe posible ataque contra AES. Por lo tanto, sigue siendo el estándar AES de cifrado preferido por los gobiernos, los bancos y los sistemas de alta seguridad de todo el mundo.

Figura 15. Programa descifrando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todas las pruebas.

La conexión entre los hilos se manejó con exclusión mutua. Todo en el programa está centralizado.

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Los algoritmos de exclusión mutua se usan en programación concurrente para evitar el uso simultáneo de recursos comunes, como variables globales, por fragmentos de código conocidos como secciones críticas.

La mayor parte de estos recursos son las señales, contadores, colas y otros datos que se emplean en la comunicación entre el código que se ejecuta cuando se da servicio a una interrupción y el código que se ejecuta el resto del tiempo. Se trata de un problema de vital importancia porque, si no se toman las precauciones debidas, una interrupción puede ocurrir entre dos instrucciones cualesquiera del código normal y esto puede provocar graves fallos.

La técnica que se emplea por lo común para conseguir la exclusión mutua es inhabilitar las interrupciones durante el conjunto de instrucciones más pequeño que impedirá la corrupción de la estructura compartida (la sección crítica). Esto impide que el código de la interrupción se ejecute en mitad de la sección crítica.

Concepto de exclusión mutua.

Consiste en que un solo proceso excluye temporalmente a todos los demás para usar un recurso compartido de forma que garantice la integridad del sistema.

Concepto de sección crítica.

Es la parte del programa con un comienzo y un final claramente marcados que generalmente contiene la actualización de una o más variables compartidas.

Para que una solución al problema de la exclusión mutua sea válida, se tienen que cumplir una serie de condiciones:

Hay que garantizar la exclusión mutua entre los diferentes procesos a la hora de acceder al recurso compartido. No puede haber en ningún momento dos procesos dentro de sus respectivas secciones críticas.

No se deben hacer suposiciones en cuanto a la velocidad relativa de los procesos en conflicto.

Ningún proceso que esté fuera de su sección crítica debe interrumpir a otro para el acceso a la sección crítica.

Cuando más de un proceso desee entrar en su sección crítica, se le debe conceder la entrada en un tiempo finito, es decir, que nunca se le tendrá esperando en un bucle que no tenga final.

REQUISITOS PARA LA EXCLUSIÓN MUTUA

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Sólo un proceso, de todos los que poseen secciones críticas por el mismo recurso compartido, debe tener permiso para entrar en ella en un momento dado.

Un proceso que se interrumpe en una sección no crítica debe hacerlo sin interferir con los otros procesos.

Un proceso no debe poder solicitar acceso a una sección crítica para después ser demorado indefinidamente, no puede permitirse el interbloqueo o la inanición.

Si ningún proceso está en su sección crítica, cualquier proceso que solicite entrar en la suya debe poder hacerlo sin demora.

No se debe suponer sobre la velocidad relativa de los procesos o el número de procesadores.

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Un proceso permanece en su sección crítica por un tiempo finito.

Una manera de satisfacer los requisitos de exclusión mutua es dejar la responsabilidad a los procesos que deseen ejecutar concurrentemente. Tanto si son programas del sistema como de aplicación, los procesos deben coordinarse unos con otros para cumplir la exclusión mutua, sin ayuda del lenguaje de programación o del sistema operativo. Estos métodos se conocen como soluciones por software.

Planteamiento del problema.

Desde la perspectiva del servidor se creó una clase llamada hilo de usuario que funciona a modo de una sesión cada vez que un cliente se conecta se crea un hilo de sesión que a su vez crea dos objetos uno para la subida de datos y otro para la descarga de datos, la entrada de datos se comunica directamente con el servidor y la salida se comunica directamente con el cliente, las operaciones de entrada con el propio servidor son solo para operar sobre la base de datos, una vez realizadas las operaciones se comunica con su propio hilo del cliente para usar la salida y comunicarse con el cliente.

De parte del cliente, justo al iniciarlo se inicia la conexión con el servidor, cada vez que se intenta iniciar sesión el cliente le pide al servidor una nueva palabra aleatoria para poder iniciar sesión una vez iniciada la sesión le pide los contactos del usuario que se acaba de logear. Posteriormente, cada vez que se quiere enviar un mensaje se envía la petición del servidor; cuando se quiere agregar un usuario también se le envía la petición al servidor, el servidor nos contesta en ambos casos con llamadas a las funciones de la interfaz por medio de la entrada.

Cabe destacar que todos los procesos son hilos con exclusión mutua para garantizar que no haya colisiones en el cliente y el servidor al enviar los datos.

La encriptación se quiso lograr colocándola en las entradas y salidas directamente mediante un objeto que a partir de la palabra aleatoria generaba el MD5 para el loggeo y también utilizaba la misma palabra aleatoria para codificar los mensajes.

Para encriptar los mensajes al inicio se propuso el algoritmo RSA lamentablemente este necesitaba transferencia de archivos, para la clave pública, por tanto la propuesta final fue

de un algoritmo AES, cuya llave de 128 bits la obteníamos de la palabra aleatoria lo cual hacia más fácil y menos pesado la encriptación.

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Diagrama cliente.

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

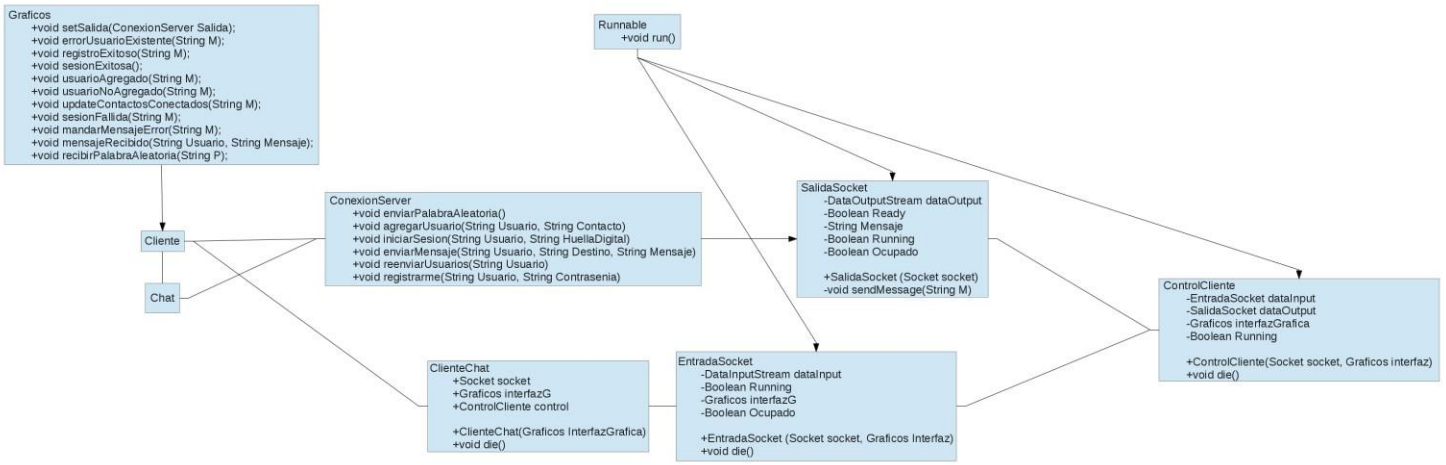
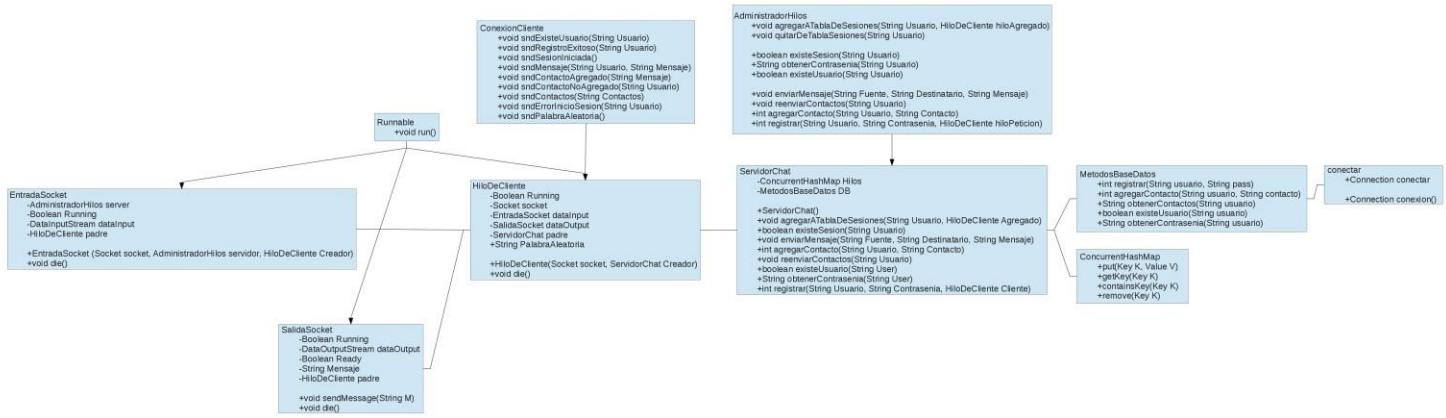


Diagrama servidor.



Nota: la clase códec que se planeaba usar en las entradas y salidas tenían un método estático para generar la palabra aleatoria; una instancia de esta misma podría crearse con la palabra aleatoria que generaba al crearse sesgaba la palabra aleatorio para crear la llave RSA. También poseía dos funciones: codificar y decodificar. Además de generar la huella digital con MD5, todo esto se imprentaba utilizando las librerías de seguridad de java, al ser estas librerías parte del JDK provisto por ORACLE podemos decir que son librerías confiables.

Lamentablemente el envío de la palabra aleatoria, la codificación de mensajes y decodificación e mensajes aumentaba ligeramente los tiempo de envío de cada mensaje lo cual impactaba en la sincronización de los equipos.

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

LOGEO

La parte del logeo se realizaba justo después de mandar la palabra aleatoria, se oculta la contraseña con la palabra aleatoria se encriptaba y se mandaba al servidor; el servidor al

Figura 15: Programa desplegando los resultados obtenidos a través del archivo proporcionado y

tienen ambos ingredientes los modificaron a través de (por codificación no se pudo realizar por las razones mencionadas anteriormente).

CONCLUSION.

Después de haber concluido este proyecto, hemos conocido como se realiza un sistema de validación con MD5. El cual es

A pesar de haber sido considerado criptográficamente seguro en un principio, ciertas investigaciones han revelado vulnerabilidades que hacen cuestionable el uso futuro del MD5. En agosto de 2004, Xiaoyun Wang, Dengguo Feng, Xuejia Lai y Hongbo Yu anunciaron el descubrimiento de colisiones de hash para MD5. Su ataque se consumó en una hora de cálculo con un clúster IBM P690.

Aunque dicho ataque era analítico, el tamaño del hash (128 bits) es lo suficientemente pequeño como para que resulte vulnerable frente a ataques de «fuerza bruta» tipo

«cumpleaños». El proyecto de computación distribuida MD5CRK arrancó en marzo de 2004 con el propósito de demostrar que MD5 es inseguro

Figura 15. Programa desplegado los resultados obtenidos a través de apoyos proporcionados y la publicación de la vulnerabilidad del equipo de Wang.

Debido al descubrimiento de métodos sencillos para generar colisiones de hash, muchos investigadores recomiendan su sustitución por algoritmos alternativos tales como SHA-1 o RIPEMD-160.

BIBLIOGRAFIA.

http://www.taringa.net/posts/info/13965115/Que-es-el-Hash-MD5-y-como-usarlo.html

http://sistemaoperativo.wikispaces.com/Exclusion+Mutua

http://neo.lcc.uma.es/evirtual/cdd/tutorial/presentacion/md5.html

http://rootear.com/seguridad/md5-como-funciona-usos

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

olegando los resultados obtenidos a través del archivo proporcionado y
después de haber realizado todos los cálculos

BENEMÉRITA UNIVERSIDAD
AUTÓNOMA DE PUEBLA

Facultad de Ciencias De La Computación

Proyecto: Calculador de Latencias

Primer Parcial

Alumno:
Quiroz Flores Alejandro

Asignatura:
Modelos de Redes

Catedrático:
Dr. Iván Olmos Pineda

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Fecha:

26 – Septiembre – 2014

Otoño 2014

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Introducción

Éste proyecto está compuesto por una serie de operaciones que determinarán los tiempos de latencia en la transmisión de paquetes entre diferentes hosts en una red representada adecuadamente por un grafo.

La latencia es la suma de los retardos en una unidad de tiempo producidos por la velocidad de transmisión y tempo de propagación de un paquete de datos a través de un canal, así como de las características de los buffers de los equipos de conectividad.

Problema a Resolver

El problema propuesto para ser resuelto por nuestro proyecto es la creación de un grafo a partir de un archivo de texto proporcionado por el usuario, el cual simboliza una red de routers. Después, se busca determinar todos los posibles caminos o rutas entre dos routers (vértices), para que así podamos calcular la latencia del envío un paquete de tamaño establecido y, finalmente, se pueda calcular el menor tiempo posible que un archivo tardaría en transferirse de un punto de la red al otro.

Especificaciones de la Implementación

El proyecto fue aplicado y programado en el lenguaje Java, lo cual resulta conveniente debido a las facilidades que proporciona para el manejo de estructuras de datos como listas ligadas o distintos tipos de arreglos.

El funcionamiento general es el siguiente:

- A partir de la lectura de archivo de texto con un formato específico (véase Anexo 1) proporcionado por el usuario e ingresado como parámetro en la ejecución del programa por línea de comandos, se genera un grafo y se obtienen los datos necesarios para el cálculo de la latencia deseada, como lo son el numero de routers en el grafo (vertices), las características de los canales utilizados (aristas), el tamaño del archivo a enviar, el router emisor y receptor, entre otros.
- Se genera una lista con todos posibles caminos entre el router inicial y el router final mediante el uso del algoritmo de Búsqueda por Profundidad (DFS).
- Se calcula la latencia de todos los caminos generados y se determina qué camino tiene el menor retardo.
- Se calcula el tiempo que le tomaría a un archivo ser enviado desde el router inicial hasta el final por el camino con la menor latencia antes determinados.

Se programaron las siguientes clases:
(Nota: Se trató de programar el proyecto entero en inglés, nombrado tanto variables como clases y métodos en dicho idioma)

- Componentes

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

- Router.java: Elemento básico del grafo. Representa a las vértices en el esquema tradicional de un grafo. Contiene un número identificador, un tiempo de cola específico y una lista de Links (enlaces) que indica los routers adyacentes, además de métodos necesarios para su manipulación.
- Link.java: Elemento de asociación de un Router con un Channel (canal). Contiene la referencia a un Router y un canal en específico.
- Channel.java: Elemento que representa las aristas en un grafo tradicional, pero más complejas, pues contienen bastantes atributos adicionales que impiden tratarlas como un simple camino con un peso determinado. Entre éstos atributos tenemos la longitud del canal en metros, velocidad de transmisión, tamaño de paquetes y el tamaño de los datos de usuario y control en ese paquete. Además contiene métodos *setters* y *getters* tradicionales.
- Graph.java: La estructura de Grafo. Posee el contador de Routers y Channels en el grafo, así como la lista de Routers en el. Además, se programaron bastantes métodos extras para su manejo, permitiendo incluso utilizar esta clase en grafos con canales direccionados.

- Operaciones
  - LatenctyOperations.java: El operador. Contiene los atributos necesarios de la lista de listas de caminos, las latencias de cada uno de los caminos, así como métodos utilizados en el cálculo de la latencia que deseamos. Entre las operaciones más importantes se encuentran:
    - calculateTotalLatency: calcula la latencia total en función del nodo inicial y final, el camino elegido como el de menor retardo y el tamaño del archivo que el usuario desea enviar. Hace uso del método *calculateChannelLatency*.
    - calculateChannelLatency: calcula la latencia de la transmisión de un paquete únicamente entre dos routers en un path específico; implementa recursividad Contiene el código nuclear del proyecto. Considera todos los casos posibles en los que el paquete puede presentar retrasos diferentes, dependiendo de circunstancias como la necesidad de segmentarse en más paquetes o si está llegando al router final.

      Como mención importante, se debe destacar que en el caso de la transmisión de paquetes con una cantidad de Datos de usuario menores a los que soporta el canal, el cálculo de su latencia se realiza con la cantidad de datos que transporta realmente y no se generaliza como un paquete "totalmente cargado", como solíamos hacer en el cálculo realizado por escrito en otros ejercicios realizados en clase. Ésta situación permite generar un cálculo más exacto en la latencia total y parcial, pues disminuye del tamaño del paquete enviado y por lo tanto, tarda menos que la generalización de enviar únicamente paquetes totalmente cargados. Además, esto

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y *disminuye ligeramente la latencia resultante en comparación a los cálculos que no consideran éste aspecto.*

- generatePacketsList: método auxiliar del cálculo de latencia en un canal. Determina la cantidad y el tamaño de los paquetes a utilizar en caso de no caber en el tamaño de paquete de un usuario.
- generatePaths_DFS: genera una lista con todos los paths entre dos vértices. El algoritmo está basado en la conocida Búsqueda por Profundidad (BPP, o DFS por sus siglas en inglés). Implementa la técnica de programación de backtracking.
- pathsToString(): genera la impresión de todos los paths encontrados.

- Principal.java: Como el nombre indica, es la clase principal, quien lee el archivo entrante y genera las instancias de todos los componentes necesarios para resolver nuestro problema.

Errores del Programa

A pesar de hacer un excelente cálculo de latencias (verificado haciendo pruebas con el archivo de texto "archivo.txt" adjunto al programa), el cálculo de los caminos es incorrecto. Por ello

Conclusión

La implementación de éste programa me permite generar dos ideas concluyentes:

El cálculo de latencias en una red de dispositivos es bastante más interesante y complicado de lo que parece, pues involucra gran cantidad de conceptos a considerar, como el tiempo de propagación y de transmisión, dependientes del las características específicas del medio por el cual queremos trasmitir datos. Éste proyecto me permitió entender perfectamente la necesidad de considerar tan específicos aspectos, con lo que creo cumplió su objetivo de representar el aprendizaje de todo un bloque de estudio de la materia y representar una calificación parcial para la misma.

Por otro lado, el proyecto mostró la necesidad de hacer uso de recursos y técnicas de programación como recursividad y backtracking que, como programadores en formación, debemos tratar de dominar cuanto antes; también es importante hacer uso más constante de clases y utilidades ya existentes en el entorno de desarrollo de Java, así como de aplicar las "buenas costumbres" en la codificación y tratar de hacer dicha codificación "universal": detalles como el uso del lenguaje Inglés o un código pensado en el costo computacional que requiere son dos simples ejemplos de tales costumbres.

Anexos

[1] Estructura del archivo de texto para la generación del grafo

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Separados por saltos de línea y comas (‘,’), el contenido del texto debe estar estructurado de la siguiente forma:

1) Datos del grafo: *vértices, enlaces*
2) Características de enlaces (n veces):

Router1, Router2, distancia(m), Capacidad(Mbps), TamañoPaquete(B), DatosControXPaquete(B)

3) Tiempos de cola de Routers: *TCola1,..,TCola_n (m veces)*
4) Vértices inicial y final para calculo de latencia: *Vi, Vf*
5) Tamaño de archivo a transmitir: *Tam archivo (GB)*

Ejemplo:

```
6,6
1,2,98,1000,1500,150
2,3,98,1000,1500,160
3,4,98,1000,1500,170
3,5,98,1000,1500,180
4,6,98,1000,1500,190
5,6,98,1000,1500,200
1,1,1,1,1,1
1,6
4
```

Figura 15. Programa desplegando los resultados obtenidos a través del archivo precompilado y después de haber realizado todos los cálculos

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Modelo de Redes

Dr. Olmos Pineda Iván

Proyecto Final

Fecha de entrega: jueves 27 de noviembre del 2014

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Equipo:

Barragán Hernández Nancy

Mellado Robles David

Meneses Casarrubias Brian

Morelos Dorantes Iván

Vázquez Juárez Jamelli

Figura 15. Programa desplegando los resultados obtenidos a través del archivo proporcionado y después de haber realizado todos los cálculos

Contenido

Introducción

La comunicación es fundamental para establecer buenas relaciones humanas, sin embargo, esta se ve afectada ya que requiere el entendimiento mutuo del emisor.

Existiendo un propósito para la comunicación y una respuesta por producirse, el comunicador desea que su comunicación tenga alta fidelidad. La palabra fidelidad es empleada aquí en el sentido de que el comunicador ha de lograr lo que desea.

Se logra una comunicación verdadera si estamos interesados en el lenguaje de la otra persona, de tal forma que esta se puede expresar libre y sinceramente, si escuchamos atentamente y observamos con conciencia y somos capaces de ponernos en el lugar del otro. Solo entonces estaremos estableciendo las bases de una buena comunicación.

Cabe destacar que el internet es una de las redes más grandes de telecomunicaciones a nivel mundial, su importancia radica en que a través de ella podemos obtener información rápida y eficaz sobre diversos temas, sin moverse de casa o del lado de su computador, esto entre infinidades de aplicaciones que podemos utilizar, entre estas aplicaciones está el chat, que designa una comunicación escrita realizada de manera instantánea mediante el uso de un software y a través de Internet entre dos, tres o más personas ya sea de manera pública a través de los llamados chats públicos (mediante los cuales cualquier usuario puede tener acceso a la conversación) o privada, en los que se comunican dos o más personas.

El chat sirve para comunicarse con grupos de personas las cuales opinan de diferentes temas, para cambiar información, entre otras cosas como simplemente saludar a los amigos.

En este trabajo se va a desarrollar un programa que sea capaz de permitir la comunicación persona a persona, garantizando un servicio seguro con encriptación de datos.

Marco Teórico

Encriptación: es el proceso mediante el cual cierta información o texto sin formato es cifrado de forma que el resultado sea ilegible a menos que se conozcan los datos necesarios para su interpretación. Es una medida de seguridad utilizada para que al momento de almacenar o transmitir información sensible ésta no

pueda ser obtenida con facilidad por terceros. Opcionalmente puede existir además un proceso de desencriptación a través del cual la información puede ser interpretada de nuevo a su estado original, aunque existen métodos de encriptación que no pueden ser revertidos.

Algunos de los usos más comunes de la encriptación son el almacenamiento y transmisión de información sensible como contraseñas, números de identificación legal, números de tarjetas de crédito, reportes administrativo-contables y conversaciones privadas, entre otros.

Como sabemos, en un Sistema de Comunicación de Datos, es de vital importancia asegurar que la Información viaje segura, manteniendo su autenticidad, integridad, confidencialidad y el no repudio de la misma entre otros aspectos.

HashMap: es una colección de objetos, (como los Arrays), pero estos no tienen orden. Cada objeto se identifica mediante algún identificador apropiado, por ejemplo un "uuid". El nombre HASH, hace referencia a una técnica de organización de archivos llamada hashing o "dispersión" en el cual se almacenan registros en una dirección del archivo que es generada por una función que se aplica sobre la llave del registro.

Socket: es un método para la comunicación entre un programa del cliente y un programa del servidor en una red. Un socket se define como el punto final en una conexión. Los sockets se crean y se utilizan con un sistema de peticiones o de llamadas de función a veces llamados interfaz de programación de aplicación de sockets (API, application programming interface).

Un socket es también una dirección de Internet, combinando una dirección IP (la dirección numérica única de cuatro partes que identifica a un ordenador particular en Internet) y un número de puerto (el número que identifica una aplicación de Internet particular, como FTP, Gopher, o WWW).

MD5 y Encriptación

5

MD5

MD5 es una técnica que permite obtener una huella digital de una serie de datos de entrada

- Con la huella, se puede verificar la integridad de los datos, es decir, comprobar que no hayan sufrido alteración

- Además, puede emplearse para la validación de usuarios o procesos, aunque no es la más usada, a comparación de otras técnicas como RSA.

Librerias utilizadas:

El **java.math.BigInteger** clase proporciona análogos de operaciones para todos los operadores enteros primitivos de Java y todos los métodos pertinentes de java.lang.Math.

También proporciona operaciones de la aritmética modular, el cálculo GCD, pruebas de primalidad, generación privilegiada, manipulación de bits, y algunas otras operaciones diversas. Todas las operaciones se comportan como si estuvieron representados BigIntegers en notación de complemento a dos.

La semántica de las operaciones aritméticas y operaciones lógicas bit a bit son similares a los de enteros operadores aritméticos de Java y los operadores enteros bit a bit de Java, respectivamente. La semántica de las operaciones de cambio se extienden los de operadores de desplazamiento de Java para permitir distancias de desplazamiento negativos.

Las operaciones de comparación realizan la firmaron comparaciones enteros. Operaciones aritméticas modulares se proporcionan para calcular los residuos, realizar la exponenciación, y calcular inversos multiplicativos. Operaciones de bits operan en un solo bit de la representación en complemento a dos de su operando.

Todos los métodos y constructores en este tiro NullPointerException clase cuando aprobaron una referencia de objeto null para cualquier parámetro de entrada.

Declaración de la clase

A continuación se presenta la declaración de java.math.BigInteger clase:

public class BigInteger

extends Número

implements Comparable < BigInteger >

Campo

Los siguientes son los campos para java.math.BigInteger clase:

static BigInteger UNO - La única constante BigInteger.

static BigInteger TEN - Los diez constante BigInteger.

static BigInteger ZERO - El cero constante BigInteger.

Contructor

- BigInteger (int signum, byte [] magnitud)

Traduce la representación signo-magnitud de un BigInteger en un BigInteger.

Esta clase MessageDigest proporciona aplicaciones de la funcionalidad de un mensaje algoritmo de resumen, como SHA-1 o SHA-256. Compendios de mensajes son funciones hash unidireccionales seguros que toman datos arbitraria tamaño y salida de un valor hash de longitud fija.

Un objeto MessageDigest comienza inicializado. Los datos se procesan a través de él utilizando los métodos actualizados. En cualquier punto de reinicio puede ser llamado para restablecer la digestión. Una vez que todos los datos que se actualizan se ha actualizado, una de las digerir métodos deberían ser llamados para completar el cálculo hash.

El digest method puede ser llamado una vez para un número dado de cambios. Después de digestión ha sido llamado, el objeto MessageDigest se restablece a su estado inicializado.

Las implementaciones son libres de implementar la interfaz Cloneable. Las aplicaciones cliente pueden probar cloneability intentando la clonación y agarrar la CloneNotSupportedException:

Tenga en cuenta que esta clase es abstracta y se extiende desde MessageDigestSpi por razones históricas. Los desarrolladores de aplicaciones sólo deberían tomar nota de los métodos definidos en este MessageDigest clase;

todos los métodos de la superclase están destinados a los proveedores de servicios criptográficos que deseen suministrar sus propias implementaciones de algoritmos de compendio de mensajes.

Se requiere que cada implementación de la plataforma Java para apoyar los siguientes estándares MessageDigest algoritmos:

- MD5
- SHA-1

- SHA-256

Estos algoritmos se describen en la sección MessageDigest de la Documentación Java Cryptography Standard Architecture Algoritmo Nombre. Consulte la documentación de la versión de su aplicación para ver si son compatibles con otros algoritmos.

Constructor

- protected MessageDigest (String algoritmo)

Crea un resumen del mensaje con el nombre del algoritmo especificado.

Métodos

- byte [] digest (byte [] input)

Realiza una actualización final de la digestión mediante la matriz de bytes, y luego completa el cálculo resumen.

- String toString ()

Devuelve una representación de cadena de este resumen del mensaje objeto.

```
import java.math.BigInteger;
import java.security.MessageDigest;


public static String getMD5(String input) {
   try {
     MessageDigest md = MessageDigest.getInstance("MD5");
      byte[] messageDigest = md.digest(input.getBytes());
```

```
      BigInteger number = new BigInteger(1, messageDigest);
      String hashtext = number.toString(16);

      // Now we need to zero pad it if you actually want the full 32
chars.

      while (hashtext.length() < 32) {

         hashtext = "0" + hashtext;

      }

      return hashtext;

   }

   catch (NoSuchAlgorithmException e) {

      throw new RuntimeException(e);

   }

}
```

¿Porque utilizamos estas librerías para la implementación de MD5?

Una de las razones por la cual elegimos las librerías **java.math.BigInteger y java.security.MessageDigest** para la implementación del algoritmo de MD5, es porque estas librerías son nativas de Java y ya que nuestro programa lo implementamos en el lenguaje Java, éstas se adecuan muy bien a él y nos da una manera fácil de importar las funciones que nos ofrecen esas librerías para la implementación del algoritmo de MD5.

Otra razón fue porque las funciones de esas librerías no son nada difíciles de utilizar, ya que toda la documentación de los métodos y atributos que nos ofrecen las podemos encontrar en la documentación de Java.

Encriptación

(Cifrado, codificación). La encriptación es el proceso para volver ilegible información considera importante. La información una vez encriptada sólo puede leerse aplicándole una clave.

Se trata de una medida de seguridad que es usada para almacenar o transferir información delicada que no debería ser accesible a terceros. Pueden ser contraseñas, números de tarjetas de crédito, conversaciones privadas, etc.

Para encriptar información se utilizan complejas fórmulas matemáticas y para desencriptar, se debe usar una clave como parámetro para esas fórmulas.

El texto plano que está encriptado o cifrado se llama criptograma.

Algoritmo criptográfico

En computación y criptografía un algoritmo criptográfico es un algoritmo que modifica los datos de un documento con el objetivo de alcanzar algunas características de seguridad como autenticación, integridad y confidencialidad.

Clasificación

Los algoritmos criptográficos se pueden clasificar en:

- **Criptografía simétrica o de clave secreta.** La criptografía simétrica (en inglés symmetric key cryptography), también llamada criptografía de clave secreta (en inglés secret key cryptography) o criptografía de una clave1 (en inglés single-key cryptography), es un método criptográfico en el cual se usa una misma clave para cifrar y descifrar mensajes. Las dos partes que se comunican han de ponerse de

acuerdo de antemano sobre la clave a usar. Una vez que ambas partes tienen acceso a esta clave, el remitente cifra un mensaje usando la clave, lo envía al destinatario, y éste lo descifra con la misma clave.

- **Criptografía asimétrica o de clave pública**. La criptografía asimétrica (en inglés asymmetric key cryptography), también llamada criptografía de clave pública (en inglés public key cryptography) o criptografía de dos claves1 (en inglés two-key cryptography), es el método criptográfico que usa un par de claves para el envío de mensajes. Las dos claves pertenecen a la misma persona que ha enviado el mensaje. Una clave es pública y se puede entregar a cualquier persona, la otra clave es privada y el propietario debe guardarla de modo que nadie tenga acceso a ella. Además, los métodos criptográficos garantizan que esa pareja de claves sólo se puede generar una vez, de modo que se puede asumir que no es posible que dos personas hayan obtenido casualmente la misma pareja de claves.

Listado de algoritmos criptográficos

Algoritmos Simétricos:

- DES
- 3DES
- RC2
- RC4
- RC5
- IDEA
- AES
- Blowfish

Algoritmos Asimétricos

- Diffie-Hellman
- RSA
- ElGamal
- Criptografía de curva elíptica

Data Encryption Standard (DES)

Es un algoritmo de cifrado, es decir, un método para cifrar información, escogido como un estándar FIPS en los Estados Unidos en 1976, y cuyo uso se ha propagado ampliamente por todo el mundo. El algoritmo fue controvertido al principio, con algunos elementos de diseño clasificados, una longitud de clave relativamente corta, y las continuas sospechas sobre la existencia de alguna puerta trasera para la National Security Agency (NSA). Posteriormente DES fue sometido a un intenso análisis académico y motivó el concepto moderno del cifrado por bloques y su criptoanálisis.

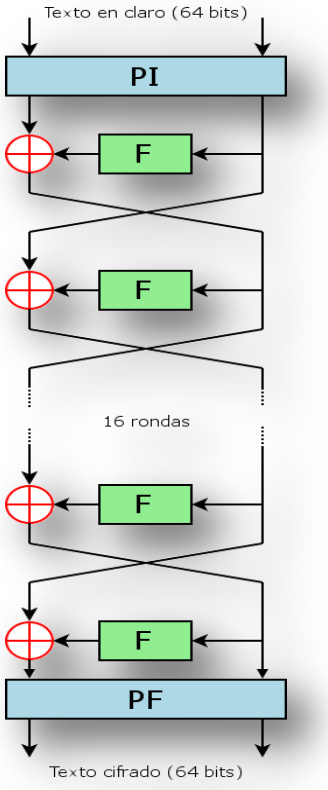
Hoy en día, DES se considera inseguro para muchas aplicaciones. Esto se debe principalmente a que el tamaño de clave de 56 bits es corto; las claves de DES se han roto en menos de 24 horas. Existen también resultados analíticos que demuestran debilidades teóricas en su cifrado, aunque son inviables en la práctica. Se cree que el algoritmo es seguro en la práctica en su variante de Triple DES, aunque existan ataques teóricos.

El algoritmo como estándar

A pesar de la polémica, DES fue aprobado como estándar federal en noviembre de 1976, y publicado el 15 de enero de 1977 como FIPS PUB 46, autorizado para el uso no clasificado de datos. Fue posteriormente confirmado como estándar en 1983, 1988 (revisado como FIPS-46-1), 1993 (FIPS-46-2), y de nuevo en 1998 (FIPS-46-3), éste último definiendo "TripleDES" (véase más abajo). El 26 de mayo de 2002, DES fue finalmente reemplazado por AES (Advanced Encryption Standard), tras una competición pública (véase Proceso de Advanced Encryption Standard). Hasta hoy día (2006), DES continúa siendo ampliamente utilizado.

Descripción

DES es el algoritmo prototipo del cifrado por bloques — un algoritmo que toma un texto en claro de una longitud fija de bits y lo transforma mediante una serie de operaciones básicas en otro texto cifrado de la misma longitud. En el caso de DES el tamaño del bloque es de 64 bits. DES utiliza también una clave criptográfica para modificar la transformación, de modo que el descifrado sólo puede ser realizado por aquellos que conozcan la clave concreta utilizada en el cifrado. La clave mide 64 bits, aunque en realidad, sólo 56 de ellos son empleados por el algoritmo. Los ocho bits restantes se utilizan únicamente para comprobar la paridad, y después son descartados. Por tanto, la longitud de clave efectiva en DES es de 56 bits, y así es como se suele especificar.



Estructura básica

La estructura básica del algoritmo aparece representada en la Figura 1: hay 16 fases idénticas de proceso, denominadas rondas. También hay una permutación inicial y final denominada PI y PF, que son funciones

inversas entre sí (PI "deshace" la acción de PF, y viceversa). PI y PF no son criptográficamente significativas, pero se incluyeron presuntamente para facilitar la carga y descarga de bloques sobre el hardware de mediados de los 70. Antes de las rondas, el bloque es dividido en dos mitades de 32 bits y procesadas alternativamente. Este entrecruzamiento se conoce como

esquema Feistel.

La estructura de Feistel asegura que el cifrado y el descifrado sean procesos muy similares — la única diferencia es que las subclaves se aplican en orden inverso cuando desciframos. El resto del algoritmo es idéntico. Esto simplifica enormemente la implementación, en especial sobre hardware, al no haber necesidad de algoritmos distintos para el cifrado y el descifrado.

El símbolo rojo "⊕" representa la operación OR exclusivo (XOR). La función-F mezcla la mitad del bloque con parte de la clave. La salida de la función-F se combina entonces con la otra mitad del bloque, y los bloques son intercambiados antes de la siguiente ronda. Tras la última ronda, las mitades no se intercambian; ésta es una característica de la estructura de Feistel que hace que el cifrado y el descifrado sean procesos parecidos.

La función (F) de Feistel

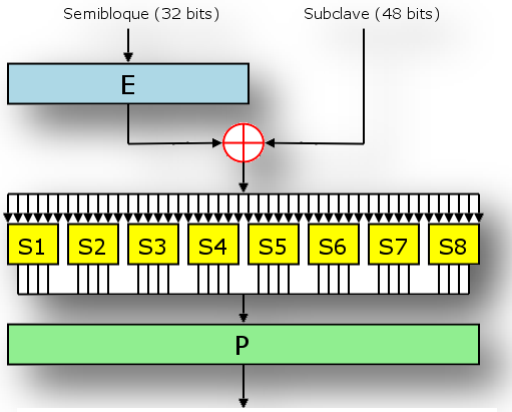


Figura 2 - La función Feistel (Función-F) de DES.

La función-F, representada en la Figura 2, opera sobre medio bloque (32 bits) cada vez y consta de cuatro pasos:

1. **Expansión** — la mitad del bloque de 32 bits se expande a 48 bits mediante la permutación de expansión, denominada E en el diagrama, duplicando algunos de los bits.

2. **Mezcla** — el resultado se combina con una subclave utilizando una operación XOR. Dieciséis subclaves — una para cada ronda — se derivan de la clave inicial mediante la generación de subclaves descrita más abajo.

3. **Sustitución** — tras mezclarlo con la subclave, el bloque es dividido en ocho trozos de 6 bits antes de ser procesados por las S-cajas, o cajas de sustitución. Cada una de las ocho S-cajas reemplaza sus seis bits de entrada con cuatro bits de salida, de acuerdo con una trasformación no lineal, especificada por una tabla de búsqueda. Las S-cajas constituyen el núcleo de la seguridad de DES — sin ellas, el cifrado sería lineal, y fácil de romper.

4. **Permutación** — finalmente, las 32 salidas de las S-cajas se reordenan de acuerdo a una permutación fija; la P-caja

   Alternando la sustitución de las S-cajas, y la permutación de bits de la P-caja y la expansión-E proporcionan las llamadas "confusión y difusión" respectivamente, un concepto identificado por Claude Shannon en los 40 como una condición necesaria para un cifrado seguro y práctico.

Generación de claves

La Figura 3 representa la generación de claves para el cifrado — el algoritmo que se encarga de proporcionar las subclaves. Primero, se seleccionan 56 bits de la clave de los 64 iniciales mediante la Elección Permutada 1 (PC-1) — los ocho bits restantes pueden descartarse o utilizarse como bits de comprobación de paridad. Los 56 bits se dividen entonces en dos mitades de 28 bits; a continuación cada mitad se trata independientemente. En rondas sucesivas, ambas mitades se desplazan hacia la izquierda uno o dos bits (dependiendo de cada ronda), y entonces se seleccionan 48 bits de subclave mediante la Elección Permutada 2 (PC-2) — 24 bits de la mitad izquierda y 24 de la derecha. Los desplazamientos (indicados por "<<<" en el diagrama) implican que se utiliza un conjunto diferente de bits en cada subclave; cada bit se usa aproximadamente en 14 de las 16 subclaves.

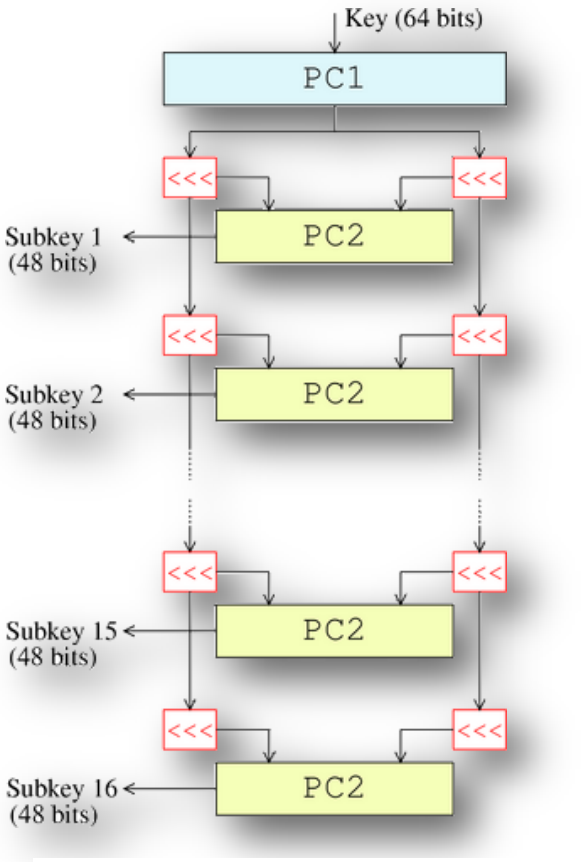


Figura 3 - La generación de clave de DES.

La generación de claves para descifrado es similar — debe generar las claves en orden inverso.

Encriptar y Desencriptar claves en Java

Java cuenta con una variedad de librerías y algoritmos para realizar la encriptación y desencriptación de datos, estas librerías se encuentran dentro de los paquetes: javax.crypto y java.security.

Dentro de los algoritmos de Criptografía más populares que podemos encontrar:

- Blowfish.
- DES.
- DESede.
- PBEWithMD5AndDES.
- PBEWithMD5AndTripleDES.
- TripleDES.

¿Por qué usar DES ante los otros algoritmos de encriptación?

Aunque en la actualidad el algoritmo DES es considerado inseguro en sus inicios fue aprobado como estándar federal en los Estado Unidos además de ser hacerse públicos los detalles del algoritmo se hicieron varios estudios de vulnerabilidad a nivel académico y por lo tanto se cuenta con bastante información acerca del funcionamiento del algoritmo. En la actualidad sigue teniendo gran popularidad y en Java cuenta con librerías nativas para implementación de aplicaciones que haga uso de este algoritmo.

En resumen se eligió DES por:

- Popularidad en la actualidad.
- Librerías nativas en java.

Desarrollo

Servidor

Lo primero que se llevó a cabo fue la construcción del servidor, donde todos los clientes se van a conectar. La clase comienza creando las estructuras que guardaran los datos de usuario como la contraseña, el nombre de usuario y los contactos.

```
private Map<Integer,String> usuarios  = new HashMap<>();
private Map<String,String> passwords  = new HashMap<>();
private Map<String,ArrayList<String>> contactos  = new HashMap<>();
private ArrayList<String> lista = new ArrayList<>();
```

```
private String pass;

private String mensajeAleatorio;
```

Lo siguiente en la lista son las variables que nos ayudarán a encriptar y desencriptar las cadenas de mensajes, estas variables son dcipher y ecipher. También definimos las variables que nos ayudarán a generar las claves de desencriptación: keyBytes y iviBytes.

```
private static Cipher dcipher, ecipher;
```

```
private byte[] keyBytes;

private byte[] iviBytes;
```

Las claves se escriben en el cliente y en servidor para garantizar que ambos generen las mismas llaves para encriptar y desencriptar los datos.

```
keyBytes = "12345678".getBytes();
iviBytes = "input123".getBytes();
```

La siguiente función es la encargada de crear las conexiones.

```
public void start() {
    keepGoing = true;
    /* iniciar servidor */
    try
    {
       // socket usado por el servidor
       ServerSocket serverSocket = new ServerSocket(port);
       // ciclo infinito para esperar conexiones
       while(keepGoing)
       {
          display("Esperando clientes en el puerto " + port + ".");

          Socket socket = serverSocket.accept();        // aceptar conexión
          // si se detuvo el servidor
          if(!keepGoing)
          break;
          ClientThread t = new ClientThread(socket);  // hilos para el cliente
          al.add(t);                                  // se guarda en el ArrayList
          t.start();
       }
       // Detener el sevidor
       try {
          serverSocket.close();
```

```
    for(int i = 0; i < al.size(); ++i) {
        ClientThread tc = al.get(i);
        try {
            tc.sInput.close();
            tc.sOutput.close();
            tc.socket.close();
        }
        catch(IOException ioE) {
            display(ioE + "");
        }
    }
}
```

```
            catch(Exception e) {
                display("Exception closing the server and clients: " + e);
            }
        }
        // si hay errores
        catch (IOException e) {
            String msg = sdf.format(new Date()) + " Exception on new ServerSocket:
" + e + "\n";
            display(msg);
        }
    }
```

La función **broadcast(String str)** recibe los mensajes encriptados, los desencripta, busca a los clientes a los que van dirigidos los mensajes, encripta nuevamente los datos y los envía.

```
private synchronized void broadcast(String message) {
        // agregar HH:mm:ss y \n al mensaje
        String [] datos = message.split(",");
        String time = sdf.format(new Date());
        String messageLf = time + " " + message + "\n";
        // mostrar mensaje en consolo o GUI
        if(sg == null)
            System.out.print(messageLf);
        else
            sg.appendRoom(messageLf);     //mostrar en la ventana del servidor

        // Ciclo en reversa en caso de que se necesite remover un cliente
```

```
        // porque se desconectó
        for(int i = al.size(); --i >= 0;) {
            ClientThread ct = al.get(i);
            // Escribimos al cliente, si no se puede, se elimina de la lista
            if(!ct.writeMsg("")) {
                al.remove(i);
                display("Cliente desconectado " + ct.username + " no existe en la
lista.");
            }else{
                if(ct.username.equals((datos[2]))){
                    for(int p = al.size(); --p >= 0;) {
                        ClientThread ct2 = al.get(p);
```

```
                    if(ct2.username.equals((datos[0]))){
                        try {
                            String time2 = sdf.format(new Date());
                            String message2 = time2 + " " + datos[0] + ": " + datos[1]
+ "\n";                     SecretKeySpec key = new SecretKeySpec(keyBytes,
"DES");                     IvParameterSpec ivSpec = new

IvParameterSpec(iviBytes);
                            ecipher = Cipher.getInstance("DES/CTR/NoPadding");
                            // inicializar los ciphers con la llave
                            ecipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
                            String encrypted = encrypt(message2);
                            ct2.writeMsg(encrypted);
                        } catch (NoSuchAlgorithmException ex) {

Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
                        } catch (NoSuchPaddingException ex) {

Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
                        } catch (InvalidKeyException ex) {

Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
                        } catch (InvalidAlgorithmParameterException ex) {

Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
                        }
                    }
```

```
 }
try {
    String time2 = sdf.format(new Date());
    String message2 = time2 + " " + datos[0] + ": " + datos[1] + "\n";
    SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
    IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
    ecipher = Cipher.getInstance("DES/CTR/NoPadding");
    // inicializar los ciphers con la llave
    ecipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
    String encrypted = encrypt(message2);
    ct.writeMsg(encrypted);
} catch (NoSuchAlgorithmException ex) {
```

```
                Logger.getLogger(Server.class.getName()).log(Level.SEVERE,
null,  ex);
            } catch (NoSuchPaddingException ex) {
                Logger.getLogger(Server.class.getName()).log(Level.SEVERE,
null,  ex);
            } catch (InvalidKeyException ex) {
                Logger.getLogger(Server.class.getName()).log(Level.SEVERE,
null, ex);
            } catch (InvalidAlgorithmParameterException ex) {
                Logger.getLogger(Server.class.getName()).log(Level.SEVERE,
null, ex);
            }
        }
    }
}
```

La siguiente clase es el hilo para cada cliente. En este hilo se reciben los mensajes de los clientes y se clasifican los tipos de mensajes. Hay 6 tipos de mensajes

- MESSAGE. Es un mensaje ordinario, el servidor desencripta los mensajes y los vuelve a encriptar antes de enviarlos.
- WHOISIN. Este tipo de mensaje indica que un cliente quiere ver los usuarios que están en línea.
- LOGOUT. Con este tipo d mensaje se da a entender que el cliente se quiere desconectar.
- REGISTRY. Este mensaje indica una petición de registro.
- LOGIN. Una petición de inicio de sesión.
- MD5. Indica que es una mensaje que tiene el formato de MD5.


```
/** Hilo para cada cliente */
```

```
class ClientThread extends Thread {

    // socket para la conexión

    Socket socket;
    ObjectInputStream sInput;
    ObjectOutputStream sOutput;

    // id unico
    int id;

    String username;
    ChatMessage cm;

    //Fecha en la que se conecta

    String date;
```

```
ClientThread(Socket socket) {
   // unico id
   id = ++uniqueId;
   this.socket = socket;
   System.out.println("Thread creando objeto Input/Output Streams");
   try
   {
      //creamos input y output
      sOutput = new ObjectOutputStream(socket.getOutputStream());
      sInput  = new ObjectInputStream(socket.getInputStream());
      username = (String) sInput.readObject();
      if(username.equals("SYSTEM")){
         System.out.println("Usuario registrandose");
      }else{
         display(username + " acaba de conectarse.");
      }
   }
   catch (IOException e) {
      display("Exception creating new Input/output Streams: " + e);
      return;
   }
   catch (ClassNotFoundException e) {
   }
   date = new Date().toString() + "\n";
}

// corre por siempre
public void run() {
   boolean keepGoing = true;
   while(keepGoing) {
```

```
// leer cadena como objeto
try {
   cm = (ChatMessage) sInput.readObject();
}
catch (IOException e) {
   display(username + " Exception reading Streams: " + e);
   break;
}
catch(ClassNotFoundException e2) {
   break;
}
```

```
String message = cm.getMessage();
//System.out.println(message);
// switch para los tipos de mensaje
switch(cm.getType()) {
   case ChatMessage.MESSAGE:
      try {
         // generar llave secreta
         //System.out.println(message+" key: "+key);
         String msg = message;
         SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
         IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
         //key = KeyGenerator.getInstance("DES").generateKey();
         dcipher = Cipher.getInstance("DES/CTR/NoPadding");
         dcipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
         String decrypted = decrypt(msg);
         broadcast(username + "," + decrypted);
      }
      catch (NoSuchAlgorithmException e) {
         System.out.println("No Such Algorithm:" + e.getMessage());
         return;
      }
      catch (NoSuchPaddingException e) {
         System.out.println("No Such Padding:" + e.getMessage());
         return;

    null, ex);
```

```
                tln("Invalid Key:" + e.getMessage()); return;
            } catch (InvalidAlgorithmParameterException ex) {
            Logger.getLogger(Server.class.getName()).log(Level.SEVERE,
            }
        break;
        case ChatMessage.LOGOUT:
            display(username + " desconectado con un mensaje de SALIR.");
            keepGoing = false;
        break;
        case ChatMessage.WHOISIN:
            try {
                SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
                IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
}
catch(InvalidKeyException e) {System.out.prin
```

```
                    ecipher = Cipher.getInstance("DES/CTR/NoPadding");
                    ecipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
                    String encrypted = encrypt("\n\nLista de usuarios conectados "
+ sdf.format(new Date()) + "\n\n");
                    writeMsg(encrypted);
                } catch (NoSuchAlgorithmException ex) {
                    Logger.getLogger(Server.class.getName()).log(Level.SEVERE,
null, ex);

                } catch (NoSuchPaddingException ex) {
                    Logger.getLogger(Server.class.getName()).log(Level.SEVERE,
null, ex);

                } catch (InvalidKeyException ex) {
                    Logger.getLogger(Server.class.getName()).log(Level.SEVERE,
null, ex);
                } catch (InvalidAlgorithmParameterException ex) {
                    Logger.getLogger(Server.class.getName()).log(Level.SEVERE,
null, ex);

                }
                // Revisa todos los usuarios conectados
                int cont=1;
                for(int i = 0; i < al.size(); ++i) {
                    ClientThread ct = al.get(i);
                    if(!ct.username.equals("SYSTEM")){
                        try {
                            SecretKeySpec key = new SecretKeySpec(keyBytes,

                            IvParameterSpec ivSpec = new
"DES");
```

```
IvParameterSpec(iviBytes);
                    ecipher = Cipher.getInstance("DES/CTR/NoPadding");
                    ecipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
                    String encrypted = encrypt(cont + ") " + ct.username + "
desde " + ct.date);

                    cont++;

                    writeMsg(encrypted);
                } catch (NoSuchAlgorithmException ex) {

Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
                } catch (NoSuchPaddingException ex) {

Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
                } catch (InvalidKeyException ex) {
```

```
Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
                    } catch (InvalidAlgorithmParameterException ex) {

Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
                    }
                }
            }
            break;
            case ChatMessage.REGISTRY:
              try {
                String[] mensj = message.split(",");
                SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
                IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
                dcipher = Cipher.getInstance("DES/CTR/NoPadding");
                dcipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
                String decrypted = decrypt(mensj[0]);
                String decrypted2 = decrypt(mensj[1]);
                String decrypted3 = decrypt(mensj[2]);
                String [] contactos2 = decrypted3.split(",");
                if(usuarios.size()>0){
                   Collection c = usuarios.values();
                   Iterator itr = c.iterator();
                   while (itr.hasNext()) {
                      if(itr.next().equals(decrypted)){
                         existe1=true;
                         break;
                      }
                   }
                   if(!existe1){
```

```
            usuarios.put(usuarios.size(), decrypted);
            passwords.put(decrypted, decrypted2);
            for(int l=0;l<contactos2.length-1;l++){
                lista.add(contactos2[l]);
            }
            contactos.put(decrypted,lista );
            writeMsg("VALIDO");
        }else{
            writeMsg("INVALIDO");
            existe1 = false;
```

```
                    //JOptionPane.showMessageDialog(ventana, "Usuario
invalido");
                    //System.out.println("hola");
                }
            }else{
                usuarios.put(usuarios.size(), decrypted);
                passwords.put(decrypted, decrypted2);
                writeMsg("VALIDO");
            }
        }
        catch (NoSuchAlgorithmException e) {
            System.out.println("No Such Algorithm:" + e.getMessage());
            return;
        }
        catch (NoSuchPaddingException e) {
            System.out.println("No Such Padding:" + e.getMessage());
            return;
        }
        catch (InvalidKeyException e) {
            System.out.println("Invalid Key:" + e.getMessage());
            return;
        } catch (InvalidAlgorithmParameterException ex) {
        Logger.getLogger(Server.class.getName()).log(Level.SEVERE,


        }
null, ex);
    break;
    case ChatMessage.LOGIN:
        try {
            String msg3 = message;
```

```
S           ey = new SecretKeySpec(keyBytes, "DES"); IvParameterSpec
e           ivSpec = new IvParameterSpec(iviBytes);
c
r           dcipher = Cipher.getInstance("DES/CTR/NoPadding");
et          dcipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
K           String decrypted = decrypt(msg3);
e
y           //System.out.println(decrypted);
S           if(usuarios.size()>0){
p
e              for(Map.Entry<String, String> entry : passwords.entrySet()){
c
k                 //System.out.printf("Key : %s and Value: %s %n",
entry.getKey(), entry.getValue());
                  if(entry.getKey().equals(decrypted)){
                     pass = entry.getValue();
```

```
                    existe1 = true;
                    mensajeAleatorio="";
                }
            }
            if(existe1){
                writeMsg("VALIDO2");
                existe1 = false;
                for(int i=0;i<5000;i++){
                    mensajeAleatorio+=rndChar();
                }
                try {
                    SecretKeySpec key2 = new SecretKeySpec(keyBytes,
"DES");
                    IvParameterSpec ivSpec2 = new
IvParameterSpec(iviBytes);
                    ecipher = Cipher.getInstance("DES/CTR/NoPadding");
                    ecipher.init(Cipher.ENCRYPT_MODE, key2, ivSpec2);
                    String encrypted = encrypt(mensajeAleatorio);
                    writeMsg(encrypted);
                } catch (NoSuchAlgorithmException ex) {

Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
                } catch (NoSuchPaddingException ex) {

Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
                } catch (InvalidKeyException ex) {
```



```
Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
```

```
                    } catch (InvalidAlgorithmParameterException ex) {

Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
                    }
                }else{
                    writeMsg("INVALIDO2");
                    //JOptionPane.showMessageDialog(ventana, "Usuario
invalido");
                    //System.out.println("hola");
                }
            }else{
                writeMsg("INVALIDO2");
            }
```

```
                }
                catch (NoSuchAlgorithmException e) {
                    System.out.println("No Such Algorithm:" + e.getMessage());
                    return;
                }
                catch (NoSuchPaddingException e) {
                    System.out.println("No Such Padding:" + e.getMessage());
                    return;
                }
                catch (InvalidKeyException e) {
                    System.out.println("Invalid Key:" + e.getMessage());
                    return;
                } catch (InvalidAlgorithmParameterException ex) {
                Logger.getLogger(Server.class.getName()).log(Level.SEVERE,


                }
null, ex);
            break;
            case ChatMessage.MD5:
                try {
                    String msg4 = message;
                    SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
                    IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
                    dcipher = Cipher.getInstance("DES/CTR/NoPadding");
                    dcipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
                    String decrypted = decrypt(msg4);
                    String preHuella ="";
                    preHuella+=mensajeAleatorio;
                    preHuella+=pass;
                    String huella = getMD5(preHuella);
                    sg.appendRoom("MD5 recibido: " + decrypted + "\n MD5
```

```
creado: " + huella + "\n\n");
                    if(huella.equals(decrypted)){
                        writeMsg("IGUALES");
                    }else{
                        writeMsg("NOIGUALES");
                    }
                }
                catch (NoSuchAlgorithmException e) {
                    System.out.println("No Such Algorithm:" + e.getMessage());
                    return;
                }
```

```
                catch (NoSuchPaddingException e) {
                    System.out.println("No Such Padding:" + e.getMessage());
                    return;
                }
                catch (InvalidKeyException e) {
                    System.out.println("Invalid Key:" + e.getMessage());
                    return;
                } catch (InvalidAlgorithmParameterException ex) {
                Logger.getLogger(Server.class.getName()).log(Level.SEVERE,


                }
null, ex);
            break;



            }
        }

        // remover servidor de la lista de clientes conectados
        remove(id);
        close();
    }
```

La función que desencripta un mensaje es la siguiente:

```
public static String decrypt(String str) {
    try {
        // decodificar con Base64
        byte[] dec = BASE64DecoderStream.decode(str.getBytes());
```

```
            byte[] utf8 = dcipher.doFinal(dec);
            return new String(utf8, "UTF-8");
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
```

Esta función recibe el mensaje encriptado y devuelve una nueva cadena con el mensaje desencriptado.

Con la función rndChar() se generan los mensajes aleatorios.

```
public char rndChar () {
```

```
        int rnd = (int) (Math.random() * 52); // usar un numero aleatorio
        char base = (rnd < 26) ? 'A' : 'a';
        return (char) (base + rnd % 26);
    }
```

Las siguientes dos funciones crean el MD5 y la encriptación respectivamente.

```
public static String getMD5(String input) {
        try {
            MessageDigest md = MessageDigest.getInstance("MD5");
            byte[] messageDigest = md.digest(input.getBytes());
            BigInteger number = new BigInteger(1, messageDigest);
            String hashtext = number.toString(16);
            // lo rellenamos para tener los 32 caracteres
            while (hashtext.length() < 32) {
                hashtext = "0" + hashtext;
            }
            return hashtext;
        }
        catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }

    public static String encrypt(String str) {
        try {
            byte[] utf8 = str.getBytes("UTF-8");
            byte[] enc = ecipher.doFinal(utf8);
```

```
            enc = BASE64EncoderStream.encode(enc);
            return new String(enc);
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
```

ServerGUI

La clase serverGUI es una interfaz gráfica en la que sólo se inicia el sevidor

```
public class ServerGUI extends JFrame implements ActionListener,
WindowListener {
    String[] arguments = new String[]{""};
    private static final long serialVersionUID = 1L;
    private JButton stopStart;
    private JTextArea chat, event;
    private JTextField tPortNumber;
    private Server server;


    ServerGUI(int port) {
        super("Mensajero Exprés - Chat");
        server = null;
        JPanel north = new JPanel();
        north.add(new JLabel("Número de Puerto: "));
        tPortNumber = new JTextField("  " + port);
        north.add(tPortNumber);
        stopStart = new JButton("Iniciar");
        stopStart.addActionListener(this);
        north.add(stopStart);
        add(north, BorderLayout.NORTH);


        JPanel center = new JPanel(new GridLayout(2,1));
        chat = new JTextArea(80,80);
        chat.setEditable(false);
        appendRoom("Chat.\n");
```

```
center.add(new JScrollPane(chat));

event = new JTextArea(80,80);
event.setEditable(false);
appendEvent("Eventos.\n");
center.add(new JScrollPane(event));
add(center);


// para cuando se cierra el servidor
addWindowListener(this);
setSize(400, 600);
setLocationRelativeTo(null);
setVisible(true);
```

```
}


void appendRoom(String str) { chat.append(str);
        chat.setCaretPosition(chat.getText().length() - 1);

}
void appendEvent(String str) { event.append(str);
        event.setCaretPosition(chat.getText().length() - 1);



}


public void actionPerformed(ActionEvent e) {

        // si está activo, paramos
        if(server != null) {

                server.stop();

                server = null;

                tPortNumber.setEditable(true);
                stopStart.setText("Iniciar");
                return;

        }
// iniciar el servidor
        int port;

        try {

                port = Integer.parseInt(tPortNumber.getText().trim());

        }

        catch(Exception er) {
                appendEvent("Puerto inválido");
                return;

        }

        // crear servidor

        server = new Server(port, this);
```

```
        // and start it as a thread
        new ServerRunning().start();
        stopStart.setText("Detener");
        tPortNumber.setEditable(false);
}


public static void main(String[] arg) {
   // start server default port 1500
```

```
    new ServerGUI(1500);
}

/*
 * Terminar el servidor si se elije el boton X
 */
public void windowClosing(WindowEvent e) {
        // si existe el servidor
        if(server != null) {
                try {
                        server.stop(); // pedirle al servidor que cierre la conexion
                }
                catch(Exception eClose) {
                }
                server = null;
        }
        // cierra la ventana
        dispose();
        System.exit(0);
}
// ignorar los otros metodos WindowListeners public
void windowClosed(WindowEvent e) {} public void
windowOpened(WindowEvent e) {} public void
windowIconified(WindowEvent e) {} public void
windowDeiconified(WindowEvent e) {} public void
windowActivated(WindowEvent e) {} public void
windowDeactivated(WindowEvent e) {}

/*
 * Hilo del servidor
 */
```

```
class ServerRunning extends Thread {
    public void run() {
        server.start();
        // el servidor falló
        stopStart.setText("Iniciar");
        tPortNumber.setEditable(true);
        appendEvent("Servidor cerrado\n");
        server = null;
    }
}
```

}

Cliente

La clase cliente utiliza sockets para conectarse al servidor y está en constante comunicación con el servidor.

```
public class Client  {

   private static Cipher dcipher;
   private byte[] keyBytes;
   private byte[] iviBytes;

   // para I/O

   private ObjectInputStream sInput;          // leer desde el socket
   private ObjectOutputStream sOutput;        // esribir en el socket

   private Socket socket;

   // si se utiliza GUI o no
   private ClientGUI cg;

   private Chat_MD5 ch;

   // servidor puerto y nombre de usuario
   private String server, username;

   private int port;

   /*

    *  Constructor llamado por console

    *  server: la direccion del servidor

    *  port: el numero de puerto

    *  username: nombre de usuario

    */
```

```
Client(String server, int port, String username) {
    // constructor en caso de que no se use GUI
    this(server, port, username, null);
    keyBytes = "12345678".getBytes();
    iviBytes = "input123".getBytes();
}
/*
* Constructor llamado cuando se usa la GUI
* en modo consola, el parametro GUI es null
*/
Client(String server, int port, String username, ClientGUI cg) {
    keyBytes = "12345678".getBytes();
```

```
        iviBytes = "input123".getBytes();
        this.server = server;
        this.port = port;
        this.username = username;
        // saber si estamos en modo GUI o no
        this.cg = cg;
    }

    /*
    * Iniciar el dialogo
    */
    public boolean start() {
        // intentar la conexion con el servidor
        try {
            socket = new Socket(server, port);
        }
        catch(Exception ec) {
            display("Error al conectarse al servidor:" + ec);
            return false;
        }

        String msg = "Conexion aceptada " + socket.getInetAddress() + ":" +
socket.getPort();
        display(msg);
        /* Crear los Streams */
        try
        {
            sInput  = new ObjectInputStream(socket.getInputStream());
            sOutput = new ObjectOutputStream(socket.getOutputStream());
        }
```

```
catch (IOException eIO) {
    display("Error creando nuevos Input/output Streams: " + eIO);
    return false;
}
// crear el hilo para escuchar desde el servior
new ListenFromServer().start();
try
{
    sOutput.writeObject(username);
}
catch (IOException eIO) {
```

```
            display("Error al iniciar : " + eIO);
            disconnect();
            return false;
        }
        // la conexión funciona
        return true;
    }

    /*
    * Enviar mensaje a la consola o a la GUI
    */
    private void display(String msg) {
    if(cg == null)
        System.out.println(msg);
    else
        cg.append(msg + "\n");
    }


    /*
    * Enviar mensaje al servidor
    */
    void sendMessage(ChatMessage msg) {
        try {
            sOutput.writeObject(msg);
        }
        catch(IOException e) {
            display("Error al escribir al servidor: " + e);
        }
    }
```

```
/*
* Cuando algo sale mal
* Cerrar los streams y desconectar
*/
private void disconnect() {
    try {
        if(sInput != null) sInput.close();
    }
    catch(Exception e) {}
    try {
        if(sOutput != null) sOutput.close();
```

```
    }
    catch(Exception e) {}
    try{
        if(socket != null) socket.close();
    }
    catch(Exception e) {}
    // informar a la GUI
    if(cg != null)
        cg.connectionFailed();


}
/*
* Para iniciar el cliente desde consola
* > java Client
* > java Client username
* > java Client username portNumber
* > java Client username portNumber serverAddress
*
* Si no se especifica el puerto, se usa el 1500
* Si la direccion de servidor no se especifica se usa "localhost"
* Si no se especifica el nombre de usuario se usa "loclhost"
* > java Client
* equivale a
* > java Client Anonymous 1500 localhost
*
*
* En consola, cuando ocurre un error, se detiene la aplicación
* En cambio con la GUI, se informa del error
```

```
*/
public static void main(String[] args) {
    // valores por defecto
    int portNumber = 1500;
    String serverAddress = "localhost";
    String userName = "Anonimo";
    // contamos el numero de parametros
    switch(args.length) {
        // > javac Client username portNumber serverAddr
        case 3:
            serverAddress = args[2];
            // > javac Client username portNumber
        case 2:
```

```
            try {
                portNumber = Integer.parseInt(args[1]);
            }
            catch(Exception e) {
                System.out.println("Numero de puerto invalido.");
                System.out.println("Debe ser is: > java Client [nombre usuario]
[puerto] [direccion servidor]");
                return;
            }
            // > javac Client username
        case 1:
            userName = args[0];
            // > java Client
        case 0:
        break;
        // numero invalido de argumentos
        default:
            System.out.println("Usage is: > java Client [nombre usuario] [puerto]
{direccion servidor]");
            return;
    }
    // crea el objeto cliente
    Client client = new Client(serverAddress, portNumber, userName);
    // probar si sirve la conexion
    if(!client.start())
    return;
    // espera mensajes del usuario
    Scanner scan = new Scanner(System.in);
    // esperar mensajes
    while(true) {
        System.out.print("> ");
```

```
// leer mensaje
String msg = scan.nextLine();
// salir
if(msg.equalsIgnoreCase("SALIR")) {
   client.sendMessage(new ChatMessage(ChatMessage.LOGOUT, ""));
   // desconectar
   break;
}
// mensaje WhoIsIn
else if(msg.equalsIgnoreCase("VER")) {
```

```
            client.sendMessage(new ChatMessage(ChatMessage.WHOISIN, ""));
        }
        else {                          // un mensaje cualquiera
            client.sendMessage(new ChatMessage(ChatMessage.MESSAGE,
msg));
        }
    }
    // desconectar
    client.disconnect();
}

/*
* una clase que espera mensajes del servidor
*/
class ListenFromServer extends Thread {
    public void run() {
        while(true) {
            try {
                String msg = (String) sInput.readObject();
                if(cg == null) {
                    System.out.println(msg);
                    System.out.print("> ");
                }
                else {
                    SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
                    IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
                    dcipher = Cipher.getInstance("DES/CTR/NoPadding");
                    dcipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
                    String decrypted = decrypt(msg);
```

```
            cg.append(decrypted);
        }
    }
    catch(IOException e) {
        display("El servidor ha cerrado la conexion: " + e);
        if(cg != null)
            cg.connectionFailed();
        break;
    }
    catch(ClassNotFoundException e2) {
    } catch (NoSuchAlgorithmException ex) {
```

```
                Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null,
ex);
            } catch (NoSuchPaddingException ex) {
                Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null,
ex);
            } catch (InvalidKeyException ex) {
                Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null,
ex);
            } catch (InvalidAlgorithmParameterException ex) {
                Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null,
            }
        }
    }
    public static String decrypt(String str) {
        try {
            byte[] dec = BASE64DecoderStream.decode(str.getBytes());
            byte[] utf8 = dcipher.doFinal(dec);
            return new String(utf8, "UTF-8");
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

Esta clase se encarga de realizar las conexiones y el chat en general, en la interfaz el usuario puede elegir un puerto al que conectarse, un nombre de usuario, escribir los mensajes, entrar y salir del chat y ver qué usuarios están conectados. La clase también tiene la función de encriptar y desencriptar mensajes.

```
public class ClientGUI extends JFrame implements ActionListener {
    private static Cipher ecipher;


    private String encrypted, encrypted2;
```

```
private byte[] keyBytes;
private byte[] iviBytes;
private String username;


private static final long serialVersionUID = 1L;

// escribir usuario y luego escribir mensaje
private JLabel label;

// para el usuario y luegos los mensajes

private JTextField tf;

// almacena la direccion del servidor y el puerto
private JTextField tfServer, tfPort;

// para iniciar/cerrar sesion y ver contactos;

private JButton login, logout, whoIsIn;

// para el chat

private JTextArea ta;

// conexion

private boolean connected;

// objeto cliente
private Client client;

// puerto por defecto

private int defaultPort;

private String defaultHost;


// Constructor que recibe un socket

ClientGUI(String host, int port) {

    super("Chat Cliente");

    keyBytes = "12345678".getBytes();

    iviBytes = "input123".getBytes();

    defaultPort = port;

    defaultHost = host;
```

```
JPanel northPanel = new JPanel(new GridLayout(3,1));
JPanel serverAndPort = new JPanel(new GridLayout(1,5, 1, 3));
tfServer = new JTextField(host);
tfPort = new JTextField("" + port);
tfPort.setHorizontalAlignment(SwingConstants.RIGHT);
serverAndPort.add(new JLabel("Direccion IP:  "));
serverAndPort.add(tfServer);
serverAndPort.add(new JLabel("Número de Puerto:  "));
serverAndPort.add(tfPort);
serverAndPort.add(new JLabel(""));
northPanel.add(serverAndPort);
```

```
        label = new JLabel("Escribe un nick debajo", SwingConstants.CENTER);
        northPanel.add(label);
        tf = new JTextField("Anonimo");
        tf.setBackground(Color.WHITE);
        northPanel.add(tf);
        add(northPanel, BorderLayout.NORTH);
        ta = new JTextArea("Bienvenido al chat\n", 80, 80);
        JPanel centerPanel = new JPanel(new GridLayout(1,1));
        centerPanel.add(new JScrollPane(ta));
        ta.setEditable(false);
        add(centerPanel, BorderLayout.CENTER);
        login = new JButton("ENTRAR");
        login.addActionListener(this);
        logout = new JButton("SALIR");
        logout.addActionListener(this);
        logout.setEnabled(false);           // tienes que inicair sesion antes de cerrar
sesion
        whoIsIn = new JButton("USUARIOS EN LÍNEA");
        whoIsIn.addActionListener(this);
        whoIsIn.setEnabled(false);          // debes iniciar sesion antes de ver tus
contactos
        JPanel southPanel = new JPanel();
        southPanel.add(login);
        southPanel.add(logout);
        southPanel.add(whoIsIn);
        add(southPanel, BorderLayout.SOUTH);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(600, 600);
        setLocationRelativeTo(null);
        setVisible(true);
        tf.requestFocus();
    }
```

```
// escribir texto en la ventana
void append(String str) {
    ta.append(str);
    ta.setCaretPosition(ta.getText().length() - 1);
}
// si la conexion falla
// reestablecemos los botones
void connectionFailed() {
    login.setEnabled(true);
```

```
    logout.setEnabled(false);
    whoIsIn.setEnabled(false);
    label.setText("Escribe tu nick debajo");
    tf.setText("Anonimo");
    tfPort.setText("" + defaultPort);
    tfServer.setText(defaultHost);
    tfServer.setEditable(false);
    tfPort.setEditable(false);
    // una vez que se ingresa el usuario ya no se vuelve a repetir la accion
    tf.removeActionListener(this);
    connected = false;
    dispose();
}

/*
* Button or JTextField clicked
*/
public void actionPerformed(ActionEvent e) {
    Object o = e.getSource();
    // el boton de salir
    if(o == logout) {
        client.sendMessage(new ChatMessage(ChatMessage.LOGOUT, ""));
        return;
    }
    // para ver los contactos
    if(o == whoIsIn) {
        client.sendMessage(new ChatMessage(ChatMessage.WHOISIN, ""));
        return;
    }
    // es un mensaje
    if(connected) {
```

```
        try {
            SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
            IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
            ecipher = Cipher.getInstance("DES/CTR/NoPadding");
            ecipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
            encrypted = encrypt(tf.getText());
            client.sendMessage(new ChatMessage(ChatMessage.MESSAGE,
encrypted));
            tf.setText("");
            return;
```

```
        }
        catch (NoSuchAlgorithmException a) {
            System.out.println("No Such Algorithm:" + a.getMessage());
            return;
        }
        catch (NoSuchPaddingException a) {
            System.out.println("No Such Padding:" + a.getMessage());
            return;
        }
        catch (InvalidKeyException a) {
            System.out.println("Invalid Key:" + a.getMessage());
            return;
        } catch (InvalidAlgorithmParameterException ex) {
            Logger.getLogger(Chat_MD5.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }



    if(o == login) {
        username = tf.getText().trim();
        if(username.length() == 0)
        return;
        String server = tfServer.getText().trim();
        if(server.length() == 0)
        return;
        String portNumber = tfPort.getText().trim();
        if(portNumber.length() == 0)
        return;
        int port = 0;
```

```
try {
    port = Integer.parseInt(portNumber);
}
catch(Exception en) {
    return;
}
client = new Client(server, port, username, this);
if(!client.start())
return;
tf.setText("");
label.setText("Escribe tu mensaje debajo");
```

```
            connected = true;
            login.setEnabled(false);
            logout.setEnabled(true);
            whoIsIn.setEnabled(true);
            // evitar que se escriba en los campos servidor y puerto
            tfServer.setEditable(false);
            tfPort.setEditable(false);
            // Action listener para cuando se escribe un mensaje
            tf.addActionListener(this);
        }

    }

    public static void main(String[] args) {
        new ClientGUI("localhost", 1500);
    }

    public static String encrypt(String str) {
        try {           // encode the string into a sequence of bytes using the named
charset
            // storing the result into a new byte array.
            byte[] utf8 = str.getBytes("UTF-8");
            byte[] enc = ecipher.doFinal(utf8);
            enc = BASE64EncoderStream.encode(enc);
            return new String(enc);
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
```

```
}
```

Principal

La clase principal tiene la función de iniciar el programa, tiene 2 botones que sirven para iniciar el servidor y para el otro para iniciar los clientes.

```
public class Principal {
    private JFrame ventana;
```

```
    private JButton servidor;
    private JButton cliente;
    private JLabel bienvenido;

    String[] argumentos = new String[]{""};


    public Principal(){

        inicializar();

    }


    public void inicializar(){

        ventana = new JFrame("Principal");

        ventana.setSize(500, 300);

        ventana.setLocationRelativeTo(null);
        ventana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ventana.setVisible(true);
        ventana.getContentPane().setBackground(new Color(204,229,255));
        ventana.setLayout(null);


        bienvenido = new JLabel();
        ventana.add(bienvenido);
        bienvenido.setText("<html><span style='font-
size:18px'>Bienvenido</span></html>");

        bienvenido.setBounds(190, 20, 220, 80);


        servidor = new JButton(); ventana.add(servidor);
        servidor.setBackground(new Color(102,255,102));
        servidor.setForeground(Color.BLACK);
        servidor.setText("Iniciar Servidor");
        servidor.setBounds(195, 110, 125, 50);
```

```
cliente = new JButton();
ventana.add(cliente);
cliente.setBackground(new Color(102,255,102));
cliente.setForeground(Color.BLACK);
cliente.setText("Iniciar Cliente");
cliente.setBounds(195, 170, 125, 50);


servidor.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        servidorMouseClicked(evt);
```

```
        }
    });

    cliente.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            clienteMouseClicked(evt);
        }
    });
}

private void servidorMouseClicked(java.awt.event.MouseEvent evt) {
    ServerGUI.main(argumentos);
}

private void clienteMouseClicked(java.awt.event.MouseEvent evt) {
    Chat_MD5.main(argumentos);
}

public static void main(String[] args){
    new Principal();
}

}
```

Esta clase tiene como objetivo hacer el registro de los clientes, así como permitirles iniciar sesión. Es una ventana que contiene 3 botones principales para ingresar la dirección del servidor al cual quieren conectarse, para hacer el registro y para iniciar sesión.

El botón de registro recibe el nombre de usuario, contraseña y los contactos del usuario. Todos los datos se almacenan en hashmaps y Arraylists. El botón de iniciar sesión comprueba que el usuario que quiere iniciar sesión esté registrado y que la contraseña sea la correcta. Para hacer esto, el usuario manda un mensaje de tipo LOGIN que le indica al servidor que debe generar un mensaje aleatorio y así el usuario pueda generar su huella MD5. Por su parte el servidor genera la huella propia y la compara con la que recibe del cliente, si son iguales, el usuario inicia sesión.

```
public class Chat_MD5 extends JFrame implements Serializable{
   String[] argumentos = new String[]{""};
```

```
private final static long serialVersionUID = 1;

private JFrame ventana;
private JButton registro;
private JButton iniciarSesion;
private JButton conexion;
private JLabel bienvenido;
private String contactos;
private String registroUsuario;
private String registroCon;
private String loginUsuario;
private String loginCon;
private JFrame frame;

private JPanel panel; private
JLabel userLabel; private
JTextField userText; private
JLabel passwordLabel;

private JPasswordField passwordText;

private JLabel contactLabel;
private JTextField contactText;
private JButton aceptar;

private JFrame frame2;
private JPanel panel2;
private JLabel userLabel2;

private JTextField userText2;

private JLabel passwordLabel2;

private JPasswordField passwordText2;

private JButton aceptar2;

private JFrame frame3;

private JPanel panel3; private
JLabel userLabel3; private
JTextField userText3; private
JButton aceptar3; private
boolean existe1 = false;

String[] arguments = new String[] {""};

private static Cipher ecipher, dcipher;
```

```
private String encrypted, encrypted2, encrypted3;


private byte[] keyBytes;
private byte[] iviBytes;
```

```
// for I/O
private ObjectInputStream sInput;          // to read from the socket
private ObjectOutputStream sOutput;        // to write on the socket
private Socket socket;

private String server;

private int port;



public Chat_MD5(){

   inicializar();

}


public void inicializar(){

   keyBytes = "12345678".getBytes();

   iviBytes = "input123".getBytes();

   port = 1500;

   server = "localhost";

   ventana = new JFrame("Ventana");
   ventana.setSize(500, 600);
   ventana.setLocationRelativeTo(null);

   ventana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   ventana.setVisible(true);
   ventana.getContentPane().setBackground(new Color(204,229,255));

   ventana.setLayout(null);



   bienvenido = new JLabel();

   ventana.add(bienvenido);

   bienvenido.setText("<html><span style='font-size:18px'>Mensajero
Exprés</span></html>");

   bienvenido.setBounds(148, 80, 220, 80);
```

```
conexion = new JButton(); ventana.add(conexion);
conexion.setBackground(new Color(102,255,102));
conexion.setForeground(Color.BLACK);
conexion.setText("Conectarse");
conexion.setBounds(195, 160, 115, 50);


registro = new JButton();

ventana.add(registro);

registro.setBackground(new Color(102,255,102));
```

```
registro.setForeground(Color.BLACK);
registro.setText("Registrarse");
registro.setBounds(195, 230, 115, 50);


iniciarSesion = new JButton();
ventana.add(iniciarSesion);
iniciarSesion.setBackground(new Color(102,255,102));
iniciarSesion.setForeground(Color.BLACK);
iniciarSesion.setText("Iniciar Sesion");
iniciarSesion.setBounds(195, 300, 115, 50);




registro.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        registroMouseClicked(evt);
    }
});


iniciarSesion.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        iniciarSesionMouseClicked(evt);
    }
});


conexion.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        conexionMouseClicked(evt);
    }
```

```
    });

    pack();
}

private void conexionMouseClicked(java.awt.event.MouseEvent evt) {
    ventana.setEnabled(false);
    frame3 = new JFrame("Conectarse");
    frame3.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    frame3.setSize(300, 150);
    frame3.setLocationRelativeTo(null);
    panel3 = new JPanel();
```

```
    frame3.add(panel3);
    panel3.setLayout(null);


    userLabel3 = new JLabel("IP");
    userLabel3.setBounds(10, 10, 80, 25);
    panel3.add(userLabel3);


    userText3 = new JTextField(20);
    userText3.setBounds(100, 10, 160, 25);
    panel3.add(userText3);


    aceptar3 = new JButton("Aceptar");
    aceptar3.setBounds(100, 80, 80, 25);
    panel3.add(aceptar3);


    aceptar3.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            aceptar3MouseClicked(evt);
        }
    });


    frame3.addWindowListener(new java.awt.event.WindowAdapter() {
    @Override
    public void windowClosing(java.awt.event.WindowEvent windowEvent) {
        ventana.setEnabled(true);
    }
    });
    frame3.setVisible(true);
}
```

```
private void aceptar3MouseClicked(java.awt.event.MouseEvent evt) {

   if(userText3.getText().trim().length()==0){
      JOptionPane.showMessageDialog(frame, "Datos incompletos");

   }else{

      System.out.println(server);

      String servidor = userText3.getText().trim();
      System.out.println(servidor);

      conectar(servidor);

      ventana.setEnabled(true);

      frame3.dispose();

   }
```

```
}

private void registroMouseClicked(java.awt.event.MouseEvent evt) {
    ventana.setEnabled(false);
    frame = new JFrame("Registro");
    frame.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    frame.setSize(300, 200);
    frame.setLocationRelativeTo(null);
    panel = new JPanel();
    frame.add(panel);
    panel.setLayout(null);

    userLabel = new JLabel("Usuario");
    userLabel.setBounds(10, 10, 80, 25);
    panel.add(userLabel);

    userText = new JTextField(20);
    userText.setBounds(100, 10, 160, 25);
    panel.add(userText);

    passwordLabel = new JLabel("Contraseña");
    passwordLabel.setBounds(10, 40, 80, 25);
    panel.add(passwordLabel);

    passwordText = new JPasswordField(20);
    passwordText.setBounds(100, 40, 160, 25);
    panel.add(passwordText);

    contactLabel = new JLabel("Contactos");
    contactLabel.setBounds(10, 70, 80, 25);
    panel.add(contactLabel);
```

```
contactText = new JTextField(20);
contactText.setBounds(100, 70, 160, 25);
panel.add(contactText);


aceptar = new JButton("Aceptar");
aceptar.setBounds(100, 110, 80, 25);
panel.add(aceptar);
```

```
        aceptar.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            aceptarMouseClicked(evt);
        }
        });

        frame.addWindowListener(new java.awt.event.WindowAdapter() {
        @Override
        public void windowClosing(java.awt.event.WindowEvent windowEvent) {
            ventana.setEnabled(true);
        }
        });

        frame.setVisible(true);
    }

    private void aceptarMouseClicked(java.awt.event.MouseEvent evt) {
        if(userText.getText().trim().length()==0 ||
passwordText.getText().trim().length()==0 ||
contactText.getText().trim().length() == 0){
            JOptionPane.showMessageDialog(frame, "Datos incompletos");
        }else{
            registroUsuario = userText.getText();
            //System.out.println(registroUsuario);
            registroCon = passwordText.getText();
            //System.out.println(registroCon);
            contactos = contactText.getText();
            ventana.setEnabled(true);
            frame.dispose();
```

```
try {
    SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
    IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
    // generate secret key using DES algorithm
    ecipher = Cipher.getInstance("DES/CTR/NoPadding");
    // initialize the ciphers with the given key
    ecipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
    encrypted = encrypt(registroUsuario);
    encrypted2 = encrypt(registroCon);
    encrypted3 = encrypt(contactos);
```

```
            sendMessage(new ChatMessage(ChatMessage.REGISTRY,
encrypted+","+encrypted2+","+encrypted3));
        }
        catch (NoSuchAlgorithmException e) {
            System.out.println("No Such Algorithm:" + e.getMessage());
            return;
        }
        catch (NoSuchPaddingException e) {
            System.out.println("No Such Padding:" + e.getMessage());
            return;
        }
        catch (InvalidKeyException e) {
            System.out.println("Invalid Key:" + e.getMessage());
            return;
        } catch (InvalidAlgorithmParameterException ex) {
            Logger.getLogger(Chat_MD5.class.getName()).log(Level.SEVERE,
null, ex);
        }

    }
}

public static String encrypt(String str) {
    try {
        // encode the string into a sequence of bytes using the named charset
        // storing the result into a new byte array.
        byte[] utf8 = str.getBytes("UTF-8");
        byte[] enc = ecipher.doFinal(utf8);
        // encode to base64
        enc = BASE64EncoderStream.encode(enc);
        return new String(enc);
```

```
    }
    catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}

private void iniciarSesionMouseClicked(java.awt.event.MouseEvent evt) {
    ventana.setEnabled(false);
    frame2 = new JFrame("Login");
```

```
frame2.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
frame2.setSize(300, 150);
frame2.setLocationRelativeTo(null);

panel2 = new JPanel();
frame2.add(panel2);
panel2.setLayout(null);


userLabel2 = new JLabel("Usuario");
userLabel2.setBounds(10, 10, 80, 25);
panel2.add(userLabel2);


userText2 = new JTextField(20);
userText2.setBounds(100, 10, 160, 25);
panel2.add(userText2);


passwordLabel2 = new JLabel("Contraseña");
passwordLabel2.setBounds(10, 40, 80, 25);
panel2.add(passwordLabel2);


passwordText2 = new JPasswordField(20);
passwordText2.setBounds(100, 40, 160, 25);
panel2.add(passwordText2);


aceptar2 = new JButton("Aceptar");
aceptar2.setBounds(100, 80, 80, 25);
panel2.add(aceptar2);


aceptar2.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        aceptar2MouseClicked(evt);
    }
```

```
    });


    frame2.addWindowListener(new java.awt.event.WindowAdapter() {
    @Override
    public void windowClosing(java.awt.event.WindowEvent windowEvent) {
        ventana.setEnabled(true);
    }
    });
    frame2.setVisible(true);
}
```

```
    private void aceptar2MouseClicked(java.awt.event.MouseEvent evt) {
        if(userText2.getText().trim().length()==0 ||
passwordText2.getText().trim().length()==0){
            JOptionPane.showMessageDialog(frame, "Datos incompletos");
        }else{
            loginUsuario = userText2.getText();
            //System.out.println(loginUsuario);
            loginCon = passwordText2.getText();
            //System.out.println(loginCon);
            ventana.setEnabled(true);
            frame2.dispose();
            //ventana.dispose();
            try {
                SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
                IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
                // generate secret key using DES algorithm
                ecipher = Cipher.getInstance("DES/CTR/NoPadding");
                // initialize the ciphers with the given key
                ecipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
                encrypted = encrypt(loginUsuario);
                encrypted2 = encrypt(loginCon);
                sendMessage(new ChatMessage(ChatMessage.LOGIN, encrypted));
            }
            catch (NoSuchAlgorithmException e) {
                System.out.println("No Such Algorithm:" + e.getMessage());
                return;
            }
            catch (NoSuchPaddingException e) {
                System.out.println("No Such Padding:" + e.getMessage());
```

```
            return;
        }
        catch (InvalidKeyException e) {
            System.out.println("Invalid Key:" + e.getMessage());
            return;
        } catch (InvalidAlgorithmParameterException ex) {
            Logger.getLogger(Chat_MD5.class.getName()).log(Level.SEVERE,
null, ex);
        }
        //ClientGUI.main(arguments);
    }
```

```
}
/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    // TODO code application logic here
    Chat_MD5 chat = new Chat_MD5();
}


public void conectar(String servidor){
    // try to connect to the server
    try {
      socket = new Socket(servidor, port);
    }
    // if it failed not much I can so
    catch(Exception ec) {
        ec.printStackTrace();
    }
    try
    {
        sInput  = new ObjectInputStream(socket.getInputStream());
        sOutput = new ObjectOutputStream(socket.getOutputStream());
        new Chat_MD5.ListenFromServer().start();
    }
    catch (IOException eIO) {
        eIO.printStackTrace();
    }
```

```
    try
    {
        sOutput.writeObject("SYSTEM");
    }
    catch (IOException eIO) {
        eIO.printStackTrace();
    }
}

void sendMessage(ChatMessage msg) {
    try {
        sOutput.writeObject(msg);
```

```
        }
        catch(IOException e) {
            e.printStackTrace();
        }
    }


    class ListenFromServer extends Thread {
        public void run() {
            while(true) {
                try {
                    String msg = (String) sInput.readObject();
                    if(msg.equals("INVALIDO")){
                        JOptionPane.showMessageDialog(ventana, "Usuario no valido");
                    }
                    if(msg.equals("VALIDO")){
                        JOptionPane.showMessageDialog(ventana, "Usuario registrado");
                    }
                    if(msg.equals("INVALIDO2")){
                        JOptionPane.showMessageDialog(ventana, "Usuario y/o
contraseña incorrectos");
                    }
                    if(msg.equals("VALIDO2")){



                    }
                    if(msg.length() > 1000){
                        String preHuella = "";
                        SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
                        IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
                        //key = KeyGenerator.getInstance("DES").generateKey();
```

```
dcipher = Cipher.getInstance("DES/CTR/NoPadding");

// initialize the ciphers with the given key
dcipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
String decrypted = decrypt(msg);
preHuella+=decrypted;

preHuella+=loginCon;

String huella = getMD5(preHuella);

try {

    SecretKeySpec key2 = new SecretKeySpec(keyBytes, "DES");
    IvParameterSpec ivSpec2 = new IvParameterSpec(iviBytes);

    // generate secret key using DES algorithm
```

```
                ecipher = Cipher.getInstance("DES/CTR/NoPadding");
                // initialize the ciphers with the given key
                ecipher.init(Cipher.ENCRYPT_MODE, key2, ivSpec2);
                encrypted = encrypt(huella);
                sendMessage(new ChatMessage(ChatMessage.MD5,

encrypted));    }

                catch (NoSuchAlgorithmException e) {
                    System.out.println("No Such Algorithm:" + e.getMessage());
                    return;
                }
                catch (NoSuchPaddingException e) {
                    System.out.println("No Such Padding:" + e.getMessage());
                    return;
                }
                catch (InvalidKeyException e) {
                    System.out.println("Invalid Key:" + e.getMessage());
                    return;
                } catch (InvalidAlgorithmParameterException ex) {

Logger.getLogger(Chat_MD5.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
            if(msg.equals("IGUALES")){
                JOptionPane.showMessageDialog(ventana, "Sesion iniciada");
                ClientGUI.main(argumentos);
                ventana.dispose();
            }
            if(msg.equals("NOIGUALES")){
                JOptionPane.showMessageDialog(ventana, "Usuario y/o
```

```
contraseña incorrectos");
                }
            }
            catch(IOException e) {
                e.printStackTrace();
            }
            // can't happen with a String object but need the catch anyhow
            catch(ClassNotFoundException e2) {
            } catch (InvalidKeyException ex) {
                Logger.getLogger(Chat_MD5.class.getName()).log(Level.SEVERE,


null, ex);
```

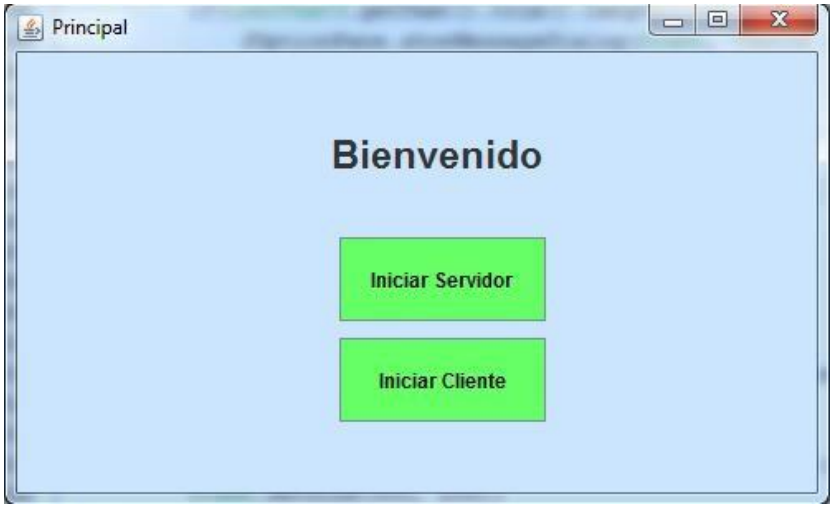
```
            } catch (InvalidAlgorithmParameterException ex) {
                Logger.getLogger(Chat_MD5.class.getName()).log(Level.SEVERE,
null,  ex);

            } catch (NoSuchAlgorithmException ex) {
null,  ex);     Logger.getLogger(Chat_MD5.class.getName()).log(Level.SEVERE,


null, ex);
            } catch (NoSuchPaddingException ex) {
                Logger.getLogger(Chat_MD5.class.getName()).log(Level.SEVERE,
            }
        }
    }


    public static String getMD5(String input) {
        try {
            MessageDigest md = MessageDigest.getInstance("MD5");
            byte[] messageDigest = md.digest(input.getBytes());
            BigInteger number = new BigInteger(1, messageDigest);
            String hashtext = number.toString(16);
            // Now we need to zero pad it if you actually want the full 32 chars.
            while (hashtext.length() < 32) {
                hashtext = "0" + hashtext;
            }
            return hashtext;
        }
        catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }
```

```
public static String decrypt(String str) {
    try {
        // decode with base64 to get bytes
        byte[] dec = BASE64DecoderStream.decode(str.getBytes());
        byte[] utf8 = dcipher.doFinal(dec);
        // create new string based on the specified charset
        return new String(utf8, "UTF-8");
    }
    catch (Exception e) {
```

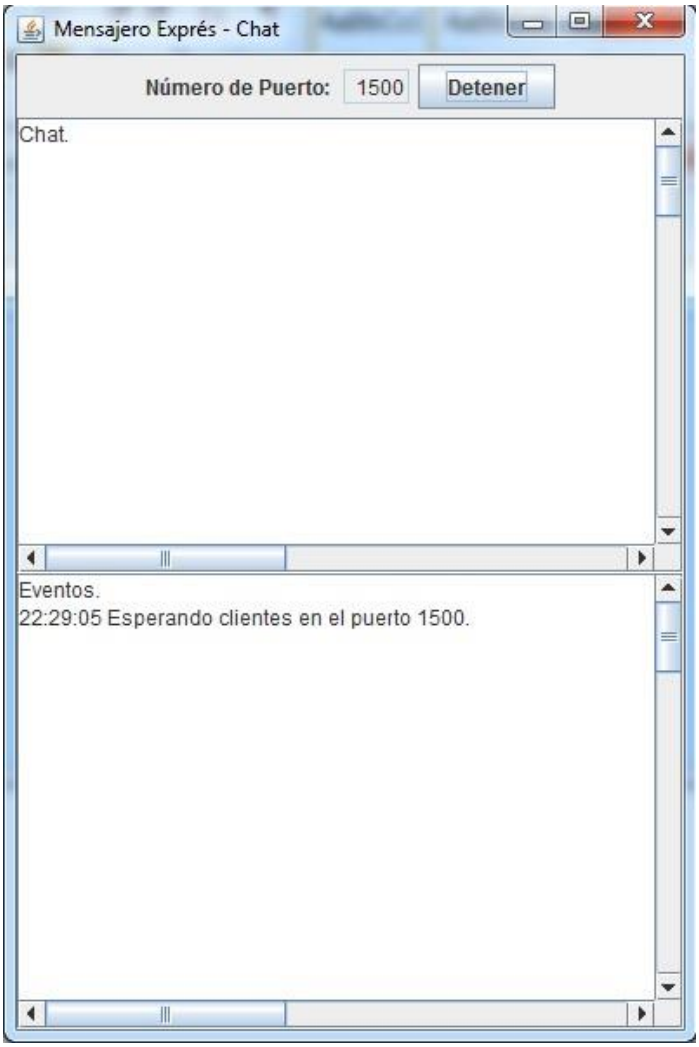
```
            e.printStackTrace();
        }
        return null;
    }
}
```

Resultados

En general se obtuvieron buenos resultados aunque no los resultados óptimos que se querían obtener desde el principio. Lo que falló fue la especificación de que los usuarios debían elegir a uno de sus contactos para platicar, aunque si puede elegir a cualquier otro que esté conectado.



Ventana principal.

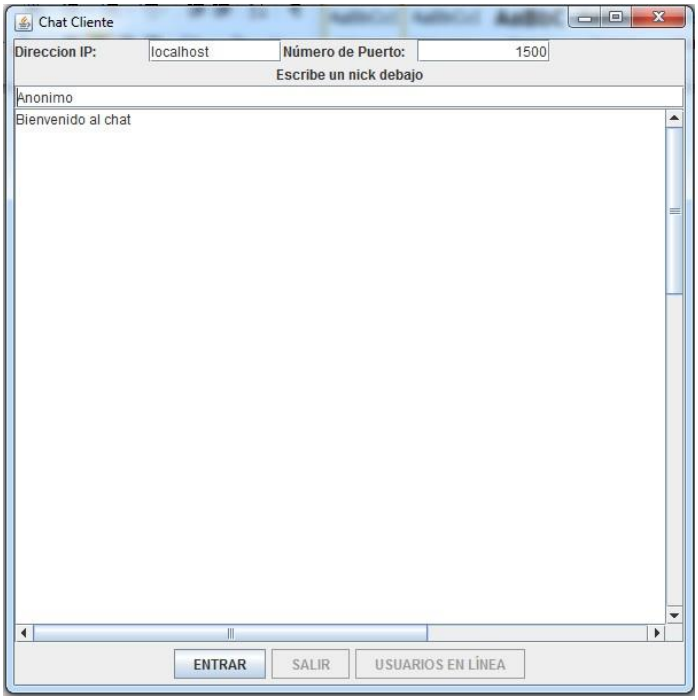


Iniciando el servidor.

Ventana del cliente.

Ventana en la que deberemos registrarnos.

Ventana para iniciar sesión.



Ventana del chat.

Conclusión

En un Sistema de Comunicación de Datos, es de vital importancia asegurar que la Información viaje segura, manteniendo su autenticidad, integridad, confidencialidad y el no repudio de la misma entre otros aspectos.

Estas características solo se pueden asegurar utilizando las Técnicas de Firma Digital Encriptada y la Encriptación de Datos, En este trabajo se usó la Encriptación de Datos.

Bibliografía

-A.S. Tanenbaum, "*Redes de Computadoras*", 4° Edición, Pearson Education, México, 2003.

-W. Stallings, "*Comunicaciones y Redes de Computadoras*", 7° Edición, Pearson Education, Madrid, 2004.

PROYECTO/EXAMEN

Modelos de Redes

DOCUMENTOTÉCNICO

En este documento se enunciaran los procesos, problemas y marcos al realizar esta práctica.

Luis Conde Rodríguez

Otoño 2014

Introducción

¿Qué es la latencia?

El término “latencia” es un término general que significa cualquier tipo de retraso en la ejecución de las operaciones seleccionadas, especialmente en los sistemas de información en sentido amplio. Entrando un poco más en los detalles, es decir pasando a los temas de redes informáticas, la “latencia” se refiere a cualquier tipo de periodos de inactividad y retrasos producidos durante la transmisión de paquetes de datos y el procesamiento de datos recibidos o preparados para ser enviados. A menudo se puede encontrar también la noción de 'lag' ('retraso' en inglés), muy popular en los círculos de aficionados a los juegos (juegos en línea a través de Internet) y las personas que utilizan el servicio de IRC intensivamente. Igualmente popular en la jerga informática es el término 'ping' que frecuentemente se usa en vez de la palabra 'lag', aunque no es completamente correcto, ya que el ping es una de las herramientas básicas para la estimación de retrasos de la transmisión en la red.

¿Cómo se calcula?

La latencia es calculada por la siguiente formula:

Latencia = Tiempo de Propagación + Tiempo de Transmisión + Tiempo de Cola.

Donde:

Tiempo de Propagación = Distancia a Recorrer/Velocidad de la luz

Tiempo de Transmisión = Tamaño del paquete/tasa de transferencia teórica.

Desarrollo

La idea de este proyecto es crear un programa que realice el cálculo de la latencia en una red.

El programa recibe un archivo con las siguientes características:

En la primera línea se nos dirá el número de nodos y arcos que conforman la red:

P/e

6,6

6 nodos y 6 arcos

Para implementarlo en código lo que se hizo fue tomar esa línea y recorrerla hasta encontrar una coma (,); una vez hallada se tomaba el valor tras ella y se encendió una bandera que tomaba el valor seguido de la misma.

En las siguientes líneas nos indicara cada uno de los arcos, así como la distancia dada en metros de uno hacia otro nodo que lo conforman, separado por comas, la velocidad en Mbps, el tamaño del paquete en bytes, y los datos de control.

P/e

1,2,9,1000,1500,200

Entonces:

Desarrollo

Existe un arco del nodo 1 al 2

El cual tiene una distancia de 9 mts

Y una velocidad de 1000 Mbps

Con 1500 bytes en tamaño de paquete

Y 200 bytes de Datos de Control

En los siguientes renglones del archivo vienen los tiempos de cola

P/e

0.0025,0.001

Cada tiempo de cola va asignado a cada nodo excepto a los nodos inicial y final.

En el siguiente renglón, se nos indica en que vértice se inicia y a que vértice se llega.

P/e

1,6

El nodo o vértice inicial es el 1

Y el nodo o vértice final es el 6.

Por ultimo, tenemos el tamaño del archivo a transferir en GB.

P/e

3.5

El archivo pesa 3.5 GigaBytes.

Implementación:

Se hizo la lectura del archivo y el poder sacar todos los datos del mismo de forma correcta, después se pudieron obtener los caminos de manera correcta, teniendo un detalle al momento de revisar los nodos debían estar ordenados y siempre comenzar por el nodo 1, lo cual supone un problema ya que el nodo de inicio no siempre es el 1.

Al desarrollarlo se fue haciendo lectura línea por línea del archivo de esa forma se obtuvo el número de nodos y arcos, cada uno de los arcos descritos fueron ingresados en una matriz para su mejor manipulación, también fue almacenado el vértice inical y final y el tamaño del archivo a transferir.

Conclusiones

Por falta de tiempo el proyecto no se logró terminar, su funcionalidad se limita a leer correctamente el archivo, pero al momento de hacer los cálculos en muchas ocasiones en las pruebas nos encontramos con errores de java como ArrayIndexOutOfBoundException

O NullPointerException por lo cual se intentaron implementar ciertos métodos que parecían dar solución parcial pero nunca total.

Para sacar los caminos del vértice inicial al final, se utilizó kruskal y un recorrido a lo profundo pero no se logró implementar como se deseaba.

En consecuencia nunca se pudo hacer que los caminos fueran correctos al momento de ingresar un a archivo aleatorio, así que al no poder calcular los caminos totales la tarea de hacer el cálculo de la latencia se volvía casi imposible. Por lo cual solo se presenta la lectura del archivo.

Para intentar solucionar este problema se dio paso a algunos algoritmos descritos en la web:

http://informesdelaconstruccion.revistas.csic.es/index.php/informesdelaconstruccion/article/viewFile/1115/1199

El principal problema es la falta de organización al momento de desarrollar software, los tiempos a pesar de ser reducidos eran suficientes, pero al no tener una disciplina de trabajo no se logró el objetivo que era el cálculo de latencia.

Conclusiones

http://informesdelaconstruccion.revistas.csic.es/index.php/informesdelaconstruccion/article/viewFile/1115/1199

http://www.speed-test.es/qu_es_la_latencia

Conclusiones

Facultad de Ciencias de la Computación

Materia: Modelos de Redes

Nombre: Daniel de San Gines García Cortez

Proyecto 1

Fecha de entrega: 26 de Septiembre de 2014

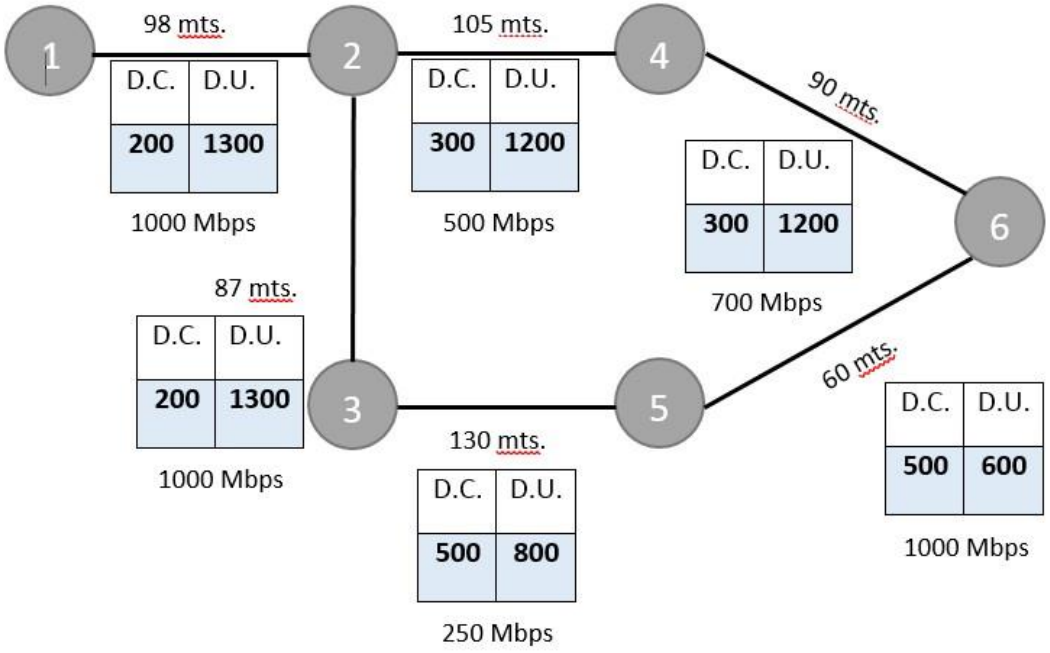
Introducción

En este primer proyecto de la materia de modelo de redes tenemos la necesidad de comprobar que opción(camino) sería la más adecuada para poder llegar de un punto “A” a un punto “B” dentro de un grafo conexo, esto lo haremos mediante programación (en esta ocasión se usará JAVA) y tendremos que utilizar un archivo donde estarán todos los datos necesarios para poder implementar el grafo y sus datos, el cual tendremos que leer dentro del programa. Calcularemos Latencias y Tiempos específicos, los cuales mencionaremos en el desarrollo.

DESARROLLO

En este proyecto tenemos la necesidad de implementar un grafo y decir cuál es el camino más óptimo para llegar al destino y porque.

Para empezar necesitamos saber cómo sería el grafo, utilizaremos como ejemplo un grafo de 6 nodos (véase en la figura grafio. 1).


1
98 mts.
2
105 mts.
4
90 mts.
6
D.C. D.U.
200 1300
1000 Mbps
D.C. D.U.
300 1200
500 Mbps
D.C. D.U.
300 1200
700 Mbps
87 mts.
D.C. D.U.
200 1300
1000 Mbps
3
130 mts.
5
60 mts.
D.C. D.U.
500 800
250 Mbps
D.C. D.U.
500 600
1000 Mbps

grafo.1

En este grafo notamos cuales datos contiene:

- Distancia de un nodo a otro (mts.)
- Los Datos de Control (D.C.)
- Los Datos de Usuario (D.U.)
- Velocidad a la que viaja la información (Mbps)

Todos estos datos se leerán de un archivo de texto, en esta ocasión el archivo contendría estos datos:

```
6,6
1,2,98,1000,1500,200
2,3,87,1000,1900,400
2,4,105,500,1500,300
3,5,130,250,1300,500
4,6,90,700,1100,400
5,6,60,1000,1100,500
0.0,0.00025,0.001,0.00070,0.0093,0.0
```

1,6

3.85

La primera línea del archivo tendrá dos campos (#Vértices,#Arcos) donde #Vértices son el número total de nodos y #Arcos es el número de conexiones.

Las otras líneas deben ser igual al #Arcos, y estos se identifican por (VI,VF,Dist,Vel, Tam_paq,DC) donde VI es el vértice inicial, VF el vértice final, Dist la distancia de un nodo a otro, Vel es la velocidad en la que viaja la información, Tam_paq es el tamaño del paquete que se enviará y DC los Datos de Control.

Después van los tiempos de cola de cada uno de los nodos (deben coincidir con #Vertices).

La penúltima línea se identifica por (Destino,Final) donde Destino es el nodo en el cual empezaremos el recorrido y Destino el nodo al cual se tiene que llegar.

Y la última línea nos dice el tamaño del archivo que se quiere compartir (dado en GB (Gigabytes)).

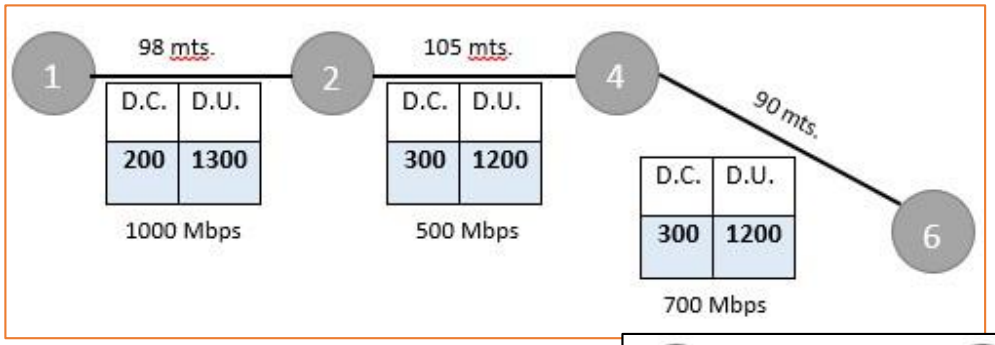
También tomaremos en cuenta algunas formulas.

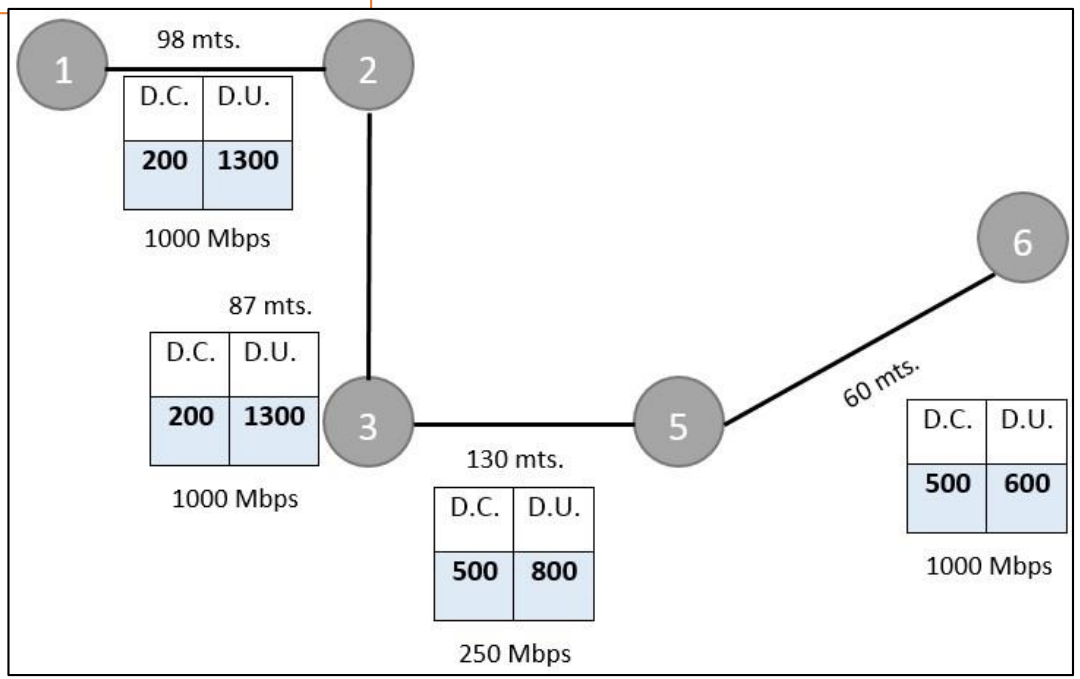
- Latencia = Tiempo de Propagación + Tiempo de Transmisión + Tiempo de Cola

- Tiempo de Propagación = Distancia a recorrer/Velocidad de la luz ($3x10^8$)
- Tiempo de Transmisión = Tamaño del Paquete(en bytes)/Velocidad (dada en Bps)

La latencia se estará multiplicando por una constante k que irá aumentando cuando se agregue otro arco al camino.

En la figura *grafo1* se ve claramente que hay dos caminos:

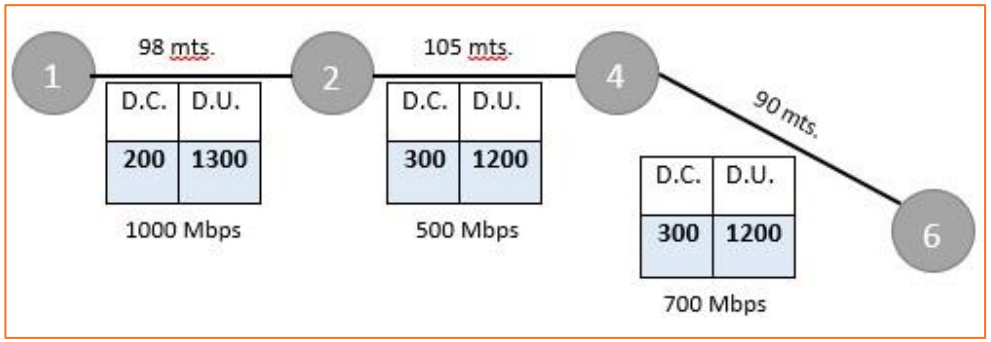




Entonces como tenemos 2 caminos tendremos que obtener 2 latencias, la latencia con menor cantidad será la más óptima para resolver el problema del tiempo de transmisión.

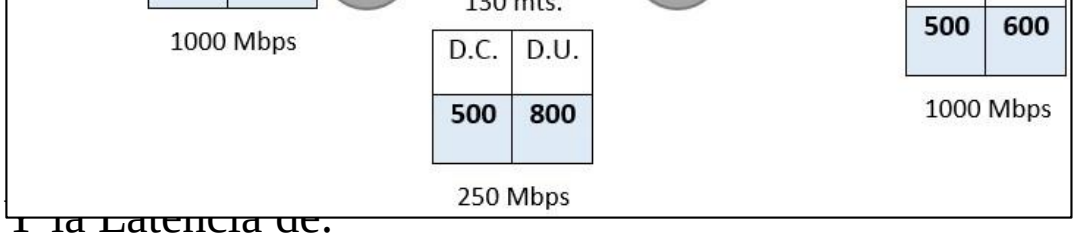
El programa que se elaboró en JAVA hace un recorrido sobre todo el grafo y solo toma los nodos que no han sido visitados (por obviedad el nodo Origen siempre se tomará en cuenta) y los mete a una lista Simplemente Ligada. Después de que llega al nodo Destino el programa devuelve la Lista Ligada, y con los tiempos de cola se obtendrán las latencias, cada Lista Ligada tendrá su propia Latencia.

Tomando en cuenta el *grafo1* la Latencia de:



sería: 0.0017496409523809525

Y la Latencia de:



Y la Latencia de:
Sería: 0.03035540666666668

Entonces la primera latencia sería me mejor opción para realizar las siguientes formulas.

Para saber cuántos paquetes se tienen que enviar del Origen al Destino debemos usar la siguiente formula:

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usuario del paq. Ini.

Como el tamaño del archivo esta dado en GB nosotros para competirlo a bytes lo multiplicamos 3 veces por 1024 (1024 * 1024 * 1024).

Ahora para sacar el Tiempo Total de Transferencia (TTT) se usa la formula:

- TTT= Latencia con menor cantidad * #Paquetes

Y es resultado se mostrará en segundo, solo lo convertimos a hora y minutos y esa sería el Tiempo Total de Transferencia.

CONCLUSIONES

Para lograr este proyecto se invirtieron aproximadamente 15 horas de programación y dando un resultado positivo.

En esta ocasión se pudo lograr la implementación del grafo, detección y devolución de caminos, resolución de Latencias, Tiempos y #paquetes. Dando por terminado esta sección del temario.

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Materia: Modelos de Redes

Integrantes:

Daniel de San Gines Garcia Cortez	201228619
Héctor Ramírez Ruíz	201205044
Cinthia Flores Mendez	200937657

Proyecto Final

Profesor: Ivan Olmos Pineda

Fecha de entrega: 28 de Noviembre de 2014

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

Introducción

En este proyecto final de la materia de Modelo de Redes tenemos la tarea de realizar un chat, el cual consta de realizar un aplicación (ap. Para abreviar) servidor y varias ap. Cliente utilizando la conexión mediante *sockets*. Se utiliza también lo que se conoce como *MD5* y *encriptación*.

Las aplicaciones tanto de cliente como servidor deben cubrir una especificación (que en el desarrollo se explica). Todo el proyecto será realizado en el lenguaje de programación JAVA.

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

DESARROLLO

Esquema General del proyecto

Para empezar a realizar este proyecto primero debemos saber qué acciones debe realizar el servidor, el cliente y todo lo demás.

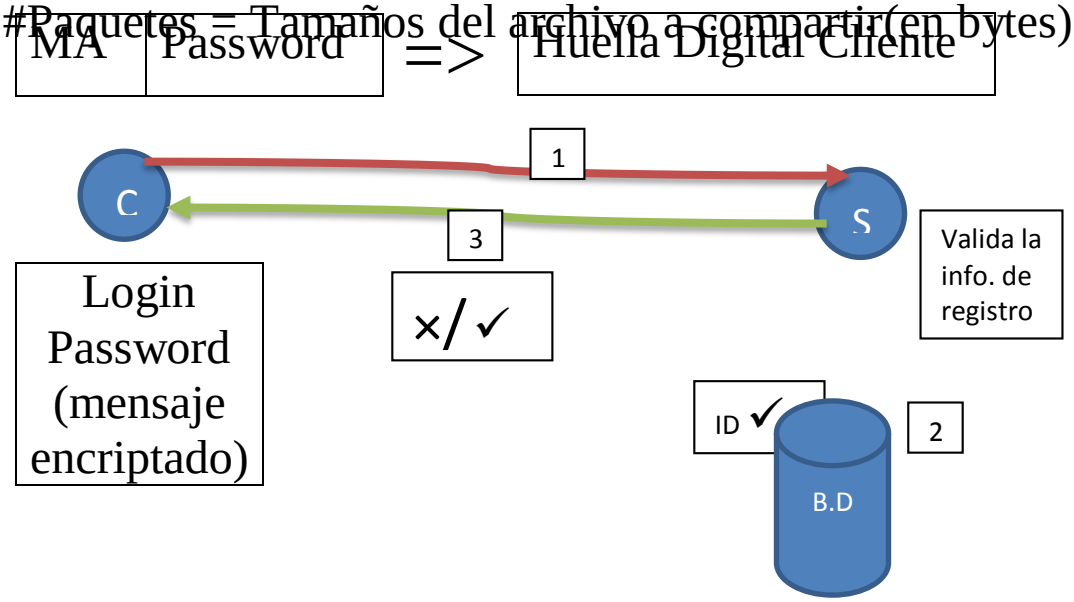
Empezando con la aplicación del cliente, esta aplicación puede hacer el alta de un nuevo cliente y hacer el login de los clientes ya registrados.

- Alta
- Login

Alta

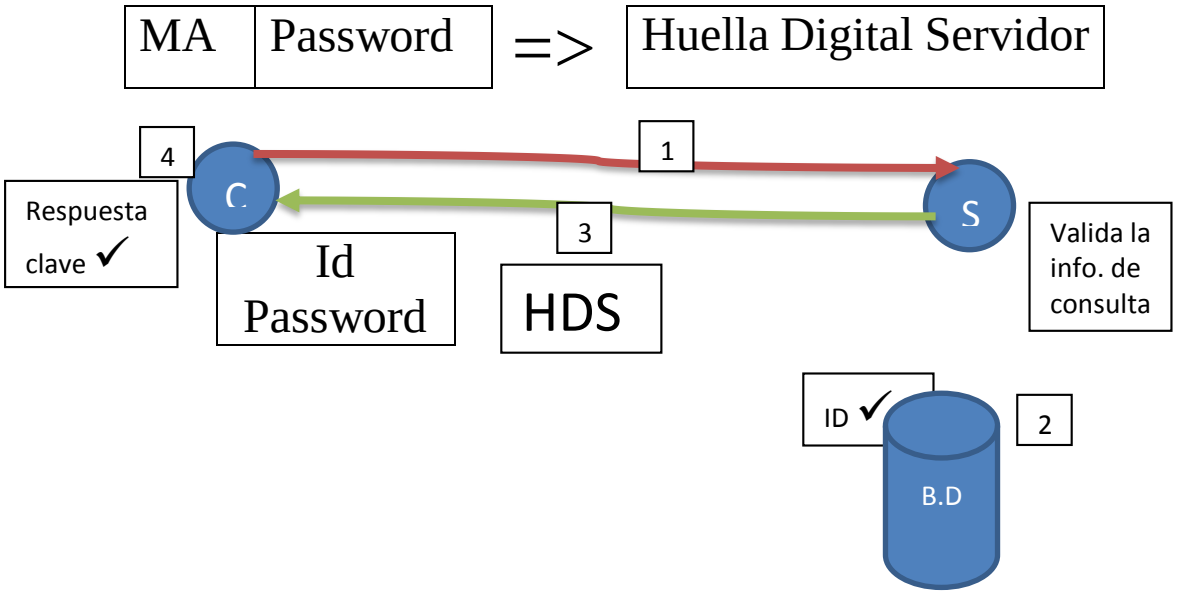
En la parte de alta el usuario pide un usuario y un password para ser ingresado en la base de datos, el cual hace mediante envió de un paquete al servidor, el paquete debe contener un mensaje aleatorio (MA) y su password, todo esto se convierte en una *huella digital del cliente*, el mensaje debe contener al menos 4 Kbytes de caracteres imprimibles, el servidor envía de regreso una confirmación después de haber realizado una alta en la base de datos, lo anterior será explicado con un pequeño dibujo.

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua



Login

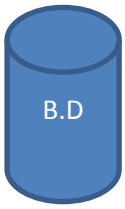
En la parte de login el usuario pide un ingresa su nombre de usuario y su password para ser verificado en la base de datos, el cual hace mediante envió de un paquete al servidor, el paquete debe ser enviado con la *huella digital del cliente*, el servidor regresa una respuesta clave el cual envía un paquete codificado de igual manera que el cliente, solo que a esta de le denomina *huella digital de servidor*, lo anterior también será explicado con un pequeño dibujo.



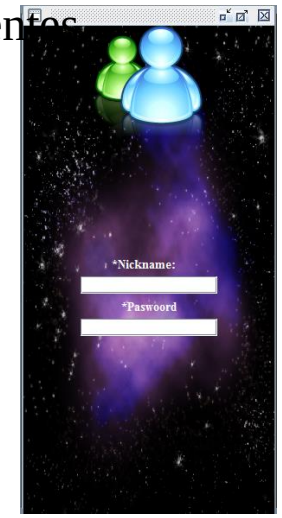
#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

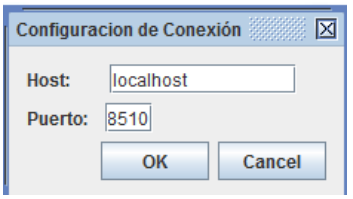
En la aplicación del servidor se mostrará la información de los usuarios a lo que se puede acceder, también será el encargado de aceptar a nuevos clientes y de hacer valida el login de los anteriores clientes.

Ap. Servidor



*Mostrar Información





Encriptación

Para progresar con el proyecto se tomó en cuenta un algoritmo de encriptación basado en MD5.

MD5 es uno de los algoritmos de reducción criptográficos diseñados por el profesor *Ronald Rivest* del *MIT (Massachusetts Institute of Technology, Instituto Tecnológico de Massachusetts)*.

Son utilizados para almacenar contraseñas y mantener cierto nivel de seguridad en las aplicaciones (y no guardar la clave como texto plano en un archivo o base de datos), a continuación se muestra la implementación de la encriptación mediante la clase MessageDigest

```
import java.security.MessageDi
gest;

2                                   import java.security.NoSuchAlgorithmExce
                                    ption;

3

4 /**

5  *

6  * @author Herman Alonso Barrates Víquez

7  */
```

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

```
8  public class StringMD {
9
10     //algoritmos
11     public static String MD2 = "MD2";
12     public static String MD5 = "MD5";
13     public static String SHA1 = "SHA-1";
14     public static String SHA256 = "SHA-256";
15     public static String SHA384 = "SHA-384";
16     public static String SHA512 = "SHA-512";
17
18     /***
19      * Convierte un arreglo de bytes a String usando valores
hexadecimales
20      * @param digest arreglo de bytes a convertir
21      * @return String creado a partir de <code>digest</code>
22      */
23     private static String toHexadecimal(byte[] digest){
24         String hash = "";
25         for(byte aux : digest) {
26             int b = aux & 0xff;
27             if (Integer.toHexString(b).length() == 1) hash += "0";
28             hash += Integer.toHexString(b);
29         }
30         return hash;
31     }
32
33     /***
34      * Encripta un mensaje de texto mediante algoritmo de resumen de
mensaje.
35      * @param message texto a encriptar
36      * @param algorithm algoritmo de encriptacion, puede ser: MD2,
MD5, SHA-1, SHA-256, SHA-384, SHA-512
37      * @return mensaje encriptado
38      */
39     public static String getStringMessageDigest(String message,
String algorithm){
40         byte[] digest = null;
41         byte[] buffer = message.getBytes();
42         try {
43             MessageDigest messageDigest =
MessageDigest.getInstance(algorithm);
44             messageDigest.reset();
45             messageDigest.update(buffer);
46             digest = messageDigest.digest();
```

```
47              } catch (NoSuchAlgorithmException ex){
48                  System.out.println("Error creando Digest");
49              }
50              return toHexadecimal(digest);
51          }
52      }
```

Descargar: StringMD.java

Para usar:

```
1  public static void main(String[] args) {
2          String mensaje = "Mensaje secreto";
3          System.out.println("Mensaje = " + mensaje);
4          System.out.println("MD2 = " +
   StringMD.getStringMessageDigest(mensaje, StringMD.MD2));
5          System.out.println("MD5 = " +
   StringMD.getStringMessageDigest(mensaje, StringMD.MD5));
6          System.out.println("SHA-1 = " +
   StringMD.getStringMessageDigest(mensaje, StringMD.SHA1));
7          System.out.println("SHA-256 = " +
   StringMD.getStringMessageDigest(mensaje, StringMD.SHA256));
8          System.out.println("SHA-384 = " +
   StringMD.getStringMessageDigest(mensaje, StringMD.SHA384));
9           System.out.println("SHA-512 = " +
    StringMD.getStringMessageDigest(mensaje, StringMD.SHA512));
10      }
```

Este fue el algoritmo que se buscó y llego a la desición de utilizar, la pregunta es…¿Por qué?.

La razón principal sobre la elección de este algoritmo es que Hay que tener en cuenta que esto no es 100% seguro, puesto que la contraseña se encripta en el servidor, entonces al enviar la contraseña desde el cliente al servidor podría ser interceptada. Para hacernos una idea, el algoritmo MD5 convierte el mensaje en un bloque múltiplo de 512 bits, (si hace falta añadirá bits por el final). Luego coge el primer bloque de 512 bits del mensaje y realiza diversas operaciones lógicas con los 128 bits de cuatro vectores iniciales ABCD de 32 bits cada uno. (Dichos vectores tendrán el valor inicial que nosotros queramos).

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

Como resultado obtiene una salida de 128 bits que se convierte en el nuevo conjunto de los 4 vectores ABCD. Se repite el algoritmo hasta procesar el último bloque del mensaje. Al terminar, el algoritmo devuelve los últimos 128 bits de estas operaciones.

Diccionario

Socket: Método para la comunicación entre un programa del cliente y un programa del servidor en una red.

MD5: técnica que permite obtener una huella digital de una serie de datos de entrada. Con la huella, se puede verificar la integridad de los datos, es decir, comprobar que no hayan sufrido alteración. Además, puede emplearse para la validación de usuarios o procesos

Encriptación: Codificación de la información de archivos o de un correo electrónico para que no pueda ser descifrado en caso de ser interceptado por alguien mientras esta información viaja por la red.

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

CONCLUSIONES

Después de haber realizado la selección de un servidor, la creación de los clientes y la base de datos, no se llegó a la conclusión del proyecto, debido a falta de especificaciones tales como:

- Funcionamiento entre cliente servidor
- Encriptación del archivo
- Sin complementación de la huella digital

Lamentablemente la conclusión del proyecto no pudo culminar, se hicieron aproximadamente 10 horas de programación tomando en cuenta 2-3 horas de programación no efectiva, es decir:

- 7 horas efectivas
- 3 horas errores

El proyecto quedó en un avance del 50% para futuras refrerencias.

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

Informe de proyecto

“Cálculo de latencias en un Grafo”

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

26 de Septiembre 2014

Introducción

El proyecto a desarrollar consiste en el cálculo de las latencias, apartir de los caminos generados desde un punto a otro en un grafo. Con el fin de aplicar nuestros conocimientos en programación y los conceptos adquiridos en la clase de modelo de redes.

Primero tenemos que recordar los conceptos a aplicar.

Latencia: Relacionado con el tiempo que toma un bit de viajar de un extremo de un medio al otro.Depende de tres factores:

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

-Tiempo de propagación del bit por el medio, que depende del tiempo de propagación de la corriente o luz por el medio, además de la distancia recorrida.

-Máxima cantidad de datos que pueden ser transmitidos por la red sin segmentarse. Al tiempo de propagación del bit en un paquete se le llama tiempo de transmisión.

-Tiempos de espera para difundirse un paquete a través de un conmutador, a demás del tráfico de la red. A este tiempo se le denomina tiempo de cola.

Latencia=TiempoPropagación + TiempoTransmisión +TiempoCola

$$\text{Tiempo de propagación} = \frac{\text{Distancia a recorrer}}{\text{Velocidad de la luz}}$$

$$\text{Tiempo de transmisión} = \frac{\text{Tamaño del paquete}}{\text{Tasa de transferencia teórica}}$$

Tasa de transferencia teórica: Máxima velocidad que se puede alcanzar considerando el ancho de banda del medio utilizado.No considera retardos de transmisión, ruido, etc.

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

Desarrollo del Programa

1.- Primero se crea la clase principal “Menú”, ahí creo un Grafo y leo el archivo de entrada linea por linea utilizando “split(",")” para separar los datos leidos al mismo tiempo que los voy insertando en nodos.

Llamo al método “todosCaminos” para calcular los caminos posibles.

Y finalmente mandó a llamar al método calcula que realiza el calculo de latencias.

```
public class Menu
{
    int Nnodos;
    int NN;
    int band=0;
    public void LeeGrafo(String arch,Grafo G1) //Lee archivo con los datos del grafo
      {

            FileInputStream fp;
            DataInputStream f;
            String linea = null;
            int token1,token2,token3,token4,token5,token6,i,j;
            int token7;
        try
         {
            fp = new FileInputStream(arch);
            f = new DataInputStream(fp);
            linea=f.readLine();

            String[] numerosComoArray = linea.split(",");
                for (int i1 = 0; i1 < numerosComoArray.length; i1++)
                {System.out.println(numerosComoArray[i1]);}

            int clineas=0;
            Nnodos=Integer.parseInt(numerosComoArray[0]);
            System.out.println(" Numero de Nodos: "+Nnodos);
            NN=Integer.parseInt(numerosComoArray[1]);
```

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

2.-Para la creación del Grafo utilizo una lista ligada donde cada nodo tiene una lista ligada que almacena sus nodos adyacentes; este se crea en la clase "Grafo".

En esta se encuentra el método "todosCaminos", que calcula los caminos de el nodo inicio al nodo final descritos en el archivo.

-Primero utlizo el método "DijkstraN" para encontrar el primer camino posible (el más corto); y apartir de este cree mi propio algoritmo que va eliminando el último nodo, seguido de agregar sus nodos adyacentes(no visitados) hasta encontrar el nodo final, y asi recursivamente hasta visitar todas las aristas y posibilidades de caminos entre estos dos nodos (inicio y fin).

```
public void todosCaminos(NodoVertice buscar,NodoVertice finaln,int bandInv)
    {   boolean band=false;boolean band1=false;int cont=0;
        DijkstraN(buscar.Vertice,finaln.Vertice);
        System.out.println("Primer Camino");
        for(int i = 0;i<todosCam.size();i++)
        {
                System.out.println(todosCam.get(i)+".....");
        }
        SinVisitar();
        while(todosCam.size()>0)
        {
            if(todosCam.get(todosCam.size()-1)==finaln.Vertice)
            {
                for(int i = 0;i<todosCam.size();i++)
                                {   MostartodosCam.Inserta2(todosCam.get(i));
                                   // System.out.println(todosCam.get(i));
                                GuardatodosCam.add(new ArrayList<Integer>());
                                GuardatodosCam.get(contCam).add(todosCam.get(i));
                                }
                contCam++;
                todosCam.remove(todosCam.get(todosCam.size()-1));
```


Además en esta clase esta el método "calcula", donde va ingresando al arreglo de caminos encontrados y realiza las

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

operaciones para calcular la latencia, que va guardando en el arreglo de “latencias”; y finalmente apartir de este ultimo arreglo escoge el de menor latencia y lo presenta al usuario.

```
for(int j=0;j<(GuardatodosCam.get(i).size())-1;j++)
{

        NodoVertice Auxiliar;
        Auxiliar=(ObtenerVertice(GuardatodosCam.get(i).get(j)));
        NodoArista Auxiliar2=Auxiliar.Arista;
        while (Auxiliar2.Vertice!=(GuardatodosCam.get(i).get(j+1)))
            {Auxiliar2 = Auxiliar2.Siguiente;}
        Tp=(Auxiliar2.distancia)/(3*(Math.pow(10,8)));
        System.out.print("Distancia " +Auxiliar2.distancia+" \n");
        System.out.print("luz " +3*(Math.pow(10,8))+ " \n");
        Tt=(double)((Auxiliar2.tamPaquete)*8)/(double)((Auxiliar2.Mbps)*1000*1000);//
        System.out.print("TamPaquete " +Auxiliar2.tamPaquete+ " \n");
        System.out.print("Mbps " +Auxiliar2.Mbps+ " \n");
        System.out.print("Datos de Control " +Auxiliar2.datosControl+ " \n");
        datos=(Auxiliar2.tamPaquete)-(Auxiliar2.datosControl);
        System.out.print("Datos " +datos+ " \n");
        Tc=(ObtenerVertice(GuardatodosCam.get(i).get(j+1)).tc);
        System.out.print("Tp " +Tp+ " \n");
        System.out.print("Tt " +Tt+ " \n");
        System.out.print("Tc " +Tc+ " \n");
        segmentosCam.get(i).add(Tp+Tt+Tc);
        System.out.print("Tp+Tt+Tc " +(Tp+Tt+Tc)+ " \n");
        if(j==0)
        {
            multiplicadorCam.get(i).add(1);
            arreglo_de_listas[j].InsertaPaquete(datos);
            System.out.print("Multiplicador" +(multiplicadorCam.get(i).get(j))+ " \n");
            System.out.print("Datos");
            arreglo_de_listas[j].Mostrar2();System.out.print("\n");
        }
        else
```

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

Informe de proyecto

“Cálculo de latencias en un Grafo”

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

Lucia Adriana Merlo Haro

26 de Septiembre 2014

Introducción

El proyecto a desarrollar consiste en el cálculo de las latencias, apartir de los caminos generados desde un punto a otro en un grafo. Con el fin de aplicar nuestros conocimientos en programación y los conceptos adquiridos en la clase de modelo de redes.

Primero tenemos que recordar los conceptos a aplicar.

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

Latencia: Relacionado con el tiempo que toma un bit de viajar de un extremo de un medio al otro.Depende de tres factores:

-Tiempo de propagación del bit por el medio, que depende del tiempo de propagación de la corriente o luz por el medio, además de la distancia recorrida.

-Máxima cantidad de datos que pueden ser transmitidos por la red sin segmentarse. Al tiempo de propagación del bit en un paquete se le llama tiempo de transmisión.

-Tiempos de espera para difundirse un paquete a través de un conmutador, a demás del tráfico de la red. A este tiempo se le denomina tiempo de cola.

Latencia=TiempoPropagación + TiempoTransmisión +TiempoCola

$$\text{Tiempo de propagación} = \frac{\text{Distancia a recorrer}}{\text{Velocidad de la luz}}$$

$$\text{Tiempo de transmisión} = \frac{\text{Tamaño del paquete}}{\text{Tasa de transferencia teórica}}$$

Tasa de transferencia teórica: Máxima velocidad que se puede alcanzar considerando el ancho de banda del medio utilizado.No considera retardos de transmisión, ruido, etc.

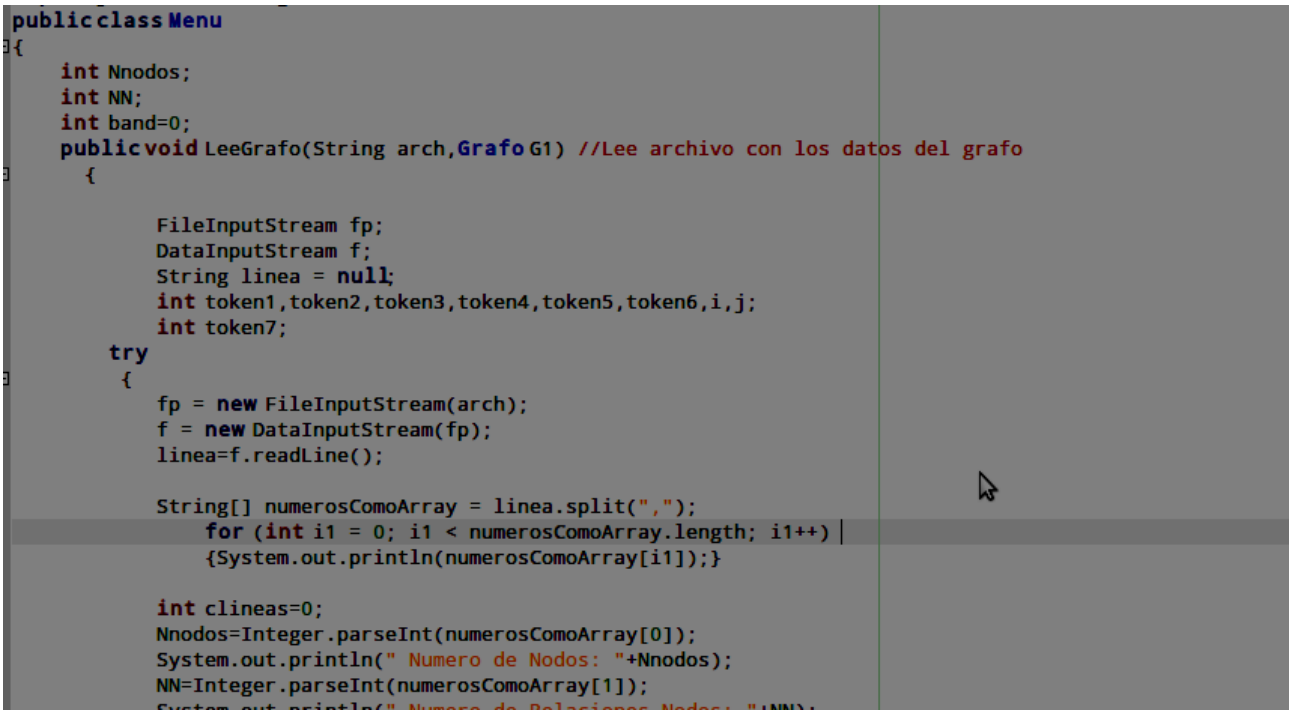
#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

Desarrollo del Programa

1.- Primero se crea la clase principal “Menú”, ahí creo un Grafo y leo el archivo de entrada linea por linea utilizando “split(",")” para separar los datos leidos al mismo tiempo que los voy insertando en nodos.

Llamo al método “todosCaminos” para calcular los caminos posibles.

Y finalmente mandó a llamar al método calcula que realiza el calculo de latencias.



```
public class Menu
{
    int Nnodos;
    int NN;
    int band=0;
    public void LeeGrafo(String arch,Grafo G1) //Lee archivo con los datos del grafo
      {

            FileInputStream fp;
            DataInputStream f;
            String linea = null;
            int token1,token2,token3,token4,token5,token6,i,j;
            int token7;
        try
         {
            fp = new FileInputStream(arch);
            f = new DataInputStream(fp);
            linea=f.readLine();

            String[] numerosComoArray = linea.split(",");
                for (int i1 = 0; i1 < numerosComoArray.length; i1++)
                {System.out.println(numerosComoArray[i1]);}

            int clineas=0;
            Nnodos=Integer.parseInt(numerosComoArray[0]);
            System.out.println(" Numero de Nodos: "+Nnodos);
            NN=Integer.parseInt(numerosComoArray[1]);
```

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

2.-Para la creación del Grafo utilizo una lista ligada donde cada nodo tiene una lista ligada que almacena sus nodos adyacentes; este se crea en la clase "Grafo".

En esta se encuentra el método "todosCaminos", que calcula los caminos de el nodo inicio al nodo final descritos en el archivo.

-Primero utlizo el método "DijkstraN" para encontrar el primer camino posible (el más corto); y apartir de este cree mi propio algoritmo que va eliminando el último nodo, seguido de agregar sus nodos adyacentes(no visitados) hasta encontrar el nodo final, y asi recursivamente hasta visitar todas las aristas y posibilidades de caminos entre estos dos nodos (inicio y fin).

```
public void todosCaminos(NodoVertice buscar,NodoVertice finaln,int bandInv)
    {   boolean band=false;boolean band1=false;int cont=0;
        DijkstraN(buscar.Vertice,finaln.Vertice);
        System.out.println("Primer Camino");
        for(int i = 0;i<todosCam.size();i++)
        {
                System.out.println(todosCam.get(i)+".....");
        }
        SinVisitar();
        while(todosCam.size()>0)
        {
            if(todosCam.get(todosCam.size()-1)==finaln.Vertice)
            {
                for(int i = 0;i<todosCam.size();i++)
                                {   MostartodosCam.Inserta2(todosCam.get(i));
                                   // System.out.println(todosCam.get(i));
                                GuardatodosCam.add(new ArrayList<Integer>());
                                GuardatodosCam.get(contCam).add(todosCam.get(i));
                                }
                contCam++;
                todosCam.remove(todosCam.get(todosCam.size()-1));
```

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

Además en esta clase esta el método “calcula”, donde va ingresando al arreglo de caminos encontrados y realiza las operaciones para calcular la latencia; que va guardando en el arreglo de “latencias”; y finalmente apartir de este ultimo arreglo escoge el de menor latencia y lo presenta al usuario.

```
for(int j=0;j<(GuardatodosCam.get(i).size())-1;j++)
{

        NodoVertice Auxiliar;
        Auxiliar=(ObtenerVertice(GuardatodosCam.get(i).get(j)));
        NodoArista Auxiliar2=Auxiliar.Arista;
        while (Auxiliar2.Vertice!=(GuardatodosCam.get(i).get(j+1)))
            {Auxiliar2 = Auxiliar2.Siguiente;}
        Tp=(Auxiliar2.distancia)/(3*(Math.pow(10,8)));
        System.out.print("Distancia " +Auxiliar2.distancia+" \n");
        System.out.print("luz " +3*(Math.pow(10,8))+ " \n");
        Tt=(double)((Auxiliar2.tamPaquete)*8)/(double)((Auxiliar2.Mbps)*1000*1000);//
        System.out.print("TamPaquete " +Auxiliar2.tamPaquete+ " \n");
        System.out.print("Mbps " +Auxiliar2.Mbps+ " \n");
        System.out.print("Datos de Control " +Auxiliar2.datosControl+ " \n");
        datos=(Auxiliar2.tamPaquete)-(Auxiliar2.datosControl);
        System.out.print("Datos " +datos+ " \n");
        Tc=(ObtenerVertice(GuardatodosCam.get(i).get(j+1)).tc);
        System.out.print("Tp " +Tp+ " \n");
        System.out.print("Tt " +Tt+ " \n");
        System.out.print("Tc " +Tc+ " \n");
        segmentosCam.get(i).add(Tp+Tt+Tc);
        System.out.print("Tp+Tt+Tc " +(Tp+Tt+Tc)+ " \n");
        if(j==0)
        {
            multiplicadorCam.get(i).add(1);
            arreglo_de_listas[j].InsertaPaquete(datos);
            System.out.print("Multiplicador" +(multiplicadorCam.get(i).get(j))+ " \n");
            System.out.print("Datos");
            arreglo_de_listas[j].Mostrar2();System.out.print("\n");
        }
        else
```

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
Ingeniería en Ciencias de la Computación
Modelos de Redes

Proyecto Final

Graciela Gaona Bernabé Erika
Leonor Basurto Munguía Lucia
Adriana Merlo Haro

Cesar Manuel Rodríguez Mendiola

26/11/2014

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

Índice general

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

ÍNDICE GENERAL 2

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

Capítulo 1

Antecedentes teóricos

1.1. Sockets

Los sockets son una forma de comunicación entre procesos que se encuentran en diferentes máquinas de una red.

Proporcionan un punto de comunicación por el cual se puede enviar o recibir información entre procesos.

Tienen un ciclo de vida dependiendo si son sockets de servidor, que esperan a un cliente para establecer una comunicación, o socket cliente que busca a un socket de servidor para establecer la comunicación.

#Paquetes = Tamaños del archivo a compartir(en bytes) / Datos de Usua

Los sockets empleados para la comunicación deberán ser del mismo tipo.

Hay dos tipos básicos de socket:

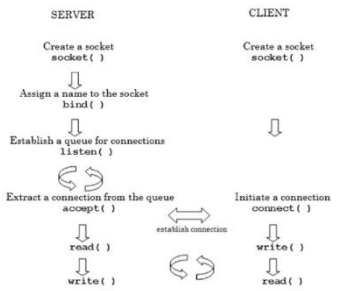
1.1.1. Stream sockets (sockets de ujo)

Estos sockets estan orientados a conexión.

Son seguros.

Cuando se utilizan estos, los datos se entregan en orden, es decir, en la misma secuencia en la que se envió. No hay duplicación de datos, además permite la comprobación de errores y el control de ujo.

Los datos se transmiten como un ujo de bytes. De una manera muy limitada, estas tomas también permiten al usuario colocar un mensaje urgente prioridad más alta por delante de los datos de la corriente actual.



Esquema de sockets orientadoa a conexión.

1.1.2. Datagram sockets (sockets de datagramas)

Estos sockets NO están orientados a conexión.

Son potencialmente poco able.

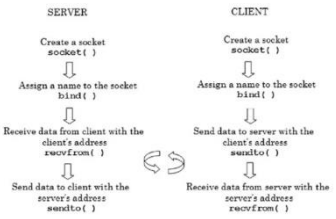
Los datos recibidos estaran fuera de orden.

No hay una conexión lógica entre los procesos de envío y recepción.

Cada datagrama se envía y se procesa independientemente. Datagramas individuales pueden tomar diferentes rutas para el mismo destino.

Con servicio sin conexión, no hay control de ujo. El control de errores, cuando se especi ca, es mínima.

Los paquetes de datagramas suelen ser pequeños y de tamañojo.



Esquema de sockets no orientadoa a conexión.

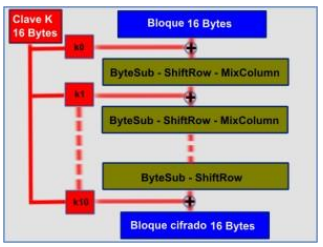
1.2. Advanced Encryption Standard (AES)

Su historia de éxito comenzó 1997, cuando el Instituto Nacional de Estándares y Tecnología, National Institute of Standards and Technology (NIST), anunció la búsqueda de un sucesor para el estándar de cifrado DES. Un algoritmo llamado "Rijndael", desarrollado por los criptólogos belgas Joan Daemen y Vincent Rijmen, fue destacado en seguridad, así como en el rendimiento y la exibilidad. Este algoritmo le ganó a varios competidores, y fue o cialmente presentado como el nuevo estándar de cifrado AES en el 2001 y se transformó en estándar efectivo en el 2002.

1.2.1. Funcionamiento

El algoritmo se basa en varias sustituciones, permutaciones y transformaciones lineales, ejecutadas en bloques de datos de 16 bytes - por lo que se le llama blockcipher. Estas operaciones se repiten varias veces, llamadas "rondas". En cada ronda, un único roundkey se calcula de la clave de encriptación, y es incorporado en los cálculos. Basado en esta estructura de bloque de AES, el cambio de un solo bit, ya sea en la clave, o en los bloques de texto simple y claro, resulta en un bloque de texto cifrado/encriptado completamente diferente - una clara ventaja sobre cifrados de ujo tradicionales.

AES utiliza claves de 128, 192 o 256 bits de longitud, sobre bloques –jos de 16 bytes, lo que da como resultado bloques cifrados en la salida del mismo tamaño que en la entrada, 16 bytes. En la imagen superior podemos ver un esquema que muestra el funcionamiento básico de este algoritmo con claves de 128 bits.



Funcionamiento del AES.

1.2.1.1. Inicio

Uno de los puntos destacados del AES es la manera en la que se aplica su clave K. En primer lugar, ésta se expande en un subconjunto formado por (k0,k1,k2,... kn), a cada una de ellas se le aplica una función round r(kn,), que realiza una operación binaria XOR con el mensaje, es decir, el bloque de entrada es cifrado por cada una de las subclaves r(kn)hasta llegar al nal.

En cada round, se llevan a cabo diferentes funciones de sustitución y permutación, por lo tanto se cambia el orden y estado inicial de los datos incluidos en el mensaje inicial. Este proceso debe ser reversible, para que el sistema sea capaz de desencriptar el mensaje.

1.2.1.2. Función round

ByteSub. Aquí cada byte del actual estado es sustituido por su correspondiente, según una tabla S de búsqueda.

1. ShiftsRows. Cada la del state es deslizada un número de posiciones determinado.

2. MixColumns. Se efectúa una transformación lineal a cada una de las columnas.

1.2.1.3. Salida

Finalmente, después de aplicar 10 rounds sobre los 16 bytes del bloque de entrada, se obtiene en la salida un bloque de igual tamaño, 16 bytes. De hecho, tanto el mensaje como las round keys mantienen una longitud de 128 bits durante todo el proceso.

1.2.2. Nivel de seguridad

El estándar de cifrado (encriptación) avanzado AES, Advanced Encryption Standard (AES), es uno de los algoritmos más seguros y más utilizados hoy en día - disponible para uso público. Está clasi - cado por la Agencia de Seguridad Nacional, National Security Agency (NSA), de los Estados Unidos para la seguridad más alta de información secreta.

Este popular estándar se usa en operaciones tan importantes como pagos electrónicos o transacciones bancarias.

El método más común de ataque hacia un cifrador por bloques consiste en intentar varios ataques sobre versiones del cifrador con un número menor de rondas. El AES tiene 10 rondas para llaves de 128 bits, 12 rondas para llaves de 192 bits, y 14 rondas para llaves de 256 bits. Hasta 2005, los mejores ataques conocidos son sobre versiones reducidas a 7 rondas para llaves de 128 bits, 8 rondas para llaves de 192 bits, y 9 rondas para llaves de 256 bits.

A modo de ejemplo: Descifrar una clave de 128 bits AES con una supercomputadora estándar del momento, llevaría más tiempo que la presunta edad del universo. ½Boxcryptor utiliza incluso claves de 256! Hasta el día de hoy, no existe posible ataque contra AES. Por lo tanto, sigue siendo el estándar AES de cifrado preferido por los gobiernos, los bancos y los sistemas de alta seguridad de todo el mundo.

En 2002, un ataque teórico, denominado "ataque XSL", fue anunciado por Nicolas Courtois y Josef Pieprzyk, mostrando una potencial debilidad en el algoritmo AES. Varios expertos criptográ cos han encontrado problemas en las matemáticas que hay por debajo del ataque propuesto, sugiriendo que los autores quizá hayan cometido un error en sus estimaciones. Si esta línea de ataque puede ser tomada contra AES, es una cuestión todavía abierta. Hasta el momento, el ataque XSL contra AES parece especulativo; es improbable que nadie pudiera llevar a cabo en la práctica este ataque.

1.3. MD5

Utilizaremos el algoritmo MD5 (Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje5) de 128 bits para encriptar la contraseña proporcionada por el usuario, a la cual se le añadirá un mensa je aleatorio generado por el servidor. Una vez encriptada se enviará al servidor que la almacenara en una base de datos.

Cuando un usuario introduce su contraseña, esta se encripta en MD5 y se compara con la base de datos; de esta forma la contraseña no está almacenada en ningún sitio además de que MD5 no permite desencriptarla.

En este tipo de algoritmo por Digestión de mensajes o Hash, se crean "Huellas Digitales" que son claves o llaves que representan un documento o conjunto de datos a partir de un mensaje y mediante ciertas operaciones matemáticas; cumplen con las siguientes características:

-Es incomprensible a simple vista.

-Cada huella es única para cada mensaje.

-Dos huellas son iguales si y solo si el mensaje original también lo es.

-Es unidireccional, es decir que no se puede reconstruir el mensaje original a partir de su huella digital.

- Facilidad de empleo e implementación.

-Su salida es una huella digital, de tamaño jo e independiente de la dimensión del documento original.

En java se usa la clase MessageDigest para obtener la encriptación MD5 y así hacer uso de las funciones criptográ cas en la plataforma Java.

1.3.1. Algoritmo

Descripción del algoritmo md5

Empezamos suponiendo que tenemos un mensaje de 'b' bits de entrada, y que nos gustaría encontrar su resumen. Aquí 'b' es un valor arbitrario entero no negativo, no tiene que ser múltiplo de ocho, y puede ser muy largo. Imaginemos los bits del mensaje escritos así:

m0 m1 ... m{b-1}

Los siguientes cinco pasos calculan el resumen del mensaje.

1.3.1.1. Adición de bits

El mensaje será extendido hasta que su longitud en bits sea congruente con 448, módulo 512. Esto es, si se le resta 448 a la longitud del mensaje tras este paso, se obtiene un múltiplo de 512. Esta extensión se realiza siempre, incluso si la longitud del mensaje es ya congruente con 448, módulo 512.

La extensión se realiza como sigue: un solo bit "1" se añade al mensaje, y después se añaden bits "0" hasta que la longitud en bits del mensaje extendido se haga congruente con 448, módulo 512. En todos los mensajes se añade al menos un bit y como máximo 512.

1.3.1.2. Longitud del mensaje

Un entero de 64 bits que represente la longitud 'b' del mensaje (longitud antes de añadir los bits) se concatena al resultado del paso anterior. En el supuesto no deseado de que 'b' sea mayor que 2^64, entonces sólo los 64 bits de menor peso de 'b' se usarán.

En este punto el mensaje resultante (después de rellenar con los bits y con 'b') se tiene una longitud que es un múltiplo exacto de 512 bits. A su vez, la longitud del mensaje es múltiplo de 16 palabras (32 bits por palabra). Con M[0 ... N-1] denotaremos las palabras del mensaje resultante, donde N es múltiplo de 16.

1.3.1.3. Inicializar el búfer MD

Un búfer de cuatro palabras (A, B, C, D) se usa para calcular el resumen del mensaje. Cada letra representa un registro de 32 bits. Estos registros se inicializan con los siguientes valores hexadecimales, los bytes de menor peso primero:

palabra A: 01 23 45 67

palabra B: 89 ab cd ef

palabra C: fe dc ba 98

palabra D: 76 54 32 10

1.3.1.4. Procesado del mensaje en bloques de 16 palabras

De nimos cuatro funciones auxiliares que toman como entrada tres palabras de 32 bits y su salida es una palabra de 32 bits.

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$

$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$

$$H(X,Y,Z) = X \oplus Y \oplus Z$$

$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

En cada posición de cada bit X actúa como un condicional: si X, entonces Z sino Y. La función Z podría haber sido de nida usando + en lugar de v ya que XY y not(x) Z nunca tendrán unos ('1') en la misma posición de bit.

Las funciones G, H e I son similares a la función F, ya que actúan bit a bit en paralelo" para producir sus salidas de los bits de X, Y y Z, en la medida que si cada bit correspondiente de X, Y y Z son independientes y no sesgados, entonces cada bit de G(X,Y,Z), H(X,Y,Z) e I(X,Y,Z) serán independientes y no sesgados.

Este paso usa una tabla de 64 elementos T[1 ... 64] construida con la función Seno. Denotaremos por T[i] el elemento i-ésimo de esta tabla, que será igual a la parte entera del valor absoluto del seno de 'i' 4294967296 veces, donde.....

Código del MD5:

```
/* Procesar cada bloque de 16 palabras. */

para i = 0 hasta N/16-1 hacer

/* Copiar el bloque 'i' en X. */

para j = 0 hasta 15 hacer

hacer X[j] de M[i*16+j].

n para /* del bucle 'j' */

/* Guardar A como AA, B como BB, C como CC, y D como DD.
*/

/* Ronda 1. */

/* [abcd k s i] denotarán la operación a = b + ((a + F(b, c, d) +
X[k] + T[i]) < < < s). */

/* Hacer las siguientes 16 operaciones. */

[ABCD 0 7 1] [DABC 1 12 2] [CDAB 2 17 3] [BCDA 3 22 4]

[ABCD 4 7 5] [DABC 5 12 6] [CDAB 6 17 7] [BCDA 7 22 8]

[ABCD 8 7 9] [DABC 9 12 10] [CDAB 10 17 11] [BCDA 11 22 12]
```

[ABCD 12 7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15 22

/* Ronda 2. */

/* [abcd k s i] denotarán la operación

a = b + ((a + G(b, c, d) + X[k] + T[i]) < < < s). */

/* Hacer las siguientes 16 operaciones. */ [ABCD 1 5 17] [DABC 6 9 18] [CDAB 11 14 19] [BCDA 0 20 20] [ABCD 5 5 21] [DABC 10 9 22] [CDAB 15 14 23] [BCDA 4 20 24] [ABCD 9 5 25] [DABC 14 9 26] [CDAB 3 14 27] [BCDA 8 20 28] [ABCD 13 5 29] [DABC 2 9 30] [CDAB 7 14 31] [BCDA 12 20 32]

/* Ronda 3. */

/* [abcd k s t] denotarán la operación

a = b + ((a + H(b, c, d) + X[k] + T[i]) < < < s). */

/* Hacer las siguientes 16 operaciones. */
[ABCD 5 4 33] [DABC 8 11 34] [CDAB 11 16 35] [BCDA 14 23 36]

[ABCD 1 4 37] [DABC 4 11 38] [CDAB 7 16 39] [BCDA 10 23 40]

[ABCD 13 4 41] [DABC 0 11 42] [CDAB 3 16 43] [BCDA 6 23 44]
[ABCD 9 4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA 2 23 48]

/* Ronda 4. */

/* [abcd k s t] denotarán la operación

a = b + ((a + I(b, c, d) + X[k] + T[i]) < < < s). */

/* Hacer las siguientes 16 operaciones. */

[ABCD 0 6 49] [DABC 7 10 50] [CDAB 14 15 51] [BCDA 5 21 52] [ABCD 12 6 53] [DABC 3 10 54] [CDAB 10 15 55] [BCDA 1 21 56]

[ABCD 8 6 57] [DABC 15 10 58] [CDAB 6 15 59] [BCDA 13 21 60]

[ABCD 4 6 61] [DABC 11 10 62] [CDAB 2 15 63] [BCDA 9 21 64]

/* Ahora realizar las siguientes sumas. (Este es el incremento de cada uno de los cuatro registros por el valor que tenían antes de que este bloque fuera inicializado.) */

A = A + AA

B = B + BB

C = C + CC

D = D + DD

1.3.1.5. Salida

El resumen del mensaje es la salida producida por A, B, C y D. Esto es, se comienza el byte de menor peso de A y se acaba con el byte de mayor peso de D.

Capítulo 2

Desarrollo

El desarrollo de esta aplicación será en un entorno grá co (netbeans), para facilitar la creación de la interfaz grá ca con la cual el nalmente el usuario tendrá que interactuar. Así mismo este entorno trabaja sobre la plataforma de java lo cual nos facilitara el desarrollo además cuenta con las librerías para encriptar mediante AES y MD5.

2.1. Conectividad

Para lograr una conectividad e ciente implementaremos los sockets orientados a conexión (Stream socket), así mismo aremos uso de los canales (Stream) para el ujo de información. Para que la aplicación soporte conexiones de multiples usuarios es necesario

2.2. Base de datos

Se necesita una base de datos para llevar el control de los usuarios que se registran, pero también llevar el control de los contactos que tiene el usuario.

Se determina que cada usuario debe de tener registrado un correo con el que se identi ca, password para poder ingresar a su cuenta, una llave que le sirve para su seguridad de inicio de sesión es decir

cada vez que intente iniciar sesión esta será la mezclar del password con el algoritmo MD5 que se compara con la que el usuario enviara , por lo cual esta puede ser nula ya que solo se usa cuando el usuario se logue y una Ip que le sirve para poderse conectar con algún usuario siempre y cuando este registrado.

Cada usuario puede o no tener contactos, cada contacto que desee agregar debe estar en la lista usuarios . Los contactos que agregan

en contactos se validan para evitar duplicación de datos, también las demás operaciones que las haga como usuario se deben de hacer como contacto.

2.2.1. Operaciones que se hace con la BD

2.2.1.1. Obtener todos los contactos de un determinado usuario

Select *from contactos where correo_usuario= _un _usuario_

Select *from contactos where correo_contacto =_un_usuario

2.2.1.2. validar correo

Se valida que el correo del usuario que no exista o un contacto que exista como usuario

Select correo_usuario from usuario where correo_usuario= _correo_nuevo_

2.2.1.3. Agregar mensaje

El mesaje se agregará cada que el usuario quiera tener conversación con alguno de sus contactos o haya tenido conversación.

Update contacto set mensaje = _mensaje_ where correo_usuario= _correo_usuario_ and correo_contacto= _correo_contacto_

Update contacto set mensaje = _mensaje_ where correo_usuario= _correo_usuario_ and correo_contacto= _correo_contacto_ or where correo_usuario= _correo_contacto_ and correo_contacto= _correo_usuario

2.2.1.4. Agregar un contacto de un determinado usuario

Insert into contactos (correo_usuario, correo_contacto) values (_correo_usuario_ , _correo_contacto_)

2.2.1.5. Agregar un usuario

Insert into contactos (correo_usuario, password, ip) values (_correo_usuario_ , _correo_contacto_ , _ip_)

2.2.1.6. Obtener el password

Obtiene el password de un determinado usuario para iniciar sesión

Select password from usuario where correo_usuario= _correo_nuevo_

2.2.1.7. Obtener mensaje

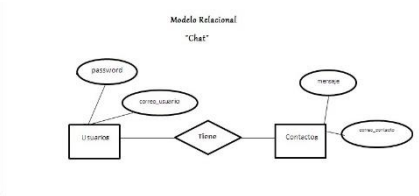
Se obtiene una determinada conversación;

Select mensaje contactos where correo_usuario= _un _usuario_ and correo_contacto=_correo_contacto_ or correo_usuario =_correo_contacto_ and correo_contato=_correo_usuario_

Capítulo 3

Resultados

3.1. Base de datos



Modelo entidad-relacion

Contactos

Field	Type	Null
correo_usuario	varchar(100)	No
correo_contacto	varchar(100)	No
mensaje	varchar(200)	yes

Tabla de contactos

Usuario

Field	Type	Null	key
correo_usuario	varchar(100)	No	
password	varchar(100)	no	

Tabla de usuarios

3.2. AES

Se inicializa el vector de bytes

```
byte[] menEncriptado = null;

Stringcadena="micccccccccccclavemuajaumicccccccccccclavemuajaumicccccccccccclavemu

byte[] cad=(cadena.getBytes());
```

Generamos una clave de 128 bits adecuada para AES

```
KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");

keyGenerator.init(128); Key key = keyGenerator.generateKey();
```

Alternativamente, una clave que queramos que tenga al menos 16 bytes y nos quedamos con los bytes 0 a 15

```
key = new SecretKeySpec(cad, 0, 16, "AES");
```

Se obtiene un cifrador AES

```
Cipher aes = Cipher.getInstance("AES/ECB/PKCS5Padding");
```

Se inicializa para encriptacion y se encripta el texto, que debemos pasar como bytes.

```
aes.init(Cipher.ENCRYPT_MODE, key);

byte[] encriptado = aes.doFinal(men.getBytes());
```

Se iniciliza el cifrador para desencriptar, con la misma clave y se desencripta

```
aes.init(Cipher.DECRYPT_MODE, key);

byte[] desencriptado = aes.doFinal(men);
```

Texto obtenido, igual al original.

```
menDesencriptado = new String(desencriptado);
```

3.3. MD5

```
import java.security.MessageDigest;
```

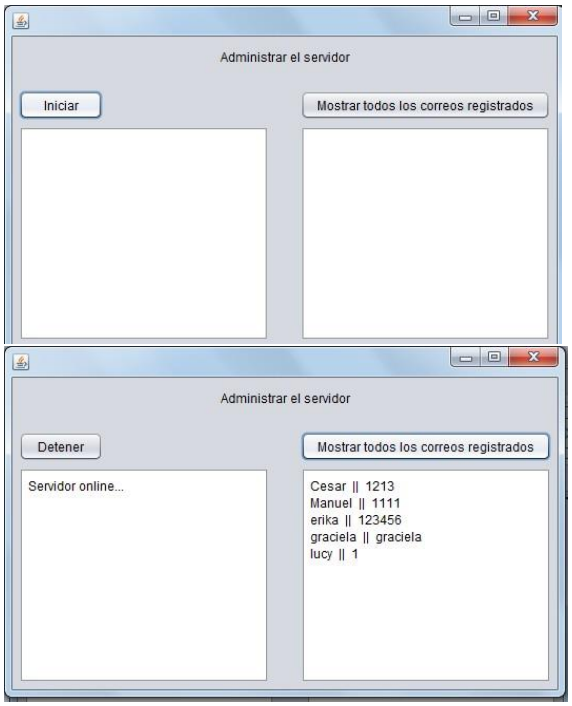
```
import java.security.NoSuchAlgorithmException;

public class TestEncriptarMD5 {

private static final char[] CONSTS_HEX = { '0','1','2','3','4','5','6','7','8','9','a','b','c','d','e','f
};

public static String encriptaEnMD5(String stringAEncriptar) {

try {

MessageDigest md = MessageDigest.getInstance("MD5");

byte[] bytes = md.digest(stringAEncriptar.getBytes());

StringBuilder strbCadenaMD5 = new StringBuilder(2 * bytes.length);
for (int i = 0; i < bytes.length; i++) {

int bajo = (int)(bytes[i] & 0x0f ); int alto = (int)((bytes[i] & 0xf0)
> > 4);
strbCadenaMD5.append(CONSTS_HEX[alto]);
strbCadenaMD5.append(CONSTS_HEX[bajo]);

}

return strbCadenaMD5.toString();

} catch (NoSuchAlgorithmException e) {

return null;

}
```

Se obtiene un MessageDigest con MessageDigest.getInstance() indicando que algoritmo de encriptación queremos (el parámetro MessageDigestAlgorithms.MD5),a continuación convertimos a bytes nuestro texto ("stringAEncriptar".getBytes()).

Ahora basta con pedirle el texto encriptado con el método digest(), este nos lo devolverá como un array de bytes. Como los bytes obtenidos pueden no ser legibles, los transformamos a números hexadecimales; esto para evitar los problemas típicos de codi cación de caracteres.

3.4. Ejecución de la aplicación

3.4.1. Servidor



3.4.2. Cliente


Cuenta
Manuel
Contraseña
****
IP Servidor
localhost
Registrar
Entrar
Mensaje
Correo invalido
Aceptar
Manuel
Agregar contacto
erika
Welcome to chat
Manuel:hola kika
erika:hola
erika:jjajaja
Manuel:bye
Salir


Entrada
Proporcione la cuenta del usuario
Cintya
Aceptar
Cancelar
Mensaje
ERROR: No encuentro un servidor online
Aceptar

Capítulo 4

Conclusiones

El uso de bibliotecas para la encriptación es sencillo y de gran potencialidad para asegurar los paquetes que uyen por la red, y si el propósito es garantizar la integridad podemos emplear el uso de las huellas digitales.

En la realización de este proyecto utilizamos AES para la encriptación y MD5 para generar huellas digitales, estos no son los únicos métodos, sin embargo, en esta ocasión fueron los que mejor encajaban.

En cuanto al ujo de mensaje a través de los sockets tenemos un solo problema que quedo por resolver y es que cuando un usuario agrega un contacto la aplicación del cliente se boque en espera de un mensaje el cual supuestamente se envía no obstante no siempre es recibido por el socket del cliente.

Capítulo 5

Bibliografía

- https://www.boxcryptor.com/es/cifrado
- http://sociedad.elpais.com/sociedad/2011/08/17/actualidad/1313532009_85021
- 5.html http://pamarke.com/como-funciona-el-cifrado-aes/
- Interprocess Communications in Linux®: The Nooks & Crannies By John Shapley Gray

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
Ingeniería en Ciencias de la Computación
Modelos de Redes

Proyecto Final

Graciela Gaona Bernabé Erika
Leonor Basurto Munguía Lucia
Adriana Merlo Haro

Cesar Manuel Rodríguez Mendiola

25

26/11/2014

Índice general

Capítulo 1

Antecedentes teóricos

1.1. Sockets

Los sockets son una forma de comunicación entre procesos que se encuentran en diferentes máquinas de una red.

Proporcionan un punto de comunicación por el cual se puede enviar o recibir información entre procesos.

Tienen un ciclo de vida dependiendo si son sockets de servidor, que esperan a un cliente para establecer una comunicación, o socket cliente que busca a un socket de servidor para establecer la comunicación.

Los sockets empleados para la comunicación deberán ser del mismo tipo.

Hay dos tipos básicos de socket:

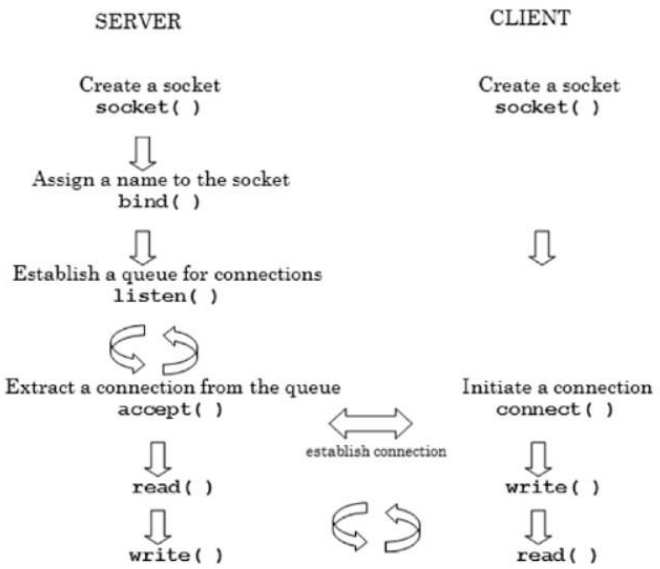
1.1.1. Stream sockets (sockets de ujo)

Estos sockets estan orientados a conexión.

Son seguros.

Cuando se utilizan estos, los datos se entregan en orden, es decir, en la misma secuencia en la que se envió. No hay duplicación de datos, además permite la comprobación de errores y el control de ujo.

Los datos se transmiten como un ujo de bytes. De una manera muy limitada, estas tomas también permiten al usuario colocar un mensaje urgente prioridad más alta por delante de los datos de la corriente actual.



Esquema de sockets orientadoa a conexión.

1.1.2. Datagram sockets (sockets de datagramas)

Estos sockets NO están orientados a conexión.

Son potencialmente poco able.

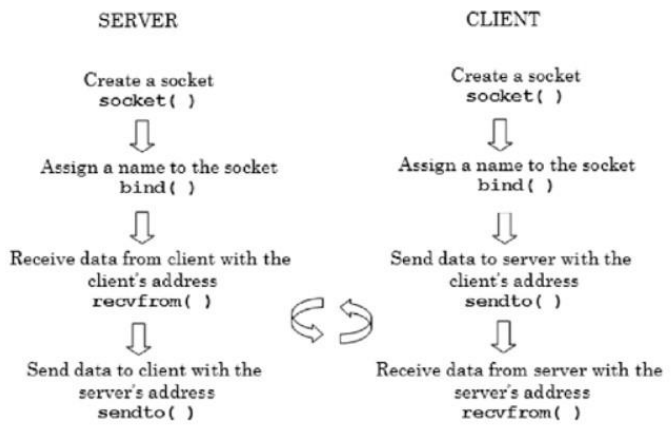
Los datos recibidos estaran fuera de orden.

No hay una conexión lógica entre los procesos de envío y recepción.

Cada datagrama se envía y se procesa independientemente. Datagramas individuales pueden tomar diferentes rutas para el mismo destino.

Con servicio sin conexión, no hay control de ujo. El control de errores, cuando se especi ca, es mínima.

Los paquetes de datagramas suelen ser pequeños y de tamaño jo.



Esquema de sockets no orientadoa a conexión.

1.2. Advanced Encryption Standard (AES)

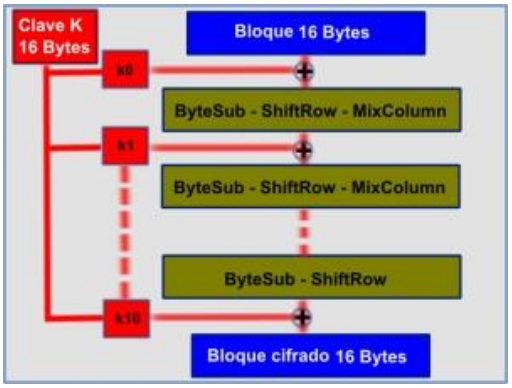
Su historia de éxito comenzó 1997, cuando el Instituto Nacional de Estándares y Tecnología, National Institute of Standards and Technology (NIST), anunció la búsqueda de un sucesor para el estándar de cifrado DES. Un algoritmo llamado "Rijndael", desarrollado por los criptólogos belgas Joan Daemen y Vincent Rijmen, fue destacado en seguridad, así como en el rendimiento y la exibilidad. Este algoritmo le ganó a varios competidores, y fue o cialmente presentado como el nuevo estándar de cifrado AES en el 2001 y se transformó en estándar efectivo en el 2002.

1.2.1. Funcionamiento

El algoritmo se basa en varias sustituciones, permutaciones y transformaciones lineales, ejecutadas en bloques de datos de 16 bytes - por lo que se le llama blockcipher. Estas operaciones se repiten varias veces, llamadas "rondas". En cada ronda, un único roundkey se calcula de la clave de encriptación, y es incorporado en los cálculos. Basado en esta estructura de bloque de AES, el cambio de un solo bit, ya sea en la clave, o en los bloques de texto simple y claro, resulta en un bloque de texto cifrado/encriptado completamente diferente - una clara ventaja sobre cifrados de ujo tradicionales.

AES utiliza claves de 128, 192 o 256 bits de longitud, sobre bloques – jos de 16 bytes, lo que da como resultado bloques cifrados en la salida

del mismo tamaño que en la entrada, 16 bytes. En la imagen superior podemos ver un esquema que muestra el funcionamiento básico de este algoritmo con claves de 128 bits.



Funcionamiento del AES.

1.2.1.1. Inicio

Uno de los puntos destacados del AES es la manera en la que se aplica su clave K. En primer lugar, ésta se expande en un subconjunto formado por (k0,k1,k2,... kn), a cada una de ellas se le aplica una función round r(kn,), que realiza una operación binaria XOR con el mensaje, es decir, el bloque de entrada es cifrado por cada una de las subclaves r(kn)hasta llegar al nal.

En cada round, se llevan a cabo diferentes funciones de sustitución y permutación, por lo tanto se cambia el orden y estado inicial de los datos incluidos en el mensaje inicial. Este proceso debe ser reversible, para que el sistema sea capaz de desencriptar el mensaje.

1.2.1.2. Función round

ByteSub. Aquí cada byte del actual estado es sustituido por su correspondiente, según una tabla S de búsqueda.

1. ShiftsRows. Cada la del state es deslizada un número de posiciones determinado.

2. MixColumns. Se efectúa una transformación lineal a cada una de las columnas.

1.2.1.3. Salida

Finalmente, después de aplicar 10 rounds sobre los 16 bytes del bloque de entrada, se obtiene en la salida un bloque de igual tamaño, 16 bytes. De hecho, tanto el mensaje como las round keys mantienen una longitud de 128 bits durante todo el proceso.

1.2.2. Nivel de seguridad

El estándar de cifrado (encriptación) avanzado AES, Advanced Encryption Standard (AES), es uno de los algoritmos más seguros y más utilizados hoy en día - disponible para uso público. Está clasi - cado por la Agencia de Seguridad Nacional, National Security Agency (NSA), de los Estados Unidos para la seguridad más alta de información secreta.

Este popular estándar se usa en operaciones tan importantes como pagos electrónicos o transacciones bancarias.

El método más común de ataque hacia un cifrador por bloques consiste en intentar varios ataques sobre versiones del cifrador con un número menor de rondas. El AES tiene 10 rondas para llaves de 128 bits, 12 rondas para llaves de 192 bits, y 14 rondas para llaves de 256 bits. Hasta 2005, los mejores ataques conocidos son sobre versiones reducidas a 7 rondas para llaves de 128 bits, 8 rondas para llaves de 192 bits, y 9 rondas para llaves de 256 bits.

A modo de ejemplo: Descifrar una clave de 128 bits AES con una supercomputadora estándar del momento, llevaría más tiempo que la presunta edad del universo. ½Boxcryptor utiliza incluso claves de 256! Hasta el día de hoy, no existe posible ataque contra AES. Por lo tanto, sigue siendo el estándar AES de cifrado preferido por los gobiernos, los bancos y los sistemas de alta seguridad de todo el mundo.

En 2002, un ataque teórico, denominado "ataque XSL", fue anunciado por Nicolas Courtois y Josef Pieprzyk, mostrando una potencial debilidad en el algoritmo AES. Varios expertos criptográ cos han encontrado problemas en las matemáticas que hay por debajo del ataque propuesto, sugiriendo que los autores quizá hayan cometido un error en sus estimaciones. Si esta línea de ataque puede ser tomada contra AES, es una cuestión todavía abierta. Hasta el momento, el ataque XSL contra AES parece especulativo; es improbable que nadie pudiera llevar a cabo en la práctica este ataque.

1.3. MD5

Utilizaremos el algoritmo MD5 (Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje5) de 128 bits para encriptar la contraseña proporcionada por el usuario, a la cual se le añadirá un mensa je aleatorio generado por el servidor. Una vez encriptada se enviará al servidor que la almacenara en una base de datos.

Cuando un usuario introduce su contraseña, esta se encripta en MD5 y se compara con la base de datos; de esta forma la contraseña no está almacenada en ningún sitio además de que MD5 no permite desencriptarla.

En este tipo de algoritmo por Digestión de mensajes o Hash, se crean "Huellas Digitales" que son claves o llaves que representan un documento o conjunto de datos a partir de un mensaje y mediante ciertas operaciones matemáticas; cumplen con las siguientes características:

-Es incomprensible a simple vista.

-Cada huella es única para cada mensaje.

-Dos huellas son iguales si y solo si el mensaje original también lo es.

-Es unidireccional, es decir que no se puede reconstruir el mensaje original a partir de su huella digital.

- Facilidad de empleo e implementación.

-Su salida es una huella digital, de tamaño jo e independiente de la dimensión del documento original.

En java se usa la clase MessageDigest para obtener la encriptación MD5 y así hacer uso de las funciones criptográ cas en la plataforma Java.

1.3.1. Algoritmo

Descripción del algoritmo md5

Empezamos suponiendo que tenemos un mensaje de 'b' bits de entrada, y que nos gustaría encontrar su resumen. Aquí 'b' es un valor arbitrario entero no negativo, no tiene que ser múltiplo de ocho, y puede ser muy largo. Imaginemos los bits del mensaje escritos así:

m0 m1 ... m{b-1}

Los siguientes cinco pasos calculan el resumen del mensaje.

1.3.1.1. Adición de bits

El mensaje será extendido hasta que su longitud en bits sea congruente con 448, módulo 512. Esto es, si se le resta 448 a la longitud del mensaje tras este paso, se obtiene un múltiplo de 512. Esta extensión se realiza siempre, incluso si la longitud del mensaje es ya congruente con 448, módulo 512.

La extensión se realiza como sigue: un solo bit "1" se añade al mensaje, y después se añaden bits "0" hasta que la longitud en bits del mensaje extendido se haga congruente con 448, módulo 512. En todos los mensajes se añade al menos un bit y como máximo 512.

1.3.1.2. Longitud del mensaje

Un entero de 64 bits que represente la longitud 'b' del mensaje (longitud antes de añadir los bits) se concatena al resultado del paso anterior. En el supuesto no deseado de que 'b' sea mayor que 2^64, entonces sólo los 64 bits de menor peso de 'b' se usarán.

En este punto el mensaje resultante (después de rellenar con los bits y con 'b') se tiene una longitud que es un múltiplo exacto de 512 bits. A su vez, la longitud del mensaje es múltiplo de 16 palabras (32 bits por palabra). Con M[0 ... N-1] denotaremos las palabras del mensaje resultante, donde N es múltiplo de 16.

1.3.1.3. Inicializar el búfer MD

Un búfer de cuatro palabras (A, B, C, D) se usa para calcular el resumen del mensaje. Cada letra representa un registro de 32 bits. Estos registros se inicializan con los siguientes valores hexadecimales, los bytes de menor peso primero:

palabra A: 01 23 45 67

palabra B: 89 ab cd ef

palabra C: fe dc ba 98

palabra D: 76 54 32 10

1.3.1.4. Procesado del mensaje en bloques de 16 palabras

De nimos cuatro funciones auxiliares que toman como entrada tres palabras de 32 bits y su salida es una palabra de 32 bits.

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$

$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$

$$H(X,Y,Z) = X \oplus Y \oplus Z$$

$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

En cada posición de cada bit X actúa como un condicional: si X, entonces Z sino Y. La función Z podría haber sido de nida usando + en lugar de v ya que XY y not(x) Z nunca tendrán unos ('1') en la misma posición de bit.

Las funciones G, H e I son similares a la función F, ya que actúan

bit a bit en paralelo" para producir sus salidas de los bits de X, Y y Z, en la medida que si cada bit correspondiente de X, Y y Z son independientes y no sesgados, entonces cada bit de G(X,Y,Z), H(X,Y,Z) e I(X,Y,Z) serán independientes y no sesgados.

Este paso usa una tabla de 64 elementos T[1 ... 64] construida con la función Seno. Denotaremos por T[i] el elemento i-ésimo de esta tabla, que será igual a la parte entera del valor absoluto del seno de

'i' 4294967296 veces, donde.....

Código del MD5:

/* Procesar cada bloque de 16 palabras. */

para i = 0 hasta N/16-1 hacer

/* Copiar el bloque 'i' en X. */

para j = 0 hasta 15 hacer

hacer X[j] de M[i*16+j].

n para /* del bucle 'j' */

/* Guardar A como AA, B como BB, C como CC, y D como DD. */

/* Ronda 1. */

/* [abcd k s i] denotarán la operación a = b + ((a + F(b, c, d) + X[k] + T[i]) < < < s). */

/* Hacer las siguientes 16 operaciones. */

[ABCD 0 7 1] [DABC 1 12 2] [CDAB 2 17 3] [BCDA 3 22 4]

[ABCD 4 7 5] [DABC 5 12 6] [CDAB 6 17 7] [BCDA 7 22 8]

[ABCD 8 7 9] [DABC 9 12 10] [CDAB 10 17 11] [BCDA 11 22 12]

[ABCD 12 7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15 22 16]

/* Ronda 2. */

/* [abcd k s i] denotarán la operación

a = b + ((a + G(b, c, d) + X[k] + T[i]) < < < s). */

/* Hacer las siguientes 16 operaciones. */ [ABCD 1 5 17] [DABC 6 9 18] [CDAB 11 14 19] [BCDA 0 20 20] [ABCD 5 5 21] [DABC 10 9 22] [CDAB 15 14 23] [BCDA 4 20 24] [ABCD 9 5 25] [DABC 14 9 26] [CDAB 3 14 27] [BCDA 8 20 28] [ABCD 13 5 29] [DABC 2 9 30] [CDAB 7 14 31] [BCDA 12 20 32]

/* Ronda 3. */

/* [abcd k s t] denotarán la operación

a = b + ((a + H(b, c, d) + X[k] + T[i]) < < < s). */

/* Hacer las siguientes 16 operaciones: */

[ABCD 5 4 33] [DABC 8 11 34] [CDAB 11 16 35] [BCDA 14 23 36]

[ABCD 1 4 37] [DABC 4 11 38] [CDAB 7 16 39] [BCDA 10 23 40]

[ABCD 13 4 41] [DABC 0 11 42] [CDAB 3 16 43] [BCDA 6 23 44] [ABCD 9 4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA 2 23 48]

/* Ronda 4. */

/* [abcd k s t] denotarán la operación

a = b + ((a + I(b, c, d) + X[k] + T[i]) < < < s). */

/* Hacer las siguientes 16 operaciones. */

[ABCD 0 6 49] [DABC 7 10 50] [CDAB 14 15 51] [BCDA 5 21 52] [ABCD 12 6 53] [DABC 3 10 54] [CDAB 10 15 55] [BCDA 1 21 56]

[ABCD 8 6 57] [DABC 15 10 58] [CDAB 6 15 59] [BCDA 13 21 60]

[ABCD 4 6 61] [DABC 11 10 62] [CDAB 2 15 63] [BCDA 9 21 64]

/* Ahora realizar las siguientes sumas. (Este es el incremento de cada uno de los cuatro registros por el valor que tenían antes de que este bloque fuera inicializado.) */

A = A + AA

B = B + BB

C = C + CC

D = D + DD

1.3.1.5. Salida

El resumen del mensaje es la salida producida por A, B, C y D. Esto es, se comienza el byte de menor peso de A y se acaba con el byte de mayor peso de D.

Capítulo 2

Desarrollo

El desarrollo de esta aplicación será en un entorno grá co (netbeans), para facilitar la creación de la interfaz grá ca con la cual el nalmente el usuario tendrá que interactuar. Así mismo este entorno trabaja sobre la plataforma de java lo cual nos facilitara el desarrollo además cuenta con las librerías para encriptar mediante AES y MD5.

2.1. Conectividad

Para lograr una conectividad e ciente implementaremos los sockets orientados a conexión (Stream socket), así mismo aremos uso de los canales (Stream) para el ujo de información. Para que la aplicación soporte conexiones de multiples usuarios es necesario

2.2. Base de datos

Se necesita una base de datos para llevar el control de los usuarios que se registran, pero también llevar el control de los contactos que tiene el usuario.

Se determina que cada usuario debe de tener registrado un correo con el que se identi ca, password para poder ingresar a su cuenta, una llave que le sirve para su seguridad de inicio de sesión es decir

cada vez que intente iniciar sesión esta será la mezclar del password con el algoritmo MD5 que se compara con la que el usuario enviara , por lo cual esta puede ser nula ya que solo se usa cuando el usuario se logue y una Ip que le sirve para poderse conectar con algún usuario siempre y cuando este registrado.

Cada usuario puede o no tener contactos, cada contacto que desee agregar debe estar en la lista usuarios . Los contactos que agregan

en contactos se validan para evitar duplicación de datos, también las demás operaciones que las haga como usuario se deben de hacer como contacto.

2.2.1. Operaciones que se hace con la BD

2.2.1.1. Obtener todos los contactos de un determinado usuario

Select *from contactos where correo_usuario= _un _usuario_

Select *from contactos where correo_contacto =_un_usuario

2.2.1.2. validar correo

Se valida que el correo del usuario que no exista o un contacto que exista como usuario

Select correo_usuario from usuario where correo_usuario= _correo_nuevo_

2.2.1.3. Agregar mensaje

El mesaje se agregará cada que el usuario quiera tener conversación con alguno de sus contactos o haya tenido conversación.

Update contacto set mensaje = _mensaje_ where correo_usuario= _correo_usuario_ and correo_contacto= _correo_contacto_

Update contacto set mensaje = _mensaje_ where correo_usuario= _correo_usuario_ and correo_contacto= _correo_contacto_ or where correo_usuario= _correo_contacto_ and correo_contacto= _correo_usuario

2.2.1.4. Agregar un contacto

Permite agregar un contacto de un determinado usuario

Insert into contactos (correo_usuario, correo_contacto) values (_correo_usuario_ , _correo_contacto_)

2.2.1.5. Agregar un usuario

Insert into contactos (correo_usuario, password, ip) values (_correo_usuario_ , _correo_contacto_ , _ip_)

2.2.1.6. Obtener el password

Obtiene el password de un determinado usuario para iniciar sesión

Select password from usuario where correo_usuario= _correo_nuevo_

2.2.1.7. Obtener mensaje

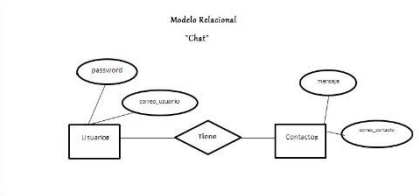
Se obtiene una determinada conversación;

Select mensaje contactos where correo_usuario= _un _usuario_ and correo_contacto=_correo_contacto_ or correo_usuario =_correo_contacto_ and correo_contato=_correo_usuario_

Capítulo 3

Resultados

3.1. Base de datos



Modelo entidad-relacion

Contactos

Field	Type	Null
correo_usuario	varchar(100)	No
correo_contacto	varchar(100)	No
mensaje	varchar(200)	yes

Tabla de contactos

Usuario

Field	Type	Null	key
correo_usuario	varchar(100)	No	
password	varchar(100)	no	

Tabla de usuarios

3.2. AES

Se inicializa el vector de bytes

```
byte[] menEncriptado = null;

Stringcadena="micccccccccccclavemuajaumicccccccccccclavemuajaumicccccccccccclavemu

byte[] cad=(cadena.getBytes());
```

Generamos una clave de 128 bits adecuada para AES

```
KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");

keyGenerator.init(128); Key key = keyGenerator.generateKey();
```

Alternativamente, una clave que queramos que tenga al menos 16 bytes y nos quedamos con los bytes 0 a 15

```
key = new SecretKeySpec(cad, 0, 16, "AES");
```

Se obtiene un cifrador AES

```
Cipher aes = Cipher.getInstance("AES/ECB/PKCS5Padding");
```

Se inicializa para encriptacion y se encripta el texto, que debemos pasar como bytes.

```
aes.init(Cipher.ENCRYPT_MODE, key);

byte[] encriptado = aes.doFinal(men.getBytes());
```

Se iniciliza el cifrador para desencriptar, con la misma clave y se desencripta

```
aes.init(Cipher.DECRYPT_MODE, key);

byte[] desencriptado = aes.doFinal(men);
```

Texto obtenido, igual al original.

```
menDesencriptado = new String(desencriptado);
```

3.3. MD5

```
import java.security.MessageDigest;
```

```
import java.security.NoSuchAlgorithmException;

public class TestEncriptarMD5 {

private static final char[] CONSTS_HEX = { '0','1','2','3','4','5','6','7','8','9','a','b','c','d','e','f
};

public static String encriptaEnMD5(String stringAEncriptar) {

try {

MessageDigest md = MessageDigest.getInstance("MD5");

byte[] bytes = md.digest(stringAEncriptar.getBytes());

StringBuilder strbCadenaMD5 = new StringBuilder(2 * bytes.length);
for (int i = 0; i < bytes.length; i++) {

int bajo = (int)(bytes[i] & 0x0f ); int alto = (int)((bytes[i] & 0xf0)
> > 4);
strbCadenaMD5.append(CONSTS_HEX[alto]);
strbCadenaMD5.append(CONSTS_HEX[bajo]);

}

return strbCadenaMD5.toString();

} catch (NoSuchAlgorithmException e) {

return null;

}
```

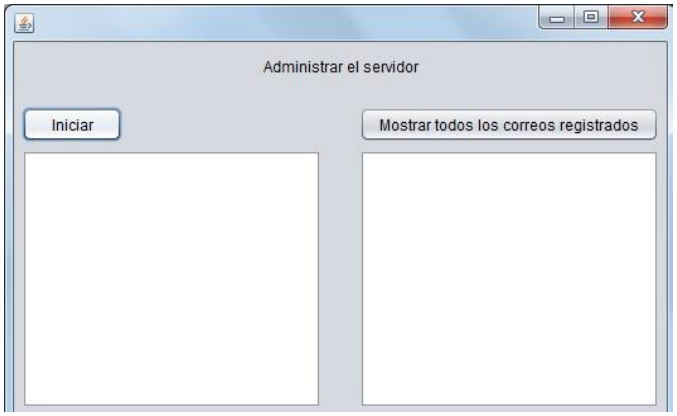
Se obtiene un MessageDigest con MessageDigest.getInstance() indicando que algoritmo de encriptación queremos (el parámetro MessageDigestAlgorithms.MD5),a continuación convertimos a bytes nuestro texto ("stringAEncriptar".getBytes()).

Ahora basta con pedirle el texto encriptado con el método digest(), este nos lo devolverá como un array de bytes. Como los bytes obtenidos pueden no ser legibles, los transformamos a números hexadecimales; esto para evitar los problemas típicos de codi cación de caracteres.

3.4. Ejecución de la aplicación

3.4.1. Servidor

Se presentan la interfaz que permite una administración basica del servidor.



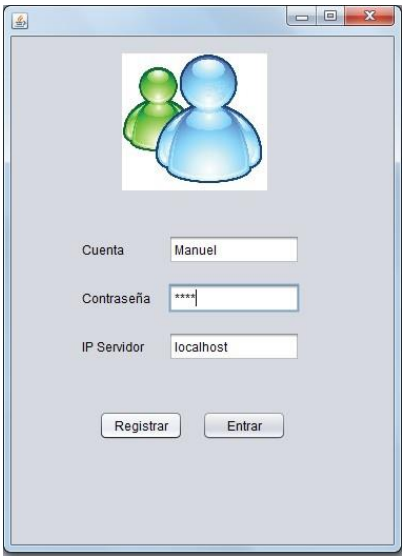
Interface inicial

En esta interface se puede inicalizar o detener el servidor, asi mismo podemos consultar todos los usuarios registrados en el chat con su respectiva conraseña.

3.4.2. Cliente

Inteface principal del cliente.

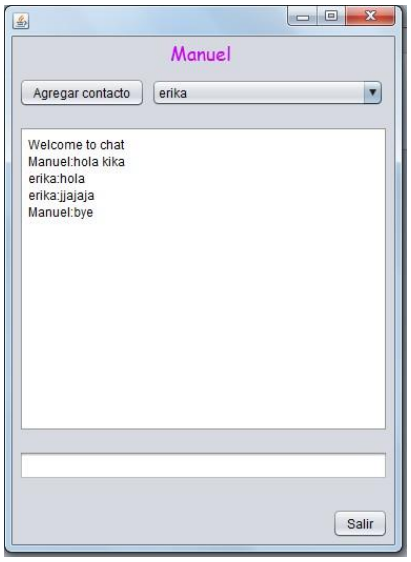
En esta interface el usuario podrá realizar su registro o en defecto podrá iniciar sesion en el chat, para cualquier de los casos anteriores, es necesario llenar todos los campos solicitados.



Interface inicial.

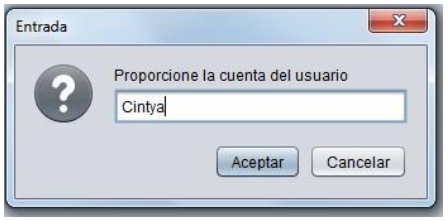
Noti cación de error

Esta es la inteface con la cual el usuario podrá enviar y recibir mensa jes. Además podra agregar un contacto.



Interface para conversaciones.

Si el usuario decea ingresar un nuevo contacto se le presentará el siguiente cuadro solicitado la cuenta(correo) del usuario que decea agragar a sus contactos.



En el caso de que el socket del cliente no encuentre un servidor de chat, se le noti cará al usuario como se muestra en la siguiente imagen.

3.5. Nota

Para poder enviar mensajes y que estos mismos queden almacenados en el historial de conversación de la base de datos Es estrictamente necesario que ambos clientes esten activos .

Capítulo 4

Conclusiones

El uso de bibliotecas para la encriptación es sencillo y de gran potencialidad para asegurar los paquetes que uyen por la red, y si el propósito es garantizar la integridad podemos emplear el uso de las huellas digitales.

En la realización de este proyecto utilizamos AES para la encriptación y MD5 para generar huellas digitales, estos no son los únicos métodos, sin embargo, en esta ocasión fueron los que mejor encajaban.

En cuanto al ujo de mensaje a través de los sockets tenemos un solo problema que quedo por resolver y es que cuando un usuario agrega un contacto.

Existe otro problema; tenemos un metodo que muestra al usuario la conversacion con uno de sus contactos, este metodo se ejecuta al inicio de la aplicación y cada vez que el usuario recibe un mensaje, intentamos llamar a este mismo metodo (Actualiza Historial) cada que el usuario seleccione un contacto pero por alguna razon desconocida no funciona correctamente, por ello decidimos omitir este paso.

Capítulo 5

Bibliografía

- https://www.boxcryptor.com/es/cifrado
- http://sociedad.elpais.com/sociedad/2011/08/17/actualidad/1313532009_850215.html
- http://pamarke.com/como-funciona-el-cifrado-aes/
- Interprocess Communications in Linux®: The Nooks & Crannies By John Shapley Gray

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Modelo de Redes

Doc. Olmos Pineda Iván

Proyecto Final

Manual de la aplicación

- Mensajero Exprés -

Fecha de entrega: jueves 27 de noviembre del 2014

Equipo:

Barragán Hernández Nancy

Mellado Robles David

Meneses Casarrubias Brian

Morelos Dorantes Iván

Vázquez Juárez Jamelli

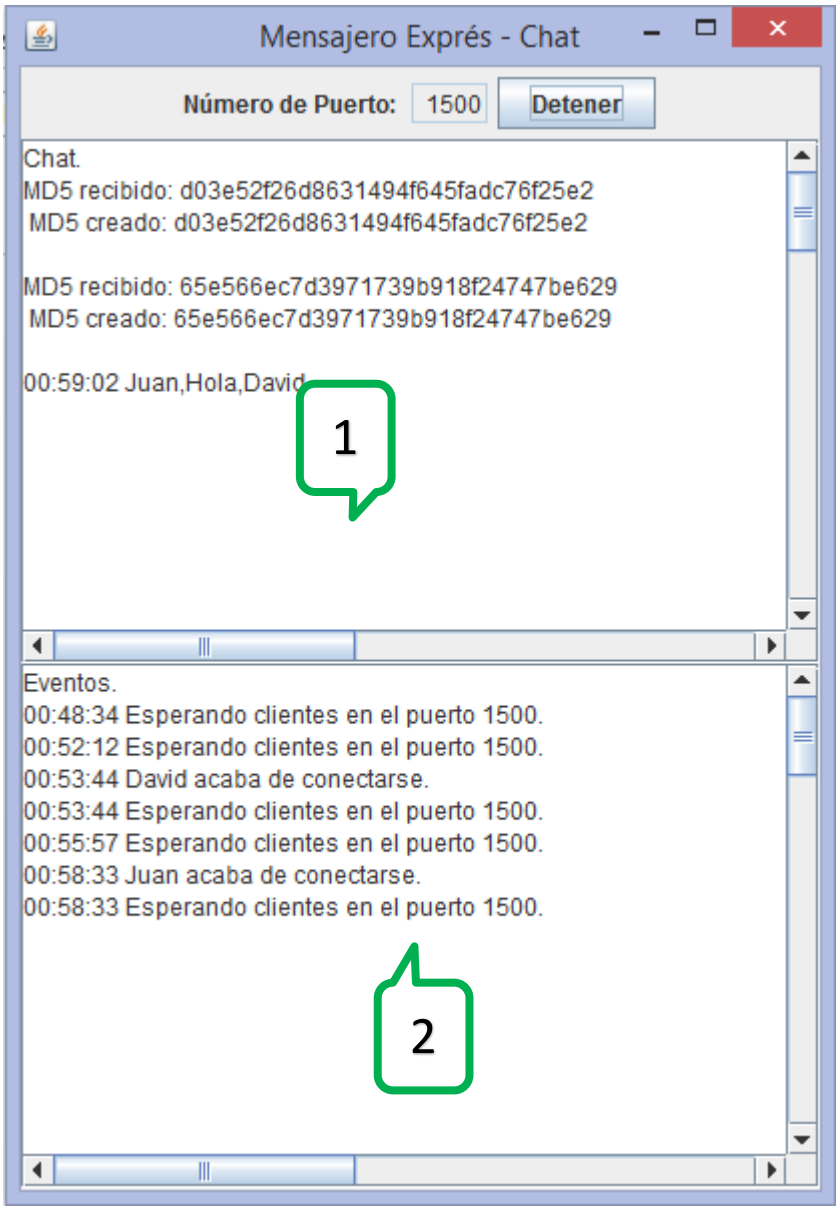
Contenido

1) Servidor
    a) Iniciar servidor.
    b) Iniciar cliente.
2) Cliente
    a) Conectarse.
    b) Registrarse.
    c) Iniciar Sesión.
    d) Envió de mensajes.

1) Servidor
    a) Iniciar servidor.

Para iniciar el servidor es necesario ejecutar la Clase “Principal”, básicamente tiene dos opciones las cuales realizan las siguientes tareas:

Inicia el servidor para poder gestionar las peticiones de los clientes como el registro, inicio de sesión y el envió de mensajes entre los clientes.



La imagen anterior muestra la interfaz gráfica del servidor una vez iniciada en el puerto seleccionado.

Esta ventana cuenta con dos partes, la primera muestra los MD5 recibidos y generados por cada cliente conectado así como los mensajes entre los clientes. La segunda parte muestra los eventos conexión registro y el momento en que se conecta cada cliente.

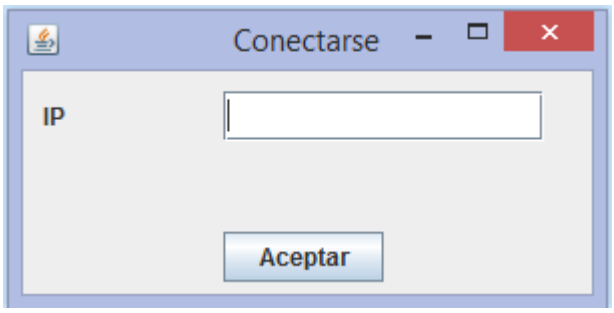
a) Iniciar cliente.

El botón anterior permite iniciar un cliente desde la misma aplicación que lanza el servido, esta parte se hablara con detalle en la parte del cliente.

2) Cliente
   a) Conectarse.

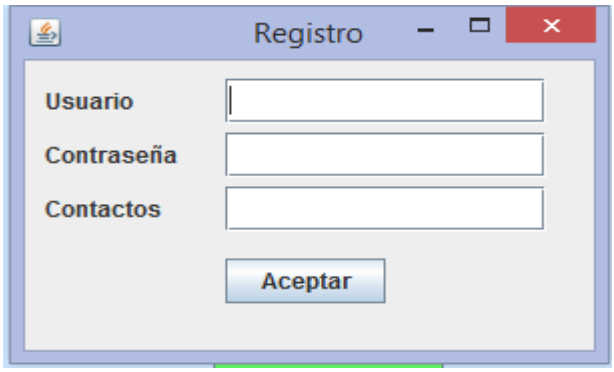
En esta ventana cuenta con 3 opciones principalmente Conectarse, Registrarse e Iniciar Sesión.

La opción Conectarse solicita la dirección IP donde se está ejecutando el servidor, al imagen siguiente ejemplifica el hecho



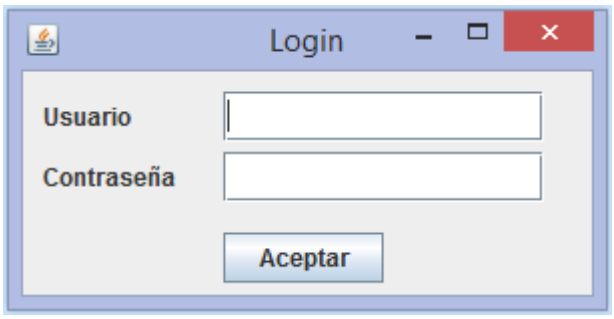
b) Registrarse.

La opción registrarse permite hacer un registro nuevo en el servidor pidiendo los siguientes datos nombre de usuario, contraseña y los nombres de sus contactos, la imagen siguiente ejemplifica lo anterior

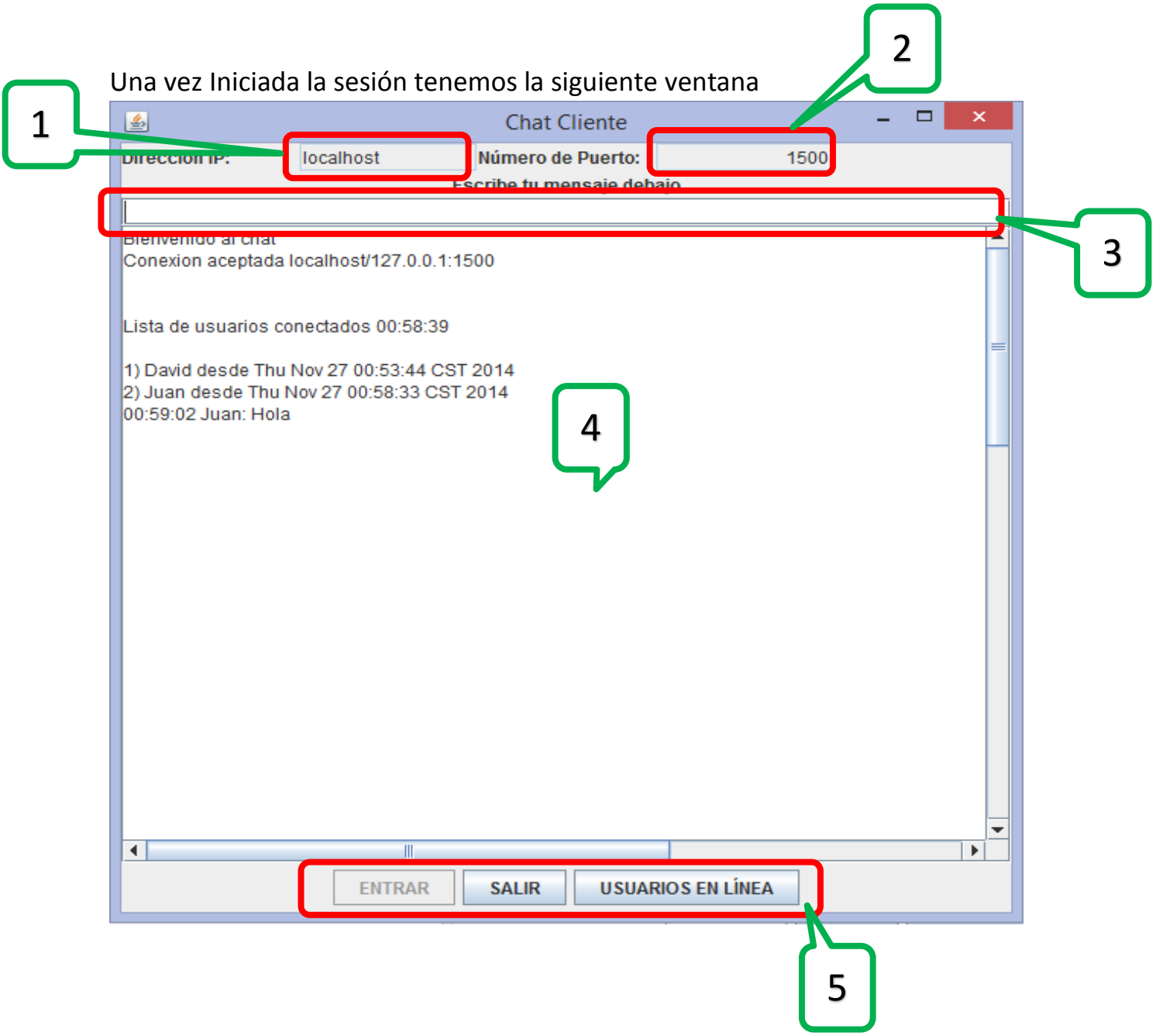


c) Iniciar Sesión.

La opción Iniciar Sesión permite hacer logueo en el servidor ejecutándose en la dirección IP anteriormente elegida, la imagen siguiente muestra los datos que nos solicitan la opción



Una vez Iniciada la sesión tenemos la siguiente ventana



1. Muestra la dirección IP del servidor al que el cliente está conectado.
2. Indica el puerto al que se encuentra conectado.

   d) Envió de mensajes.

3. Este capto de texto permite enviar mensajes a el contacto especificado de la siguiente forma:

   "Mensaje ","nombre de usuario"

El formato para el envío de mensajes en el campo de texto es el siguiente; mensaje a enviar seguido de una coma y a continuación el nombre del destinatario, por ultimo para enviarlo basta con pulsar enter.

4. En esta área de texto se muestran los mensajes enviados y recibidos por el cliente así como ciertos eventos generados por el cliente.
5. Por ultimo en esta área se localiza tres botones con las opciones de Entrar, Salir y Usuarios en Línea
   a) El Botón entrar permite empezar una sesión de chat.
   b) El Botón Salir permite salir permite terminar la sesión con el servidor.
   c) El Botón “Usuarios en Línea” muestra los usuarios conectado al servido en el momento.

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Modelo de Redes

Doc. Olmos Pineda Iván

Proyecto Final

Fecha de entrega: jueves 27 de noviembre del 2014

Equipo:

Barragán Hernández Nancy

Mellado Robles David

Meneses Casarrubias Brian

Morelos Dorantes Iván

Vázquez Juárez Jamelli

Índice

Introducción

La comunicación es fundamental para establecer buenas relaciones humanas, sin embargo, esta se ve afectada ya que requiere el entendimiento mutuo del emisor.

Existiendo un propósito para la comunicación y una respuesta por producirse, el comunicador desea que su comunicación tenga alta fidelidad. La palabra fidelidad es empleada aquí en el sentido de que el comunicador ha de lograr lo que desea.

Se logra una comunicación verdadera si estamos interesados en el lenguaje de la otra persona, de tal forma que esta se puede expresar libre y sinceramente, si escuchamos atentamente y observamos con conciencia y somos capaces de ponernos en el lugar del otro. Solo entonces estaremos estableciendo las bases de una buena comunicación.

Cabe destacar que el internet es una de las redes más grandes de telecomunicaciones a nivel mundial, su importancia radica en que a través de ella podemos obtener información rápida y eficaz sobre diversos temas, sin moverse de casa o del lado de su computador, esto entre infinidades de aplicaciones que podemos utilizar, entre estas aplicaciones está el chat, que designa una comunicación escrita realizada de manera instantánea mediante el uso de un software y a través de Internet entre dos, tres o más personas ya sea de manera pública a través de los llamados chats públicos (mediante los cuales cualquier usuario puede tener acceso a la conversación) o privada, en los que se comunican dos o más personas.

El chat sirve para comunicarse con grupos de personas las cuales opinan de diferentes temas, para cambiar información, entre otras cosas como simplemente saludar a los amigos.

En este trabajo se va a desarrollar un programa que sea capaz de permitir la comunicación persona a persona, garantizando un servicio seguro con encriptación de datos.

Marco Teórico

Encriptación: es el proceso mediante el cual cierta información o texto sin formato es cifrado de forma que el resultado sea ilegible a menos que se conozcan los datos necesarios para su interpretación. Es una medida de seguridad utilizada para que al momento de almacenar

o transmitir información sensible ésta no pueda ser obtenida con facilidad por terceros. Opcionalmente puede existir además un proceso de desencriptación a través del cual la información puede ser interpretada de nuevo a su estado original, aunque existen métodos de encriptación que no pueden ser revertidos.

Algunos de los usos más comunes de la encriptación son el almacenamiento y transmisión de información sensible como contraseñas, números de identificación legal, números de tarjetas de crédito, reportes administrativo-contables y conversaciones privadas, entre otros.

Como sabemos, en un Sistema de Comunicación de Datos, es de vital importancia asegurar que la Información viaje segura, manteniendo su autenticidad, integridad, confidencialidad y el no repudio de la misma entre otros aspectos.

HashMap: es una colección de objetos, (como los Arrays), pero estos no tienen orden. Cada objeto se identifica mediante algún identificador apropiado, por ejemplo un "uuid". El nombre HASH, hace referencia a una técnica de organización de archivos llamada hashing o "dispersión" en el cual se almacenan registros en una dirección del archivo que es generada por una función que se aplica sobre la llave del registro.

Socket: es un método para la comunicación entre un programa del cliente y un programa del servidor en una red. Un socket se define como el punto final en una conexión. Los sockets se crean y se utilizan con un sistema de peticiones o de llamadas de función a veces llamados interfaz de programación de aplicación de sockets (API, application programming interface).

Un socket es también una dirección de Internet, combinando una dirección IP (la dirección numérica única de cuatro partes que identifica a un ordenador particular en Internet) y un número de puerto (el número que identifica una aplicación de Internet particular, como FTP, Gopher, o WWW).

MD5 y Encriptación

MD5

MD5 es una técnica que permite obtener una huella digital de una serie de datos de entrada

- Con la huella, se puede verificar la integridad de los datos, es decir, comprobar que no hayan sufrido alteración
- Además, puede emplearse para la validación de usuarios o procesos, aunque no es la más usada, a comparación de otras técnicas como RSA.

Librerias utilizadas:

El **java.math.BigInteger** clase proporciona análogos de operaciones para todos los operadores enteros primitivos de Java y todos los métodos pertinentes de java.lang.Math.

También proporciona operaciones de la aritmética modular, el cálculo GCD, pruebas de primalidad, generación privilegiada, manipulación de bits, y algunas otras operaciones diversas. Todas las operaciones se comportan como si estuvieron representados BigIntegers en notación de complemento a dos.

La semántica de las operaciones aritméticas y operaciones lógicas bit a bit son similares a los de enteros operadores aritméticos de Java y los operadores enteros bit a bit de Java, respectivamente. La semántica de las operaciones de cambio se extienden los de operadores de desplazamiento de Java para permitir distancias de desplazamiento negativos.

Las operaciones de comparación realizan la firmaron comparaciones enteros. Operaciones aritméticas modulares se proporcionan para calcular los residuos, realizar la exponenciación, y calcular inversos multiplicativos. Operaciones de bits operan en un solo bit de la representación en complemento a dos de su operando.

Todos los métodos y constructores en este tiro NullPointerException clase cuando aprobaron una referencia de objeto null para cualquier parámetro de entrada.

Declaración de la clase

A continuación se presenta la declaración de java.math.BigInteger clase:

public class BigInteger

extends Número

implements Comparable < BigInteger >

Campo

Los siguientes son los campos para java.math.BigInteger clase:

static BigInteger UNO - La única constante BigInteger.

static BigInteger TEN - Los diez constante BigInteger.

static BigInteger ZERO - El cero constante BigInteger.

Contructor

- BigInteger (int signum, byte [] magnitud)

Traduce la representación signo-magnitud de un BigInteger en un BigInteger.

Esta clase MessageDigest proporciona aplicaciones de la funcionalidad de un mensaje algoritmo de resumen, como SHA-1 o SHA-256. Compendios de mensajes son funciones hash unidireccionales seguros que toman datos arbitraria tamaño y salida de un valor hash de longitud fija.

Un objeto MessageDigest comienza inicializado. Los datos se procesan a través de él utilizando los métodos actualizados. En cualquier punto de reinicio puede ser llamado para restablecer la digestión. Una vez que todos los datos que se actualizan se ha actualizado, una de las digerir métodos deberían ser llamados para completar el cálculo hash.

El digest method puede ser llamado una vez para un número dado de cambios. Después de digestión ha sido llamado, el objeto MessageDigest se restablece a su estado inicializado.

Las implementaciones son libres de implementar la interfaz Cloneable. Las aplicaciones cliente pueden probar cloneability intentando la clonación y agarrar la CloneNotSupportedException:

Tenga en cuenta que esta clase es abstracta y se extiende desde MessageDigestSpi por razones históricas. Los desarrolladores de aplicaciones sólo deberían tomar nota de los métodos definidos en este MessageDigest clase; todos los métodos de la superclase están

destinados a los proveedores de servicios criptográficos que deseen suministrar sus propias implementaciones de algoritmos de compendio de mensajes.

Se requiere que cada implementación de la plataforma Java para apoyar los siguientes estándares MessageDigest algoritmos:

- MD5
- SHA-1
- SHA-256

Estos algoritmos se describen en la sección MessageDigest de la Documentación Java Cryptography Standard Architecture Algoritmo Nombre. Consulte la documentación de la versión de su aplicación para ver si son compatibles con otros algoritmos.

Constructor

- protected MessageDigest (String algoritmo)

Crea un resumen del mensaje con el nombre del algoritmo especificado.

Métodos

- byte [] digest (byte [] input)

Realiza una actualización final de la digestión mediante la matriz de bytes, y luego completa el cálculo resumen.

- String toString ()

Devuelve una representación de cadena de este resumen del mensaje objeto.

```
import java.math.BigInteger;
import java.security.MessageDigest;


public static String getMD5(String input) {
   try {
     MessageDigest md = MessageDigest.getInstance("MD5");
```

```
      byte[] messageDigest = md.digest(input.getBytes());
      BigInteger number = new BigInteger(1, messageDigest);
      String hashtext = number.toString(16);
      // Now we need to zero pad it if you actually want the full 32 chars.
      while (hashtext.length() < 32) {
         hashtext = "0" + hashtext;
      }
      return hashtext;
   }
   catch (NoSuchAlgorithmException e) {
      throw new RuntimeException(e);
   }
}
```

¿Porque utilizamos estas librerías para la implementación de MD5?

Una de las razones por la cual elegimos las librerías **java.math.BigInteger y java.security.MessageDigest** para la implementación del algoritmo de MD5, es porque estas librerías son nativas de Java y ya que nuestro programa lo implementamos en el lenguaje Java, éstas se adecuan muy bien a él y nos da una manera fácil de importar las funciones que nos ofrecen esas librerías para la implementación del algoritmo de MD5.

Otra razón fue porque las funciones de esas librerías no son nada difíciles de utilizar, ya que toda la documentación de los métodos y atributos que nos ofrecen las podemos encontrar en la documentación de Java.

Encriptación

(Cifrado, codificación). La encriptación es el proceso para volver ilegible información considera importante. La información una vez encriptada sólo puede leerse aplicándole una clave.

Se trata de una medida de seguridad que es usada para almacenar o transferir información delicada que no debería ser accesible a terceros. Pueden ser contraseñas, números de tarjetas de crédito, conversaciones privadas, etc.

Para encriptar información se utilizan complejas fórmulas matemáticas y para desencriptar, se debe usar una clave como parámetro para esas fórmulas.

El texto plano que está encriptado o cifrado se llama criptograma.

Algoritmo criptográfico

En computación y criptografía un algoritmo criptográfico es un algoritmo que modifica los datos de un documento con el objetivo de alcanzar algunas características de seguridad como autenticación, integridad y confidencialidad.

Clasificación

Los algoritmos criptográficos se pueden clasificar en:

- **Criptografía simétrica o de clave secreta.** La criptografía simétrica (en inglés symmetric key cryptography), también llamada criptografía de clave secreta (en inglés secret key cryptography) o criptografía de una clave1 (en inglés single-key cryptography), es un método criptográfico en el cual se usa una misma clave para cifrar y descifrar mensajes. Las dos partes que se comunican han de ponerse de acuerdo de antemano sobre la clave a usar. Una vez que ambas partes tienen acceso a esta clave, el remitente cifra un mensaje usando la clave, lo envía al destinatario, y éste lo descifra con la misma clave.
- **Criptografía asimétrica o de clave pública**. La criptografía asimétrica (en inglés asymmetric key cryptography), también llamada criptografía de clave pública (en inglés public key cryptography) o criptografía de dos claves1 (en inglés two-key cryptography), es el método criptográfico que usa un par de claves para el envío de mensajes. Las dos claves pertenecen a la misma persona que ha enviado el mensaje. Una clave es pública y se puede entregar a cualquier persona, la otra clave es privada y el propietario debe guardarla de modo que nadie tenga acceso a ella. Además, los métodos criptográficos garantizan que esa pareja de claves sólo se puede generar una vez, de modo que se puede asumir que no es posible que dos personas hayan obtenido casualmente la misma pareja de claves.

Listado de algoritmos criptográficos

Algoritmos Simétricos:

- DES
- 3DES
- RC2
- RC4
- RC5
- IDEA
- AES
- Blowfish

Algoritmos Asimétricos

- Diffie-Hellman
- RSA
- ElGamal
- Criptografía de curva elíptica

Data Encryption Standard (DES)

Es un algoritmo de cifrado, es decir, un método para cifrar información, escogido como un estándar FIPS en los Estados Unidos en 1976, y cuyo uso se ha propagado ampliamente por todo el mundo. El algoritmo fue controvertido al principio, con algunos elementos de diseño clasificados, una longitud de clave relativamente corta, y las continuas sospechas sobre la existencia de alguna puerta trasera para la National Security Agency (NSA). Posteriormente DES fue sometido a un intenso análisis académico y motivó el concepto moderno del cifrado por bloques y su criptoanálisis.

Hoy en día, DES se considera inseguro para muchas aplicaciones. Esto se debe principalmente a que el tamaño de clave de 56 bits es corto; las claves de DES se han roto en menos de 24 horas. Existen también resultados analíticos que demuestran debilidades teóricas en su cifrado, aunque son inviables en la práctica. Se cree que el algoritmo es seguro en la práctica en su variante de Triple DES, aunque existan ataques teóricos.

El algoritmo como estándar

A pesar de la polémica, DES fue aprobado como estándar federal en noviembre de 1976, y publicado el 15 de enero de 1977 como FIPS PUB 46, autorizado para el uso no clasificado de datos. Fue posteriormente confirmado como estándar en 1983, 1988 (revisado como FIPS-46-1), 1993 (FIPS-46-2), y de nuevo en 1998 (FIPS-46-3), éste último definiendo "TripleDES" (véase más abajo). El 26 de mayo de 2002, DES fue finalmente reemplazado por AES (Advanced Encryption Standard), tras una competición pública (véase Proceso de Advanced Encryption Standard). Hasta hoy día (2006), DES continúa siendo ampliamente utilizado.

Descripción

DES es el algoritmo prototipo del cifrado por bloques — un algoritmo que toma un texto en claro de una longitud fija de bits y lo transforma mediante una serie de operaciones básicas en otro texto cifrado de la misma longitud. En el caso de DES el tamaño del bloque es de 64 bits. DES utiliza también una clave criptográfica para modificar la transformación, de modo que el descifrado sólo puede ser realizado por aquellos que conozcan la clave concreta utilizada en el cifrado. La clave mide 64 bits, aunque en realidad, sólo 56 de ellos son empleados por el algoritmo. Los ocho bits restantes se utilizan únicamente para comprobar la paridad, y después son descartados. Por tanto, la longitud de clave efectiva en DES es de 56 bits, y así es como se suele especificar.

Estructura básica

La estructura básica del algoritmo aparece representada en la Figura 1: hay 16 fases idénticas de proceso, denominadas rondas. También hay una permutación inicial y final denominada PI y PF, que son funciones inversas entre sí (PI "deshace" la acción de PF, y viceversa). PI y PF no son criptográficamente significativas, pero se incluyeron presuntamente para facilitar la carga y descarga de bloques sobre el hardware de mediados de los 70. Antes de las rondas, el bloque es dividido en dos mitades de 32 bits y procesadas alternativamente. Este entrecruzamiento se conoce como esquema Feistel.

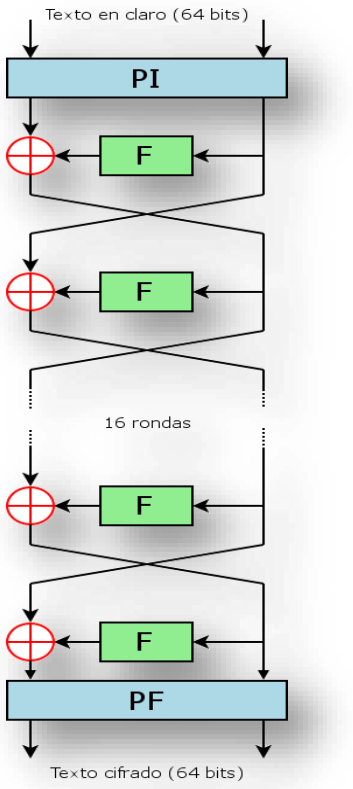


Figura 1- La estructura general de Feistel en DES

La estructura de Feistel asegura que el cifrado y el descifrado sean procesos muy similares — la única diferencia es que las subclaves se aplican en orden inverso cuando desciframos. El resto del algoritmo es idéntico. Esto simplifica enormemente la implementación, en especial sobre hardware, al no haber necesidad de algoritmos distintos para el cifrado y el descifrado.

El símbolo rojo "⊕" representa la operación OR exclusivo (XOR). La función-F mezcla la mitad del bloque con parte de la clave. La salida de la función-F se combina entonces con la otra mitad del bloque, y los bloques son intercambiados antes de la siguiente ronda. Tras la última ronda, las mitades no se intercambian; ésta es una característica de la estructura de Feistel que hace que el cifrado y el descifrado sean procesos parecidos.

La función (F) de Feistel

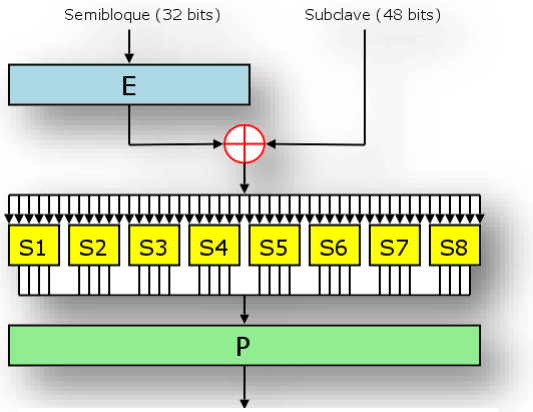


Figura 2 - La función Feistel (Función-F) de DES.

La función-F, representada en la Figura 2, opera sobre medio bloque (32 bits) cada vez y consta de cuatro pasos:

1. **Expansión** — la mitad del bloque de 32 bits se expande a 48 bits mediante la permutación de expansión, denominada E en el diagrama, duplicando algunos de los bits.
2. **Mezcla** — el resultado se combina con una subclave utilizando una operación XOR. Dieciséis subclaves — una para cada ronda — se derivan de la clave inicial mediante la generación de subclaves descrita más abajo.
3. **Sustitución** — tras mezclarlo con la subclave, el bloque es dividido en ocho trozos de 6 bits antes de ser procesados por las S-cajas, o cajas de sustitución. Cada una de las ocho S-cajas reemplaza sus seis bits de entrada con cuatro bits de salida, de acuerdo con una trasformación no lineal, especificada por una tabla de búsqueda. Las S-cajas constituyen el núcleo de la seguridad de DES — sin ellas, el cifrado sería lineal, y fácil de romper.
4. **Permutación** — finalmente, las 32 salidas de las S-cajas se reordenan de acuerdo a una permutación fija; la P-caja
   Alternando la sustitución de las S-cajas, y la permutación de bits de la P-caja y la expansión-E proporcionan las llamadas "confusión y difusión" respectivamente, un concepto identificado por Claude Shannon en los 40 como una condición necesaria para un cifrado seguro y práctico.

Generación de claves

La Figura 3 representa la generación de claves para el cifrado — el algoritmo que se encarga de proporcionar las subclaves. Primero, se seleccionan 56 bits de la clave de los 64 iniciales mediante la Elección Permutada 1 (PC-1) — los ocho bits restantes pueden descartarse o utilizarse como bits de comprobación de paridad. Los 56 bits se dividen entonces en dos mitades de 28 bits; a continuación cada mitad se trata independientemente. En rondas sucesivas, ambas mitades se desplazan hacia la izquierda uno o dos bits (dependiendo de cada ronda), y entonces se seleccionan 48 bits de subclave mediante la Elección Permutada 2 (PC-2) — 24 bits de la mitad izquierda y 24 de la derecha. Los desplazamientos (indicados por "<<<" en el diagrama) implican que se utiliza un conjunto diferente de bits en cada subclave; cada bit se usa aproximadamente en 14 de las 16 subclaves.

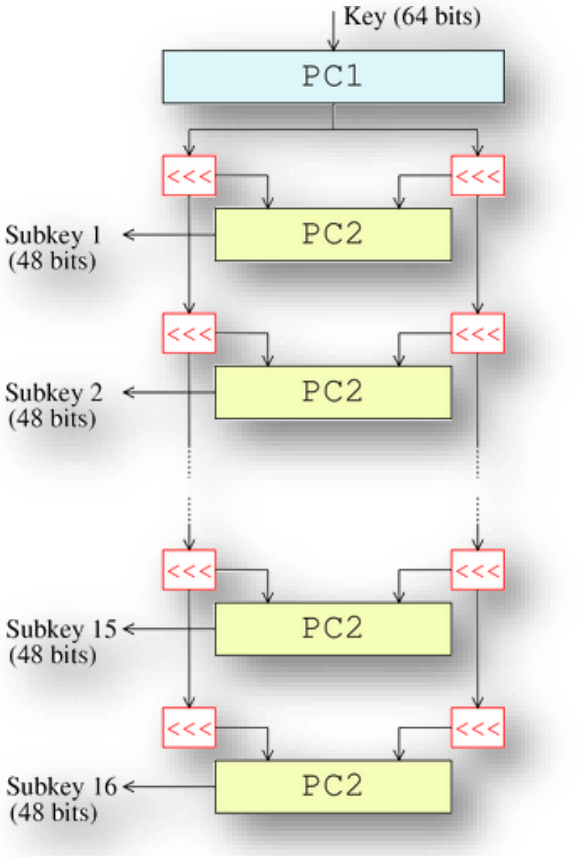


Figura 3 - La generación de clave de DES.

La generación de claves para descifrado es similar — debe generar las claves en orden inverso.

Encriptar y Desencriptar claves en Java

Java cuenta con una variedad de librerías y algoritmos para realizar la encriptación y desencriptación de datos, estas librerías se encuentran dentro de los paquetes: javax.crypto y java.security.

Dentro de los algoritmos de Criptografía más populares que podemos encontrar:

- Blowfish
- DES
- DESede
- PBEWithMD5AndDES
- PBEWithMD5AndTripleDES
- TripleDES

¿Por qué usar DES ante los otros algoritmos de encriptación?

Aunque en la actualidad el algoritmo DES es considerado inseguro en sus inicios fue aprobado como estándar federal en los Estado Unidos además de ser hacerse públicos los detalles del algoritmo se hicieron varios estudios de vulnerabilidad a nivel académico y por lo tanto se cuenta con bastante información acerca del funcionamiento del algoritmo. En la actualidad sigue teniendo gran popularidad y en Java cuenta con librerías nativas para implementación de aplicaciones que haga uso de este algoritmo.

En resumen se eligió DES por:

- Popularidad en la actualidad
- Librerías nativas en java

Desarrollo

Resultados

Conclusión

En un Sistema de Comunicación de Datos, es de vital importancia asegurar que la Información viaje segura, manteniendo su autenticidad, integridad, confidencialidad y el no repudio de la misma entre otros aspectos.

Estas características solo se pueden asegurar utilizando las Técnicas de Firma Digital Encriptada y la Encriptación de Datos, En este trabajo se usó la Encriptación de Datos.

Bibliografía

-A.S. Tanenbaum, "*Redes de Computadoras*", 4° Edición, Pearson Education, México, 2003.

-W. Stallings, "*Comunicaciones y Redes de Computadoras*", 7° Edición, Pearson Education, Madrid, 2004.

BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA

MODELO DE REDES

DR. IVAN OLMOS PINEDA

JUAN CARLOS GOMEZ MORALES

FECHA DE ENTREGA 26/09/14

INTRODUCCION

Latencia es el tiempo de posposición entre datos de envío y recepción. A diferencia de ancho de banda que teóricamente puede ser aumentado infinitamente, latencia rápidamente golpea una pared de ladrillo impuesta por las leyes de la física, en este caso la velocidad de la luz.

PROGRAMA (CODIGO)

```
import java.io.*;
import java.util.StringTokenizer;
class grafos
{
```

```
int NN;

int M[][] = new int[20][20];


public void escribematriz()
{
        int i,j;
        System.out.println(" NODOS = "+NN);
        System.out.println(" Matriz de Adyacencias");
        System.out.print(" ");
        for(i=1;i<=NN;i++)System.out.print("  "+i);
        System.out.println();

        for(i=1;i<=NN; i++)
        {
                System.out.print(i+"  ");
                for (j=1;j<=NN;j++)
                System.out.print(M[i][j]+"  ");
                System.out.println();
        }
}

//Lee archivo con los datos del grafo
```



```
public void LeeGrafo(String arch)
```

```
{

        FileInputStream fp;
        DataInputStream f;
        String linea = null;
        int token1,token2,i,j;
try
   {
        fp = new FileInputStream(arch);
        f = new DataInputStream(fp);
        linea=f.readLine();

        NN=Integer.parseInt(linea);
        System.out.println(" Numero de Nodos: "+NN);

        // Inicializamos la matriz con ceros

        for (i=1;i<=NN;i++)
            for(j=1;j<=NN; j++)
               M[i][j]=0;

        // Leemos el archivo línea por línea

        do {
             linea = f.readLine();
             if (linea!=null){
```

```
StringTokenizer tokens=new StringTokenizer(linea);
  token1 = Integer.parseInt(tokens.nextToken());
token2 = Integer.parseInt(tokens.nextToken());


 // escribimos en pantalla los datos leidos transformados en números

   System.out.println(token1+" "+token2);

 // almacenamos en la matriz

M[token1][token2]=1;
M[token2][token1]=1;
                                        }
}while(linea != null);
fp.close();
        }


        catch (FileNotFoundException exc)
        {
            System.out.println ("El archivo " + arch + " no fue encontrado ");
        }

        catch (IOException exc)
        {
```

```
            System.out.println (exc);
        }

    }

    public void grado ()
    {
                int i,j=0;
                int resultado=0;
        int fila[] = new int[100];
        for (i=1;i<=NN;i++){
                    for(j=1;j<=NN; j++){

                 if (M[i][j] == 1)
                        fila[i] = fila[i]+1;
                        //System.out.println("Imprime Filas"+i);
                  //System.out.println(fila[i]);

                    }
                 }
                //resultado[]fila[];
                resultado = fila[0];
for(i=1; i<fila.length; i++)
{
if(fila[i] > resultado)
{
```

```
resultado = fila[i];
}
}
System.out.println( resultado);
}

  public void llameGrado(){
        System.out.println("GRADO:");
        grado (M);
  }

  public static void main(String[] ar)
  {

    grafos G=new grafos();
    G.LeeGrafo("entrada.dat");
    G.escribematriz();
    G.llameGrado();
  }
  }
```

PROFESOR:

Ivan Olmos Pineda

INTEGRANTES:

Jordy Joaquin Cuan Robledo 201103624

Roberto Jimenez Coronado 201104380

Alejandro Quiroz Flores 201128811

Rafael Pérez Aguirre 201116854

Janeth Salas Gómez 201121267

06/11/14

Introducción

Hace algunas decadas no existian modelos o estandares para realizar el cableado de algun sistema de telecomunicaciones. Cada sistema tenia especificos requerimientos de acuerdo a las caracteristicas del cableado que utilizaran. Por ejemplo, los sistemas telefonicos utilizaban cables multipares, los equipos de computo requerian cableados especiales de las marcas de dichos equipos. Estos y más cables que utilizara el sistema compartian la misma instalación, con lo cual no habia protección de aislamiento ante las interferencias que pudieran surgir.

Con el paso del tiempo las tecnologias de los sistemas de información se hicieron más robustas, las empresas comenzaron a utilizar cada vez más de estos sistemas, todos ellos utilizaban distintos tipos de cables, y formas de instalación. Si la empresa necesitaba alguna modificacíon o, cambio de tecnologia, era necesario volver a cambiar toda la infraestructura.

A mediados de los 80 se comprendio la necesidad de realizar un estandar que contemplara todos los requerimientos de cableado para los sistemas de comunicaciones. En 1985 la CCIA (Computer Communications Industry Association) solicito a la EIA (Electronic Industries Alliance) efectuar un estandar pertinente a los sistemas de cableado.

El comité "TR-41" fue el encargado de desarrollar estandares para el cableado de los sistemas. Su objetivo fue desarrollar estandares que fueran independientes de las tecnogias de los sistemas de comunicaciones asi como de los fabricantes.

El trabajo que realizaron fue la realización y aceptación de estandares acerca de las infraestructuras de cableado para diferentes aplicaciones, las cuales estan categorizadas en normas comunes, normas locales y normas de componentes.

Planteamiento del problema

La facultad de Cs. De la Computación cuenta con 4 edificios.

- Edificio 104A: 23 cubiculos.

- Edificio 104B: sólo contiene un salón.
- Edificio 104C: tiene 4 niveles, un sotano y 3 pisos, de los cuales en el primer piso se encuentran 4 modulos, cada uno con 36 host. En el segundo nivel se encuentra el salón de usos multiples.
- Edificio 104D tiene 3 niveles, en cada nivel se encuentran 4 salones.

Nota: Los espacios de los edificios aquí presentadas no son los reales, son los propuestos por el profesor unicamente con fines practicos.

Justificación

Obedeciendo a la norma ANSI/TIA/EIA-568 contemplamos la utilizacion de fibra monomodo que permite la dispersión modal mínima, dicha de otro manera, permite tener un gran ancho de banda a grandes distancias. Este, en combinacion con cable UTP categoria 6A, se excederán en capacidad de canal para ofrecer la actual tecnoglogia por nuesotro proveedor de servicio de internet debido a los posibles cambios tecnologicos a futuro, los cuales podrán ser soportados por el sistema estructurado que en este documento proponemos.

Desarrollo

Cableado estructurado. La clasificacion de cableado estructurado que estamos proponiendo es de categoria 6A ya que solo estamos contemplando cable UTP 6A y fibra monomodo con una velocidad maxima de 1000Mbps que soporta topologias gigabit ethernet.

Para usos prácticos para la instalación del cableado, elejimos el estandar ANSI/TIA/EIA-568.

Pero antes de esto detallaremos algunos conceptos básicos.

Requisitos Generales

La estructura de cableado debe abordar:

- Voz
- Datos
- Vídeo
- Seguridad
- Plenario de audio para el hogar
- Sistema de control

Estándares de cableado eléctrico

ANSI-TIA-EIA 606

Esta norma establece las especificaciones para la administración de un cableado

La administración de los cableados requiere una excelente documentación

Debe permitir diferenciar por dónde viaja voz, datos, video, señales de seguridad, audio, alarmas, etcétera.

La documentación puede llevarse en papel, pero en redes complejas es mejor asistirse con una solución computarizada

Además, en ciertos ambientes se realizan cambios a menudo en los cableados, por esto la documentación debe ser fácilmente actualizable

ANSI-TIA-EIA 607

Esta norma especifican como se debe hacer la conexión del sistema de tierras (los sistemas de telecomunicaciones requieren puestas a tierra confiables).

Los gabinetes y los protectores de voltaje son conectados a una barra de cobre (busbar) con "agujeros" (de 2" x 1/4")

Estas barras se conectan al sistema de tierras (grounding backbone) mediante un cable de cobre cubierto con material aislante (mínimo número 6 AWG, de color verde o etiquetado de manera adecuada)

Este backbone estará conectado a la barra principal del sistema de telecomunicaciones (TMGB, de 4" x 1/4") en la acometida del sistema de telecomunicaciones. El TMGB se conectará al sistema de tierras de la acometida eléctrica y a la estructura de acero de cada piso.

Acometida de Entrada

Es la parte de la instalación de enlace que une la red de distribución de la empresa eléctrica con la instalación propia de la vivienda. Es propiedad de la empresa eléctrica y suele haber una por cada edificio. La acometida normal de una única vivienda es monofásica, de dos hilos, uno activo (fase) y el otro neutro, a 230 voltios, dependiendo del país. En el caso de un edificio de varias viviendas la acometida normal será trifásica, de cuatro hilos, tres activos o fases y uno neutro, siendo en este caso la tensión entre las fases 400 V y de 230 V entre fase y neutro.

Las acometidas pueden ser subterráneas o aéreas, dependiendo del tipo de distribución de la zona:

Acometida Aérea: Es la que va desde el poste hasta la vivienda, en recorrido visto, a una altura mínima de 6 m para el cruce de la calle.

Acometida Subterránea: Así se llama a la parte de la instalación que va bajo tierra desde la red de distribución pública hasta la unidad funcional de protección o caja, instalada en la vivienda.

Cuarto de Equipo

El cuarto de equipos es un espacio centralizado para los equipos de telecomunicaciones (Ej. PBX, Equipos de Cómputo, Switch), que sirven a los ocupantes del edificio. Este cuarto, únicamente debe guardar equipos directamente relacionados con el sistema de telecomunicaciones y sus sistemas de soporte. La norma que estandariza este subsistema es la EIA/TIA 569.

Cuarto de telecomunicaciones/comunicaciones

Un cuarto de telecomunicaciones es el área en un edificio utilizada para el uso exclusivo de equipo asociado con el sistema de cableado de telecomunicaciones. El espacio del cuarto de comunicaciones no debe ser compartido con instalaciones eléctricas que no sean de telecomunicaciones. El cuarto de telecomunicaciones debe ser capaz de albergar equipo de telecomunicaciones, terminaciones de cable y cableado de interconexión asociado. El diseño de cuartos de telecomunicaciones debe considerar, además de voz y datos, la incorporación de otros sistemas de información del edificio tales como televisión por cable (CATV), alarmas, seguridad, audio y otros sistemas de telecomunicaciones. Todo edificio debe contar con al menos un cuarto de telecomunicaciones o cuarto de equipo. No hay un límite máximo en la cantidad de cuartos de telecomunicaciones que pueda haber en un edificio.

Cableado Horizontal

El cableado Horizontal es el cableado que se extiende desde el armario de telecomunicaciones o Rack hasta la estación de trabajo. Es muy dificultoso remplazar el cableado Horizontal, por lo tanto es de vital importancia que se consideren todos los servicios de telecomunicaciones al diseñar el cableado Horizontal antes de comenzar

con él. Imagínese una situación en la cual usted a diseñado y construido una red, y en la práctica detecta que se produce gran cantidad de errores en los datos debido a un mal cableado. En esa situación usted debería invertir gran cantidad de dinero en una nueva instalación que cumpla con las normas de instalación de cableado estructurado vigente, lo que le asegura una red confiable.

El cableado horizontal deberá diseñarse para ser capaz de manejar diversas aplicaciones de usuario incluyendo:

- Comunicaciones de voz (teléfono).
- Comunicaciones de datos.
- Redes de área local.

El sistema de cableado horizontal incluye:

A) Los cables de empalme de interconexión (o puentes) que comprenden la terminación de conexión horizontal entre diferentes vías.

B) Cable que se extiende desde la toma hasta el rack (Cable Horizontal).

C) Toma de telecomunicaciones.

D) El cable perteneciente al área de trabajo.

E) Pese a que no pertenecer al cableado Horizontal se incluye en el gráfico, este es el cableado Backbone.

F) Terminaciones Mecánicas Figura -2-.

Cableado vertical

El cableado vertical (o de "backbone") es el que interconecta los distintos armarios de comunicaciones. Éstos pueden estar situados en plantas o habitaciones distintas de un mismo edificio o incluso en edificios colindantes. En el cableado vertical es usual utilizar fibra óptica o cable UTP, aunque en algunos casos se puede usar cable coaxial.

La topología que se usa es en estrella existiendo un panel de distribución central al que se conectan los paneles de distribución horizontal. Entre ellos puede existir sólo un panel intermedio.

En el cableado vertical están incluidos los cables del "backbone", los mecanismos en los paneles principales e intermedios, los latiguillos usados para el parcheo, los mecanismos que terminan el cableado vertical en los armarios de distribución horizontal

Área de trabajo

El área de trabajo comprende todo lo que se conecta a partir de la roseta de conexión hasta los propios dispositivos a conectar (ordenadores e impresoras fundamentalmente). Están también incluidos cualquier filtro, adaptador, etc., que se necesite. Éstos irán siempre conectados en el exterior de la roseta. Si el cable se utiliza para compartir voz, datos u otros servicios, cada uno de ellos deberá de tener un conector diferente en la propia roseta de conexión.

Al cable que va desde la roseta hasta el dispositivo a conectar se le llama latiguillo y no puede superar los 3 metros de longitud.

Cableado de Campus

Cables que conectan el distribuidor de campo con el edificio al que se distribuye. Los cables del backbone de campo pueden conectar también a los distribuidores del edificio.

Herramientas necesarias para la Acometida de Entrada

Nivel de burbuja - Alicates de corte, Plomada - Martillo o maceta de albañil, tenazas de albañil - Cortafríos - Cuerda de trazar Paletín de albañil - Llana y espátula - Barreño para yeso-Destornilladores – Pelahilos.

Materiales que se emplean en el cuarto de telecomunicaciones

Servidor, Router, Switch, Rack, Path Panel, Cross Connect, UTP Cat6a.

HERRAMIENTAS

Cable horizontal

RJ45, Ponchador RJ45, Cable horizontal para datos, Escalerilla, Cubierta o regleta, probadores.

Área de trabajo.

Computadora, Jumpers, Registros, Conector de energía, Teléfono, conector de voz y de datos, mesa.

ANSI/TIA/EIA-568 Cableado de telecomunicaciones para edificios comerciales

El estándar ANSI/TIA/EIA-568 y sus recientes actualizaciones especifican los requerimientos de un sistema integral de cableado, independiente de las aplicaciones y de los proveedores, para los edificios comerciales.

Se estima que la "vida productiva" de un sistema de cableado para edificios comerciales debe ser de 15 a 25 años. En este período, las tecnologías de telecomunicaciones seguramente cambien varias veces. Es por esto que el diseño del cableado debe prever grandes anchos de banda, y ser adecuado tanto a las tecnologías actuales como a las futuras.

El estándar especifica:

- Requerimientos mínimos para cableado de telecomunicaciones dentro de un ambiente de oficina, para distintas tecnologías de cables (cobre y fibra).
- Topología y distancias recomendadas.
- Parámetros de desempeño de los medios de comunicación (cables de cobre, fibra).

Este estándar ha tenido varias versiones; nosotros decidimos ocupar el último estándar publicado por la TIA:

ANSI/TIA/EIA 568-C. Es una revisión del ANSI/TIA/EIA 568-B, publicado entre 2001 y 2005. El nuevo estándar consolida los documentos centrales de las recomendaciones originales y todos los "adendum", pero cambia la organización, generando una recomendación "genérica" o "común" a todo tipo de edificios.

Está armado en varias partes:

- **ANSI/TIA/EIA 568-C.0**

  Tiene como objetivo permitir la planificación y la instalación de un sistema de cableado estructurado para todo tipo de instalaciones. Esta norma específica un sistema que soporte cableados de telecomunicaciones genéricos en un entorno multi-producto y multiproveedor. Varios de los conceptos originalmente indicados en la recomendación

Se definen los siguientes componentes:

- ❖ Subsistema de cableado 1 Es el cableado que se tiende desde las áreas de trabajo (escritorios) hasta el primer nivel de distribución, llamado "Distribuidor A" (por ejemplo, la sala de telecomunicaciones del piso en edificios comerciales).
- ❖ Subsistema de cableado 2 Es el cableado que se tiende desde el Distribuidor A hasta un segundo nivel de distribución, llamado "Distribuidor B".
- ❖ Subsistema de cableado 3 Es el cableado que se tiende desde el Distribuidor B hasta el distribuidor principal del edificio, llamado "Distribuidor C".
- ❖ Distribuidor A Es el primer nivel de distribución, donde se concentran las áreas de trabajo.
- ❖ Distribuidor B Es un nivel de distribución intermedio, entre el primer nivel de distribución y el distribuidor principal de cableado. En caso que el Distribuidor A no exista, las áreas de trabajo se conectan directamente a este distribuidor.
- ❖ Distribuidor C Es el distribuidor principal del edificio
- ❖ Equipo de salida (Equipment outlet) Lugar donde se ubican los puestos o áreas de trabajo, escritorios, etc.

- **ANSI/TIA/EIA 568-C.1**

  Provee información acerca del planeamiento, instalación y verificación de cableados estructurados para edificios comerciales. Los aspectos de la anterior recomendación ANSI/TIA/EIA 568-B.1 que aplican únicamente a este tipo de edificios fueron detallados y actualizados en esta nueva recomendación.

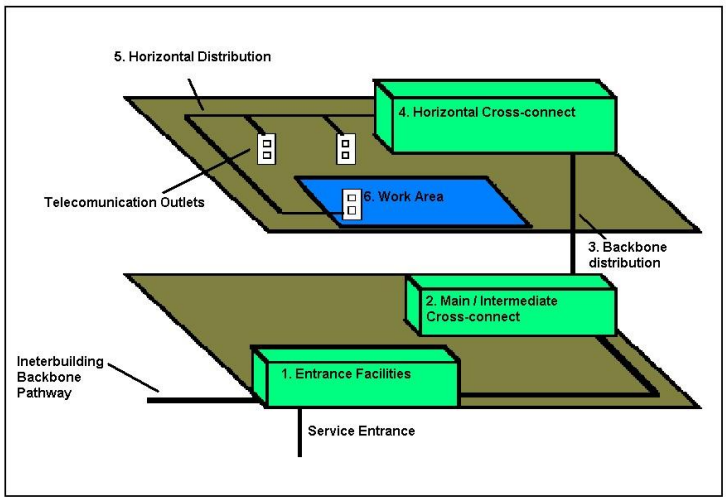
El estándar identifica seis componentes funcionales:

- Instalaciones de Entrada (o "Acometidas")
- Distribuidor o repartidor principal y secundarios (Main / Intermediate Cross - Connect)
- Distribución central de cableado ("Back-bone distribution")
- Distribuidores o repartidores Horizontales (Horizontal Corss-Connect)
- Distribución Horizontal de cableado (Horizontal Distribution)
- Áreas de trabajo
- Estos componentes se relacionas con los de la recomendación genérico 568-C.0, de la siguiente manera:

Nomenclatura 568-C.0	**Nomenclatura 568-C.1**
o Distributor C	Main Crossconnect (MC)
o Distributor B	Intermediate Crossconnect (IC)
o Distributor A	Horizontal Crossconnect (HC)

- Equipment Outlet — Telecommunication Outlet
- Cabling Subsystem 3 — Interbuilding Backbone Cabling
- Cabling Subsystem 2 — Intrabuilding Backbone Cabling
- Cabling Subsystem 1 — Horizontal Cabling

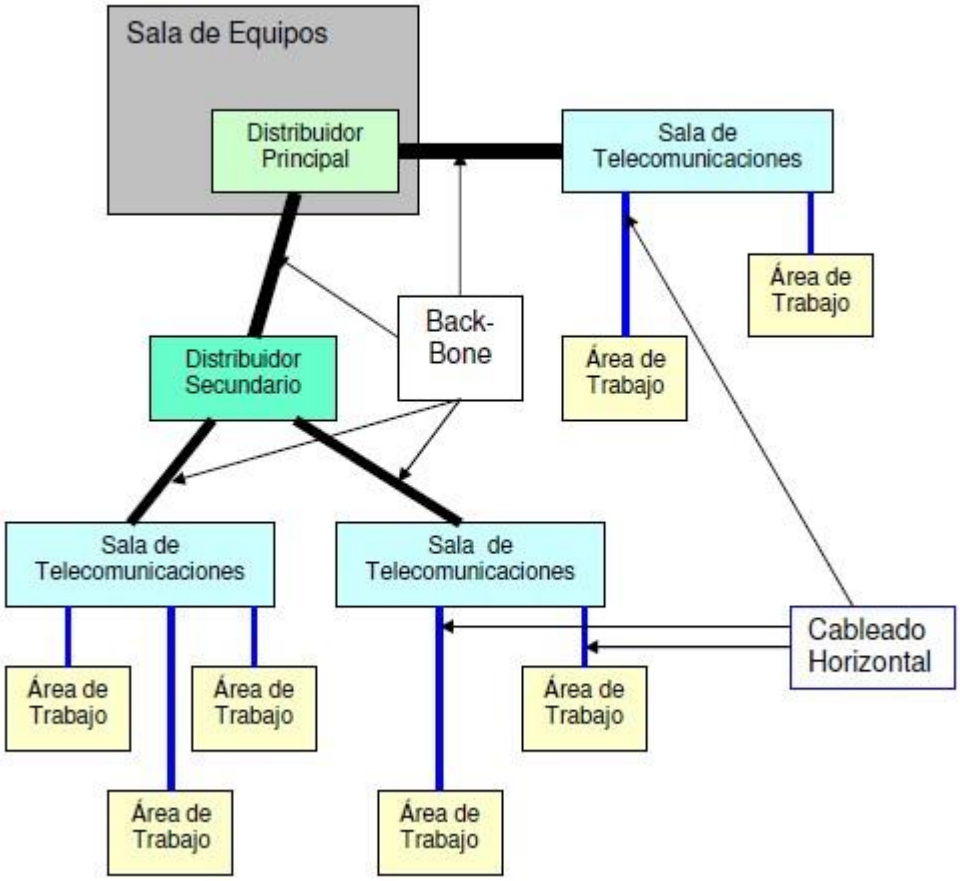
El siguiente diagrama muestra la ubicación de cada componente de la recomendación 568-C.1



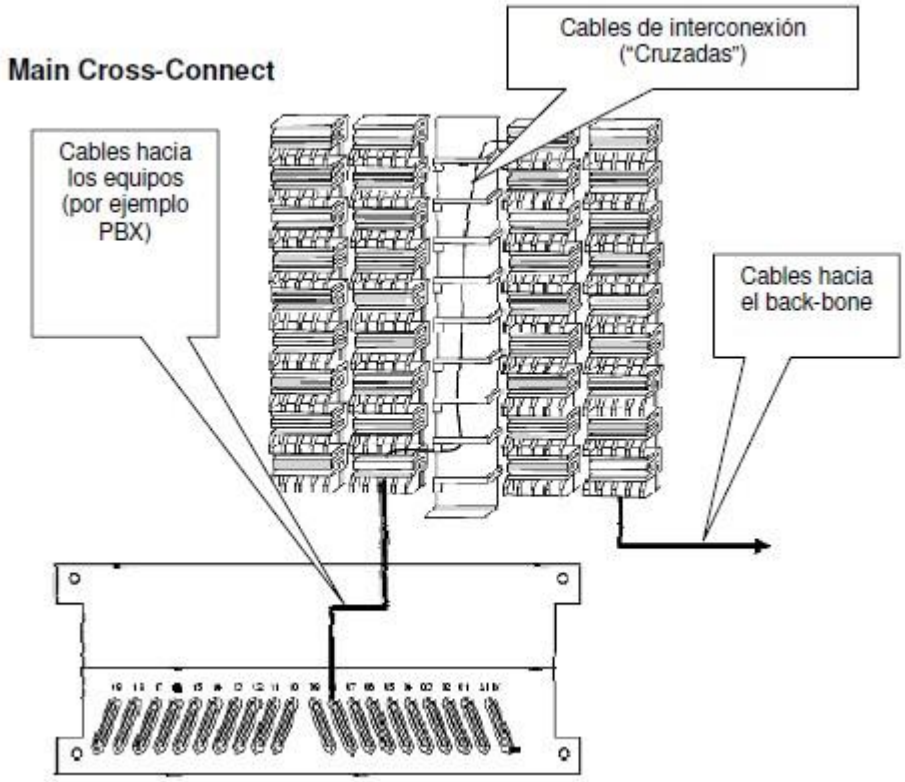
Distribuidor o repartidor principal y secundarios (Main / Intermediate Cross-Connect)

La estructura general del cableado se basa en una distribución jerárquica del tipo "estrella", con no más de 2 niveles de interconexión. El cableado hacia las "áreas de trabajo" parte de un punto central, generalmente la "Sala de Equipos". Aquí se ubica el Distribuidor o Repartidor principal de cableado del edificio. Partiendo de éste distribuidor principal, para llegar hasta las áreas de trabajo, el cableado puede pasar por un Distribuidor o Repartidor secundario y por una Sala de Telecomunicaciones.

El estándar no admite más de dos niveles de interconexión, desde la sala de equipos hasta la sala de Telecomunicaciones. Estos dos niveles de interconexión brindan suficiente flexibilidad a los cableados de back-bone.



El distribuidor o repartidor principal (a veces llamado MDF = "Main Distributoin Frame") puede estar constituido por "regletas", "patcheras" u otros elementos de interconexión. Generalmente está dividido en dos áreas, una a la que llegan los cables desde los equipos centrales (por ejemplo PBX) y otra a la que llegan los cables de distribución central (back-bone).



Distribución central de cableado ("Back-bone distribution")

Los sistemas de distribución central de cableado incluyen los siguientes componentes:

- Cables montantes
- Repartidores principales y secundarios
- Terminaciones mecánicas
- Cordones de interconexión o cables de cruzadas para realizar las conexiones entre distintos cables montantes

El esquema de la distribución central de cableado debe seguir la jerarquía en forma de estrella.

El estándar admite los siguientes cables para el Back-Bone:

- Cables UTP de 100 ohm (par trenzado sin malla)
- Cables de Fibra óptica multimodo de 50/125 µm
- Cables de Fibra ótpica multimodo de 62.5/125 µm
- Cables de Fibra ótpica monomodo

- Cable STP-A de 150 ohm (par trenzado con malla).

A diferencia de los servicios telefónicos clásicos, los servicios de datos (o de telefonía IP) generalmente no requieren de pares de cobre desde la sala de equipos. Este tipo de servicios generalmente puede soportarse mediante el tendido de Fibras Ópticas, desde la sala de equipos (o centro de cómputos) hasta los armarios de telecomunicaciones. Por esta razón, los tendidos de back-bone.

Generalmente se componen de cables UTP y de cables de Fibras ópticas, en número apropiada para las necesidades presentes y previsiones futuras.

Las distancias máximas para los cables montantes dependen de las aplicaciones (telefonía, datos, video, etc.) que deban transmitirse por ellas. Como reglas generales, el estándar establece las distancias máximas presentadas a continuación:

Tipo de Cable	Sala de Telecomunicaciones hasta Distribuidor Principal	Sala de Telecomunicaciones hasta Distribuidor Secundario	Distribuidor Secundario hasta Distribuidor Principal
UTP	800 m	300 m	500 m
Fibras ópticas Multimodo	2.000 m	300 m	1.700 m
Fibras ópticas Monomodo	3.000 m	300 m	2.700 m

Distribuidores o repartidores Horizontales (Horizontal Corss-Connect)

La función principal de los repartidores horizontales es la de interconectar los cables horizontales (ya sean de cobre o fibra óptica, provenientes de las áreas de trabajo) con los cables montantes (provenientes de la sala de equipos). Eventualmente, en la Sala de Telecomunicaciones, puede haber equipos de telecomunicaciones, los que son incorporados al repartidor horizontal para su interconexión hacia la sala de equipos (a través del back-bone) y/o hacia las áreas de trabajo (a través del cableado horizontal).

Di s posi ti vos de múl ti pl es c onec tores de tel ecomuni c aci ones (" Mul ti -User

Tel ec ommuni c ati ons Outlet Assembl y")

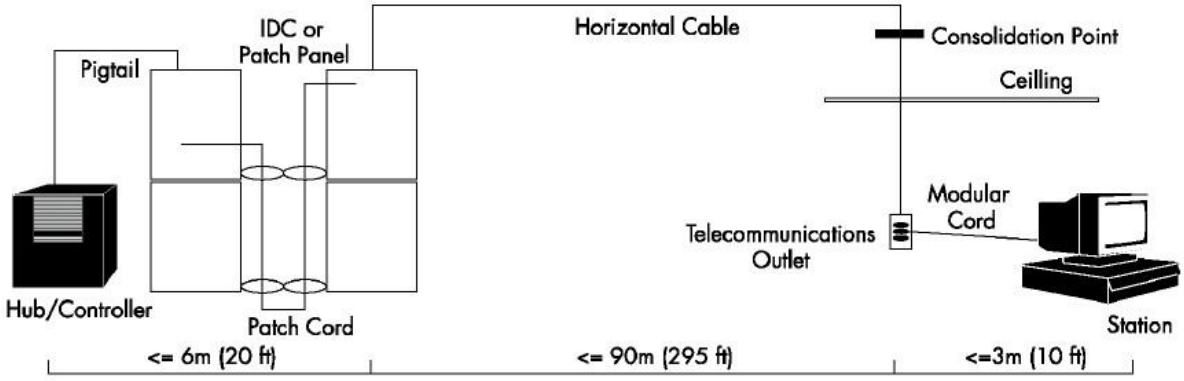
Un mismo "Dispositivo de múltiples conectores de telecomunicaciones" puede tener hasta 12 conectores. Las distancias máximas desde los "Dispositivo de múltiples conectores de telecomunicaciones" hasta las áreas de trabajo pueden variar, de acuerdo a las distancias de los cables horizontales que llegan a estos dispositivos, de manera que las distancia total ("punta a punta") no supere los 100 m. La siguiente tabla indica las distancias máximas admisibles, en función de los tramos marcados como "A", "B" y "C" en la figura anterior:

Tramo "A" (m)	Tramo "B" (m)	Tramo "C" (m)	Distancia total (m)
5	90	5	100
5	85	9	99
5	80	13	98
5	75	17	97
5	70	22	97

Puntos de consolidación

Los "puntos de Consolidación" son lugares de interconexión entre cableado horizontal proveniente del repartidor horizontal y cableado horizontal que termina en las áreas de trabajo o en los "Dispositivo de múltiples conectores de telecomunicaciones".

Dado que el cableado horizontal es "rígido", la idea es tener un punto intermedio que permita, en caso de reubicaciones de oficinas (y por lo tanto de áreas de trabajo), re-cablear únicamente parte del cableado horizontal (el que va desde el punto de consolidación hasta las nuevas áreas de trabajo).



Los puntos de consolidación, son útiles para prever futuros cambios en los lugares de las áreas de trabajo; la distancia total de cable, desde el área de trabajo, hasta el armario de telecomunicaciones (incluyendo el pasaje por el punto de consolidación) no debe exceder los 90 m.

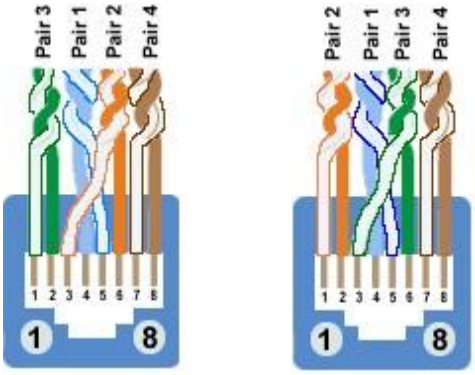
Áreas de trabajo

Se recomienda que la distancia del cordón de interconexión no supere los 5 m.

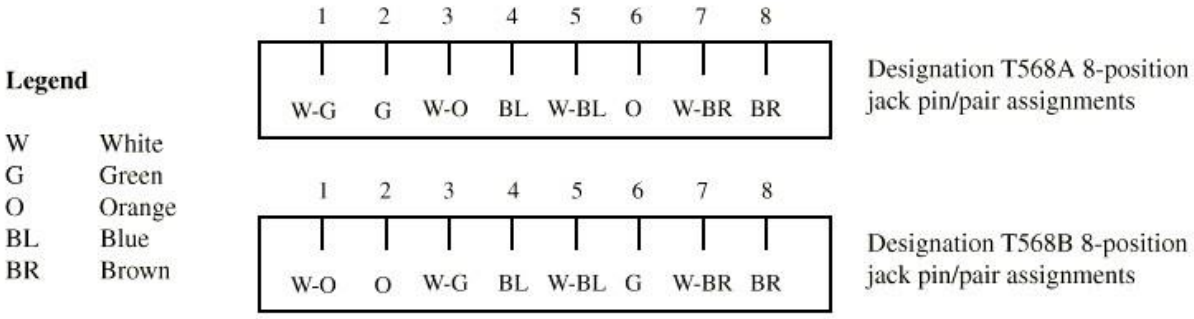
Los cables UTP son terminados en los conectores de telecomunicaciones en "jacks" modulares de 8 contactos, en los que se admiten dos tipos de conexiones, llamados T568A y T568B. Esta denominación no debe confundirse con el nombre de la norma ANSI/TIA/EIA 568-A o ANSI/TIA/EIA 568-B, ya que representan cosas bien diferentes. La

norma actualmente vigente es la ANSI/TIA/EIA 568-B, en la que se admiten dos formas de conectar los cables en los conectores modulares.

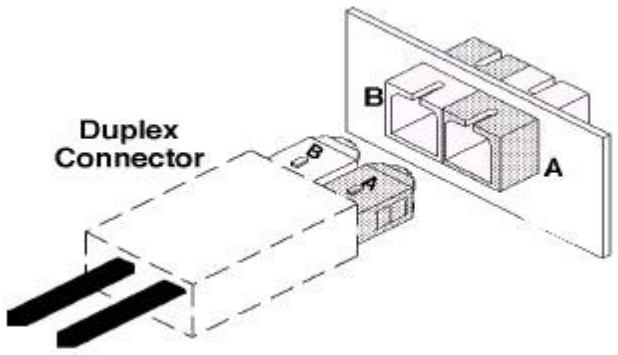
Estas dos formas de conexión son las que se denominan T568A y T568B.



T568A **T568B**



Los cables de fibra óptica son terminados en el área de trabajo en conectores dobles, es decir, que permiten la terminación de dos hilos de fibra.



Se recomienda utilizar el conector 568SC, pero se admiten otros tipos de conectores de dimensiones adecuadas. La figura muestra un conector del tipo 568SC y un cordón de interconexión de fibra óptica con su correspondiente terminación 568SC

- **ANSI/TIA/EIA 568-C.2**

Detalla los requerimientos específicos de los cables de pares trenzados balanceados, a nivel de sus componentes y de sus parámetros de transmisión

El estándar reconoce las categorías 3, 4, 5, 5e, 6 y 6ª de cable, nosotros elegimos la categoría:

Categoría 6A

La categoría 6A fue recientemente estandarizada, en marzo de 2008, en la recomendación TIA 568-B.2-10. Aplica a cables UTP de 100Ω y sus componentes de conexión, soportando aplicaciones de hasta 500MHz de ancho de banda, diseñado para 10 Giga bit Ethernet. Fue incluida dentro de la recomendación 568-C.

En marzo de 2007 fue aprobada la guía TIA/EIA TSB-155 [3], la que especifica métodos para evaluar el soporte de aplicaciones 10GBase-T en sistemas de cableados Categoría 6. Esta guía indica cómo realizar medidas en el rango extendido de frecuencias de 250 a 500 MHz., así como requerimientos adicionales de AXT (Alien Cross Talk) necesarios para soportar aplicaciones de 10 GBase-T.

Dado que los sistemas categoría 6 no fueron diseñados originalmente para llegar a estas frecuencias, las distancias máximas soportadas (en aplicaciones de hasta 500 MHz) pueden ser menores a 100 m (por ejemplo, se especifica que el rango de funcionamiento puede variar de 37 a 100m, dependiendo de varios factores).

En marzo de 2008 fue aprobada la recomendación ANSI/TIA/EIA 568-B.2-10, la que especifica la característica de los cables UTP y los componentes de conexión para trabajar a frecuencias de hasta 500 MHz., necesarios para soportar aplicaciones de 10

GBase-T, hasta 100 m de distancia.

Es de hacer notar que las categorías indican los parámetros de transmisión de los cables y los componentes de interconexión en función del "ancho de banda" medido en MHz, y no en bits por segundo.

Los cables reconocidos para el cableado horizontal deben tener 4 pares trenzados balanceados, sin malla (UTP = Unshielded Twisted Pair). Los conductores de cada par

deben tener un diámetro de 22 AWG a 24 AWG.

Características mecánicas de los cables para cableado horizontal

- El diámetro de cada cable no puede superar los 1.22 mm
- Los cables deben ser de 4 pares únicamente. No se admite para el cableado horizontal cables de más o menos pares. (Notar que si se admiten cables "multipares" para los backbones)
- Los colores de los cables deben ser los siguientes:
    - Par 1: Azul-Blanco , Azul **(W-BL)(BL)**
    - Par 2: Naranja-Blanco , Naranja **(W-O)(O)**
    - Par 3: Verde-Blanco , Verde **(W-G)(G)**
    - Par4: Marrón-Blanco , Marrón **(W-BR)(BR)**
- El diámetro completo del cable debe ser menor a 6.35mm
- Debe admitir una tensión de 400 N
- Deben permitir un radio de curvatura de 25.4 mm (1") sin que los forros de los cables sufran ningún deterioro.

Características eléctricas de los cables para cableado horizontal

- La resistencia "en continua" de cada conductor no puede exceder los 9.38Ω por cada 100 m a 20 °C.
- La diferencia de resistencias entre dos conductores del mismo par no puede

superar en ningún caso un 5%.

- La capacitancia mutua de cualquier par de cables, medida a 1 kHz no puede exceder los 6.6 nF en 100 m de cable para Categoría 3 y 5.6 nF en 100 m de cable para Categoría 5e.
- La capacitancia desbalanceada, entre cualquier cable y tierra, medida a 1 kHz, no puede exceder los 330 pF en 100 m de cable.
- La impedancia característica del cable debe ser de 100Ω +/- 15% en el rango de las frecuencias de la categoría del cable

- **ANSI/TIA/EIA 568-C.3**

Este estándar especifica las características de los componentes y los parámetros de transmisión para un sistema de cableado de fibra óptica (cables, conectores, etc)

incluyendo aspectos mecánicos, ópticos y requisitos de compatibilidad, para fibras multimodo de 50/125 µm y 62.5/125 µm y fibras monomodo.

Cables de Fibra Óptica

Los cables de fibra óptica pueden ser descritos como guías de onda para la luz.

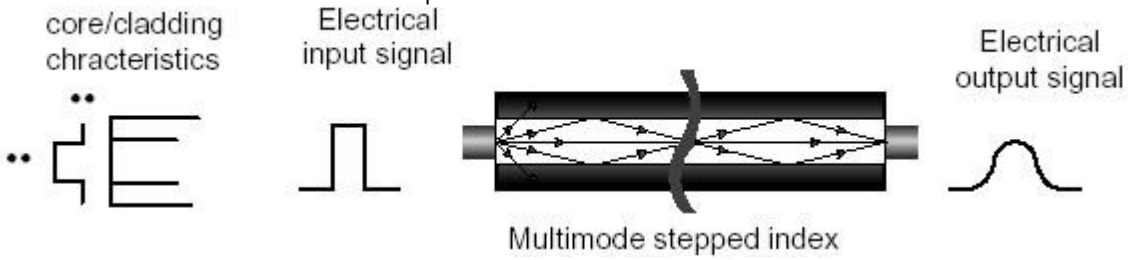
Son construidos con un núcleo de vidrio (o plástico para aplicaciones de distancias cortas) rodeado de un revestimiento también de vidrio ("cladding") con índice de refracción menor al núcleo.

Las fibras ópticas se categorizar en dos grupos:

- **Fibras Multimodo**

  La luz viaja dentro del núcleo de la fibra como una onda dentro de una guía de ondas. Las "ventanas" (longitudes de onda) y los materiales de las fibras se han elegido de manera que la luz forme "ondas estacionarias" dentro de la fibra.

  En fibras en las que el núcleo es suficientemente grande (del orden de los 50 µm) pueden existir varias ondas estacionarias, cada una en un "modo" de oscilación. Este tipo de fibras se conocen como "**multimodo**". Existen dos tecnologías de fabricación para este tipo de fibras. En la primera, hay una clara separación entre el núcleo y el cladding, como se muestra en la siguiente figura. El diámetro del núcleo está perfectamente determinado, y es del orden de los 50 µm. Este tipo de fibras se conocen como "Step Index".



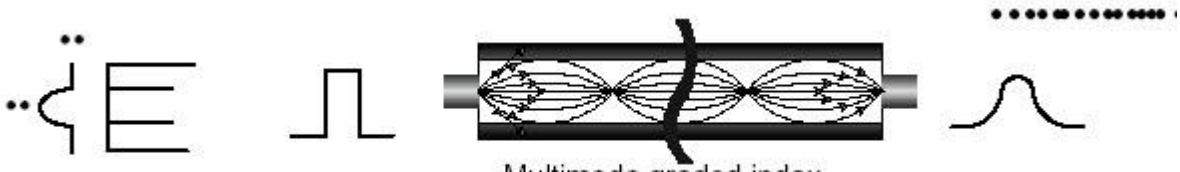
  Dado que la luz siempre está confinada dentro del núcleo, la velocidad es la misma para todos los modos; ésta dispersión es conocida como "dispersión modal" limita el ancho de banda utilizable de la fibra óptica.

Para mejorar esta situación, es posible fabricar fibras ópticas de "índice gradual". En estas fibras, el índice de refracción cambia en forma gradual, desde el núcleo hasta el cladding.

Dado que la velocidad de propagación depende del índice de refracción, en los momentos en los que la luz se encuentra más alejada del núcleo, se desplaza

más rápido. Esto compensa la diferencia de tiempos de los distintos "modos",

disminuyendo por lo tanto la dispersión modal y aumentando el ancho de banda utilizable de la fibra.



Multimode graded index

Las fibras multimodo comerciales se conocen generalmente por el diámetro del núcleo y el cladding. Las más comunes son 50/125 µm y 62.5/125 µm.

Las ventanas utilizadas en las fibras multimodo son las de 850 nm y 1300 nm, con emisores del tipo LED.

❖ Fibras Monomodo

Las fibras monomodo se diferencias de las multimodo esencialmente en el diámetro del núcleo. A diferencia de las multimodo, que tienen núcleos del orden

de los 50 µm, los núcleos de las fibras monomodo son de 8 a 9 µm. Estos diámetros

tan pequeños no permiten que la luz viaje en varios "modos", sino que solo puede existir un camino dentro del núcleo. Al existir únicamente un modo, la dispersión modal es mínima, lo que permite tener un gran ancho de banda aún a distancias grandes.

Las fibras monomodo comerciales tienen diámetros de 9/125 µm. Las ventanas utilizadas son las de 1300 nm y 1550 nm, con emisores del tipo LASER.

Dado que las fibras monomodo son más caras que las multimodo, al igual que los emisores requeridos, su uso se restringe generalmente a aplicaciones de grandes distancias (más de 50 km), siendo rara vez utilizadas dentro de edificios.

Características de transmisión

Según el estándar ANSI/TIA/EIA 568-B.3 Las cables de fibra óptica deben cumplir con los siguientes requerimientos:

Tipo de cable	Longitud de onda	Máxima atenuación (dB/km)	Mínima capacidad de transmisión de información (MHz . km)
Multimodo de 50/125 µm	850	3.5	500
	1300	1.5	500
Multimodo de 62.5/125 µm	850	3.5	160
	1300	1.5	500
Monomodo de interior	1310	1.0	N/A
	1550	1.0	N/A
Monomodo de interior	1310	0.5	N/A
	1550	0.5	N/A

Características físicas

Las cables de fibra óptica admitidos por ANSI/TIA/EIA 568-B.3 son multimodo de

50/125 µm y 62.5/125 µm y fibras monomodo.

Los cables para interiores deben soportar un radio de curvatura de 25 mm. Los cables de 2 o 4 hilos de interior, al momento de tenderlos, deben soportar una radio de

curvatura de 50 mm bajo una tensión de 222 N (50 lbf). Todos los cables deben soportar un radio de curvatura de 10 veces el diámetro externo del cable sin tensión y 15 veces

el diámetro externos bajo la tensión de tendido.

Los cables para exterior deben tener protección contra el agua y deben soportar una tensión de tenido mínima de 2670 N (600 lbf). Todos los cables de exterior deben soportar un radio de curvatura de 10 veces el diámetro externo del cable sin tensión y 20 veces el diámetro externos bajo la tensión de tendido.

Conectores

De acuerdo al estándar ANSI/TIA/EIA 568-B.3, los conectores para fibras multimodo deben ser de color beige. Los conectores para fibras monomodo deben ser de color azul.

El estándar tomo como ejemplo el conector 568SC, pero admite cualquier otro que cumpla las especificaciones mínimas.

Los conectores de fibra utilizan 2 "hilos" de fibra (ya que la transmisión sobre fibra es generalmente unidireccional. Cada hilo de fibra se termina en un conector, que deben estar claramente marcados como "A" y "B" respectivamente. Las cajas de conexión de fibra en las áreas de trabajo deben tener como mínimo 2 conectores, y deben permitir un radio de curvatura mínimo de 25 mm.

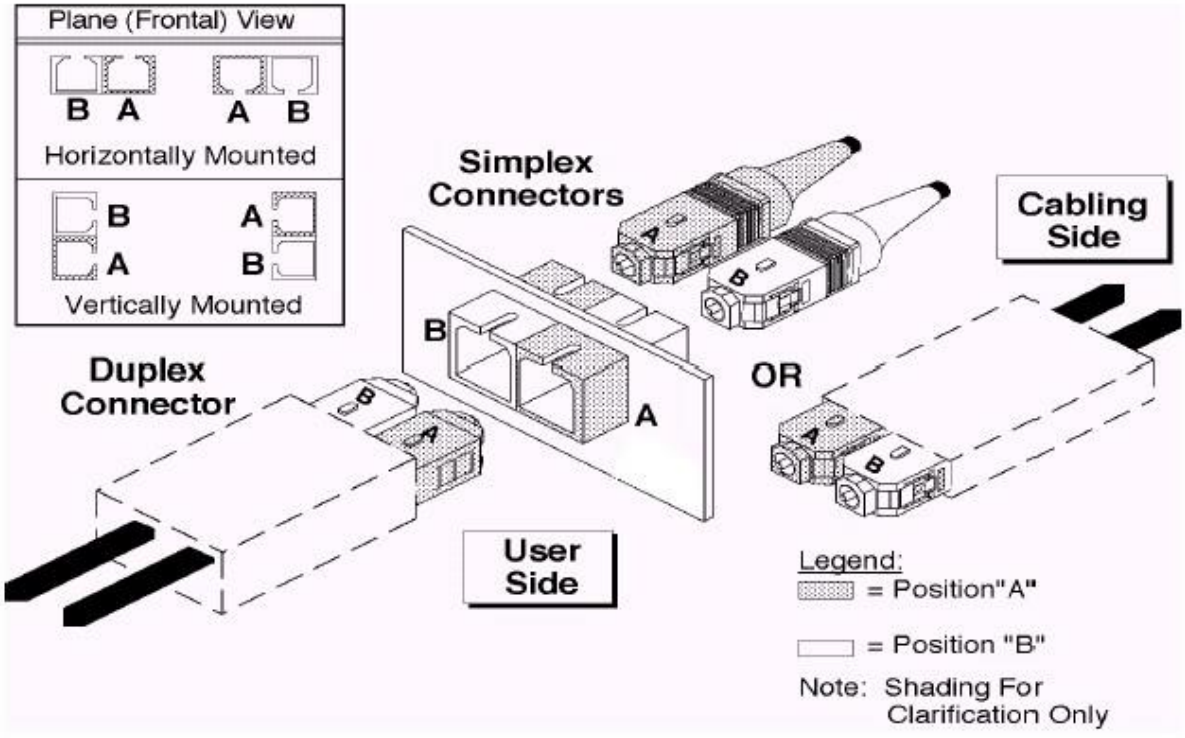
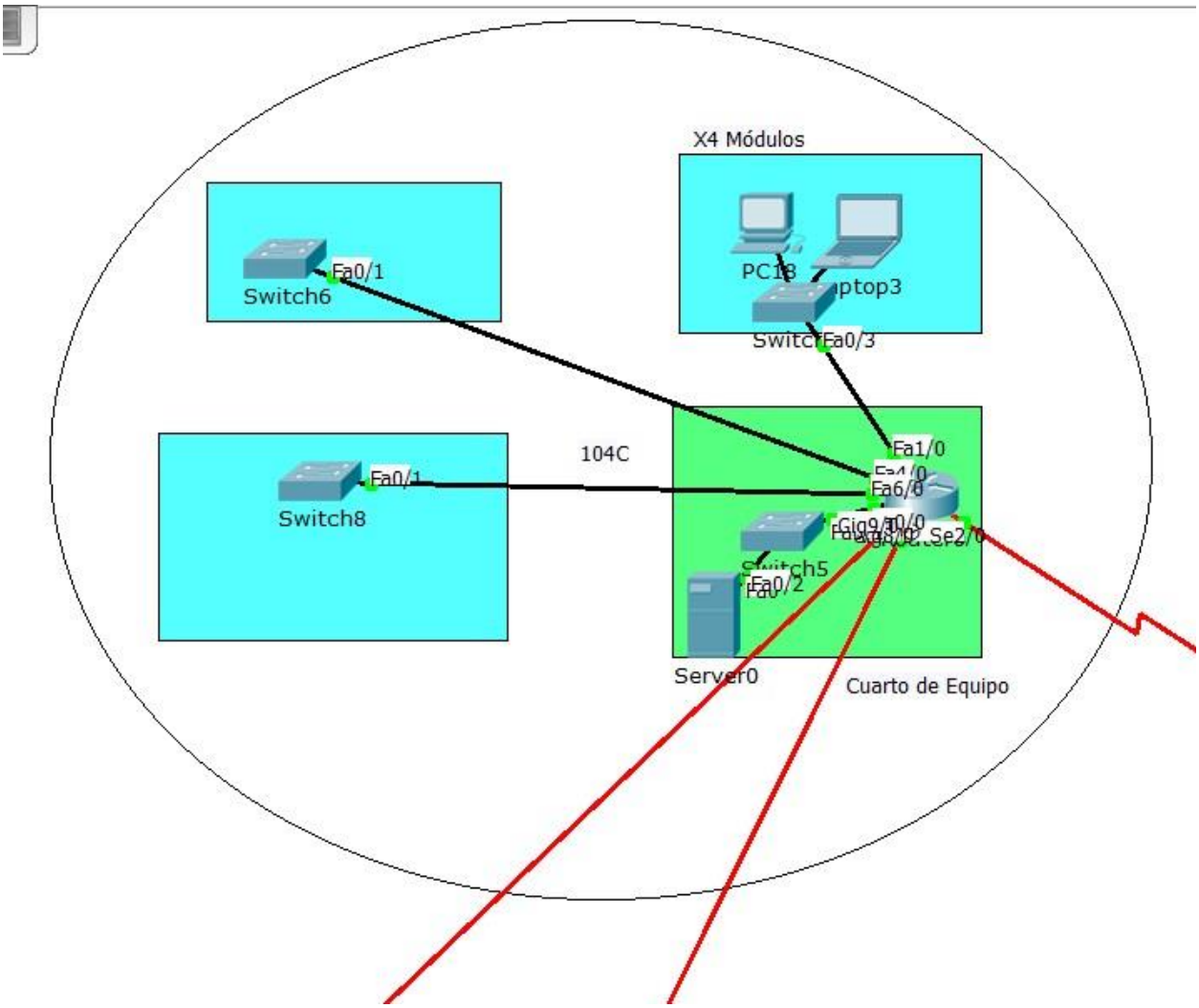


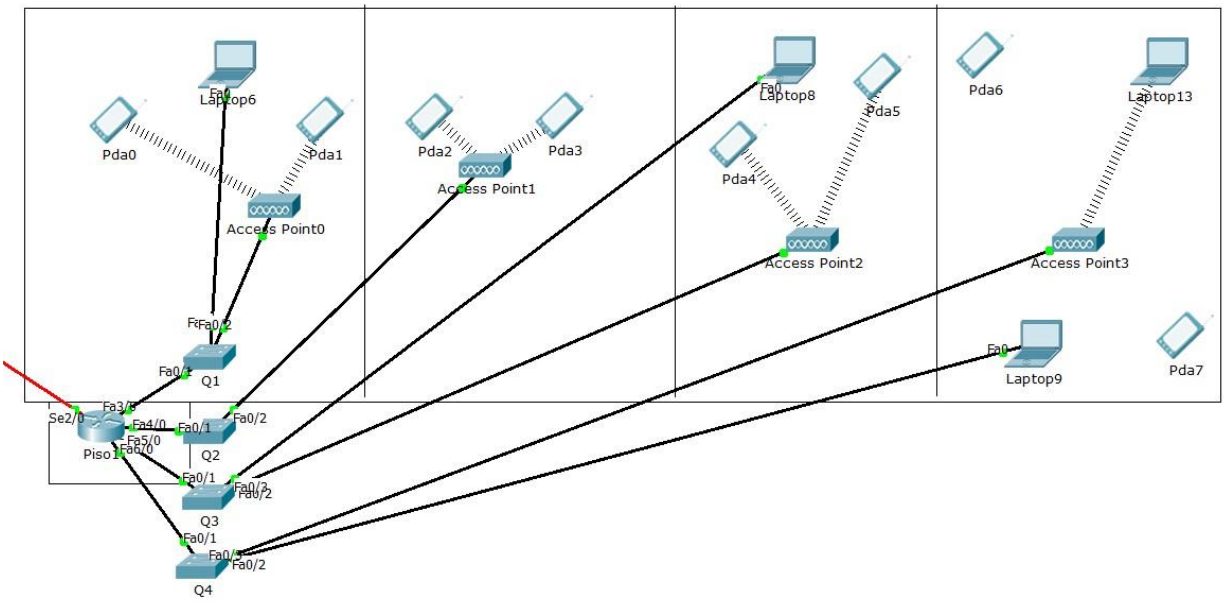
Figure 1 – Position A and B configuration of a 568SC

Diagrama (lógico)

Se hizo una simulación de la estructura general del cableado planteado en las instalaciones de la Facultad de Ciencias de la Computación, simulado con la herramienta Packet Tracer.

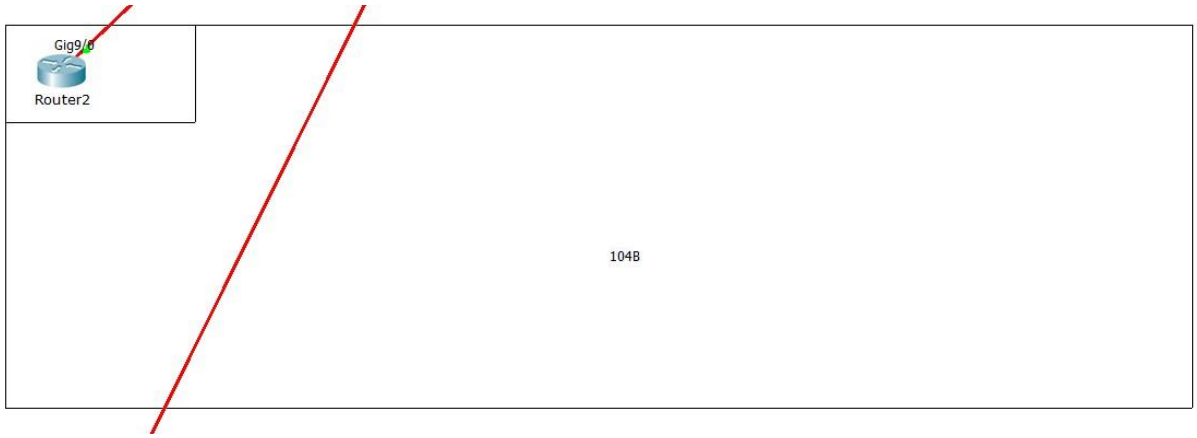


Img Diagrana 104C

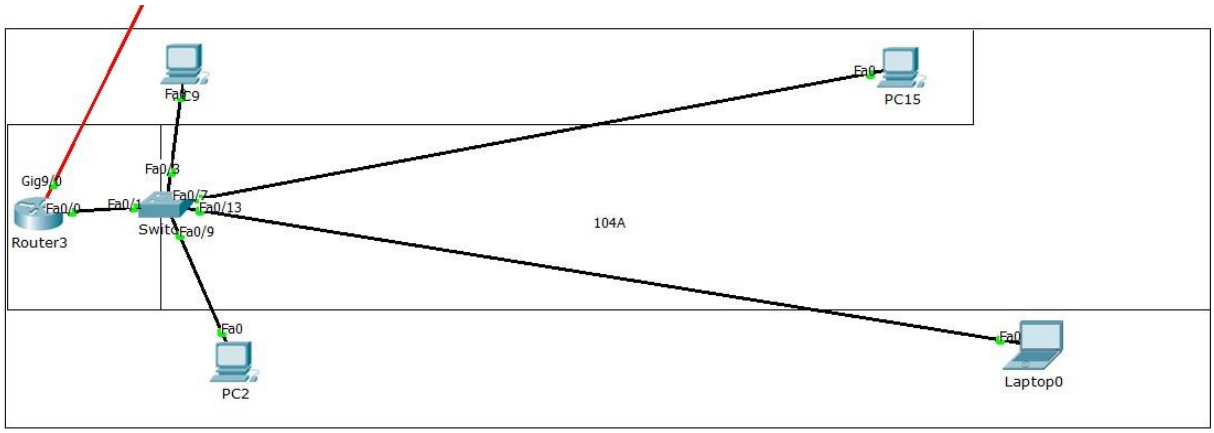


Esta configuración se realiza para los tres pisos. Incluye posisionamiento de Access points y

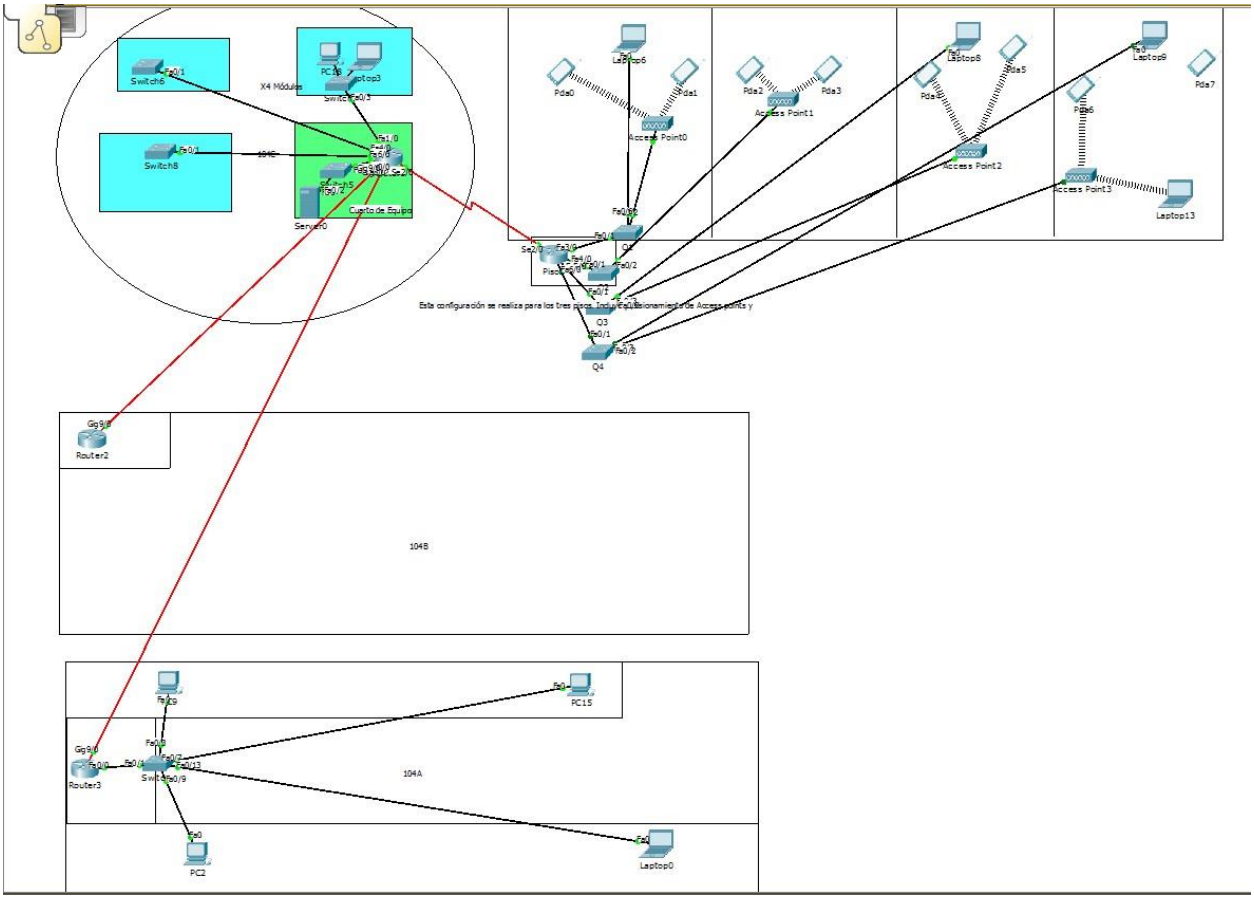
Img Diagrana 104D



Img Diagrana 104B



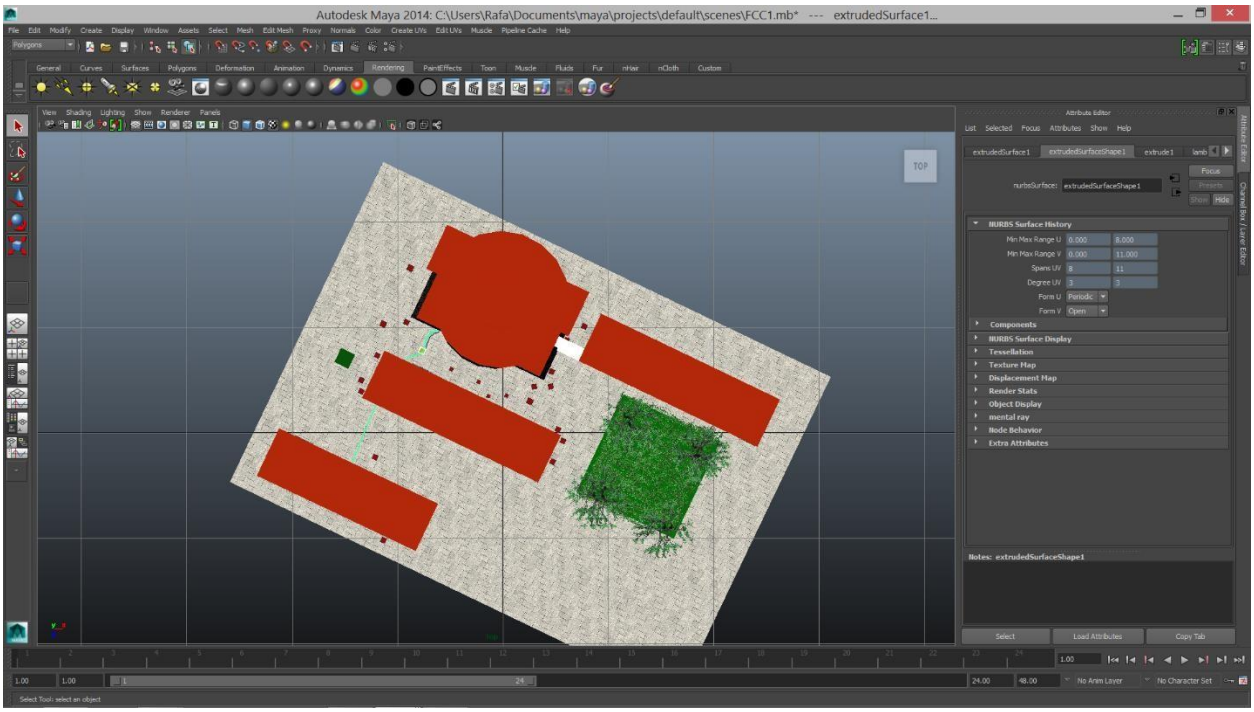
Img Diagrana 104A



Img Diagrana general

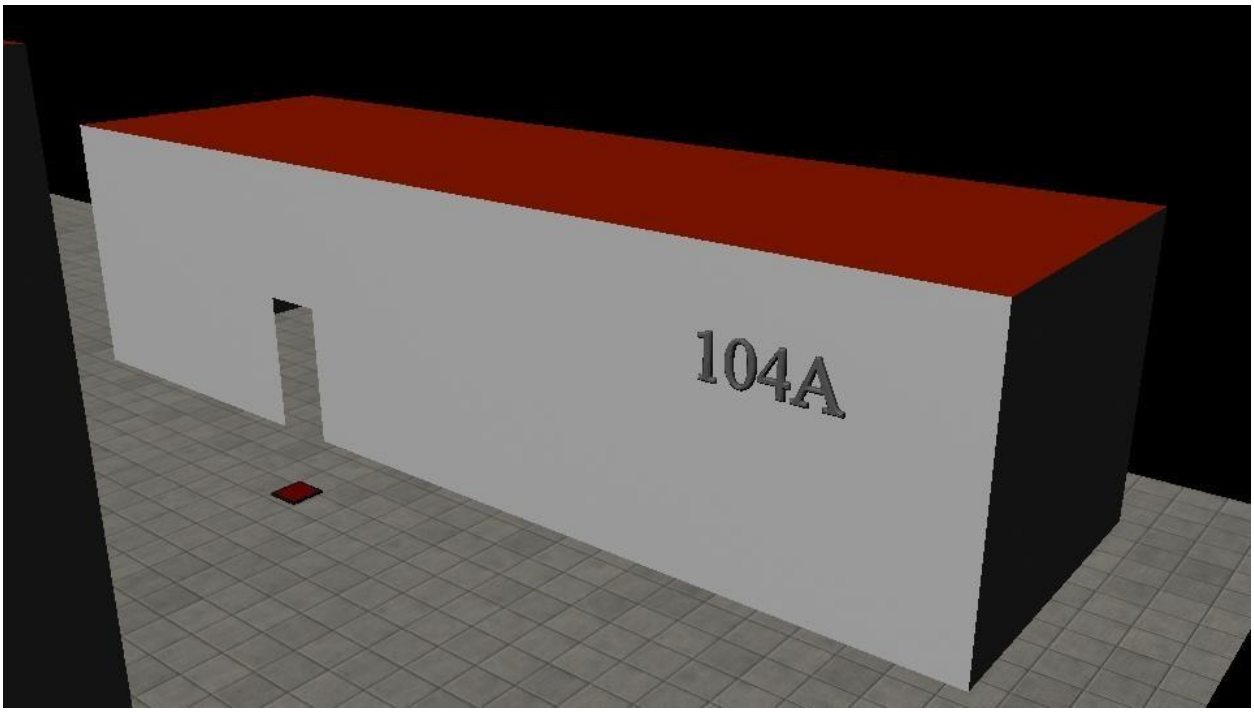
Modelado (visual)

Basándonos en la estructura arquitectónica y considerando los diferentes registros plubiales, eléctricos y de fibra ópticas, éstos últimos provenientes del CIU, propusimos los el siguiente modelado:



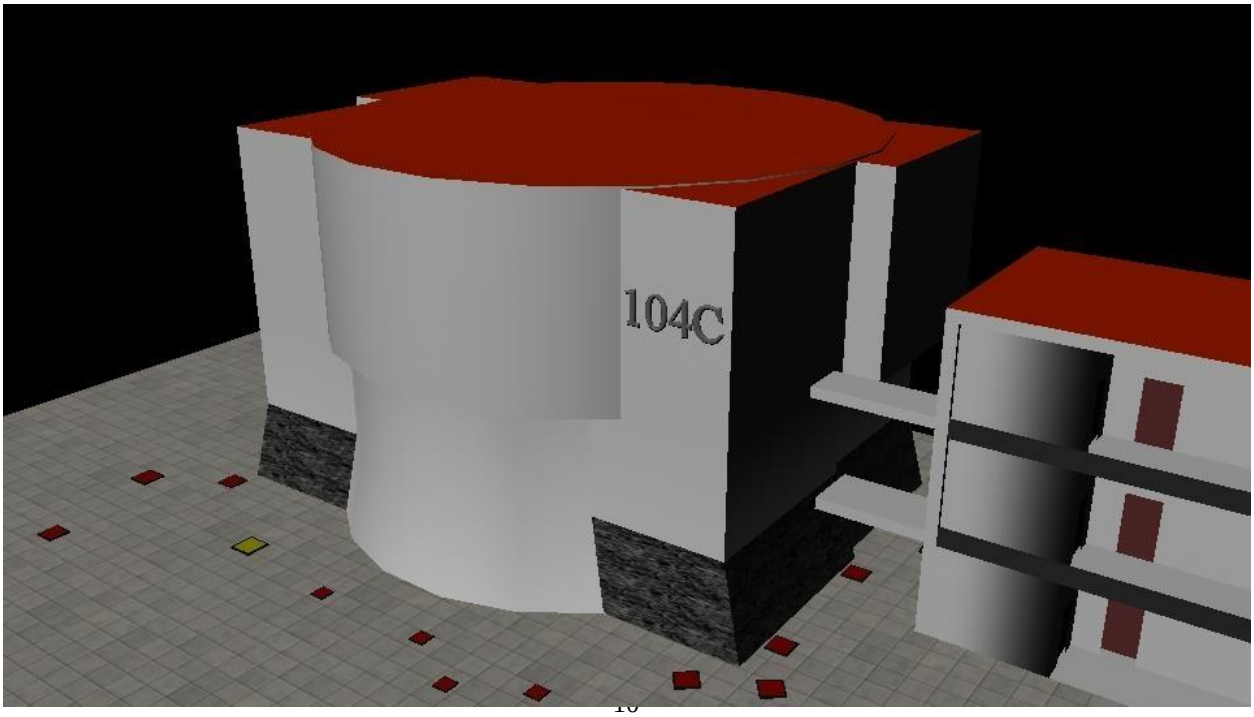
Img M1. Vista satelital

Img M2. Vista de edificios A, B, C y D

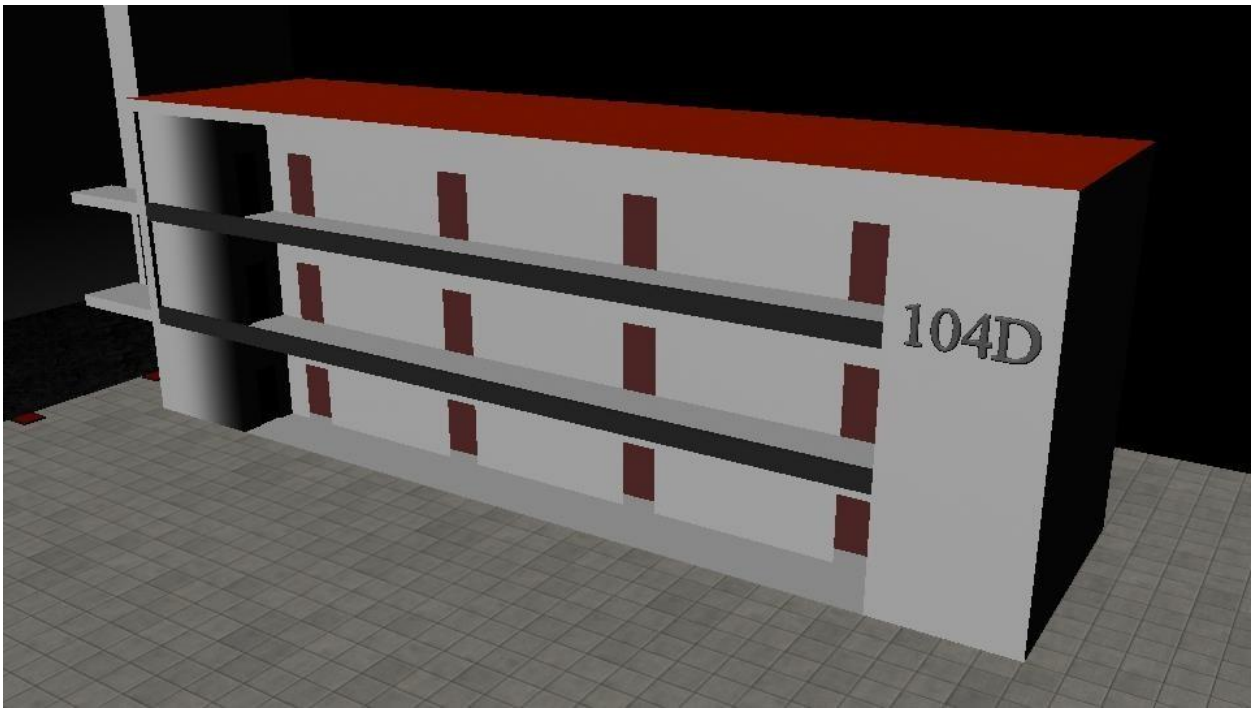


Img M3. Edificio 104 A

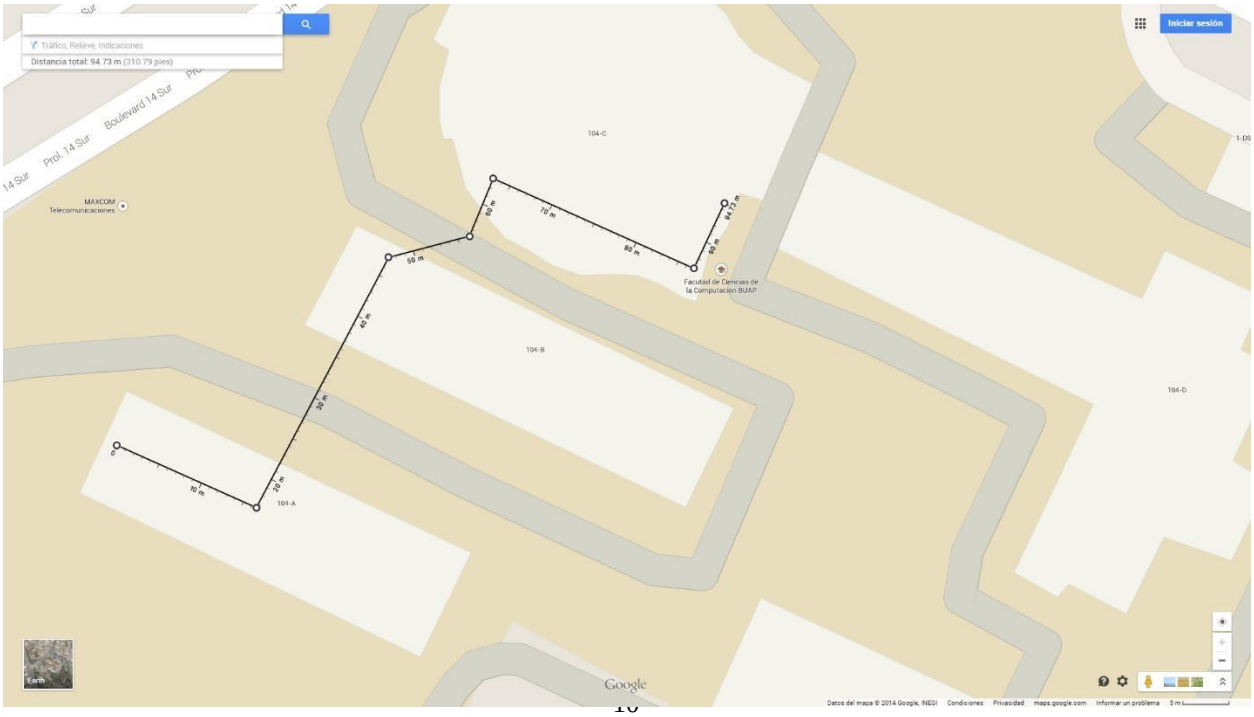
Img M4. Edificio 104 B



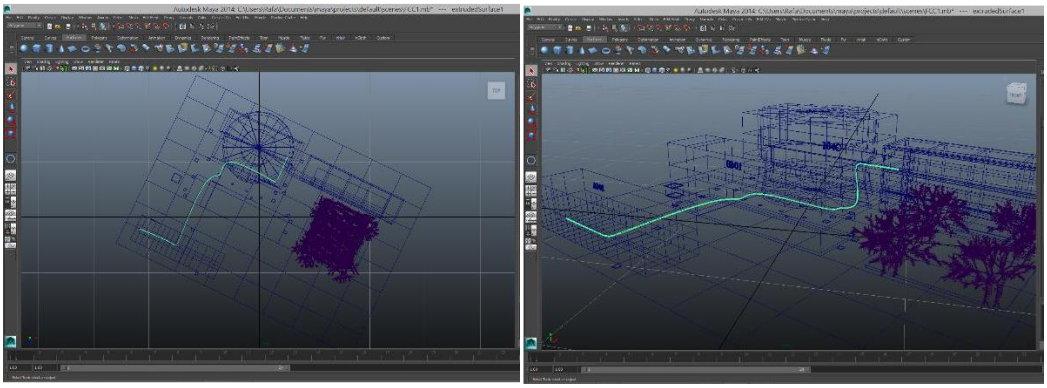
Img M5. Edificio 104 C



Img M6. Edificio 104 D

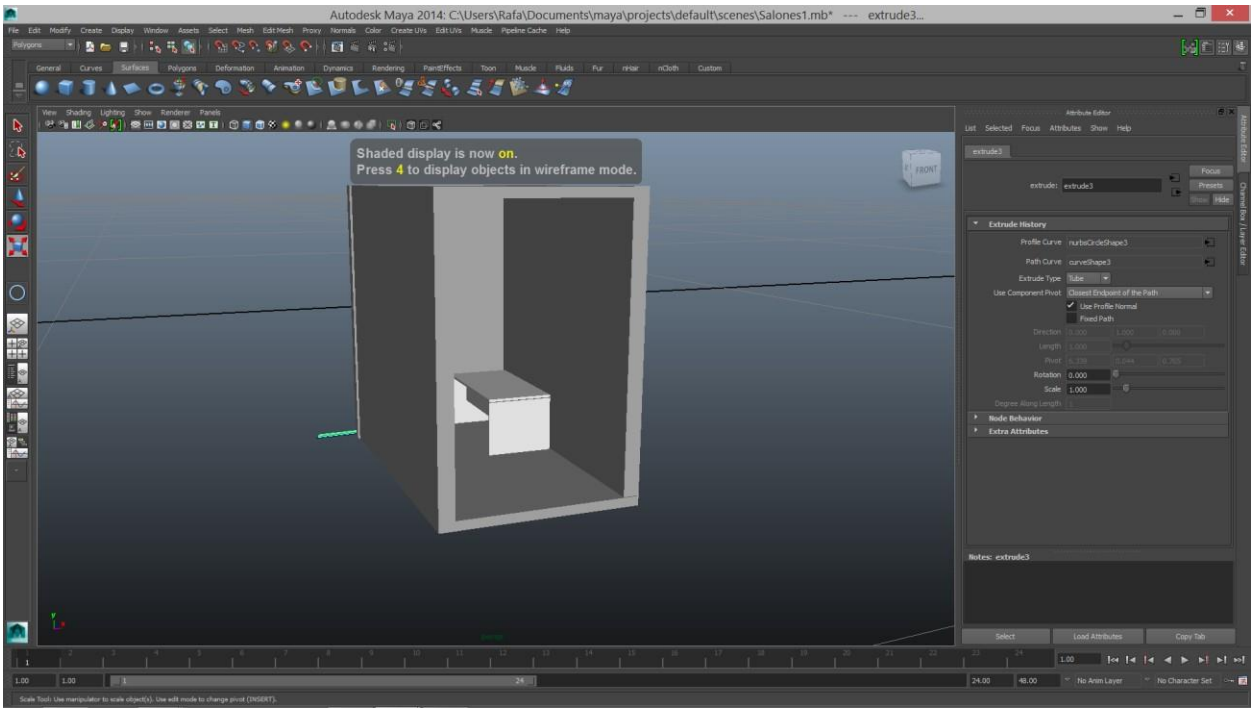


Img M7. Medición de distancias entre edificios (con ayuda de Google Maps)

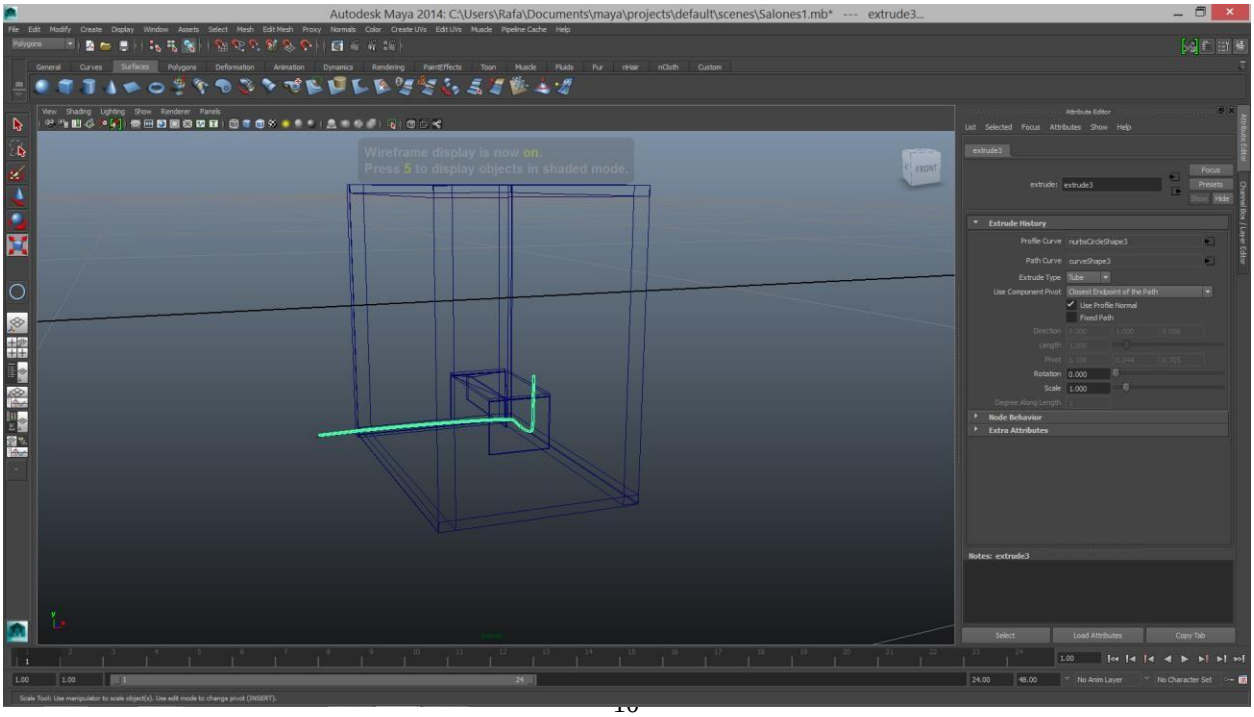


Img M8. Cableado de Edificios A – B – C

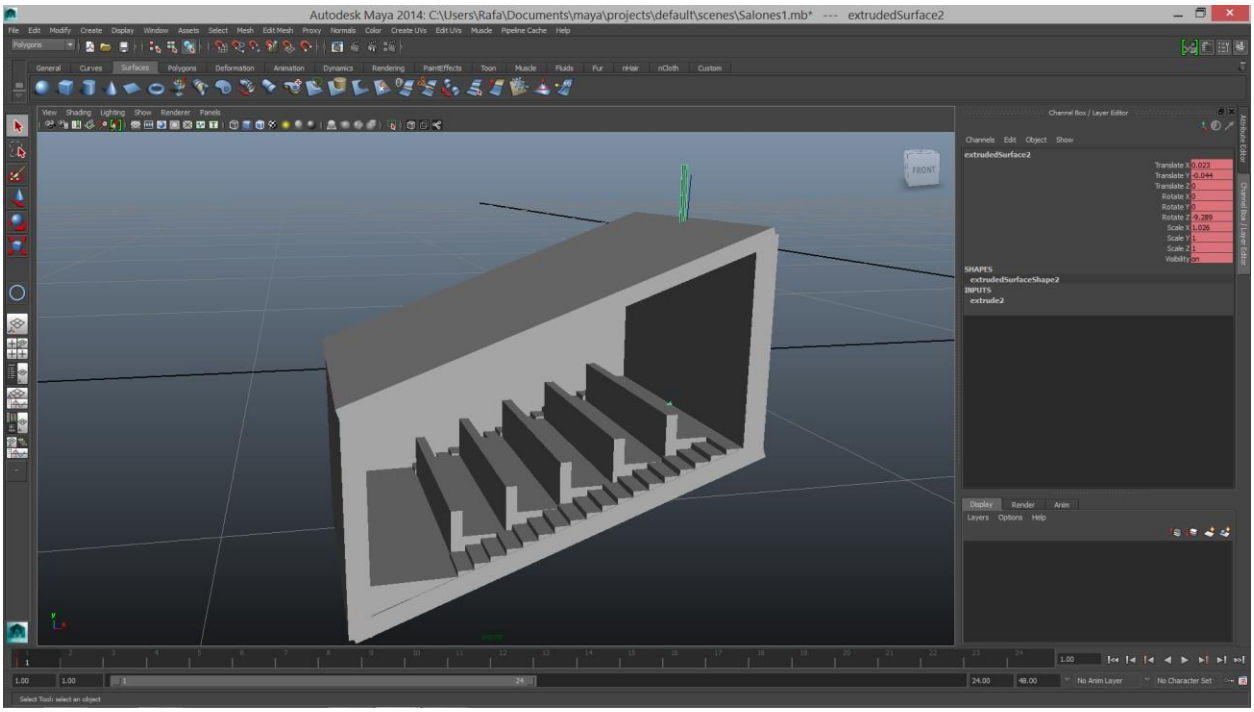
Img M9. Cableado de edificios 104 A – B – C



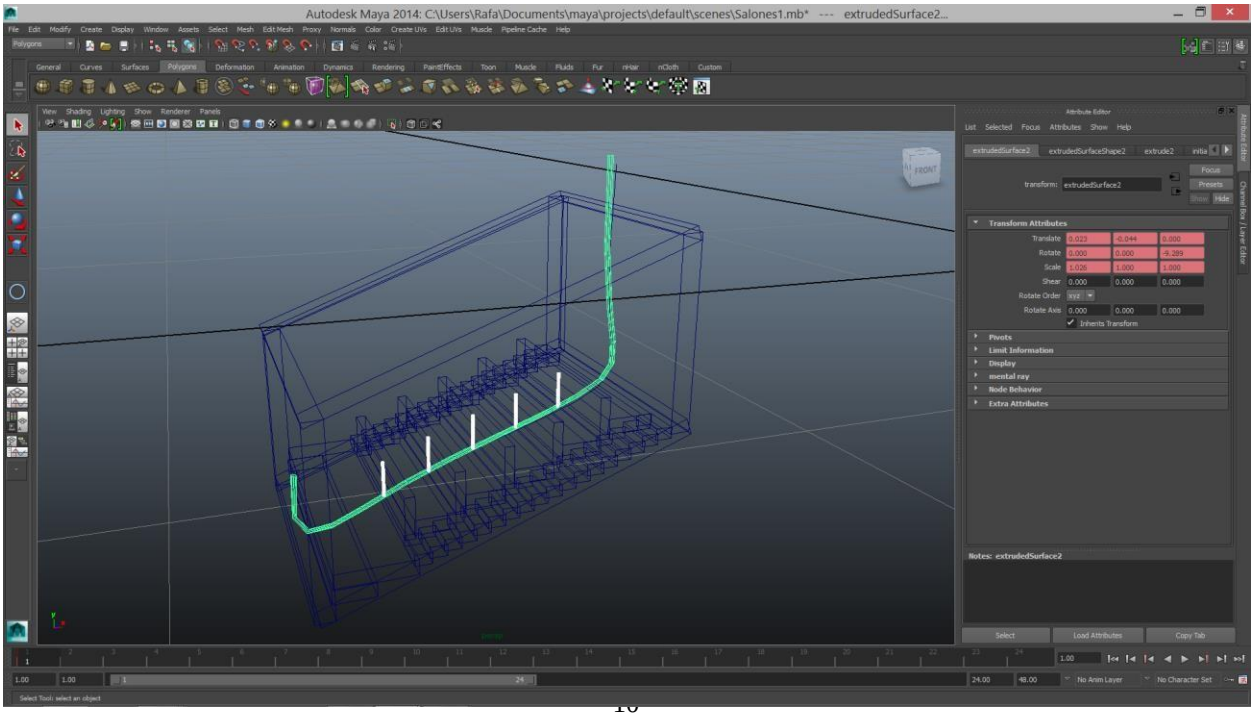
Img M10 Modelado de cubos en edificio 104ª



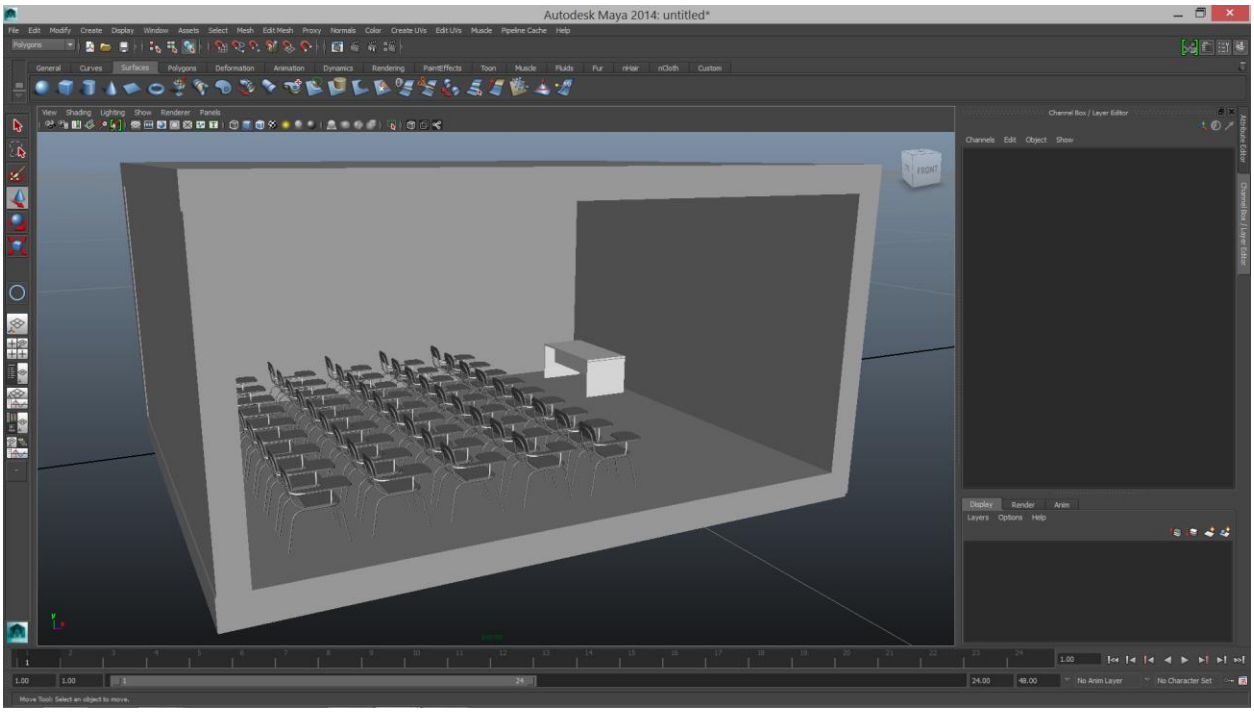
Img M11 Cableado de cubos en edificio 104A



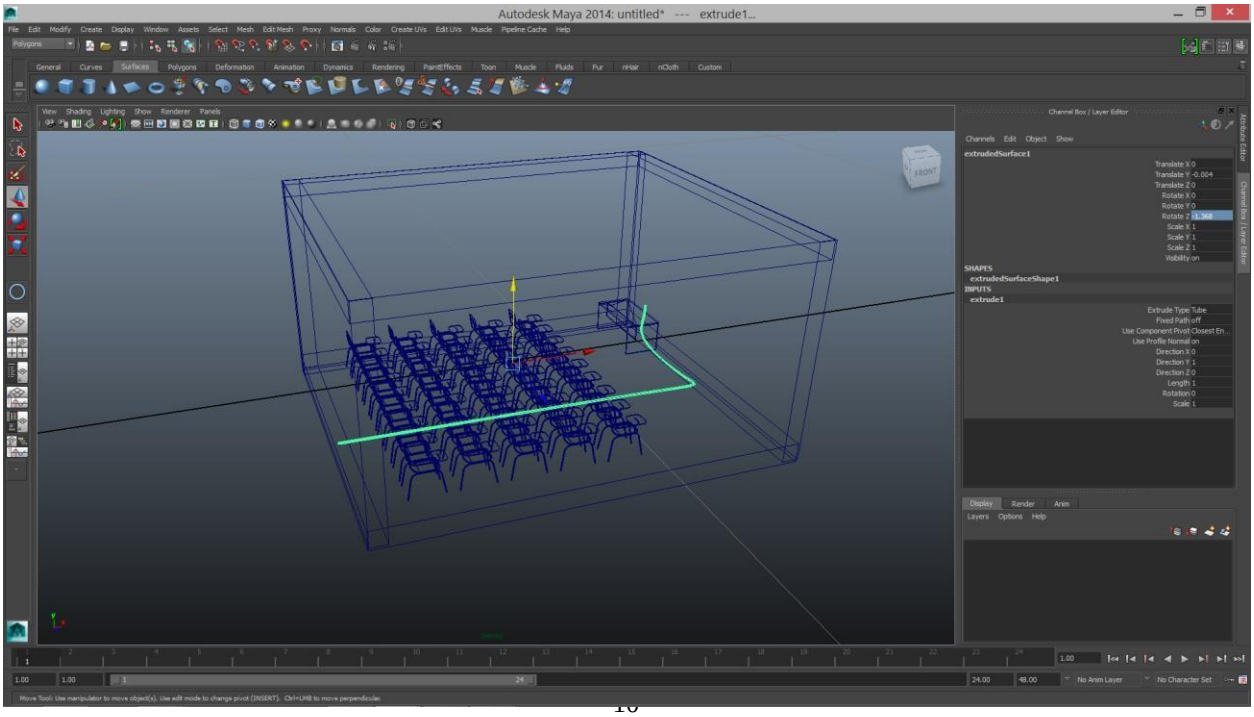
Img M10 Salones de edificio 104C



Img M11 Cableado de módulos en edificio 104C



Img M12. Salones de edificio 104D



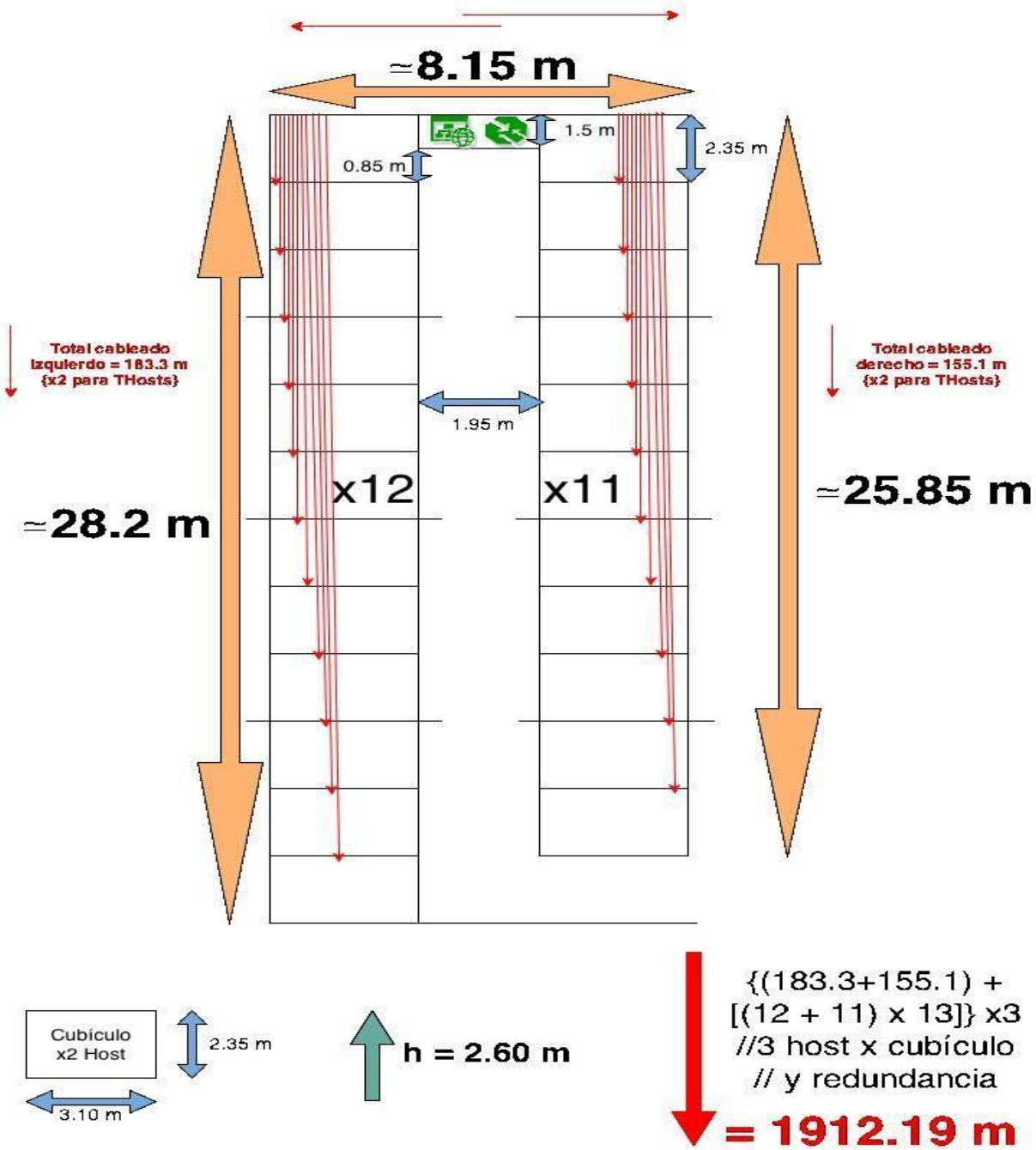
Img M13. Cableado de salones

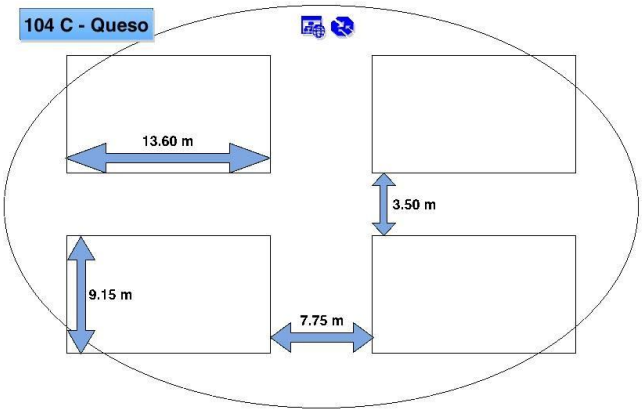
Mediciones

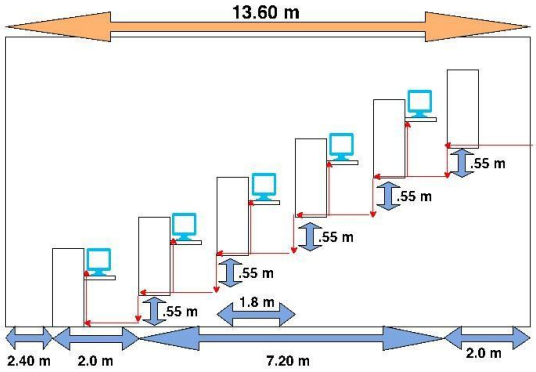
Las mediciones se llebaron a cabo dentro de la facultad con la ayuda de un flexometro. Se tomaron las medidas de cada edificio, salones, alturas, paredes, escalones, entre otros, para determinar las distancias exactas que recorrerian los cables dentro de la instalación.

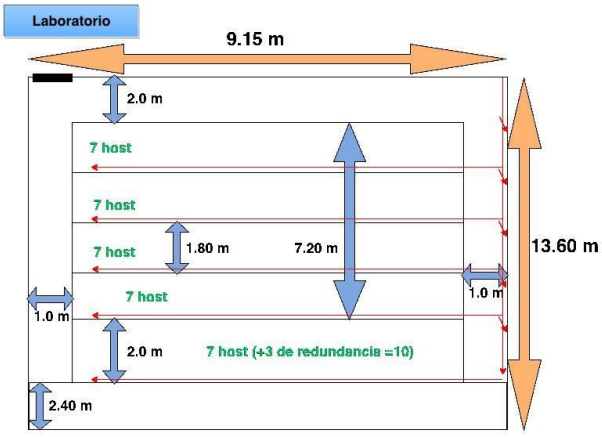
Acontinuación se representan las mediciones (sin escala) que sirvió para determinar las cantidades de cable a utilizar:

104 A


≃8.15 m
1.5 m
2.35 m
0.85 m
Total cableado
Izquierdo = 183.3 m
{x2 para THosts}
Total cableado
derecho = 155.1 m
{x2 para THosts}
1.95 m
x12
x11
≃28.2 m
≃25.85 m
Cubículo
x2 Host
2.35 m
3.10 m
h = 2.60 m
{(183.3+155.1) +
[(12 + 11) x 13]} x3
//3 host x cubículo
// y redundancia
= 1912.19 m

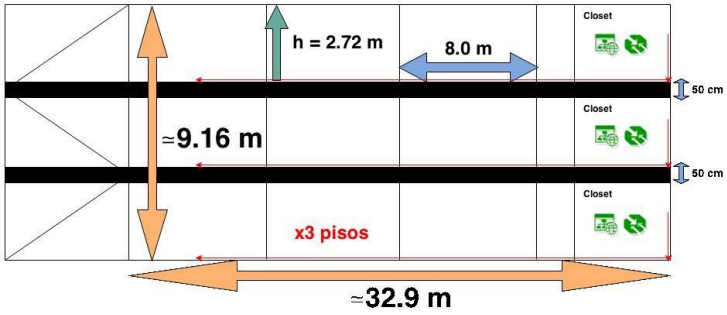




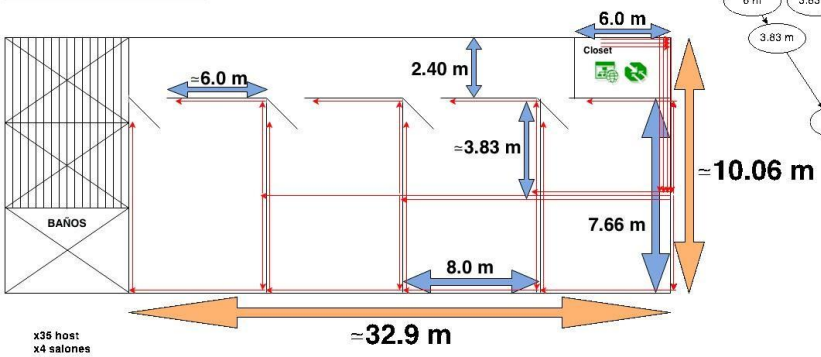


Pared: 2.0 m + 1.8 m + .55 m = 3.8 m x 50 ==> 271.5 Mts (1.8 m + .55 m) x 40 ==> 94 Mts (1.8 m + .55 m) x 30 ==> 70.5 Mts (2.35 m) x 20 ==> 47 Mts (2.35 m) x 10 ==> 23.5 Mts TOTAL Pared => 506.5 m	Terminales: distancia = 7.15 + 1 separación entre computadoras = 1.2 Sumatoria (1 a 7) de: 1.2 * i + 1.0 => 40.6 m Tenemos 5 filas: 40.6 x 5 = 203 m TOTAL Laboratorio ==> 709.5 m (Son 4 labs)

104 D - Vista frontal

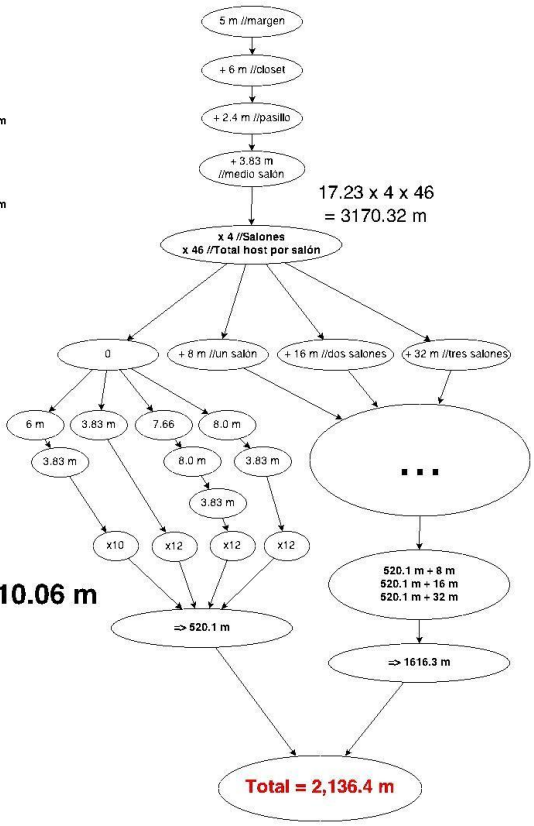


104 D - Vista superior



x35 host
x4 salones
x30%

=> 48 x 4



Presupuesto

Equipo	Cantidad	Precio Unitario	Precio Total	
Router	6	$369.00	$ 2,214.00 USD	$ 2,214.00
Acces Point	12	$220.00	$ 2,640.00 USD	$ 2,640.00
Switch 48	18	$293.00	$ 5,274.00 USD	$ 5,274.00
Switch 24	5	$153.00	$ 765.00 USD	$ 765.00
Patch core	186	$215.78	$ 40,135.00 MN	$ 3,050.26
Face Plate	269	$22.80	$ 6,133.20 MN	$ 466.12
Patch panel 48	18	$ 8.218.94	$ 147,940.92 MN	$ 11,243.51
Patch panel 24	5	$4,445.02	$ 22,225.10 MN	$ 1,689.11
Rack	6	$6,057.78	$ 36,346.68 MN	$ 2,762.35
Organizador horizontal	12	$39.00	$ 468.00 USD	$ 468.00

Organizador vertical	6	$2,701.00	$ 16,206.00 USD	$ 16,206.00
Ventilador	12	$94.12	$ 1,129.44 USD	$ 1,129.44
Acondicionadora de aire	1	$11,946	$ 11,946 USD	$ 11,946.00
Charola sencilla	4	$210	$ 840 MN	$ 63.84
Rollo de velcro	4	$162.50	$ 650.00 MN	$ 49.40
Kit de pararrallo	1	$1,983	$ 1,983 USD	$ 1,983.00
Tuberia flexium	500m	$2.60	$ 1,300 USD	$ 1,300.00
Ethernet converter	4	$34.99	$ 139.96 USD	$ 139.96
Servidor	1	$14,771	$ 14,771 USD	$ 14,771.00
Fibra monomodo (36)	250m	$1.60	$ 400.00 USD	$ 400.00
			Total dolares	$ 78,560.99
			Total Pesos	$ 1 070 152.84

Conclusiones

Esta práctica envuelve un grado de complejidad alto gracias a los diversos detalles en los que hay que prestar especial atención.[11]

El cableado estructural le da una seguridad al usuario final muy alta de no interrupción de servicio debido a la buena distribución del equipo, la redundancia, entre otras cosas.

Cabe mencionar que éste tipo de cableado no es apto para varias instituciones dado que el costo de los equipos, remodelaciones, componentes, etc., se pueden elevar a cifras exorbitantes.

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Ingeniería en Ciencias de la Computación

2do. examen parcial

Para la asignatura de

Modelos de Redes

Propuesta de cableado estructurado para la Facultad de Ciencias de la Computación BUAP

Catedrático

M.C. Iván Olmos Pineda

Otoño 2014

Introducción

El diseño y la correcta implementación de una red de computadoras garantizan que el tráfico de información dentro de ella se efectúe de forma correcta y por lo tanto aumente el índice de satisfacción de sus usuarios.

Desde su inicio la implementación de redes de computadoras ha pasado por una serie de evoluciones y mejoras desarrolladas para paliar los inconvenientes entre una tecnología y otra. Fue también gracias a esta evolución en los medios de transmisión y protocolos que se pudo implementar una serie de normas a seguir para garantizar calidad en las conexiones desde su instalación en el espacio físico hasta la recepción de los paquetes en las terminales.

El cableado estructurado se define como el tendido o instalación de cables (pudiendo estos ser de distintos tipos, ya sea par trenzado, fibra óptica o cable coaxial) dentro de un edificio, adecuándose a la planta física con el fin de implantar una red de área local en todo el espacio de trabajo.

Planteamiento del problema

Esta parte del trabajo se centra en dotar de una infraestructura de cableado estructurado al edificio 104D, que, para fines prácticos ha sido distribuido en solamente tres plantas con 4 salones cada uno. Para este esquema se contempla que cada salón cuente con un rack de comunicaciones destinado para 30 terminales.

Esta parte del edificio cuenta con una extensión de 31.56 metros de largo de frente y 2.35 mts. De alto por cada nivel, así como 8 metros de ancho.

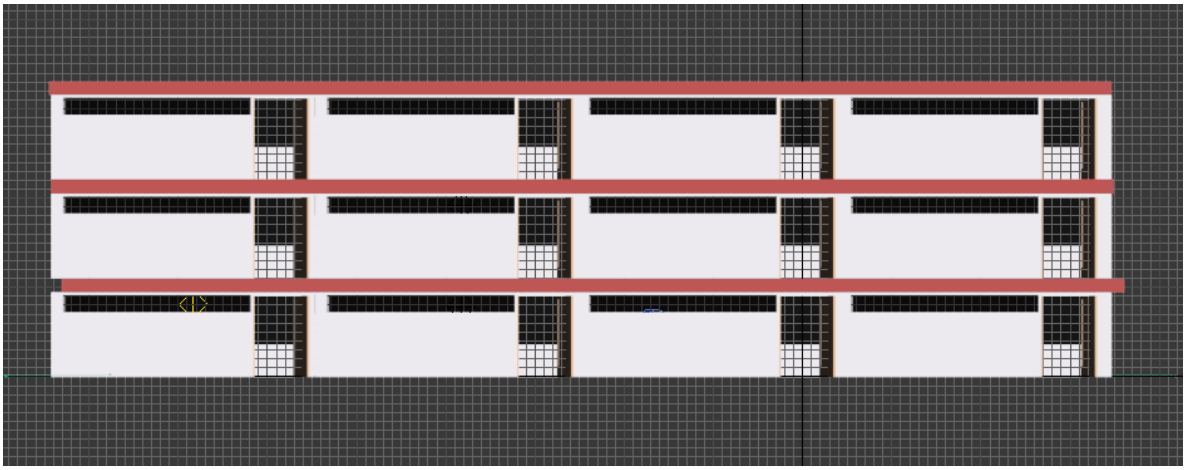


Fig. 1 Vista frontal del edificio.

Estructura física

La planta física del edificio 104D consta de 3 pisos con 4 salones cada uno con una superficie aproximada de 252.48 m² por planta. La figura 2 muestra una definición aproximada de la planta física que se describe, cabe destacar que las condiciones de las 3 plantas son similares.

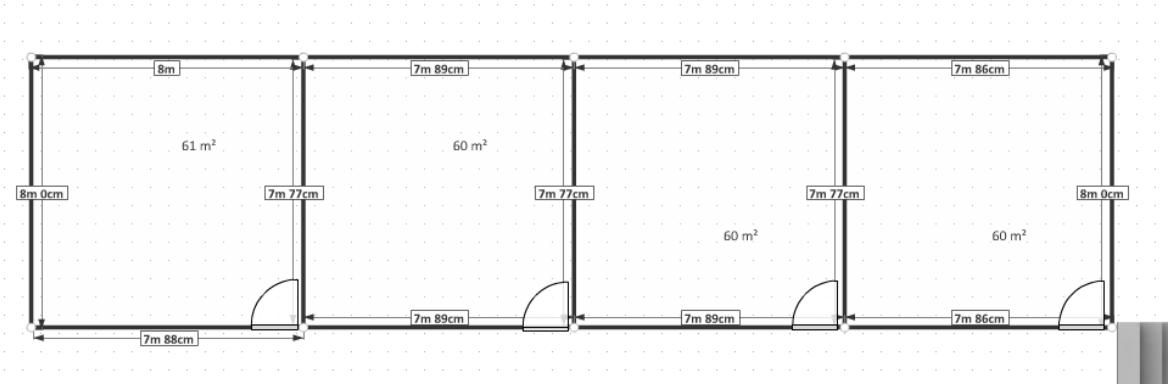


Fig. 2

Contando con esta información, precisamos lo siguiente:

- En este edificio los salones se encuentran dotados de un techo falso y todas las paredes son de ladrillo, razón por la cual es necesario contar con herramientas adecuadas.
- El centro principal de cableado no se encuentra en el edificio, sino que será necesario conectar al Site ubicado en el edificio 104B que se encuentra frente al 104D. (fig. 4)
- El cableado en las áreas de trabajo será distribuido por medio de canaletas.
- Cada salón contará con un centro de cableado que distribuirá a las 30 terminales de cada área de trabajo dando un total de 480 terminales por edificio.

La distribución aproximada del equipo y terminales en el área de trabajo queda según la figura 3.

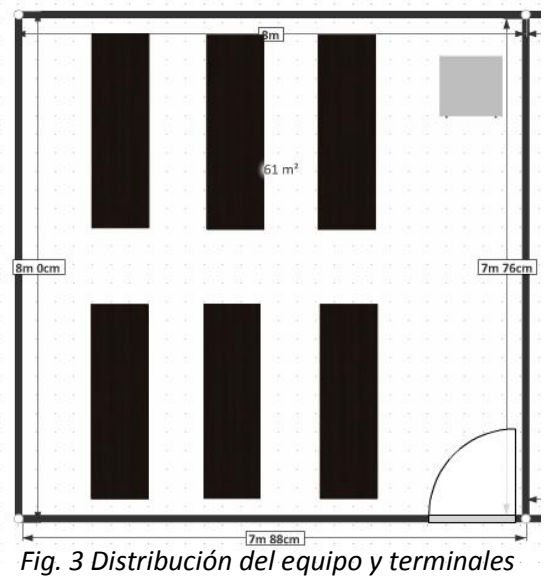


Fig. 3 Distribución del equipo y terminales

Se colocarán 5 equipos por mesa de trabajo y el rack de comunicaciones estará ubicado a lado de cada columna que soporta el peso estructural del edificio desde sus cimientos.

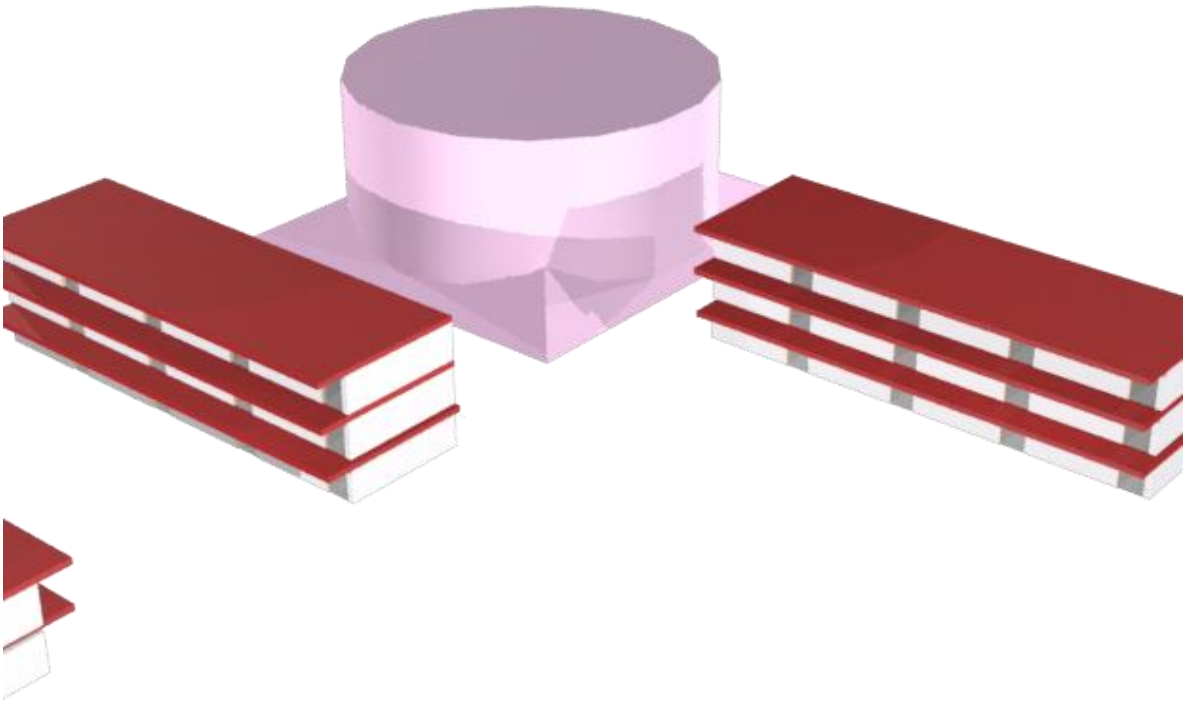


Fig. 4 Ubicación de los edificios 104D y 104B

Materiales y métodos de conexión

Para poder efectuar la conexión de todas las áreas de trabajo, además del rack se requiere de equipo.

En este caso se eligió el switch ESW500 de Cisco, ya que cuenta con soporte para las terminales requeridas y además permite agregar la conectividad por fibra óptica. Tomando en cuenta lo anterior se requiere un total de 12 switches, uno para cada área de trabajo.

Utilizaremos el esquema de cableado horizontal, el cual, por su versatilidad y características se ajusta a las dimensiones de cada área de trabajo.

Dado que sólo requerimos de servicio de datos, prescindiremos de indicar un esquema de conexión a servicios de voz, centrándonos sólo en lo anterior como lo muestra la figura 5.

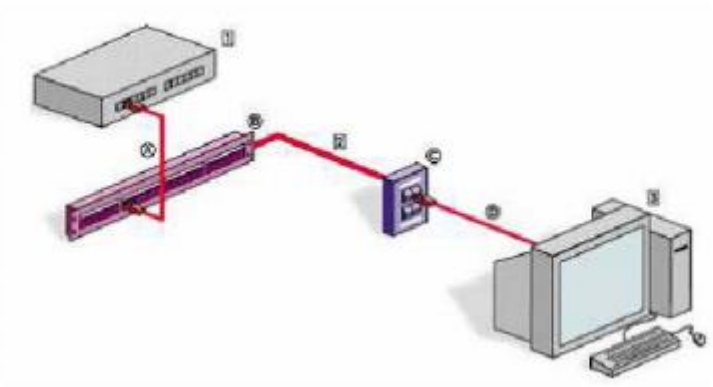


Figura. 5 Esquema de conexión de cableado horizontal

Cables y tomas

Para realizar el tendido de la red de datos utilizaremos cable par trenzado UTP categoría 6, el cual cuenta con las especificaciones para proveer de servicio a cada terminal con una buena velocidad. Para esto se utilizará un aproximado de 220 metros de cable y 30 jacks de conexión a pared, dando un total de 120 jacks y 2640 metros de cable UTP Categoría 6 para cubrir todas las áreas de trabajo.

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
Ingeniería en Ciencias de la Computación
Modelos de Redes

Examen del Parcial 2

Graciela Gaona Bernabé Erika
Leonor Basurto Munguía Lucia
Adriana Merlo Haro

Cesar Manuel Rodríguez Mendiola

12

05/10/2014

Índice general

Capítulo 1

Antecedentes teóricos

El cableado estructurado consiste en el tendido de cables de par trenzado UTP/STP en el interior de un edi cio con el propósito de implantar una red de área local. Suele tratarse de cable de par trenzado de cobre, para redes de tipo IEEE 802.3.

Permitir la comunicación virtual con cualquier dispositivo en cualquier lugar y en cualquier momento. Un buen diseño soporta la implementación de distintas soluciones independientes. Se pretende estandarizar los sistemas de comunicación Permite realizar el cableado sin conocer los equipos de comunicación de datos que lo utilizarán

El tendido de los cables es sencillo de administrar

Las fallas son menores y más fáciles de localizar que en los sistemas

POST (Plain Old Telephone System)

Permite el crecimiento futuro

Facilita al cliente el manejo y administración de los servicios conectados

El cableado estructurado se realiza bajo estándares internacionales de las industrias

Para minimizar la perdida de paquetes las direcciones IP deberán ser asignadas en un solo sentido, es decir, de forma ascendente o descendente pero nunca mezclar ambas.

1.1. De niciones

1.1.1. Patch Cord

Cable de red de conexión directa o cruzada.

1.1.2. Placa con servicios

Placa que soporta los conectores RJ45.

1.1.3. Panel de parcheo (Patch-Panels)

Estructuras metálicas con placas de circuitos que permiten la interconexión entre equipos. Poseen una determinada cantidad de puertos (RJ-45 End-Plug), donde cada puerto se asocia a una placa de circuito.

1.1.4. Rack

Estructura de metal muy resistente, generalmente de forma cuadrada de aproximadamente 3 mts de alto por 1 mt de ancho, en donde se colocan los equipos regeneradores de señal y los Patch-Panels, estos son a justados al rack sobre sus ori cios laterales mediante tornillos.

1.1.5. Cableado vertical (backbone)

También conocido como cableado troncal, permite la interconexión entre los distribuidores de cableado de las distintas plantas en un edi-

cio, o entre distintos edi cios en un campus. Tiene una topología es de estrella jerárquica, aunque también suelen utilizarse las topologías de bus o de anillo.

1.1.6. Cable Rollover (Cable de Consola)

Se utiliza para conectar una PC al router. Puede tener hasta 7.5mts

1.1.7. Switch

Dispositivo digital lógico de interconexión de equipos que opera en la capa de enlace de datos del modelo OSI. Su función es interconectar dos o más segmentos de red, de manera similar a los puentes de red, pasando datos de un segmento a otro de acuerdo con la dirección MAC de destino de las tramas en la red.

1.1.8. Router

Dispositivo que proporciona conectividad a nivel de red o nivel tres en el modelo OSI. Su función principal consiste en enviar o encaminar paquetes de datos de una red a otra, es decir, interconectar subredes, entendiendo por subred un conjunto de máquinas IP que se pueden comunicar sin la intervención de un encaminador (mediante bridges), y que por tanto tienen pre jos de red distintos.

1.1.9. Punto de Acceso (AP)

Tambien conocidos como WAP por sus siglas en ingles Wirless Access Point; dispositivo que interconecta dispositivos de comunicación alámbrica para formar una red inalámbrica. Normalmente un WAP también puede conectarse a una red cableada, y puede transmitir datos entre los dispositivos conectados a la red cable y los dispositivos inalámbricos. Muchos WAPs pueden conectarse entre sí para formar una red aún mayor, permitiendo realizar roaming. Conector de Cruce (Cross Connect) Es un grupo de puntos de conexión montados en una pared o en un Rack, usado como terminaciones mecánicas para la administración del cableado del edi cio.

1.1.10. Canaleta

Ducto adherido a la pared o piso por medio del cual pasara el cable (del tipo que sea)

1.1.11. Área de trabajo

Lugar donde se encuentra el personal laborando con los dispositivos nales

1.1.12. Dispositivo nal

Dispositivos con los cuales interactua el personal, por ejemplo; PCs, impresoras, etc.

1.1.13. Closet de comunicación

Lugar donde se concentran todas las conexiones necesarias para el área de trabajo

1.1.14. Cuarto de equipo

lugar donde se localizan todos los dispositivos generalmente de alto costo.

1.1.15. Acometida

Consiste en cables, accesorios de conexión, dispositivos de protección, y demás equipo necesario para conectar el edi cio a servicios externos. Puede contener el punto de demarcación. Ofrecen protección eléctrica establecida por códigos eléctricos aplicables. Deben ser diseñadas de acuerdo a la norma EIA/TIA-569-A. Los requerimientos de instalación son:

ˆ Precauciones en el manejo del cable

ˆ Evitar tensiones en el cable

ˆLos cables no deben enrutarse en grupos muy apretados

ˆ Utilizar rutas de cable y accesorios apropiados 100 ohms UTP y STP

ˆ No giros con un angulo menor de 90 grados ni mayor de 270.

1.2. Sistema de puesta a tierra

El sistema de puesta a tierra y puenteo establecido en estándar ANSI/TIA/EIA-607 es un componente importante de cualquier sistema de cableado estructurado moderno. El gabinete deberá disponer de una toma de tierra, conectada a la tierra general de la instalación eléctrica, para efectuar las conexiones de todo equipamiento. El conducto de tierra no siempre se halla indicado en planos y puede ser único para ramales o circuitos que pasen por las mismas ca jas de pase, conductos ó bandejas. Los cables de tierra de seguridad serán puestos a tierra en el subsuelo.

Capítulo 2

Desarrollo

2.1. Cableado Estructurado

2.1.1. Edi cio 104 A

En el edi cio 104 A se proporcionara servicio de red alámbrica para las dos las de cubos, para cada una se asigna un red especi ca y para ellos requerimos de dos switch y un router que se conectara a la Acometida que se encuentra en el edi cio 104B pero estos se ubicaran en el cuarto de equipo que esta al fondo del edi cio 104 A ver g.

9. Cada cubo tendrá una placa de servicio con 3 conectores RJ45 a una distancia de 1.05m de cada separación de cubo-cubo ver g.10, en cada cubo hay dos maquinas por lo que la placa de servicio tendría

2 conectores RJ45 pero bajo las reglas de cableado estructurado se debe de dejar un tanto de redundancia, entonces en colocaremos 3 conectores RJ45 para cada cubo. El cableado que comunica a cada placa de servicio, estará colocado dentro de la pared. El router principal que conecta al router del edi cio 104A tiene una distancia de

60 metros, atravesando parte de los cubos y la acometida ver g. 11.

8

9

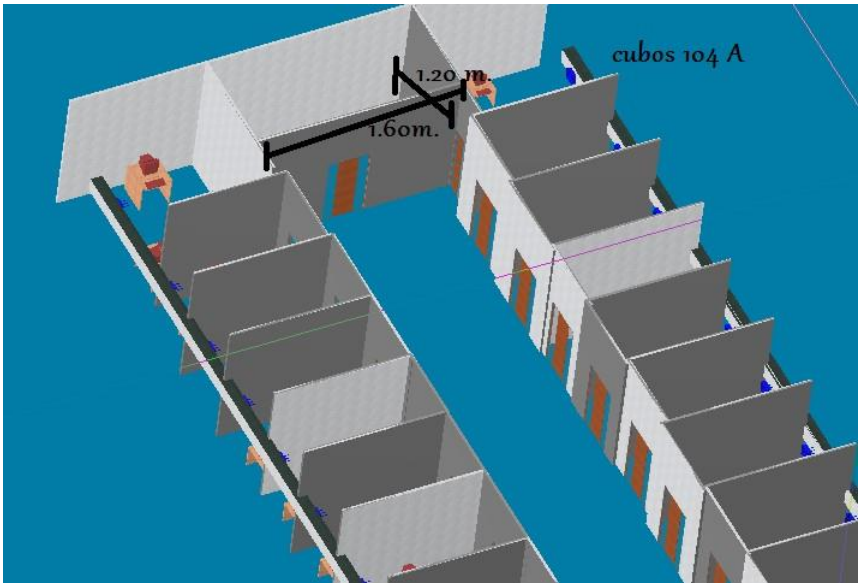


Fig.9 cuarto de comunicación 104A

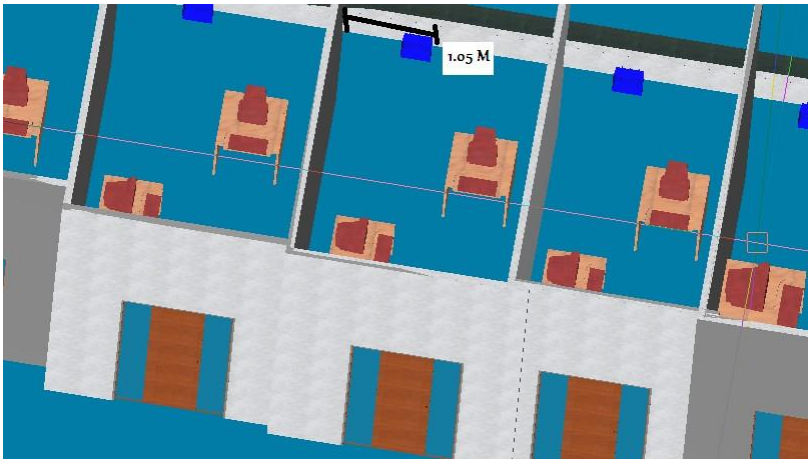


Fig.10 Ubicación de la placa de servicio de cada cubo

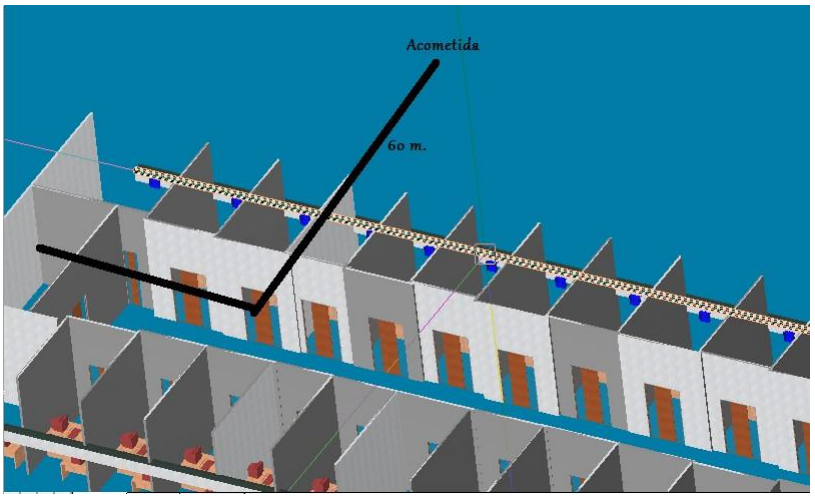


Fig.11 cuarto de comunicación 104 A - Acometida

2.1.2. Edi cio 104C (Queso)

Para la conexión a internet de los laboratorios, se asigna una dirección de red a cada laboratorio (en el apartado diseño de las redes se detalla), por lo que requerimos un switch, router y un Access Point para cada modulo.

Cada laboratorio tendrá un closet de comunicación, a la entrada, al fondo se ubicara y guardara un rack, switch y un router.

La ubicación del Access Point será en techo al centro del salón, la conexión de este es de 7x5 hacia el closet de comunicación como se observa en la g. 1.

En cada la de computadoras hay 6 PCs estáticas, para ello se colocaran 4 placas de servicios de 2 conectores RJ45, ya que son 30 PCs que se conectaran a la red por lo que conexiones de RJ45 son 30, pero las reglas de cableado estructurado marca que se debe de dejar el 30 % de redundancia, por lo tanto el total de conexiones RJ45 son de 39 , la ubicación de la primera placa se colocara a 1 m, la segunda a 2 m, la tercera a 2 m, la cuarta a 2m y la ultima a 1 metro ver g.

2. La distancia del cable de conexión del closet de comunicación hacia la PCs estarán colocados dentro de la pared, junto a las escaleras y son 40 cables UTP categoría 5e, para las las y las placas de servicio. El cableado que va hacia la primera la es de 11m, la segunda es de

9m, la tercera es de 7m, la cuarta es de 5m y la última es de 3m con una distancia a la última placa de 8 m. La dimensión del closet de

comunicación es de 1.20 x 1.60 m. ver g. 3.

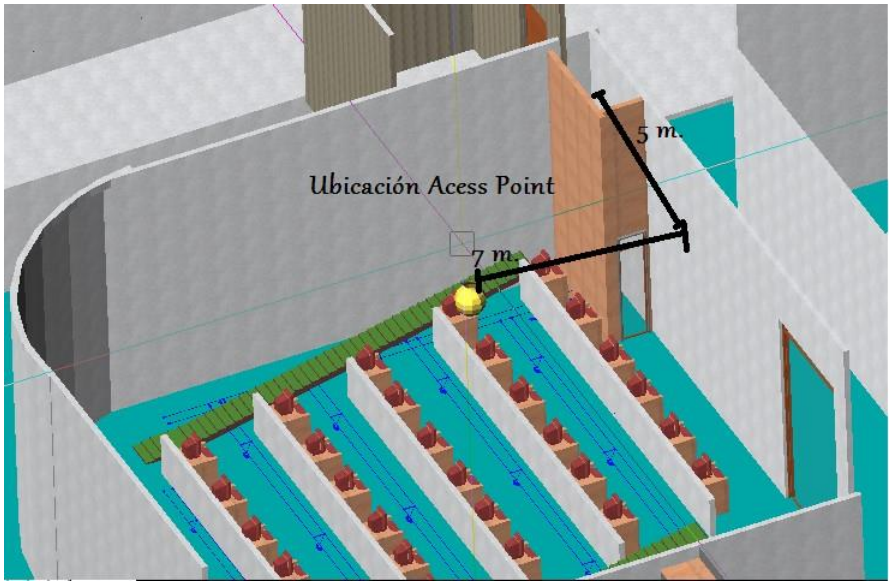


Fig.1 Ubicación del Access Point hacia el closet de comunicación

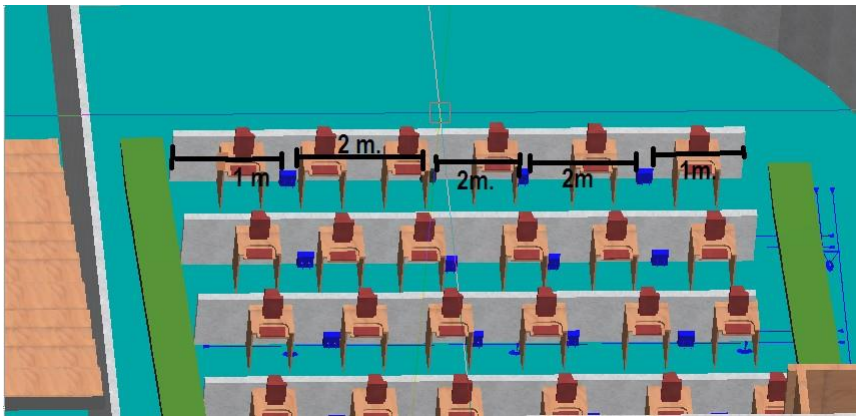


Fig. 2 Ubicación de las placas de servicio (laboratorios)

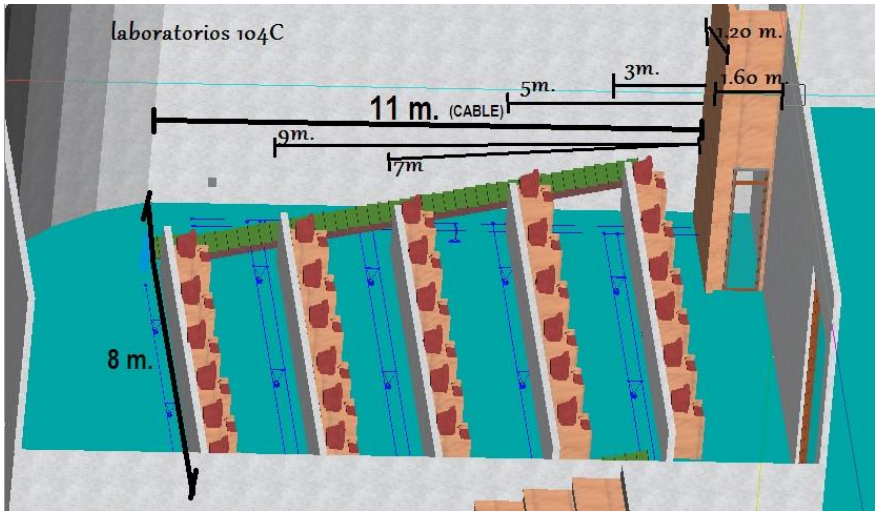


Fig. 3 Longitud de cable UTP hacia las placas de servicio y medida del closet de comunicación (laboratorios)

La distancia de conexión del cada closet de comunicación hacia la Acometida entrara y saldrá por el laboratorio 2, el cable ira en el suelo y con protección.

Las medidas del cableado que sale del laboratorio 4 son de 10 m del closet de comunicación hacia el laboratorio 2, con 2 metros hacia el pasillo donde va la protección del cable, del 1 es de 13 m del closet de comunicación hacia la protección, del 3 es de 11metros y el cableado del cuarto de comunicación del edi cio 104D es de 20 m hacia el modulo 102, ver g. 4. Ahora por el laboratorio 2 saldrá el cableado de los módulos hacia la Acometida pasando a lo largo de la hasta la ultima ventana con una distancia de 10 m de largo x 6.5 m de bajada y 12 m a la Acometida ver g.5.

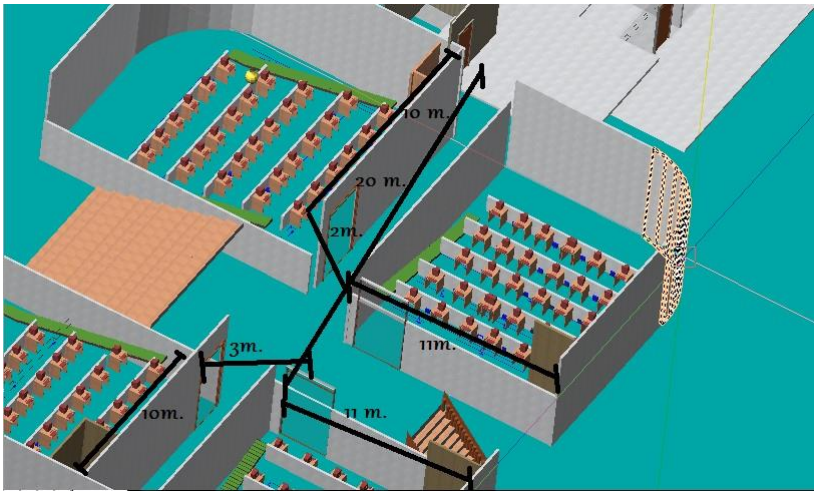


Fig. 4 cableado fuera de los laboratorios

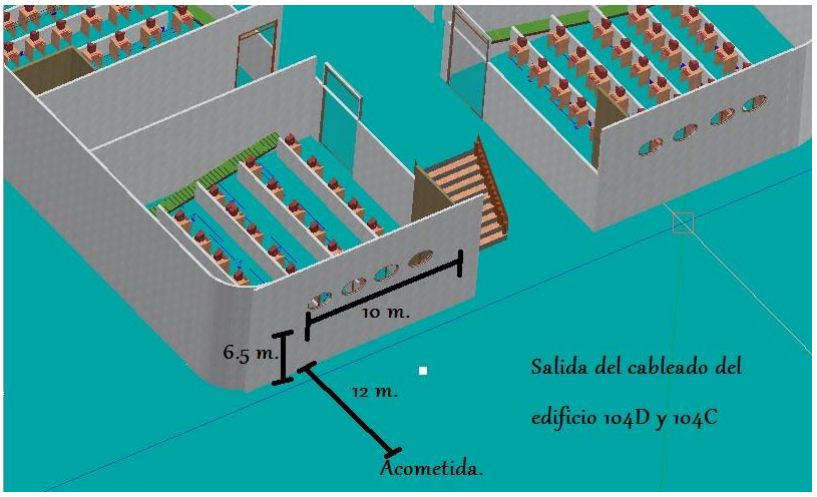


Fig. 5 Cableado de los laboratorios y del edi cio 104D hacia la Acometida

2.1.3. Edi cio 104 D

Para cada salón se les da un red especi ca, un servicio de internet alámbrico e inalámbrico y para ello cada salón tiene un switch y Access Point asociado, por lo que cada salón tiene 4 placas de servicio de 2 RJ45, y un el closet de comunicación que se ubicara detrás de la puerta a 50 cm de la pared, con una dimensión de 50cm. La ubicación del Access Point se ubicara 4.5 m de la pared hacia la mitad del salón a 4 metros. Las placas de servicio se colocaran a mitad de cada pared

con una altura de 50 cm del piso hacia arriba. La primera placa se encuentra a un metro del pizarrón la segunda a la mitad de la pared estas les llegara el cable del lado del Access Point ver g.6, las dos últimas: una se encuentra a lado de la puerta y la otra a mitad de pared esta se conectan hacia closet de comunicación ver g.7.

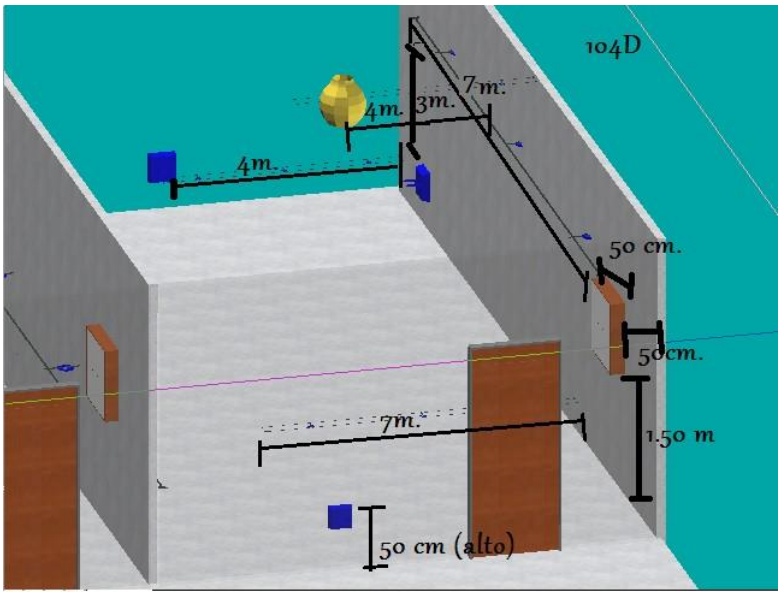


Fig.6 Ubicación del Closet comunicación, placas y Access Point (salón 104D)

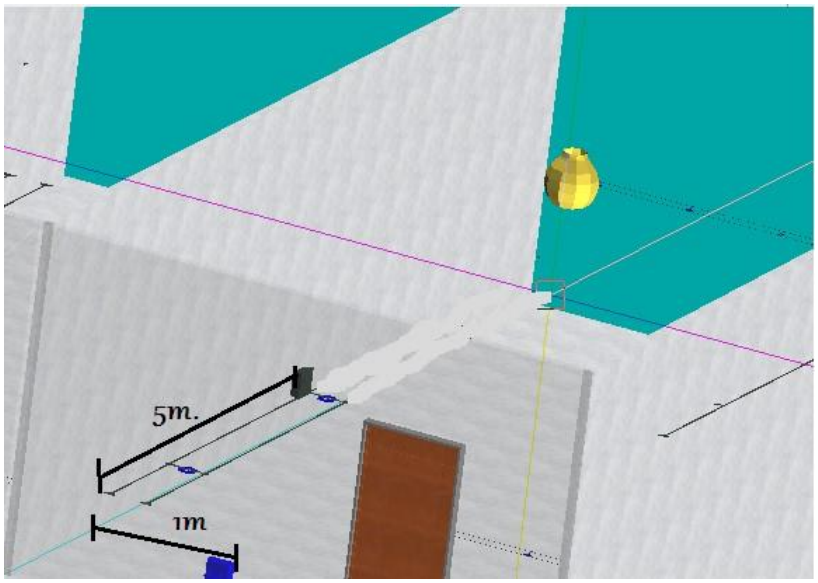


Fig.7 Ubicación de las últimas 2 placas de servicio: una a lado de la entrada y otra atrás. (salón 104D)

Ahora la conexión del closet de comunicación de cada salón, se conectara hacia los router que están asignados a cada piso del edi co que estarán ubicados en un cuarto de comunicación que se ubicara al lado del puente que esta en entre el edi cio 104C y 104D, la dimensión del cuarto de comunicación será de 2.5m x 1m. El cableado que será asignado a cada piso bajara hacia el puente, que por allí ira en un tubo de protección al cuarto de equipo ver g. 8.

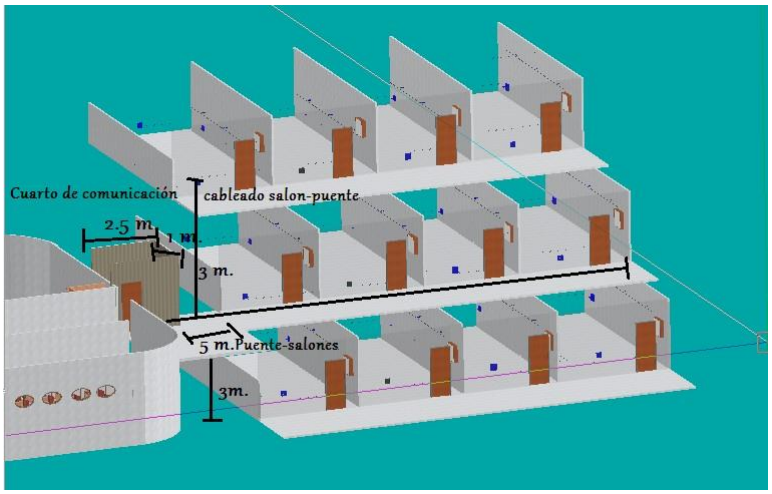


Fig.8 Edi cio 104D

2.2. Diseño de las redes

Teniendo en mente una mayor velocidad y por supuesto una futura expansión (si fuese necesaria) de red, se propone implementar una red distinta para cada área de trabajo, para esto colocaremos un dispositivo switch por cada área de trabajo. Para cada área de trabajo estaremos implementando una dirección IP compuesta de la siguiente manera: Numero de facultad . Numero de edi cio . Numero de salón (Área de trabajo) . Numero de host

Por ejemplo.

Sabemos que la Facultad de Ciencias de la Computación (FCC) es identi cada por el número 104. Para localizar una IP en especí co, no tendríamos inconvenientes al identi car la siguiente información:

Numero de facultad: 104

Numero de edi cio: 3 (104C -Queso-)

Numero de salón (laboratorio): 4

Numero de host : 27

Con los datos anteriormente obtenidos podemos construir la dirección IP 104.3.4.27, la cual, corresponde al host 27, ubicado en el laboratorio tres, del edi cio C, en la FCC.

Por lo anterior estaríamos montando un total de 26 redes distribuidas de la siguiente manera:

2 Redes para el edi cio 104A.

4 Redes para el edi cio 104C (una por laboratorio).

12 Redes para el edi cio 104D(una por salón).

8 Redes para intercomunicar a todas las anteriores.

Para toda área de trabajo se empleará un 30 % de redundancia.

2.2.1. Redes del edi cio 104A

En este edi cio se localizan los 23 cubículos de profesores, cada cubículo alo ja a 2 PCs de escritorio por lo cual, considerando el 30 % de redundancia, debemos de colocar 3 puntos de acceso a la red.

Implementaremos 2 redes una por cada lado para no saturar una sola salida, las redes son:

2.2.1.1. 104.1.1.0

Proporcionara servicio de internet a 12 cubículos, 36 puertos.

2.2.1.2. 104.1.2.0

Proporcionara servicio de internet a 11 cubículos restantes, 33 puertos.

Cada puerto es un canal independiente que se conecta al panel del parcheo, de este último se conecta directamente a un switch. Esto

es por cada red. Ambas redes se conectaran al router, al igual que los panel de parcheo y switches, está ubicado dentro del closet de comunicación local.

El router general se conecta al panel de parcheo de la acometida.

```
interface FastEthernet0/0
 ip address 192.168.5.2 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet1/0
 ip address 104.1.1.1 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet2/0
 ip address 104.1.2.1 255.255.255.0
 duplex auto
 speed auto
!
router rip
!
ip classless
ip route 104.3.1.0 255.255.255.0 192.168.5.1
ip route 104.3.2.0 255.255.255.0 192.168.5.1
ip route 104.3.3.0 255.255.255.0 192.168.5.1
ip route 104.3.4.0 255.255.255.0 192.168.5.1
ip route 192.168.1.0 255.255.255.0 192.168.5.1
ip route 192.168.2.0 255.255.255.0 192.168.5.1
ip route 192.168.3.0 255.255.255.0 192.168.5.1
ip route 192.168.4.0 255.255.255.0 192.168.5.1
ip route 192.168.254.0 255.255.255.0 192.168.5.1
ip route 192.168.6.0 255.255.255.0 192.168.5.1
ip route 192.168.7.0 255.255.255.0 192.168.5.1
ip route 192.168.8.0 255.255.255.0 192.168.5.1
ip route 104.4.1.0 255.255.255.0 192.168.5.1
ip route 104.4.2.0 255.255.255.0 192.168.5.1
ip route 104.4.3.0 255.255.255.0 192.168.5.1
ip route 104.4.4.0 255.255.255.0 192.168.5.1
ip route 104.4.5.0 255.255.255.0 192.168.5.1
ip route 104.4.6.0 255.255.255.0 192.168.5.1
ip route 104.4.7.0 255.255.255.0 192.168.5.1
ip route 104.4.8.0 255.255.255.0 192.168.5.1
ip route 104.4.9.0 255.255.255.0 192.168.5.1
ip route 104.4.10.0 255.255.255.0 192.168.5.1
ip route 104.4.11.0 255.255.255.0 192.168.5.1
ip route 104.4.12.0 255.255.255.0 192.168.5.1
```

Con guracion del router general 104A

2.2.2. Redes del edi cio 104C

En este edi cio se localizan los 4 laboratorios, cada cubículo alo ja a 30 PCs de escritorio por lo cual, considerando el 30 % de redundancia, debemos de colocar 39 puntos de acceso a la red. Además soporta conexiones inalámbricas a lo sumo 35 laptops.

Implementaremos 4 redes una por cada laboratorio, las redes y con-guraciones por cada router son :

2.2.2.1. 104.3.1.0

```
ip dhcp excluded-address 104.3.1.1
!
ip dhcp pool L1
 network 104.3.1.0 255.255.255.0
 default-router 104.3.1.1
!
!
!
!
!
!
!
!
!
!
!
!
interface FastEthernet0/0
 ip address 104.3.1.1 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet1/0
 ip address 192.168.1.2 255.255.255.0
 duplex auto
 speed auto
!
 version 2
!
ip classless
ip route 104.3.2.0 255.255.255.0 192.168.1.1
ip route 104.3.3.0 255.255.255.0 192.168.1.1
ip route 104.3.4.0 255.255.255.0 192.168.1.1
ip route 104.1.1.0 255.255.255.0 192.168.1.1
ip route 104.1.2.0 255.255.255.0 192.168.1.1
ip route 192.168.2.0 255.255.255.0 192.168.1.1
ip route 192.168.3.0 255.255.255.0 192.168.1.1
ip route 192.168.4.0 255.255.255.0 192.168.1.1
ip route 192.168.5.0 255.255.255.0 192.168.1.1
ip route 104.4.1.0 255.255.255.0 192.168.1.1
ip route 104.4.2.0 255.255.255.0 192.168.1.1
ip route 104.4.3.0 255.255.255.0 192.168.1.1
ip route 104.4.4.0 255.255.255.0 192.168.1.1
ip route 104.4.5.0 255.255.255.0 192.168.1.1
ip route 104.4.6.0 255.255.255.0 192.168.1.1
ip route 104.4.7.0 255.255.255.0 192.168.1.1
ip route 104.4.8.0 255.255.255.0 192.168.1.1
ip route 104.4.9.0 255.255.255.0 192.168.1.1
ip route 104.4.10.0 255.255.255.0 192.168.1.1
ip route 104.4.11.0 255.255.255.0 192.168.1.1
ip route 104.4.12.0 255.255.255.0 192.168.1.1
ip route 192.168.6.0 255.255.255.0 192.168.1.1
ip route 192.168.7.0 255.255.255.0 192.168.1.1
ip route 192.168.8.0 255.255.255.0 192.168.1.1
ip route 192.168.254.0 255.255.255.0 192.168.1.1
```

2.2.2.2. 104.3.2.0

```
ip dhcp excluded-address 104.3.2.1
!
ip dhcp pool L2
 network 104.3.2.0 255.255.255.0
 default-router 104.3.2.1
!
!
!
!
!
!
!
!
!
!
!
!
!
interface FastEthernet0/0
 ip address 104.3.2.1 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet1/0
 ip address 192.168.2.2 255.255.255.0
 duplex auto
 speed auto
!
ip classless
ip route 104.3.1.0 255.255.255.0 192.168.2.1
ip route 104.3.3.0 255.255.255.0 192.168.2.1
ip route 104.3.4.0 255.255.255.0 192.168.2.1
ip route 192.168.1.0 255.255.255.0 192.168.2.1
ip route 192.168.3.0 255.255.255.0 192.168.2.1
ip route 192.168.4.0 255.255.255.0 192.168.2.1
ip route 104.1.1.0 255.255.255.0 192.168.2.1
ip route 104.1.2.0 255.255.255.0 192.168.2.1
ip route 192.168.5.0 255.255.255.0 192.168.2.1
ip route 104.4.1.0 255.255.255.0 192.168.2.1
ip route 192.168.254.0 255.255.255.0 192.168.2.1
ip route 192.168.6.0 255.255.255.0 192.168.2.1
ip route 104.4.2.0 255.255.255.0 192.168.2.1
ip route 104.4.3.0 255.255.255.0 192.168.2.1
ip route 104.4.4.0 255.255.255.0 192.168.2.1
ip route 104.4.5.0 255.255.255.0 192.168.2.1
ip route 104.4.6.0 255.255.255.0 192.168.2.1
ip route 104.4.7.0 255.255.255.0 192.168.2.1
ip route 104.4.8.0 255.255.255.0 192.168.2.1
ip route 104.4.9.0 255.255.255.0 192.168.2.1
ip route 104.4.10.0 255.255.255.0 192.168.2.1
ip route 104.4.11.0 255.255.255.0 192.168.2.1
ip route 104.4.12.0 255.255.255.0 192.168.2.1
ip route 192.168.7.0 255.255.255.0 192.168.2.1
ip route 192.168.8.0 255.255.255.0 192.168.2.1
```

2.2.2.3. 104.3.3.0

```
ip dhcp excluded-address 104.3.3.1
!
ip dhcp pool L3
 network 104.3.3.0 255.255.255.0
 default-router 104.3.3.1
!
!
!
!
!
!
!
!
!
!
!
!
interface FastEthernet0/0
 ip address 104.3.3.1 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet1/0
 ip address 192.168.3.2 255.255.255.0
 duplex auto
 speed auto
!
ip classless
ip route 104.3.1.0 255.255.255.0 192.168.3.1
ip route 104.3.2.0 255.255.255.0 192.168.3.1
ip route 104.3.4.0 255.255.255.0 192.168.3.1
ip route 192.168.1.0 255.255.255.0 192.168.3.1
ip route 192.168.2.0 255.255.255.0 192.168.3.1
ip route 192.168.4.0 255.255.255.0 192.168.3.1
ip route 192.168.5.0 255.255.255.0 192.168.3.1
ip route 104.1.1.0 255.255.255.0 192.168.3.1
ip route 104.1.2.0 255.255.255.0 192.168.3.1
ip route 104.4.1.0 255.255.255.0 192.168.3.1
ip route 104.4.2.0 255.255.255.0 192.168.3.1
ip route 104.4.3.0 255.255.255.0 192.168.3.1
ip route 104.4.4.0 255.255.255.0 192.168.3.1
ip route 104.4.5.0 255.255.255.0 192.168.3.1
ip route 104.4.6.0 255.255.255.0 192.168.3.1
ip route 104.4.7.0 255.255.255.0 192.168.3.1
ip route 104.4.8.0 255.255.255.0 192.168.3.1
ip route 104.4.9.0 255.255.255.0 192.168.3.1
ip route 104.4.10.0 255.255.255.0 192.168.3.1
ip route 104.4.11.0 255.255.255.0 192.168.3.1
ip route 104.4.12.0 255.255.255.0 192.168.3.1
ip route 192.168.6.0 255.255.255.0 192.168.3.1
ip route 192.168.7.0 255.255.255.0 192.168.3.1
ip route 192.168.8.0 255.255.255.0 192.168.3.1
ip route 192.168.254.0 255.255.255.0 192.168.3.1
```

2.2.2.4. 104.3.4.0

2.2.2.

```
 duplex auto
 speed auto
!
interface FastEthernet1/0
 ip address 192.168.4.2 255.255.255.0
 duplex auto
 speed auto
ip classless
ip route 104.3.1.0 255.255.255.0 192.168.4.1
ip route 104.3.2.0 255.255.255.0 192.168.4.1
ip route 104.3.3.0 255.255.255.0 192.168.4.1
ip route 192.168.1.0 255.255.255.0 192.168.4.1
ip route 192.168.2.0 255.255.255.0 192.168.4.1
ip route 192.168.3.0 255.255.255.0 192.168.4.1
ip route 104.1.2.0 255.255.255.0 192.168.4.1
ip route 104.1.1.0 255.255.255.0 192.168.4.1
ip route 192.168.5.0 255.255.255.0 192.168.4.1
ip route 192.168.254.0 255.255.255.0 192.168.4.1
ip route 192.168.6.0 255.255.255.0 192.168.4.1
ip route 192.168.7.0 255.255.255.0 192.168.4.1
ip route 192.168.8.0 255.255.255.0 192.168.4.1
ip route 104.4.1.0 255.255.255.0 192.168.4.1
ip route 104.4.2.0 255.255.255.0 192.168.4.1
ip route 104.4.3.0 255.255.255.0 192.168.4.1
ip route 104.4.4.0 255.255.255.0 192.168.4.1
ip route 104.4.5.0 255.255.255.0 192.168.4.1
ip route 104.4.6.0 255.255.255.0 192.168.4.1
ip route 104.4.7.0 255.255.255.0 192.168.4.1
ip route 104.4.8.0 255.255.255.0 192.168.4.1
ip route 104.4.9.0 255.255.255.0 192.168.4.1
ip route 104.4.10.0 255.255.255.0 192.168.4.1
ip route 104.4.11.0 255.255.255.0 192.168.4.1
ip route 104.4.12.0 255.255.255.0 192.168.4.1
!
```

Para las conexiones inalámbricas se colocara un AP en el techo al centro del laboratorio, debidamente limitado su radio de expansión. Cada puerto es un canal independiente que se conecta al panel del parcheo, de este último se conecta directamente a un switch, posteriormente

se conecta a un router, esto es dentro del panel de comunicación. Así mismo para cada red. Los 4 routers generales serán conectados directamente al panel de parcheo de la acometida.

2.2.3. Redes del edi cio 104D

En este edi cio se localizan los 12 salones de clase, cada salón alo ja a 35 PCs.

Mediante una encuesta realizada en la facultad de ciencias de la computación, se determinó que los alumnos pre eren tener conectividad inalámbrica dentro del salón a pesar de que la alámbrica es más rápida y segura. Por ello se instalaran 6 puertos estáticos alrededor del salón en puntos estratégicos, y un AP en el techo al centro del mismo con una capacidad de a lo sumo 30 laptops conectadas a el.

El AP será con gurado para que proporcione servicio solo dentro del salón, por lo cual los accesos a la red estáticos serán colocados cerca de las esquinas, con esto garantizamos la conectividad en cualquier parte del salón.

Cada punto de acceso es un canal independiente que al igual que el AP se conectan al panel de parcheo y este a su vez se conecta al switch, todo esto ubicado dentro del closet de comunicación. Esto es para cada uno de los 4 salones.

De cada closet de comunicación sale un canal que se conecta al panel de parcheo y a su vez al router general, este último se conectara directamente al panel de parcheo de la acometida. Esto es por cada nivel del edi cio en cuestión.

El router asignado a cada nivel del edi cio, proporcionara el servicio de DHCP

2.2.3.1. 104.4.1.0, 104.4.2.0, 104.4.3.0,104.4.4.0

Con guracion del router asignado al nivel 1:

```
!
hostname Router
!
!
!
!
ip dhcp excluded-address 104.4.1.1
ip dhcp excluded-address 104.4.2.1
ip dhcp excluded-address 104.4.3.1
ip dhcp excluded-address 104.4.4.1
ip dhcp excluded-address 104.3.1.1
!
ip dhcp pool Q1
 network 104.4.1.0 255.255.255.0
 default-router 104.4.1.1
ip dhcp pool Q2
 network 104.4.2.0 255.255.255.0
 default-router 104.4.2.1
ip dhcp pool Q3
 network 104.4.3.0 255.255.255.0
 default-router 104.4.3.1
ip dhcp pool Q4
 network 104.4.4.0 255.255.255.0
 default-router 104.4.4.1
ip dhcp pool L1
 network 104.3.1.0 255.255.255.0
 default-router 104.3.1.1
!
!
!
interface FastEthernet3/0
 ip address 104.4.3.1 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet4/0
 ip address 104.4.4.1 255.255.255.0
 duplex auto
 speed auto
!
router rip
!
ip classless
ip route 104.4.5.0 255.255.255.0 192.168.6.1
ip route 192.168.7.0 255.255.255.0 192.168.6.1
ip route 104.4.6.0 255.255.255.0 192.168.6.1
ip route 104.4.7.0 255.255.255.0 192.168.6.1
ip route 104.4.8.0 255.255.255.0 192.168.6.1
ip route 104.4.9.0 255.255.255.0 192.168.6.1
ip route 104.4.10.0 255.255.255.0 192.168.6.1
ip route 104.4.11.0 255.255.255.0 192.168.6.1
ip route 104.4.12.0 255.255.255.0 192.168.6.1
ip route 192.168.8.0 255.255.255.0 192.168.6.1
ip route 104.3.2.0 255.255.255.0 192.168.6.1
ip route 192.168.254.0 255.255.255.0 192.168.6.1
ip route 192.168.2.0 255.255.255.0 192.168.6.1
ip route 104.1.1.0 255.255.255.0 192.168.6.1
ip route 104.1.2.0 255.255.255.0 192.168.6.1
ip route 104.3.1.0 255.255.255.0 192.168.6.1
ip route 104.3.3.0 255.255.255.0 192.168.6.1
ip route 104.3.4.0 255.255.255.0 192.168.6.1
ip route 192.168.1.0 255.255.255.0 192.168.6.1
ip route 192.168.3.0 255.255.255.0 192.168.6.1
ip route 192.168.4.0 255.255.255.0 192.168.6.1
ip route 192.168.5.0 255.255.255.0 192.168.6.1
!
```

2.2.3.2. 104.4.5.0, 104.4.6.0, 104.4.7.0,104.4.8.0

Con guracion del router asignado al nivel 2:

```
 ip dhcp excluded-address 104.4.5.1
 ip dhcp excluded-address 104.4.6.1
 ip dhcp excluded-address 104.4.7.1
 ip dhcp excluded-address 104.4.8.1
 !
 ip dhcp pool Q5
  network 104.4.5.0 255.255.255.0
  default-router 104.4.5.1
 ip dhcp pool Q6
  network 104.4.6.0 255.255.255.0
  default-router 104.4.6.1
 ip dhcp pool Q7
  network 104.4.7.0 255.255.255.0
  default-router 104.4.7.1
 ip dhcp pool Q8
  network 104.4.8.0 255.255.255.0
  default-router 104.4.8.1
interface FastEthernet0/0
 ip address 192.168.7.2 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet1/0
 ip address 104.4.5.1 255.255.255.0
 duplex auto
 speed auto
!


!
ip classless
ip route 104.4.1.0 255.255.255.0 192.168.7.1
ip route 192.168.6.0 255.255.255.0 192.168.7.1
ip route 104.4.2.0 255.255.255.0 192.168.7.1
ip route 104.4.3.0 255.255.255.0 192.168.7.1
ip route 104.4.4.0 255.255.255.0 192.168.7.1
ip route 104.4.9.0 255.255.255.0 192.168.7.1
ip route 104.4.10.0 255.255.255.0 192.168.7.1
ip route 104.4.11.0 255.255.255.0 192.168.7.1
ip route 104.4.12.0 255.255.255.0 192.168.7.1
ip route 192.168.8.0 255.255.255.0 192.168.7.1
ip route 104.1.1.0 255.255.255.0 192.168.7.1
ip route 104.1.2.0 255.255.255.0 192.168.7.1
ip route 104.3.1.0 255.255.255.0 192.168.7.1
ip route 104.3.2.0 255.255.255.0 192.168.7.1
ip route 104.3.3.0 255.255.255.0 192.168.7.1
ip route 104.3.4.0 255.255.255.0 192.168.7.1
!
!
```

2.2.3.3. 104.4.9.0, 104.4.10.0, 104.4.11.0,104.4.12.0

Con guracion del router asignado al nivel 3:

```
!
ip dhcp excluded-address 104.4.9.1
ip dhcp excluded-address 104.4.10.1
ip dhcp excluded-address 104.4.11.1
ip dhcp excluded-address 104.4.12.1
!
ip dhcp pool Q9
 network 104.4.9.0 255.255.255.0
 default-router 104.4.9.1
ip dhcp pool Q10
 network 104.4.10.0 255.255.255.0
 default-router 104.4.10.1
ip dhcp pool Q11
 network 104.4.11.0 255.255.255.0
 default-router 104.4.11.1
ip dhcp pool Q12
 network 104.4.12.0 255.255.255.0
 default-router 104.4.12.1
!
```

```
!
ip classless
ip route 104.4.1.0 255.255.255.0 192.168.8.1
ip route 104.4.2.0 255.255.255.0 192.168.8.1
ip route 104.4.3.0 255.255.255.0 192.168.8.1
ip route 104.4.4.0 255.255.255.0 192.168.8.1
ip route 104.4.5.0 255.255.255.0 192.168.8.1
ip route 104.4.6.0 255.255.255.0 192.168.8.1
ip route 104.4.7.0 255.255.255.0 192.168.8.1
ip route 104.4.8.0 255.255.255.0 192.168.8.1
ip route 192.168.6.0 255.255.255.0 192.168.8.1
ip route 192.168.7.0 255.255.255.0 192.168.8.1
ip route 104.1.1.0 255.255.255.0 192.168.8.1
ip route 104.1.2.0 255.255.255.0 192.168.8.1
ip route 104.3.1.0 255.255.255.0 192.168.8.1
ip route 104.3.2.0 255.255.255.0 192.168.8.1
ip route 104.3.3.0 255.255.255.0 192.168.8.1
ip route 104.3.4.0 255.255.255.0 192.168.8.1
ip route 192.168.254.0 255.255.255.0 192.168.8.1
ip route 192.168.1.0 255.255.255.0 192.168.8.1
ip route 192.168.2.0 255.255.255.0 192.168.8.1
ip route 192.168.3.0 255.255.255.0 192.168.8.1
ip route 192.168.4.0 255.255.255.0 192.168.8.1
ip route 192.168.5.0 255.255.255.0 192.168.8.1
!
```

2.2.4. Routers principales

Los routers principales estarán ubicados dentro de la acometida, dichos routers soportan comunicación por bra óptica multimodo a una velocidad de un Ten GB, así mismo son capaces de balancear las cargas de salida para no saturar demasiado una sola salida.

Para balancear las cargas de red en los routers principales, asumimos el peor de los casos en los que se puede encontrar cada red.

- La red 104.1.1.0 a lo sumo proporcionará servicio a 36 PCs
- La red 104.1.2.0 a lo sumo proporcionará servicio a 33 PCs
- La red 104.3.1.0 a lo sumo proporcionará servicio a 70 PCs de las cuales 40 alámbricas y 30 inalámbricas
- La red 104.3.2.0 a lo sumo proporcionará servicio a 70 PCs de las cuales 40 alámbricas y 30 inalámbricas
- La red 104.3.3.0 a lo sumo proporcionará servicio a 70 PCs de las cuales 40 alámbricas y 30 inalámbricas
- La red 104.3.4.0 a lo sumo proporcionará servicio a 70 PCs de las cuales 40 alámbricas y 30 inalámbricas
- La red 104.4.1.0 a lo sumo proporcionará servicio a 38 PCs de las cuales 8 alámbricas y 30 inalámbricas
- La red 104.4.2.0 a lo sumo proporcionará servicio a 38 PCs de las cuales 8 alámbricas y 30 inalámbricas

- La red 104.4.3.0 a lo sumo proporcionará servicio a 38 PCs de las cuales 8 alámbricas y 30 inalámbricas
- La red 104.4.4.0 a lo sumo proporcionará servicio a 38 PCs de las cuales 8 alámbricas y 30 inalámbricas
- La red 104.4.5.0 a lo sumo proporcionará servicio a 38 PCs de las cuales

8 alámbricas y 30 inalámbricas

- La red 104.4.6.0 a lo sumo proporcionará servicio a 38 PCs de las cuales 8 alámbricas y 30 inalámbricas

- La red 104.4.7.0 a lo sumo proporcionará servicio a 38 PCs de las cuales 8 alámbricas y 30 inalámbricas

- La red 104.4.8.0 a lo sumo proporcionará servicio a 38 PCs de las cuales 8 alámbricas y 30 inalámbricas

- La red 104.4.9.0 a lo sumo proporcionará servicio a 38 PCs de las cuales 8 alámbricas y 30 inalámbricas

- La red 104.4.10.0 a lo sumo proporcionará servicio a 38 PCs de las cuales 8 alámbricas y 30 inalámbricas

- La red 104.4.11.0 a lo sumo proporcionará servicio a 38 PCs de las cuales 8 alámbricas y 30 inalámbricas

- La red 104.4.12.0 a lo sumo proporcionará servicio a 38 PCs de las cuales 8 alámbricas y 30 inalámbricas

En resumen:

Laboratorios	
Filas	5
PCs por fila	6
Total	30
Alambricas	30
30%	10
Inalambricas	30
Total	70
4 Laboratorios	280

Salones	
Alambricas	6
30%	2
Inalambricas	30
Total	38
4 Salones	152
3 Pisos	456

Cubiculos	
Alambricas	2
30%	1
Inalambricas	0
Total	3
23 Cubiculos	69

Con lo anterior tendríamos un total de 805 hosts

2.2.4.1. Router principal 1

Proveerá de servicios a las redes

- 104.1.1.0

- 104.1.2.0

- 104.2.1.0

- 104.2.2.0
- 104.2.3.0
- 104.2.4.0

Y tendrá la siguiente con guración:

```
router rip
!
ip classless
ip route 104.3.1.0 255.255.255.0 192.168.1.2
ip route 104.3.2.0 255.255.255.0 192.168.2.2
ip route 104.3.3.0 255.255.255.0 192.168.3.2
ip route 104.3.4.0 255.255.255.0 192.168.4.2
ip route 104.1.1.0 255.255.255.0 192.168.5.2
ip route 104.1.2.0 255.255.255.0 192.168.5.2
ip route 104.4.1.0 255.255.255.0 192.168.254.2
ip route 104.4.2.0 255.255.255.0 192.168.254.2
ip route 104.4.3.0 255.255.255.0 192.168.254.2
ip route 104.4.4.0 255.255.255.0 192.168.254.2
ip route 104.4.5.0 255.255.255.0 192.168.254.2
ip route 104.4.6.0 255.255.255.0 192.168.254.2
ip route 104.4.7.0 255.255.255.0 192.168.254.2
ip route 104.4.8.0 255.255.255.0 192.168.254.2
ip route 104.4.9.0 255.255.255.0 192.168.254.2
ip route 104.4.10.0 255.255.255.0 192.168.254.2
ip route 104.4.11.0 255.255.255.0 192.168.254.2
ip route 104.4.12.0 255.255.255.0 192.168.254.2
!
```

2.2.4.2. Router principal 2

Proveerá de servicios a las redes

- 104.4.1.0
- 104.4.2.0

- 104.4.3.0
- 104.4.4.0
- 104.4.5.0
- 104.4.6.0
- 104.4.7.0
- 104.4.8.0
- 104.4.9.0
- 104.4.10.0
- 104.4.11.0
- 104.4.12.0

Y tendrá la siguiente con guración:

```
interface FastEthernet0/0
 ip address 192.168.6.1 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet1/0
 ip address 192.168.7.1 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet2/0
 ip address 192.168.8.1 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet3/0
 no ip address
 duplex auto
 speed auto
 shutdown
!
interface FastEthernet4/0
 ip address 192.168.254.2 255.255.255.0
 duplex auto
 speed auto
!
```

```
 shutdown
!
ip classless
ip route 104.4.1.0 255.255.255.0 192.168.6.2
ip route 104.4.5.0 255.255.255.0 192.168.7.2
ip route 104.4.2.0 255.255.255.0 192.168.6.2
ip route 104.4.3.0 255.255.255.0 192.168.6.2
ip route 104.4.4.0 255.255.255.0 192.168.6.2
ip route 104.4.6.0 255.255.255.0 192.168.7.2
ip route 104.4.7.0 255.255.255.0 192.168.7.2
ip route 104.4.8.0 255.255.255.0 192.168.7.2
ip route 104.4.9.0 255.255.255.0 192.168.8.2
ip route 104.4.10.0 255.255.255.0 192.168.8.2
ip route 104.4.11.0 255.255.255.0 192.168.8.2
ip route 104.4.12.0 255.255.255.0 192.168.8.2
ip route 104.3.2.0 255.255.255.0 192.168.254.1
ip route 192.168.2.0 255.255.255.0 192.168.254.1
ip route 104.1.1.0 255.255.255.0 192.168.254.1
ip route 104.1.2.0 255.255.255.0 192.168.254.1
ip route 104.3.1.0 255.255.255.0 192.168.254.1
ip route 104.3.3.0 255.255.255.0 192.168.254.1
ip route 104.3.4.0 255.255.255.0 192.168.254.1
'
```

2.3. Especi caciones de Material

2.3.1. Router principal

CCR1016 es Router de grado industrial, súper rápido con el 16 core de CPU. La mejor opción si se necesitan muchos millones de paquetes por segundo. El Cloud Core Router es impulsado por RouterOS, un sistema operativo con todas las funciones de enrutamiento que se ha mejorado de forma continua durante quince años. El enrutamiento dinámico, punto de acceso, rewall, MPLS, VPN, calidad de servicio avanzada, balanceo de carga y la unión, la con guración en tiempo real y monitoreo son algunas funciones compatibles con RouterOS. El dispositivo está disponible con una ca ja de montaje en rack 1U, cuenta con doce puertos Ethernet Gigabit, un cable de consola serial y un puerto USB. El CCR1016-12G tiene dos ranuras SODIMM, por defecto se envía con 2 GB de RAM, pero no tiene ningún límite de memoria en RouterOS (acepta y puede utilizar 16 GB o más). Viene con la fuente de alimentación y la ca ja de montaje en rack 1U.

ESPECIFICACIONES

- Código del producto: CCR1016-12G
- CPU frecuencia nominal: 1.2 GHz
- Número de núcleos de CPU: 16

- Tamaño de RAM: 2 GB
- Arquitectura: Tile GX
- Puertos 10/100 Ethernet: 0

- Puertos Ethernet 10/100/1000: 12

- Slots MiniPCI: 0

- Slots MiniPCI-e: 0

ˆ Cantidad de puertos USB: 1

ˆ Power Jack: 1 ˆ PoE: No

ˆ Monitor de Voltaje: Sí

ˆ Monitor de temperatura PCB: No

ˆ Monitor de temperatura de la CPU: Sí

ˆ Dimensiones: 355x145x55mm

ˆ Sistema Operativo: RouterOS v6 (64 bits)

ˆ Nivel de licencia: Level 6

ˆ Ganancia de la antena DBI : No

ˆ Monitor de corriente: Sí

ˆ CPU: TLR4-01680CG-12CE-A3a

ˆ Consumo de energía máximo: 38W

2.3.2. Routers

Router Cisco 4 puertos Ethernet 10/100 Modelo: RV042

Puertos e Interfaces	
Conexión WAN	Ethernet (RJ-45)
Ethernet LAN (RJ-45) cantidad de puertos	4
Protocolos	
Protocolos de red compatibles	PPPoE, PPTP
DHCP, cliente	
DHCP, servidor	
Soporte DMZ	
Red	
Estándares de red	IEEE 802.3, IEEE 802.3ab
Seguridad	
Algoritmos de seguridad soportados	128-bit AES, 256-bit AES, 3DES
Método de autenticación	MD5/SHA1
Seguridad con cortafuegos	SPI
Soporte VPN	PPTP, L2TP, IPsec
Características de LAN Ethernet	
Tecnología de cableado Ethernet	10/100BASE-T(X)
Ethernet LAN, velocidad de transferencia de datos	10, 100 Mbit/s
Otras características	
Dimensiones (Ancho x Profundidad x Altura)	130 x 200 x 38,5 mm

2.3.3. Transformadores

Convertidor TP-Link RJ45 Gigabit/ bra óptica multimodo

TP-Link MC200CM Convertidor de medios que transforma una señal 1000Base-SX de bra óptica en una señal 1000Base-T de cobre (RJ45) y viceversa. Está equipado con un conector SC-Dúplex de bra óptica que permite establecer una conexión Gigabit de alta velocidad a una distancia de 550 m por medio de bra óptica multimodo 50/125 La señal emitida respeta la norma 1000Base-SX Shortwave de onda corta de 850 nm de transmisión y recepción de datos.

Funciones:

- Compatible con los estándares IEE 802.3ab e IEEE 802.3z.

- Conectores de bra óptica SC (cuadrados) y RJ45 Gigabit.

- Funciona a 1000 Mbps Full Dúplex por cable Cat. 5e ó Cat. 6

- Puerto RJ45 Auto MID/MID-X

- Indicadores LED de estado y actividad de red

- Alimentación externa (5 V CC/2 A) - 0,55 km de distancia por bra 50/125µm, 0,22 km por bra 62,5/125 µm - Dimensiones: 94,5x73x27 mm

2.3.4. Switches

Cisco Gigabit Ethernet Switch Catalyst 4948, 48 Puertos, 96 Gbit/s, 32.768 Entradas

Cisco Catalyst 4948 10GE-S Switch. Tasa de transferencia (máx): 1 Gbit/s, Capacidad de conmutación: 96 Gbit/s, Velocidad de transferencia (paquete): 72 Mpps. Tabla de direcciones MAC: 32768 entradas. Protocolos de gestión: SNMP, HTTP, Telnet, RMON, Protocolo de conmutación: ACL, ARP, BOOTP, ICMP, IGMP, IP, IPv6, IPX, RADIUS, SSH, TCP, UDP, DHCP, TFTP, Protocolo de trasmisión de datos: Ethernet, Fast Ethernet, Gigabit Ethernet. Peso: 2 kg. Intervalo de humedad relativa para funcionamiento: 10 - 90 %, Intervalo de temperatura operativa: 0 - 40 °C, Intervalo de temperatura de almacenaje: -40 - 75 °C

Velocidad de reloj	1666 MHz
Puertos de entrada y salida (E/S)	48 x 10/100/1000Base-T (RJ-45) 4 x SFP 1 x Serial RJ-45
Cantidad de puertos	48
Tabla de direcciones MAC	32768 entradas
Estándares de red	IEEE 802.1D, IEEE 802.1p, IEEE 802.1Q, IEEE 802.1s, IEEE 802.1w, IEEE 802.1x, IEEE 802.3, IEEE 802.3ad, IEEE 802.3u, IEEE 802.3z
Intervalo de temperatura operativa	0 - 40 °C
Intervalo de temperatura de almacenaje	-40 - 75 °C
Intervalo de humedad relativa para funcionamiento	10 - 90 %
Peso	2 kg
RAM mínima	256 MB
Memoria Flash	16 MB
Seguridad	UL 60950-1, CAN/CSA-C22.2 No. 60950-1, EN 60950-1, IEC 60950-1, AS/NZS 60950, IEC 60825-1, IEC 60825-2, EN 60825-1, EN 60825-2, 21 CFR 1040
Bidireccional completo (Full dúplex)	✓
Capacidad de conmutación	96 Gbit/s
Velocidad de transferencia (paquete)	72 Mbps
Tasa de transferencia (máx.)	1 Gbit/s
Protocolo de conmutación	ACL, ARP, BOOTP, ICMP, IGMP, IP, IPv6, IPX, RADIUS, SSH, TCP, UDP, DHCP, TFTP
Protocolos de gestión	SNMP, HTTP, Telnet, RMON
Protocolo de trasmisión de datos	Ethernet, Fast Ethernet, Gigabit Ethernet
Indicadores LED	✓
Velocidad de transferencia de datos	1000 Mbit/s
Plataforma de gestión	Cisco CLI, Cisco Works LMS
Dimensiones (Ancho x Profundidad x Altura)	439 x 401 x 44.5 mm

Switch Inteligente Gigabit Ethernet para Montaje en Rack de 16 puertos

Flexibilidad extrema Dieciséis puertos 10/100/1000 Mbps autosensitivos que automáticamente detectan las velocidades óptimas de la red.

Fácil Administración Su administración inteligente basada en Web le permite una con guración sencilla.

Garantía de por Vida

Los estrictos estándares de fabricación aseguran la máxima calidad en todos los productos INTELLINET NETWORK SOLUTIONS. Todos los artículos cuentan con una Garantía de por Vida, el compromiso de calidad más fuerte que alguien puede ofrecer.

Estándares

- IEEE 802.1d (Protocolo Spanning tree)
- IEEE 802.1p (Prioridad de Trá co)

- IEEE 802.1q (VLAN Basada en etiquetas)
- IEEE 802.3 (10Base-T Ethernet)
- IEEE 802.3u (100Base-TX Fast Ethernet)

- IEEE 802.3ab (Gigabit Ethernet para par trenzado)

- IEEE 802.3ad (Link Aggregation)

- IEEE 802.3x (control de ujo para modo full dúplex)

General

ˆ Medios soportados

- 10Base-T Cat3,4, 5 UTP/STP RJ-45

- 100Base-TX Cat5 UTP/STP RJ-45

- 1000Base-T Cat5e UTP/ STP RJ-45

ˆ Filtrado de paquetes/ tasa de envío:

- 1,488,000 pps (1000 Mbps)

- 148,000 pps (100 Mbps)

- 14,880 pps (10 Mbps)

ˆ Tabla de direcciones MAC: 8192 entradas

ˆ Velocidad de backplane: 32 Gbps

ˆ Arquitectura de conmutación: almacenaje y envío

ˆ Certi cados EMI y seguridad: FCC clase A, marca CE

Opciones de Con guración 44

ˆ Velocidad de enlace de cada puerto: 10 Mbps, 100 Mbps, 1000 Mbps o autonegociación

ˆ Control de ujo encendido/ apagado por puerto

- VLAN

- Con guración de Tromentas de Broadcast con tasa ICMP, tasa de broadcast, tasa de multicast, tasa de unicast

- Gabinete metálico, 19", montaje en rack, 1 U

- Dimensiones: 440 (Largo) x 205 (Ancho) x 44 (Alto)

- Peso: 3.6 kg

ˆ Temperatura de operación: 0° C - 40° C

ˆ Factor de humedad: 10 % 90 % RH, sin condensación

ˆ Temperatura de almacenamiento: -40° C 70° C

2.3.5. Panel de parcheo

48 puertos 5e

ˆ Terminadores sólidos y con ables para cableado que presente jack con un mínimo de 3 mm a 50 micro pulgadas de baño de oro

ˆ Compatible con herramientas de ponchado 110 y Krone

ˆ Soporta cable trenzado sólido y multi lar de calibres 22 y 26 AWG

ˆ Para uso con cable de par trenzado (UTP) Cat5e

ˆ Ideal para aplicaciones de red Ethernet, fast Ethernet y Gigabit Ethernet

ˆ Conecta 48 puertos RJ-45 a la red ˆ Ocupa 2U de alto en el rack

ˆ Cumple con los estándares de montaje en rack de 19"

ˆ Garantía de por Vida

Especi caciones:

Estándares

ˆ IEEE 802.3 (10Base-T Ethernet)

ˆ IEEE 802.3u (100Base-TX Fast Ethernet)

ˆ IEEE 802.3ab (1000Base-T Gigabit Ethernet)

Certi caciones

ˆ Lista UL

ˆ Cumple con RoHS

Diseño

ˆ Dimensiones: 482 mm (Ancho) x 88 mm (Alto) x 48 mm (Profundo)

ˆ Distancia horizontal entre los hoyos izquierdo y derecho del montaje: 465 mm

ˆ Distancia vertical entre los hoyos superior e inferior de montaje: 75.8 mm

ˆ Peso: 1.27 kg

ˆ Temperatura de operación: -40 80°C

Contenido del paquete

ˆ Panel de parcheo Cat5e

ˆ 4 tornillos de montaje

ˆ Guia de instalación rápida

Panel de parcheo 12 puertos 5e

Modelo: 350-312

Panel de parcheo de 12 puertos Categoría 5e, para montaje en panel o pared. Está construido con altos estándares de calidad y cumple con todas las normas o ciales. Mide 25,4 cm de frente (10,16 pulgadas) x 5,6 cm de alto (2,24 pulgadas) x 3 cm de espesor (1,20 pulgadas).

La alta calidad de su nuevo panel de parcheo hace que sea la elección perfecta para cualquier Instalación IT. Fácil y rápidamente puede armar e identi car todos los puertos de la red.

Está disponible con 12 puertos (350-312), 24 puertos (350-324) y 48 puertos (350-348). Fabricado en acero de bajo carbón calibre 16, color negro.

ˆ Cumple con los estándares de categoría 5e ANSI/TIA/EIA-568-B.2.

ˆ Cumple con el estándar UL.[41]

ˆ Compatible con sistemas de cableado de menor velocidad.

ˆ Provee incremento de ancho de banda para utilizar pintura electrostática de alta duración.

ˆ Acero de bajo carbón de 1 mm.

ˆ Perforaciones y dimensiones para su montaje en rack.

ˆ Retardante al fuego UL 94 V-0.

ˆ Conductores sólidos terminados en 22-26 AWG.

ˆ Compatible con las herramientas de impacto 110.

ˆ Conectores retardantes al fuego UL-94 V-0.

ˆ Contactos laminados de oro de 50 micro pulgadas.

IEEE 802.3

- 10BaseT

- 100BaseT

- 1000BaseT Gigabit Ethernet

2.3.6. Access point

Access Point para Montaje en Techo de Alta Potencia 300N PoE

- 300 Mbps, 2T2R MIMO, Soporte PoE, Puente, Repetidor, Múltiples SSIDs y VLANs, 26 dBm, 400 mW

- Color: Blanco

- Ítem: 525251

El Access Point para Montaje en Techo de Alta Potencia 300N PoE Intellinet Network Solutions es lo último en tecnología inalámbrica. Aprovecha las ventajas de los avances en tecnología, una red inalámbrica puede ahora alcanzar velocidades mejoradas.

Especi caciones

Estándares

- IEEE 802.1d (Protocolo Spanning tree)

- IEEE 802.11b (WLAN, 11 Mbps)

- IEEE 802.11g (WLAN, 54 Mbps)

- IEEE 802.11n (300 Mbps WLAN)

- IEEE 802.1x (Control de acceso a la red)

- IEEE 802.3 (10Base-T Ethernet)

- IEEE 802.3af (PoE)

- IEEE 802.3u (100Base-TX Fast Ethernet)

General

ˆ Puertos LAN: 1 RJ45 10/100 Mbps datos y un puerto de salida de alimentación(puerto PD)

ˆ Puertos LAN con Auto MDI/MDI-X

ˆ Memoria Flash: 2 Mb

ˆ Memoria: 16 Mb SDRAM

ˆ Chipsets: Realtek RTL8196C y RTL8192CE

ˆ Certi caciones: FCC Clase B, Marca CE, RoHS

Inalámbrico

ˆ Rango de frecuencias:

- 2.400

- 2.483 GHz

ˆ Técnicas de modulación:

- 802.11b: Distribución de Espectro de Secuencia Directa (DSSS): DBPSK, DQPSK, CCK

- 802.11g: Multiplexaje por Dívisión de Frecuencia Ortogonal (OFDM): BPSK, QPSK, 16QAM, 64QAM

- 802.11n: Multiplexaje por Dívisión de Frecuencia Ortogonal (OFDM): BPSK, QPSK, 16QAM, 64QAM

ˆ Tasas de transferencia:

- IEEE 802.11b (11 Mbps, 5.5 Mbps, 2 Mbps, 1 Mbps)

- IEEE 802.11g (54 Mbps, 48 Mbps, 36 Mbps, 24 Mbps, 18 Mbps, 12 Mbps, 9 Mbps, 6 Mbps)

- IEEE 802.11n (MCS0-15: hasta 300 Mbps)

ˆ Potencia de salida:

- OFDM: 23 dBm +/- 1.5 dBm (300 Mbps, 200 mW máx.)

- OFDM: 24 dBm +/- 1.5 dBm (250 Mbps, 40 mW máx.)

- CCK: 26 dBm +/- 1 dBm (11 Mbps, 400 mW máx.)

- Up to 20 dBm (100 mW in Eurpean Union

ˆ Modos de operación inalámbricos:

- Access Point

- Estación Infraestructura (Cliente AP)

- Puente punto a punto

- Puente punto multipunto

- Puente WDS - Repetidor Universal

ˆ Seguridad inalámbrica:

- Encriptación WEP (64/128 bit)

- WPA TKIP

- WPA2 AES

- WPA2 mezclado

- WPA RADIUS

- 802.1x Autenticación

- Control de acceso a clientes mediante ltrado de direcciones MAC

ˆ Antenas:

- 2 antenas planas con conector F invertido PIFA con 3 dBi cada una

- 2T2R modo MIMO (2 transmisores, 2 receptores)

LEDs

- Alimentación
- WLAN Enlace/ Actividad
- LAN Enlace/ Actividad

Diseño

- Dimensiones: 160 (Ancho) x 127 (Largo) x 35 (Alto) mm
- Peso: 0.8 kg
- Temperatura de operación: 0° C 40° C
- Factor de humedad: 10 % 90 % RH, sin condensación
- Temperatura de almacenamiento: -20° C 60° C

Alimentación

- Adaptador de corriente externo: 5 V DC, 2.0 A
- Consumo de potencia: 4 Watts máximo
- Puerto PD para PoE: 48 V DC a 0,2 A

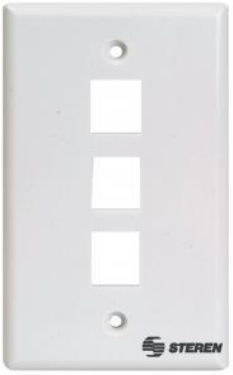
2.3.7. Tapas con servicios

Tapas de 2 entradas y 3 entradas

46

Placa Keystone de dos cavidades, para conector jack RJ45, color blanco

Placa para registro de 2 cavidades, tipo Keystone, para conector hembra (jack) de Categoría 5e y 6, en color blanco.



Placa Keystone de tres cavidades, para conector jack RJ45, color blanco (310-203WH)

Placa para registro de 3 cavidades, tipo Keystone, para conector hembra (jack) de Categoría 5e y 6, en color blanco.

- Cumple con los estándares:

  ANSI/TIA/EIA-568-B.2

  IEEE 802.3 ab

  UL

  CSA

  FCC part 68.5

  IEC-60603-7-2

- Conectores RJ45 y RJ11 montados en circuito impreso

- Indicadores de colores para inserción

- Conexión bajo la norma 568 A y B

2.3.8. Rack

Rack vertical de acero

Sencillo 42UR con puerta ventilada.

Los gabinetes Optronics de 42UR, son diseñados para alo jar equipo de telecomunicaciones con medida estándar de 19 , con la nalidad de administrar, organizar y proteger los sistemas de red. Los gabinetes Optronics de 42UR cumplen con los estándares ANSI/EIA/ RS-310-D, IEC297-2, DIN41497 y DIN41494, donde especi can el diseñado para alo jar equipos u accesorios para de telecomunicaciones.

Los gabinetes Optronics de 42UR tienen una fabricación de acero en calibre 12, 14 y 16 para sus diferentes componentes. Su estructura es con able y sólida, soportan grandes pesos estáticos, lo cual permite una gran versatilidad de con guración en el interior del gabinete, para lograr una efectiva y fácil instalación de accesorios o equipos activos, esto dependerá de las necesidades del proyecto que pretenda desarrollar.

Los gabinetes Optronics de 42UR están construidos de acuerdo a las necesidades de los cuartos de telecomunicaciones ya que contienen paneles que se desmontan en la parte superior e inferior para la entrada del cableado, siendo esta una ventaja para evitar perforaciones adicionales innecesarias, contiene 2 paneles laterales desmontables para facilitar con ello la instalación de los equipos y/o para dar mantenimiento de los mismos. Este tipo de gabinete consta de dos con guraciones sencillo o profundo, esta característica da la oportunidad de poder introducir en el gabinete equipo más amplio o dejar mayor espacio para la administración del cable según sea el caso del gabinete.

Los gabinetes Optronics de 42UR se encuentran disponible en 2 versiones con puerta de vidrio templado la cual permite visualizar los equipos albergados dentro del gabinete sin la necesidad de abrirlo o con puerta ventilada la cual consta de un diseño con perforaciones lo que permite brindar ventilación al gabinete y evitar que se sobrecalienten los equipos, las dos puertas cuentan con cerradura con dos llaves, la apertura de las puertas es en un ángulo de 180º. La puerta trasera del gabinete Optronics contiene ranuras de ventilación para mantener en un óptimo funcionamiento a los dispositivos albergados en el cuarto de telecomunicación.

Características

ˆ Tienen la capacidad de albergar y organizan cualquier tipo de equipo u organizador estándar de 19 .

ˆ Cuenta con una puerta delantera en cristal templado o ventilado con cerradura y un juego de llaves.

ˆ Tiene una puerta trasera ventilada con cerradura y un juego de llaves.

ˆ Paneles desmontables en la parte superior e inferior del gabinete para las acometidas del cable.

ˆ Tornillos niveladores para piso irregular y lograr una jación en el gabinete cuando se encuentre en el sitio adecuado.

ˆRuedas que permiten desplazar el gabinete dentro de los cuartos de comunicaciones.

Especificaciones	Sencillo
Material de fabricación	Acero SPCC en calibre de 2mm para el perfil del montaje, 1.5mm en ángulos y 1.2 los demás complementos
Certificación	ANSI/EIA RS-310-DIEC297-2DIN41494
Dimensión externa	Ancho 600mm Profundo 600mm Alto 2050mm
Dimensión interna	Ancho 19" Profundo 500mm Alto 42U
Iluminación	1 kit de iluminación tubo fluorescente color blanco, 120VCA 60Hz - (no incluye)
Ventilación	2 extractores de 120 VCA 60HZ (No incluye)
Puertas	• Puerta delantera: Cristal templado con cerradura y dos llaves, rango de apertura 180°
Paneles	• Paneles laterales: Desmontables con posibilidad de instalar cerradura • Paneles superior/inferior: Para la entrada de cableado
Barra de contactos	Barra de 19" con 3 contactos dobles para 120-240VCA 10A, longitud de cable 1.8m (no incluye)

2.3.9. Canaleta

Canaleta 22mm x 12mm x 29mm

Clave: CANAL22X12X29

Canaleta Auto adherible, 2.2cm x 1.2cm x 2.9m

Siguen las especi caciones de la sig. tabla:

CANTIDAD DE CABLES QUE ACEPTA UNA CANALETA						
ALTURA (mm)	DIMENSIONES A x B (mm) A = Ancho B = Alto	Comunicación	Coaxial	Fibra Óptica		
		UTP (5.5 mm)	RG 58 (4.8 mm)	RG 59 (6.3 mm)	Fibra Óptica (2.8 mm)	Fibra Óptica Multipar (8.3 mm)
7	13x7	1	1	1	1	
12	20x12	3	4	2	7	1
	32x12	5	6	3	11	2
	32x12 cd	4	5	3	10	2
16	60x16	12	13	8	28	4
	60x16 cd	10	11	8	26	4
20	20x20	6	7	4	12	2
	75x20 cd	19	20	13	40	6
25	25x25	8	9	5	18	3
	40x25	13	14	8	29	4
	40x25 cd	12	13	8	27	4
40	40x40	20	21	13	46	7
	60x40	30	31	20	70	10
	60x40 cd	28	29	20	68	10
	100x40	50	51	32	116	17

2.3.10. Conectores RJ-45

Plugs Modulares RJ45, Cat5e, Paquete con 100

UTP, 3 puntas, para cable sólido, 100 plugs en bote

Item: 502399

Features:

Plug modular RJ45 Cat5e

- Contactos con chapa de oro de 15 µ
- Terminal de 3 puntas para cable sólido
- Ajuste para cable redondo
- Para aplicaciones de par trenzado sin blindaje

ˆ Diseño estándar 8P8C, compatible con todos los conectores de red RJ45

ˆ Completamente compatible con las especi caciones de las Categorías 3, 4, 5 y 5e

ˆ Garantía de por vida

Especi caciones

Estándares

ˆ Categoría 5 (ANSI/TIA/EIA-568-A)

ˆ Categoría 5e (TIA/EIA-568-B)

Material

ˆ Aislante: policarbonato catalogado UL 94V-2

ˆ Contactos: fósforo bronce

Baño de oro

ˆ 15 micro pulgadas

Contenido del paquete

ˆ 100 Plugs Modulares RJ45 Cat5e en bote

Cable UTP 5E

La Bobina de Cable Cat5e INTELLINET NETWORK SOLUTIONS la infraestructura que proporciona altas velocidades de transmisión a las aplicaciones de datos, voz y vídeo - hasta 350 MHz y más.

El Cable Cat5e cumple con los estándares EIA/TIA y se encuentra clasi cado como Retardante de Flama.

Características

ˆ Clasi cación de Retardante de Flama: CM en muros; permitido en aplicaciones residenciales riser de la familia 1-2 , o en montajes con canaleta o escalerilla a prueba de fuego; no permitido en plenums

ˆ Probado bajo estándares: Cumple o excede los estándares actuales de Cat5e

ˆ Bitácora de uso en la Ca ja: Registra el cable restante disponible para la siguiente instalación

Garantía de por Vida

Los estrictos estándares de fabricación aseguran la máxima calidad en todos los productos INTELLINET NETWORK SOLUTIONS. Todos los artículos cuentan con una Garantía de por Vida, el compromiso de calidad más fuerte que alguien puede ofrecer.

Features

Par Trenzado Sin Blindar

ˆ Puede ser usado en riser con canaleta o eje a prueba de fuego

ˆ Probado bajo estándares para cumplir con certi cación 3P

ˆ Probado bajo estándares, cumple o excede lo propuesto por la Cat5e

ˆ Bitácora de uso en la ca ja, informa sobre el cable restante para la siguiente instalación ˆ Garantía de por vida

Especi caciones

General

ˆ Color Blanco

- 4 Pares Trenzados Sin Blindar
- Calibre 24 AWG
- Núcleo sólido

ˆ Forro de PVC

ˆ Probado a 350 MHz para proporcionar un ancho de banda y desempeño mejorados

ˆ Cumple o excede con las especi caciones de Cat5e

ˆ Ideal para 10Base-T, 100Base-TX y 1000base-TX

ˆ Clasi cado como Retardador de ama: CM Para muro; permitido en aplicaciones residenciales de la familia 1-2, o en aplicaciones riser con canaleta/a prueba de fuego; no permitida en plenums

ˆ Aprobada por UL

ˆ Certi cado 3P

Especi caciones técnicas

ˆ Pruebas de canal CAT5e TIA, probado a 350 MHz a 80 metros:

- Pérdidas en el retorno (db) = 16.3

- Atenuación (db/100) = 45.0

- Near-End Crosstalk (db) = 30.1

- Power Sum-NEXT (db) = 28.1 - Equal Level-FEXT (db) = 12.9

- Power Sum-ELFEXT (db) = 9.9

2.3.11. Cable UTP 6E

La Bobina de Cable Cat6 INTELLINET NETWORK SOLUTIONS permite una mejor relación señal a ruido con pérdidas mínimas - lo

que se traduce en redes más veloces y con ables para las aplicaciones de hoy.

El Cable Cat6 cumple con los estándares EIA/TIA y se encuentra clasi cado como Retardante de Flama.

ˆ Clasi cación de Retardante de Flama: CM en muros; permitido en aplicaciones residenciales riser de la familia 1-2 , o en montajes con canaleta o escalerilla a prueba de fuego

ˆ Probado bajo estándares: Cumple o excede los estándares actuales de CAT6

ˆ Bitácora de uso en la caja

Garantía de por Vida ** (Directa con el fabricante)

Los estrictos estándares de fabricación aseguran la máxima calidad en todos los productos INTELLINET NETWORK SOLUTIONS. Todos los artículos cuentan con una Garantía de por Vida, el compromiso de calidad más fuerte que alguien puede ofrecer.

ˆ UTP, Par Trenzado Sin Blindaje

ˆ Puede ser usada en aplicaciones riser con canaleta o eje prueba de fuego

ˆ Probado bajo estándares, cumple o excede lo propuesto por los estándares de Cat6

ˆ Bitácora de uso en la caja, informa sobre el cable restante para la siguiente instalación ˆ Garantía de por vida

Especi caciones:

General

ˆ Color Gris

ˆ 4 Pares Trenzados Sin Blindar con separador de pares

ˆ Calibre 23 AWG

ˆ Núcleo sólido

ˆ Forro de PVC

- Probado a 250 MHz
- Cumple o excede con las especi caciones de Cat6
- Ideal para transporte de datos, audio y vídeo

ˆ Retardador de ama: CM

ˆ Para muro; permitido en aplicaciones residenciales de la familia 1-2, o en aplicaciones riser con canaleta/a prueba de fuego; no permitida en plenum

ˆ Aprobada por UL

Especi caciones técnicas

Pruebas de canal CAT6 TIA, probada a 250 MHz:

ˆ Pérdidas en el retorno (db) = 17.32

ˆ Atenuación (db/100) = 33.04

ˆ Near-End Crosstalk (db) = 38.3

ˆ Power Sum-NEXT (db) = 36.3

ˆ Equal Level-FEXT (db) = 19.8

ˆ Power Sum-ELFEXT (db) = 16.8

2.3.12. Tubos de PVC

Tubo pvc de 3/4 de pulgada

Tubo pvc 1 pulgada 62

Tubo pvc de 1 1/2 pulgada

Tubo pvc de 1 1/4 pulgada

Tubo pvc 2 pulgadas

Tubo pvc 7 pulgadas

Tubo galvanizado de 3 pulgadas

Todos los tubos son de tipo conduit además siguen las especi cacio-nes de la sig. tabla:

CANTIDAD DE CABLES QUE ACEPTA UNA TUBERÍA PVC						
DIÁMETRO INTERNO		DIÁMETRO EXTERNO DEL CABLE (mm)				
Tubería en (mm)	Tubería en (Pulgadas)	3.3	4.6	5.6	6.1	7.4
15.8	0.5	1	1	0	0	0
20.9	0.75	6	5	4	3	2
26.6	1	8	8	7	6	3
35.1	1.25	16	14	12	10	6
40.9	1.5	20	18	16	15	7
52.5	2	30	26	22	20	14
62.7	2.5	45	40	36	30	17
77.9	3	70	60	50	40	20

2.3.13. Ca jas para Switch

64

Gabinete Intellinet, Montaje en Pared, 19", 6U, 600x450 Modelo: 203906

ˆ Especí camente diseñado para aplicaciones de espacio reducido

ˆ Se monta directamente en los muros

ˆ Carga estática máxima de 60 kg

ˆ Canales de cableado y ori cios de entrada de aire en las partes superior y de fondo

ˆ Grado de protección: IP20

ˆ Rollo de acero en frío de alta calidad SPCC

ˆ Grosor de la lámina: 2.0 mm para los rieles de montaje y 1.2 mm para lo demás

ˆ Terminado de la super cie: desengrasante, decapado, con spray fosfórico, electricidad estática, terminado en polvo

ˆ De conformidad con los estándares ANSI/EIA RS-310-D, IEC297-2, DIN41491 (Parte 1, Parte 7), DIN41494

ˆ Se envía completamente ensamblado General

ˆ 19" ˆ 6 U ˆ Negro ˆ Acero

ˆ Montaje en pared 65

Accesorios

ˆ 30 juegos de tornillería

Dimensiones externas

- 351 (Altura) x 600 (Ancho) x 450 (Profundidad) mm (13.8 x 23.6 x 17.7 in.)

Otras características

- Dimensiones (Ancho x Profundidad x Altura) 600 x 450 x 351 mm

ˆ Montaje en rack6U

CaractéristicasTipoMontado en la pared

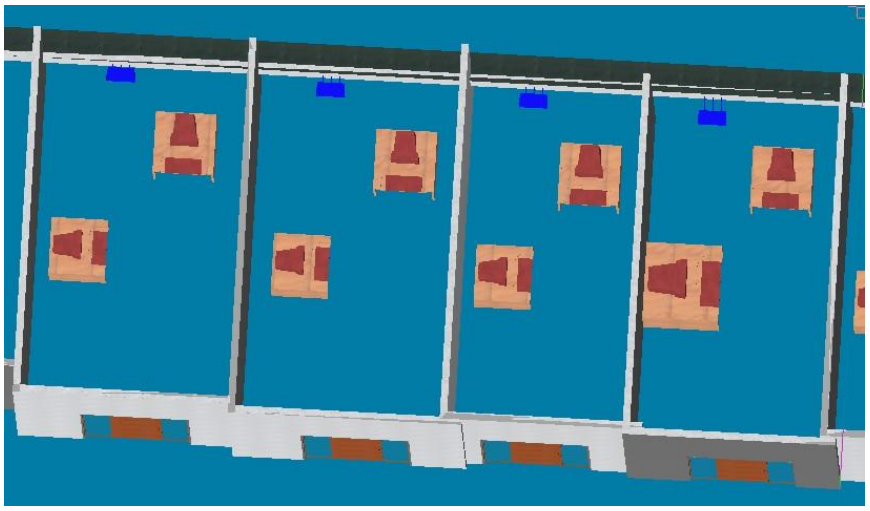
Certi cación ANSI/EIA, IEC, DIN Tamaño48, 26 cm (19\")

Color del producto Negro

Capítulo 3

Resultados

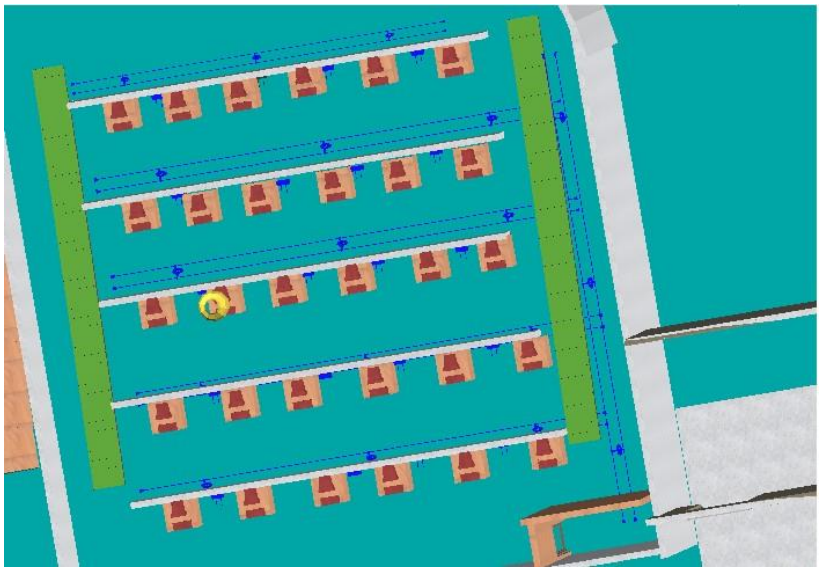
3.1. Cableado



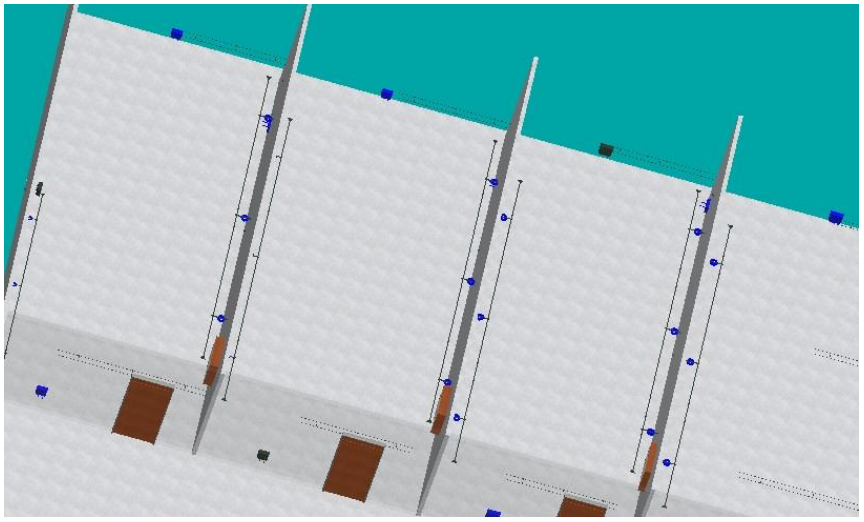
Modelado de los cubículos.

60

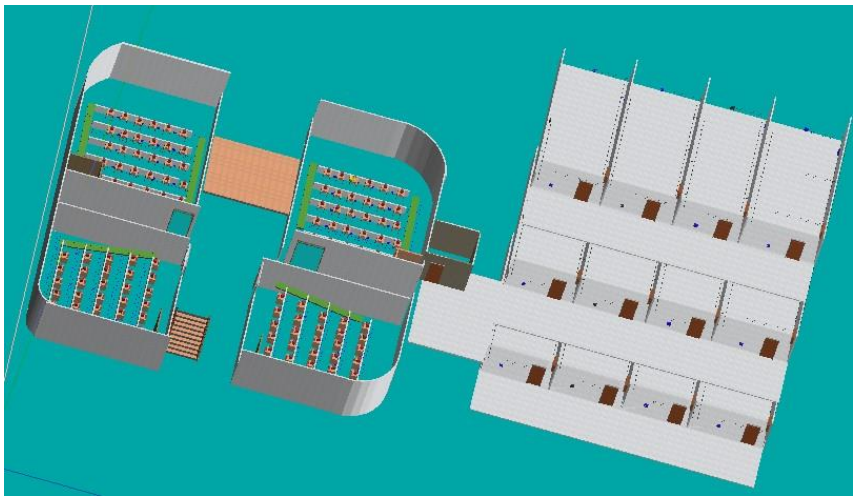
69



Modelado de un laboratorio.

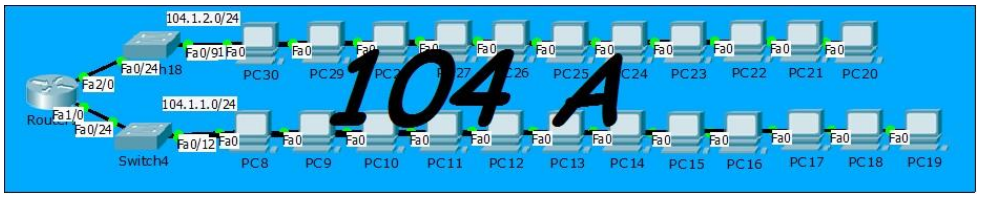


Modelado de los salones.

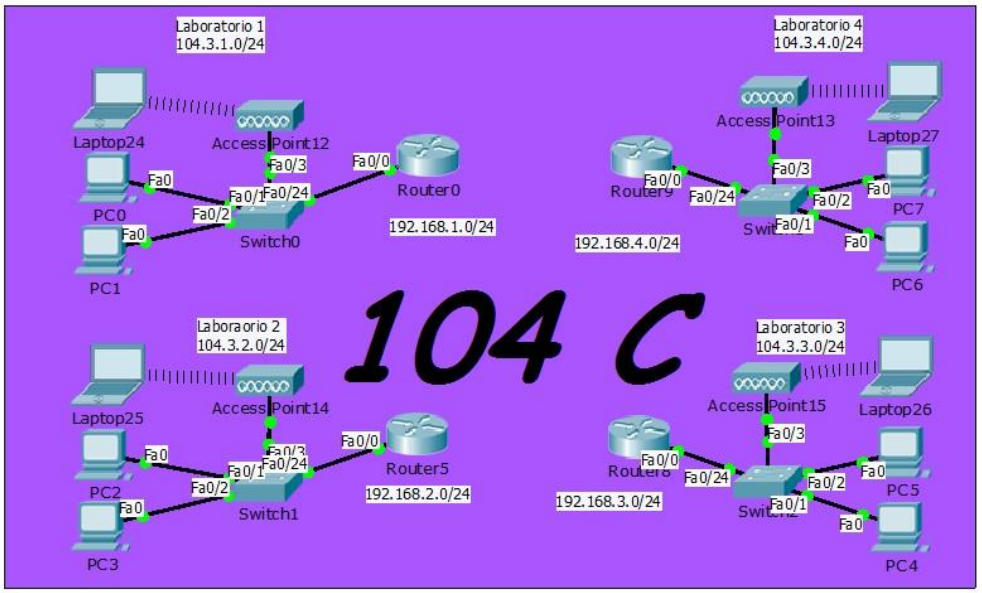


Modelado general.

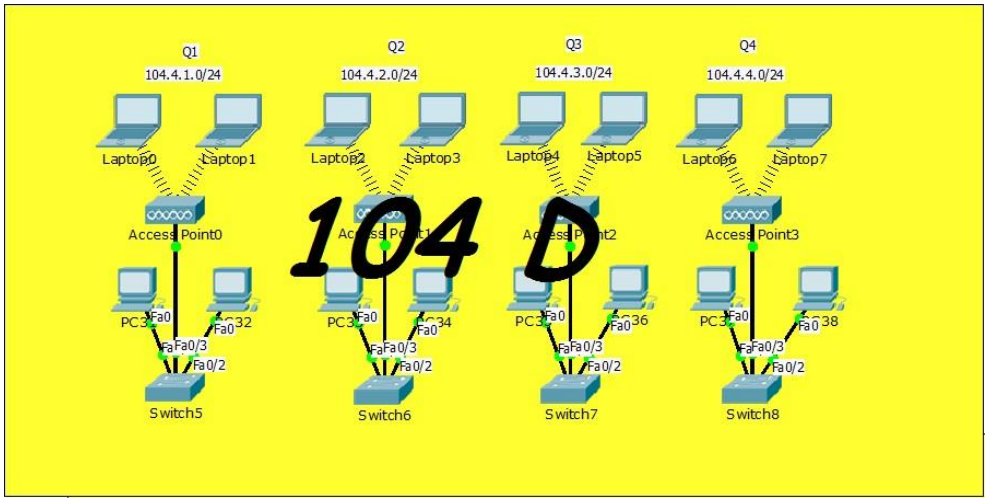
3.2. Redes



Redes de los cubículos.

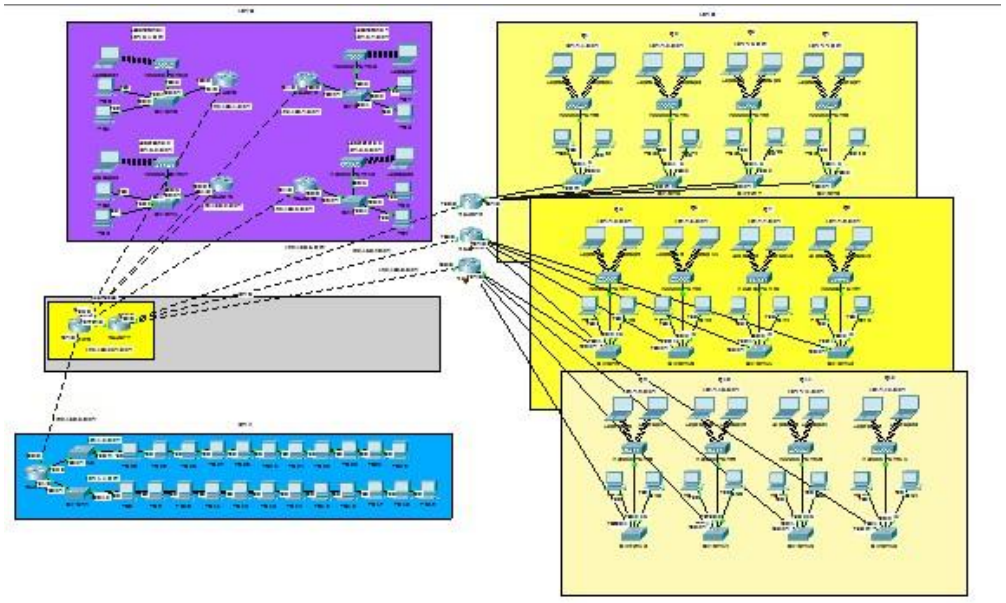


Redes de los laboratorios



Redes del primer nivel de salones

El modelado anterior se mantiene para los 8 salones restantes (2 niveles)



Modelado de todas las redes en conjunto

3.3. Material

3.3.1. Edi cio 104-A

- 1 rack

- 2 paneles de parcheo 48 puertos

- 2 Switch 48 puertos

- 1 Router 4 puertos

- 23 placas de 3 entradas

- 282 conectores RJ-45

- 1554 metros cable UTP-5E

- 100 metros tubo pvc de 1 1/4 de pulgada

3.3.2. Edi cio 104-B

Acometida

- 2 Routers de 12 puertos

- 2 transformadores de bra óptica a cable utp

- 2 racks - Ventilación

Salida para Acometida

- 60 metros de bra óptica

- 70 metros de tubo pvc de 3/4 de pulgada

- 70 metros de cable UTP 6E

3.3.3. Edi cio 104-C

Material por laboratorio:

- 1 Switch 48 puertos

- 1 Panel de Parcheo de 48 puertos

- 1 Router 4 puertos

- 1 rack

- 1 Access point

- 15 metros de canaleta

- 20 placas de 2 entradas

- 164 conectores RJ-45

- 820 metros cable UTP E5

- 8 metros tubo pvc 2 pulgadas

- 12 metros tubo pvc de 1 1/2 pulgadas

- 50 metros tubo pvc de 1 pulgada

Material por edi cio

- 4 Switch 48 puertos

- 4 Panel de parcheo 48 puertos

- 4 Routers 4 puertos

- 4 racks - 4 Access point

- 60 metros de canaleta

- 80 placas de 2 entradas

- 656 conectores RJ-45

- 3280 metros cable UTP 5E

- 32 metros tubo pvc de 2 pulgadas

- 48 metros de tubo pvc de 1 1/2 pulgadas

- 200 metros de tubo pvc de 1 pulgada

Salida para Acometida

- 323 metros cable UTP 6E

- 108 metros tubo pvc 1 pulgada

3.3.4. Edi cio 104-D

Material por salón:

- 1 Access Point

- 4 Placas para conector RJ45 de 2 entradas

- 1 Switch 12 puertos

- 1 panel de parcheo de 12 puertos

- 115 metros cable UTP 5E

- 34 conectores RJ-45

- 11 metros de canaleta

- 50 metros de tubo pvc (interior) de 3/4 de pulgada

Material por edi cio:

- 12 Switch 12 puertos

-12 ca jas para Switch

- 12 paneles de parcheo de 12 puertos

- 12 Access point

- 48 placas de 2 entradas

- 1380 metros cable UTP 5E

- 132 metros canaleta

- 408 conectores RJ-45

- 600 metros tubo pvc 3/4 de pulgada (interior)[70]

Exterior del edi cio:

- 156 metros tubo pvc 3/4 de pulgada (exterior)

- 6 metros tubo galvanizado de 3 pulgadas

- 473 (468+5puente) metros 6E

Cuarto de comunicación

- 1 rack

- 3 Router de 4 puertos

- Ventilación

3.4. Precios de dispositivos

Routers principales (12 puertos)

-2 Routers

- Cloud Core Router CCR1016-12G

Precio Unitario: $9,050.54

Total: $18 101.08

Routers (4 puertos)

-8 Routers Router

Cisco 4 puertos Ethernet 10/100

Modelo: RV042

Precio Unitario: $2,479.00

Total: $19,832.00

Transformadores

-2 Transformadores

Convertidor TP-Link RJ45 Gigabit/ bra óptica multimodo

TP-Link MC200CM

Precio Unitario: $965.59

Total: $1,931.18

Switch (48 puertos)

- 6 Switch 48 puertos

Cisco Gigabit Ethernet Switch Catalyst 4948, 48 Puertos, 96 Gbit/s, 32.768 Entradas Precio Unitario: $9,281.00

Total: $55,686.00

Switch (16 puertos)

-12 Switch 16 puertos

Switch Inteligente Gigabit Ethernet para Montaje en Rack de 16 puertos

Precio Unitario: $1,532.00

Total: $18,384.00

Panel de parcheo

- 6 paneles de parcheo 48 puertos

Intellinet Panel de Parcheo Cat5e, 2U, 48 Puertos 513579

Precio Unitario:$ 971.00 + 99.9 envio

Total: $6,425.4

Panel de parcheo

- 12 paneles de parcheo 12 puertos

Panel de parcheo de 12 puertos Categoría 5e, para montaje en panel o pared. Modelo: 350-312

Precio Unitario: $ 280.00

Total: $3,360

Access point

- 16 access point

Intellinet

Precio Unitario:$1028.00

Total: $16 448 + envio

Tapas de 2 entradas

- 128 tapas de 2 entradas

Steren

Mayoreo $5.80

Total: $742.4

Tapas de 3 entradas

- 23 conectores de 3 entradas

Steren

Mayoreo $5.80

Total: $133.4

Racks

72

- 8 racks

Rack vertical de acero

Sencillo 42UR con puerta ventilada

Precio Unitario:$5773.37

Total: $46186.96

Canaleta

Auto adherible 2.9m largo X 2.2 cm X 1.2 cm

Precio Unitario: $ 22.00

Total: 67 paquetes $1474 + envio

Conectores RJ-45

Plugs Modulares RJ45, Cat5e, Paquete con 100

Intellinet, paquete $268

Total: 14 paquetes $3752 + envio

Cable UTP 5E

Intellinet 305 metros

Precio Unitario: $1697

Total: 21 paquetes $35637 + envio

Cable UTP 6

Intellinet 305 metros

Precio Unitario: $1799

Total: 3 paquetes $5397 + envio

74

Ca jas para Switch

- 12 ca jas para Switch

Gabinete Intellinet, Montaje en Pared, 19'', 6U, 600x450

Modelo: 203906

Precio Unitario:$3,249.00

Total: $38,988

Tubo pvc 3/4 pulgada

3 metros $12.00

Total: 276 paquetes: $3312.00

Tubo pvc 1 pulgada

3 metros $22.00

Total: 103 paquetes: $2266

Tubo pvc 1 1/4 pulgada

3 metros $38.00

Total: 34 paquetes: $1292

Tubo pvc 1 1/2 pulgada

3 metros $48.00

Total: 16 paquetes: $768

Tubo pvc 2 pulgadas

3 metros $99.90

Total: 11 paquetes: $1 098.9

76

Tubo galvanizado 3 pulgadas

6 metros $307.12

Total: $307.12

3.5. Costo Total

$281 522.44

Capítulo 4

Concluciones

El simula el modelado de la red nos permitió determinar si la con - guración lógica es correcta o no, en su defecto, podemos resolver el problema de forma inmediata pues el uso de las herramientas facilita localizar el problema e incluso de forma grá ca.

La realización del enrutamiento de forma estática es muy tediosa sin embargo, nos permite llevar un mejor control del tránsito sobre la red.

73

79

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
Ingeniería en Ciencias de la Computación
Modelos de Redes

Examen del Parcial 1

Cesar Manuel Rodríguez Mendiola

26/09/2014

Antecedentes teóricos

Para entender un poco mejor el tema, primero de namos algunos términos:

Tasa de transferencia teórica: Máxima velocidad que se puede alcanzar considerando el ancho de anda del medio utilizado. Determinado por el fabricante

Latencia: Es la suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red.

Otros factores que in uyen en la latencia de una red son:

ˆEl tamaño de los paquetes transmitidos.

ˆ El tamaño de los bu ers dentro de los equipos de conectividad. Ellos pueden producir un Retardo Medio de Encolado.

Tiempo de Propagación: Tiempo empleado para enviar un bit desde un origen hasta un destino sobre un medio en especí co.

Tiempo de transmisión: Cantidad máxima de datos que pueden enviarse a través de una red sin tener que segmentarse en subpaquetes.

Tiempo de cola: Tiempo empleado para difundir un paquete a través de un dispositivo, además del trá co en la red.

Una vez teniendo en claro los puntos anteriores nos será fácil comprender la siguiente fórmula para determinar la latencia en nuestra red, con la cual podríamos seleccionar adecuadamente los dispositivos que la conformaran.

Entonces tenemos que:

Donde:

$$Latencia = TP + TT + TC$$

2

TP: Tiempo de Propagación $\approx \frac{distancia}{velocidadelaluz}$

TT: Tiempo de Transmición $\approx \frac{tamañodelpaquete}{tasadetransmicionteorica}$

TC: Tiempo de Cola

Desarrollo

El desarrollo de esta aplicación será en un entorno grá coco (netbeans), para facilitar la creación de la interfaz grá ca con la cual el usuario tendra que interactuar. Así mismo, este entorno resulta ser amigable para el manejo de cuadros de diálogos, mediante los cuales el usuario realizara la búsqueda del archivo correspondiente al grafo con el que se desea trabajar.

He decidido trabajar en netbeans por comodidad del lenguaje (java).

Trataremos de desarrollar esta aplicación de tal forma que se optimice el uso de memoria, parametrizada para realizar cambios si fuese necesario de manera rápida y sencilla.

Para optimizar el uso de memoria emplearemos listas ligadas para las adyacencias en lugar de las matrices de adyacencias; pues como ya sabemos al utilizar matrices se desperdicia bastante memoria cuando se tiene un gran número de vértices y pocas aristas. Con esto tendríamos un uso de memoria de k1(n+a) + 2k2a en vez de tener k2n2.

La parametrización la logramos al usar variables globales pero ojo, no todas deben ser globales y/o publicas pues esto también implicaría gastar recursos de memoria.

Por lo anterior representaremos las adyacencias como nodos con los siguientes atributos:

- int nombre;
- int distancia;
- double velocidad;

- int tamPaquete;
- int datControl;
- int datUsuario;

^ nodo ligaAdyacencia;

Para determinar todos los caminos diseñaremos un algoritmo hibrido basado en BPP (Busqueda Primero en Profundidad) y Dikstra. Veamos cómo queda el seudocódigo:

```
Todos los caminos( VertInicial){

    Marcar VertInicial como visitado;

    V = primer adyacencia de VertInicial;

    Para todos los caminos adyacentes a V hacer{

    Si V es igual a VerticeFinal entonces Guardar camino();

    Sino Si V no esta visitado entonces;

    Todos los caminos( V);

    }

    Dsmarcar V como visitado;

    }
```

Para calcular las latencias de cada camino tendríamos algo parecido al siguiente seudocódigo:

```
Latencia(Vi, tam){

    si v no es el n de camino entonces{
                                    5
    double PaquetesCompletos = 0;

    double Resto = 0;

    PaquetesCompletos = tam/datUsuario;
```

Resto = t-(PaquetesCompletos*(double)v.datUsuario);

Si el Resto > 0 entonces PaquetesCompletos++;

TiempoPropagacion = v.distancia/VelocidadLuz;

```
TiempoTransmicion = v.tamPaquete/Convert_Mb_to_B(v.velocidad);

TC=TiemposCola[v.nombre-1];

Si v.nombre != VerticeFinal entonces

(TiempoPropagacion + TiempoTransmicion + TC)*PaquetesCompletos;

Si Resto > 0 entonces{

PaquetesCompletos ;

Latencia(v.ligaAdyacencia,Resto,w);

}

Si PaquetesCompletos > 0entonces {

Latencia(v.ligaAdyacencia,v.datUsuario,w);

Latencias[w] += lat;

}

}
```

Para calcular el total del paquetes necesarios para enviar el archivo a través de la red solo se divide el tamaño de archivo entre el tamaño del paquete del siguiente nodo del mejor camino

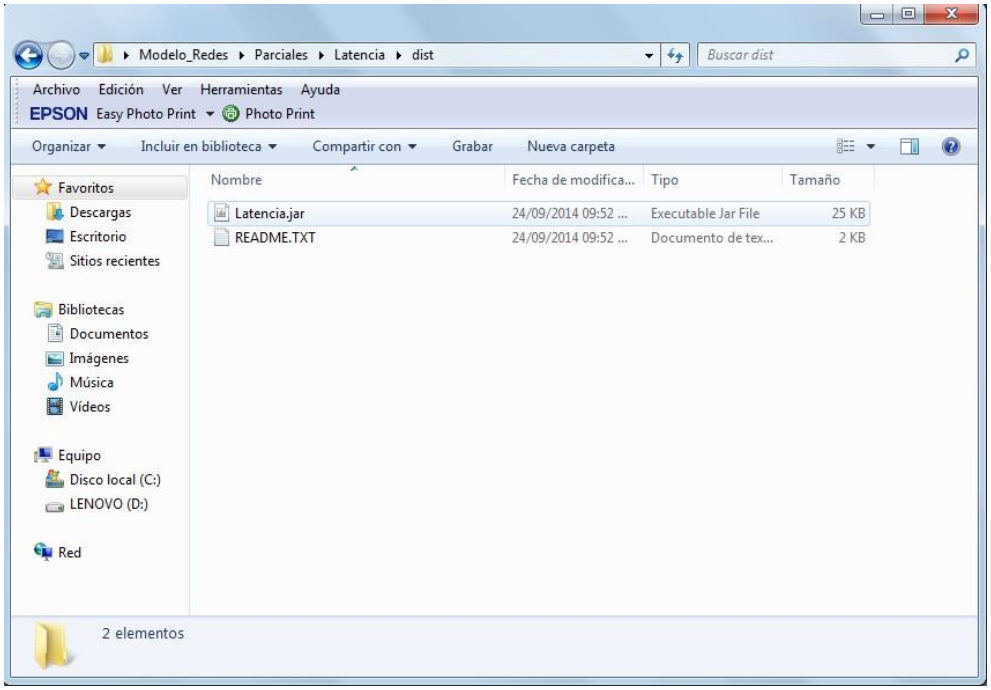
Finalmente el tiempo total de transmisión lo podemos determinar multiplicando el total de paquetes por el tiempo de latencia del mejor camino

Resultados

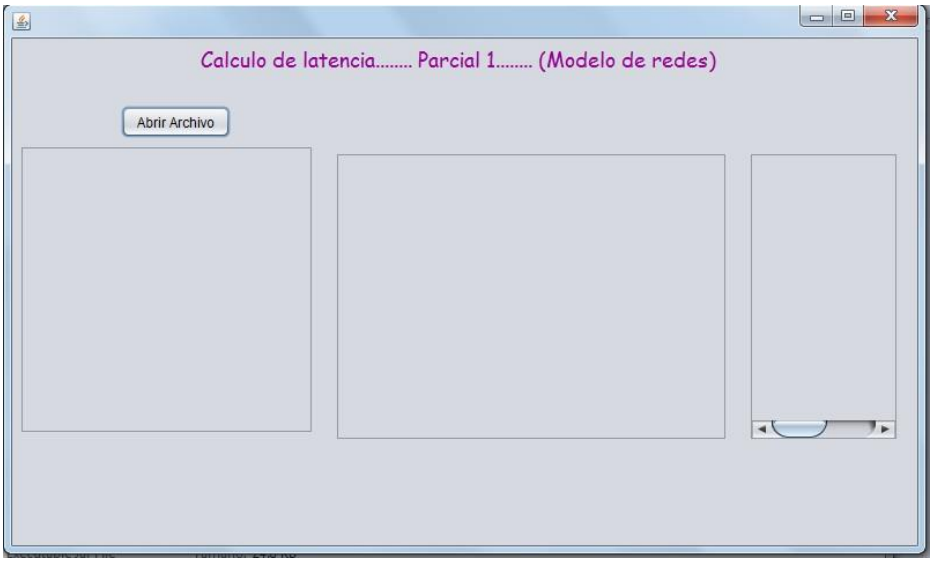
La aplicación ha sido probada por lo menos con 3 grafos diferente, apesar de que los grafos contienen ciclos ha sido posible detreminar todos los caminos posible satisfactoriamente.

Aqui se muestra la ejecución de la aplicación al igual que los resultados obtenidos en algunas pruebas.

Ejecutando la aplicción.

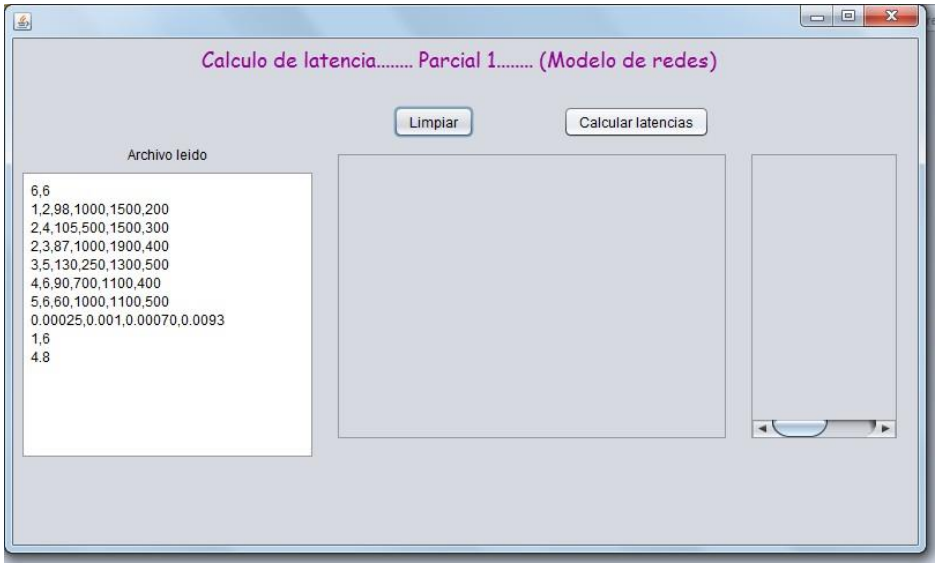


Ejecutamos la aplicaión.

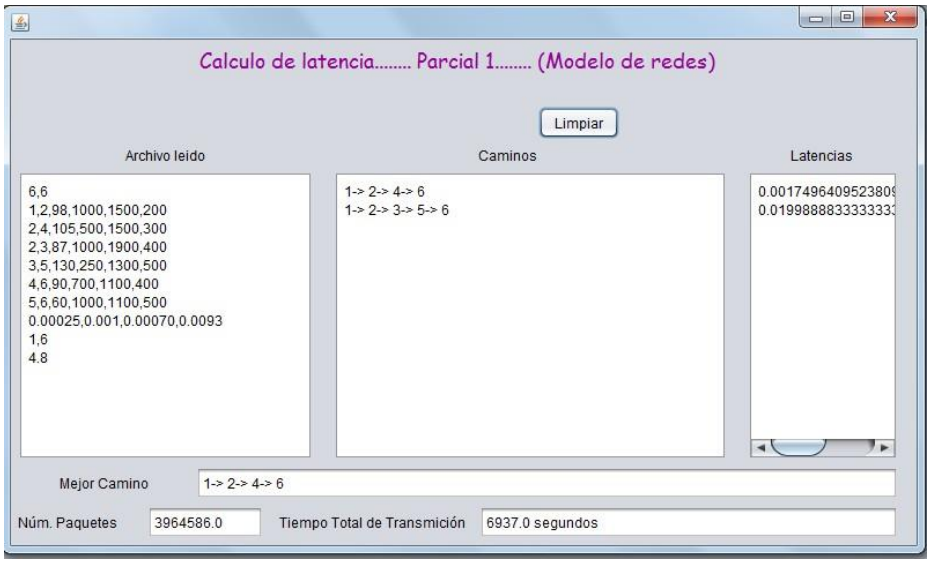


Ventana principal de nuestra aplicación en ejecución.

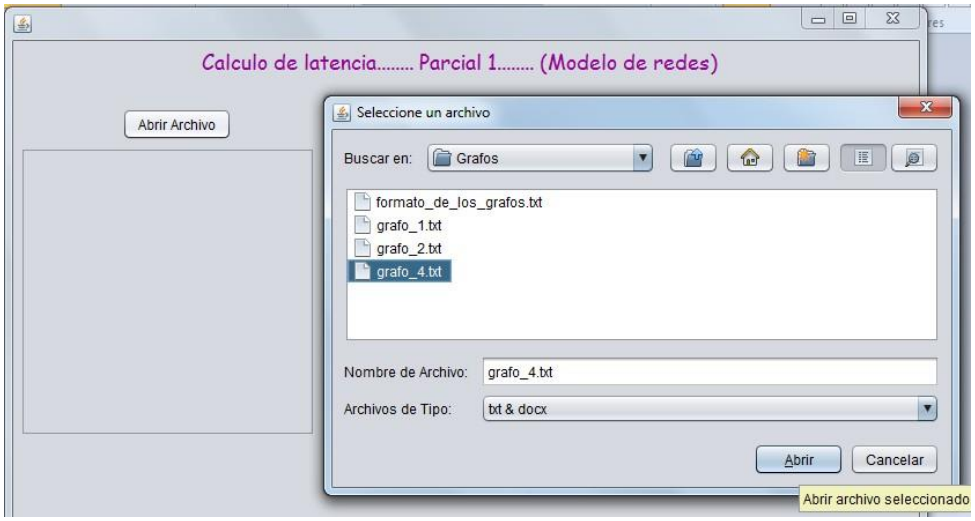
Seleccionamos el archivo correspondiente al grafo.



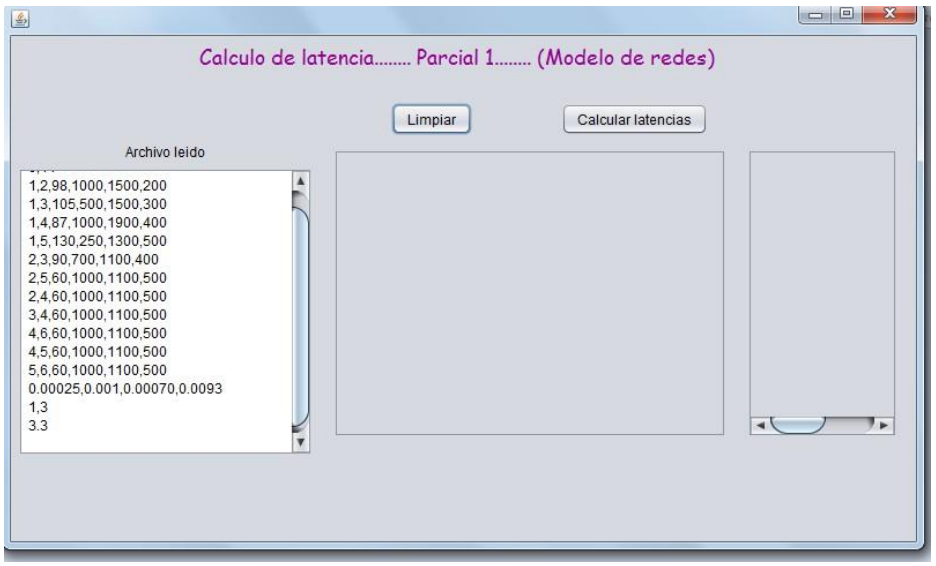
Se mustra el archivo leido y posteriormente cliquemos en el boton Calcular latencias .



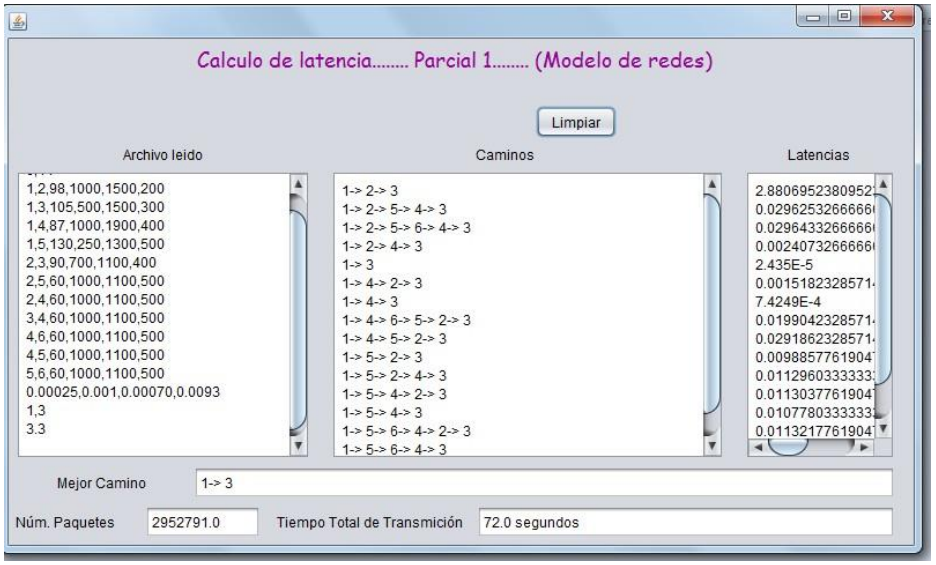
Se presentan todos los resultados.



Limpiamos pantalla y abrimos otro archivo.



Se mustra el archivo leido y posteriormente cliquemos en el boton Calcular latencias



Se presentan todos los resultados

Probamos la aplicación con otro archivo.

Como podemos observar la aplicación nos permite conocer todos los caminos posibles, así mismo la latencia correspondiente a cada uno de estos ultimos.

Tambien nos presenta el mejor camino, es decir, el de menor latencia, en otras palabras el camino más rapido.

Además calcula el numero total de paquetes requeridos para transmitir un archivo en la red. Con todos estos datos calcula y presenta el tiempo total de transmicion.

El cálculo de latencias es de suma importancia para poder determi- nar cuan efectiva es nuestro modelado de la red sin la necesidad de montarla físicamente, es decir, teóricamente podríamos saber cuánto tiempo le tomaría a la red aproximadamente en transferir datos desde un host, servidor, etc. hasta otro en determinada red.

Dichos cálculos son fundamentales para determinar la topología y características de los dispositivos a utilizar para que nuestra red sea de calidad y cumpla los requerimientos establecidos por el cliente.

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Modelos de Redes

Nombre: Héctor Ramírez Ruíz

Título: Primer examen parcial

Programa para calcular tiempo total de transmisión

12

Fecha de entrega: Viernes, 26 de Septiembre de 2014

Introducción

Como examen parcial se nos pidió realizar un programa en Java/C/C++ que realice los cálculos de latencia, número de paquetes y tiempo total de transmisión de una red de computadoras simulada.

El programa debe recibir como entrada únicamente un archivo de texto en el que vendrán especificados todos los datos que forman la red y los necesarios para poder hacer los cálculos.

Como salida debe mostrar todos los caminos posibles del elemento origen de la red al elemento origen (especificado en el archivo) y, además elegir el camino más apto determinado por su latencia. Además, teniendo ya el “mejor camino”, debe enlistarse el número de paquetes y el tiempo de transferencia total de éste.

Desarrollo del tema

La mejor manera de representar una red de computadoras y a la vez, poder hacer cálculos para saber por ejemplo, cuál es el mejor camino de un origen a un destino es llevando el concepto a un lenguaje de programación. A través de una computadora será mucho más fácil realizar esos cálculos que son bastante útiles para la materia. Así, en el primer examen parcial desarrollamos un programa que simulará una red de computadoras en Java, para mi caso.

Los datos necesarios se ingresan a través de un archivo de texto de la manera siguiente:

1. Número_de_vértices,número_de_arcos
2. Esto se hace de acuerdo al número de arcos:

   Vértice_inicial,vértice_final,distancia(mts.),velocidad_teórica(Mbps),tamaño_del_paquete(Bytes),datos_de_control(Bytes)
3. Esto va de acuerdo al número de vértices:

   Tiempo_de_cola1,Tiempo_de_cola2,Tiempo_de_cola3,…,

   Tiempo_de_colan
4. Vértice_origen,vértice_destino
5. Tamaño_del_archivo(Gigabytes)

Con estos datos, se puede calcular el tiempo de transmisión que se tomará el paquete en llegar del elemento de la red origen al destino.

Tenemos que obtener la latencia y, el número de paquetes previamente.

Para esto, utilizamos todos los datos que se nos dan. La red de computadoras la vemos como un grafo, obviamente, a un grafo ya se le puede manipular mucho más fácilmente.

Lo primero es leer el archivo, se hace línea por línea.

Lo que hago es hacer una lista de adyacencias, leo cada vértice y lo ligo hacia abajo. Después, ya en la segunda parte del archivo leo cada nodo o genero una lista ligada en el vértice correspondiente. El nodo ya tendrá como atributos los datos necesarios.

Los demás datos (la parte 3, 4 y 5) del archivo los guardó en variables para su futuro uso.

Después, deben obtenerse todos los caminos posibles del vértice origen al vértice destino que viene especificado. Para esto, yo intenté usar el recorrido a profundidad pero, sólo me generó un camino y, no pude modificarlo para hacerlo.

Intenté con Dijkstra pero, no fui capaz de modificarlo para encontrar y guardar todos los caminos.

Generaba siempre un solo camino, pude hacer que imprimiera los caminos del ejemplo en el que me basaba pero, obviamente no funcionará con otro archivo.

Después de obtener los caminos, se podría calcular los datos que se nos piden y, elegir en base a eso, el mejor camino.

Conclusiones

El uso de un lenguaje de programación para esta materia es muy útil, fue reflejado en este “proyecto”. Realizar cálculos como éstos a mano es muy costoso, muy propenso a fallar y, para modelos grandes una tarea muy tediosa. Es aquí, donde las computadoras nos sirven de mucho. En particular, este programa nos facilitará mucho las cosas.

Nota: Mi error fue en el backtracking del recorrido a profundidad, que no pude implementarlo a pesar, de no ser tan complicado. En el de Dijkstra a decir verdad no supe donde modificar. Debo admitir (sin excusarme) que mis bases en programación no son las mejores.

15


MD5

Tabla de contenido

1

Introducción

El problema a resolver en esta aplicación es el uso del encriptador nombrado MD5, para autentificar el acceso de los usuarios a un servidor, esto para volver seguro el envio de datos y no comprometer las contraseñas.

La criptografía es la técnica que protege documentos y datos. Funciona a través de la utilización de cifras o códigos para escribir algo secreto en documentos y datos confidenciales que circulan en redes locales o en internet. Su utilización es tan antigua como la escritura. Los romanos usaban códigos para ocultar sus proyectos de guerra de aquellos que no debían conocerlos, con el fin de que sólo las personas que conocían el significado de estos códigos descifren el mensaje oculto.

A partir de la evolución de las computadoras, la criptografía fue ampliamente divulgada, empleada y modificada, y se constituyó luego con algoritmos matemáticos. Además de mantener la seguridad del usuario, la criptografía preserva la integridad de la web, la autenticación del usuario así como también la del remitente, el destinatario y de la actualidad del mensaje o del acceso.

En criptografía, MD5 (abreviatura de Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje 5) es un algoritmo de reducción criptográfico de 128 bits ampliamente usado.

MD5 es uno de los algoritmos de reducción criptográficos diseñados por el profesor Ronald Rivest del MIT (Massachusetts Institute of Technology, Instituto Tecnológico de Massachusetts). Fue desarrollado en 1991 como reemplazo del algoritmo MD4 después de que Hans Dobbertin descubriese su debilidad.

A pesar de su amplia difusión actual, la sucesión de problemas de seguridad detectados desde que, en 1996, Hans Dobbertin anunciase una colisión de hash plantea una serie de dudas acerca de su uso futuro. Codificación .

La codificación del MD5 de 128 bits es representada típicamente como un número de 32 dígitos hexadecimal. El siguiente código de 28 bytes ASCII será tratado con MD5 y veremos su correspondiente hash de salida:

- MD5("Esto sí es una prueba de MD5") = e99008846853ff3b725c27315e469fbc

- Un simple cambio en el mensaje nos da un cambio total en la codificación hash, en este caso cambiamos dos letras, el «sí» por un «no».
    - MD5("Esto no es una prueba de MD5") = dd21d99a468f3bb52a136ef5beef5034
- Otro ejemplo sería la codificación de un campo vacío:
    - MD5("") = d41d8cd98f00b204e9800998ecf8427e

Desarrollo del problema

IMPLEMENTACION DEL ALGORITMO “MD5”

Descripción del algoritmo

- Se manda el mensaje en el buffer a convertir, se guarda un conteo del tamaño del mensaje en un contador de bytes del buffer. Se comprueba si ya se ha cumplido el tamaño.
- El método “concatenación final” su objetivo es, obtener la longitud del mensaje en bits, se hacen corrimientos hacia la derecha para obtener su valor en bytes, y enmascarar para obtener los bits menos significativos.
- Se hace el cálculo para obtener el índice en el buffer para colocar las últimas palabras y si es una cadena no ha superado la longitud de 512 bits inserta desde una posición que es múltiplo de 512 bits menos 64 bits.

  //Creación del último bloque que depende del tamaño

  Bits [8]

  Mascara (Obtención de los bits más significativos)
  Índice, longitudMensaje.

  Bits < - conteoFinal de bytes en el buffer * 8 (Obtención de bits)
  Bits [0] <- bits AND Mascara

  Bits [1] <- (bits >>8) AND Mascara
  Bits [2] <- (bits >>16) AND Mascara
  Bits [3] <- (bits >>24) AND Mascara
  Bits [4] <- (bits >>32) AND Mascara
  Bits [5] <- (bits >>40) AND Mascara
  Bits [6] <- (bits >>48) AND Mascara
  Bits [7] <- (bits >>56) AND Mascara

Se colocan las palabras en el buffer a convertir.

- Al inicializar los buffers se almacenan en “Little - endian” se cambiaran de la siguiente manera 5

A	0x67452301L	0x01234567L
B	0xEFCDAB89L	0x89ABCDEFL
C	0x98BADCEFL	0XFEDCBA98L
D	0X10325476L	0x76543210L

- Para crear los bloques mandamos el buffer final, y este de dividirá en el 16 bloques de 32 bits (4 bytes), se hace una operación de concatenamiento y corrimiento de bits para obtener de 4 en cuatro el valor de cada bloque.

  //crear bloque

  Recibimos Buffer []
  Bloque [16];
  Contador <-0;

  Para i <-1 hasta "tamaño del Buffer" i <- i + 4

  Hacer

  Bloque [contador++] <- Buffer[i] $+256^0$
  Buffer [i + 1] * 256^1+
  Buffer [i + 2] * 256^2 +
  Buffer [i + 2] * 256^3 ;

  Fin Para;

  Devolver bloque;

- Se hacen las cuatro rondas de operaciones, estas son 16 iteraciones, que las funciones constan de la siguiente estructura.

$$FF(a,b,c,d,M_j,s,t_i)\_denota\_a = b + ((a + F(b.c.d) + M_j + t_i) <<< s)$$
$$GG(a,b,c,d,M_j,s,t_i)\_denota\_a = b + ((a + G(b.c.d) + M_j + t_i) <<< s)$$
$$HH(a,b,c,d,M_j,s,t_i)\_denota\_a = b + ((a + H(b.c.d) + M_j + t_i) <<< s)$$
$$II(a,b,c,d,M_j,s,t_i)\_denota\_a = b + ((a + I(b.c.d) + M_j + t_i) <<< s)$$

  Donde:

  - Las rotaciones a la izquierda se hacen de la siguiente manera

    A = (A << Mj) | (A>>> (32L – Mj)

    Esto para poder hacer corrimientos y mantener la función de la rotación hacia la izquierda.

Todos estos se enmascaran con un valor 0xFFFFFFFL para descartar lo demás

- Los valores de *ti* s la parte entera de 2 ^32 *ABS (sin (i)), en donde i está en radianes. La cual se considera esta tabla de valores

```
A = F(A, B, C, D, x[ 0], S11, 0xd76aa478L); /* 1 */
D = F(D, A, B, C, x[ 1], S12, 0xe8c7b756L); /* 2 */
C = F(C, D, A, B, x[ 2], S13, 0x242070dbL); /* 3 */
B = F(B, C, D, A, x[ 3], S14, 0xc1bdceeeL); /* 4 */
A = F(A, B, C, D, x[ 4], S11, 0xf57c0fafL); /* 5 */
D = F(D, A, B, C, x[ 5], S12, 0x4787c62aL); /* 6 */
C = F(C, D, A, B, x[ 6], S13, 0xa8304613L); /* 7 */
B = F(B, C, D, A, x[ 7], S14, 0xfd469501L); /* 8 */
A = F(A, B, C, D, x[ 8], S11, 0x698098d8L); /* 9 */
D = F(D, A, B, C, x[ 9], S12, 0x8b44f7afL); /* 10 */
C = F(C, D, A, B, x[10], S13, 0xffff5bb1L); /* 11 */
B = F(B, C, D, A, x[11], S14, 0x895cd7beL); /* 12 */
A = F(A, B, C, D, x[12], S11, 0x6b901122L); /* 13 */
D = F(D, A, B, C, x[13], S12, 0xfd987193L); /* 14 */
C = F(C, D, A, B, x[14], S13, 0xa679438eL); /* 15 */
B = F(B, C, D, A, x[15], S14, 0x49b40821L); /* 16 */
```

Etapa1

```
A = G(A, B, C, D, x[ 1], S21, 0xf61e2562L); /* 17 */
D = G(D, A, B, C, x[ 6], S22, 0xc040b340L); /* 18 */
C = G(C, D, A, B, x[11], S23, 0x265e5a51L); /* 19 */
B = G(B, C, D, A, x[ 0], S24, 0xe9b6c7aaL); /* 20 */
A = G(A, B, C, D, x[ 5], S21, 0xd62f105dL); /* 21 */
D = G(D, A, B, C, x[10], S22, 0x2441453L); /* 22 */
C = G(C, D, A, B, x[15], S23, 0xd8a1e681L); /* 23 */
B = G(B, C, D, A, x[ 4], S24, 0xe7d3fbc8L); /* 24 */
A = G(A, B, C, D, x[ 9], S21, 0x21e1cde6L); /* 25 */
D = G(D, A, B, C, x[14], S22, 0xc33707d6L); /* 26 */
C = G(C, D, A, B, x[ 3], S23, 0xf4d50d87L); /* 27 */
B = G(B, C, D, A, x[ 8], S24, 0x455a14edL); /* 28 */
A = G(A, B, C, D, x[13], S21, 0xa9e3e905L); /* 29 */
D = G(D, A, B, C, x[ 2], S22, 0xfcefa3f8L); /* 30 */
C = G(C, D, A, B, x[ 7], S23, 0x676f02d9L); /* 31 */
B = G(B, C, D, A, x[12], S24, 0x8d2a4c8aL); /* 32 */
```

Etapa 2

```
A = H(A, B, C, D, x[ 5], S31, 0xfffa3942L); /* 33 */
D = H(D, A, B, C, x[ 8], S32, 0x8771f681L); /* 34 */
C = H(C, D, A, B, x[11], S33, 0x6d9d6122L); /* 35 */
B = H(B, C, D, A, x[14], S34, 0xfde5380cL); /* 36 */
A = H(A, B, C, D, x[ 1], S31, 0xa4beea44L); /* 37 */
D = H(D, A, B, C, x[ 4], S32, 0x4bdecfa9L); /* 38 */
C = H(C, D, A, B, x[ 7], S33, 0xf6bb4b60L); /* 39 */
B = H(B, C, D, A, x[10], S34, 0xbebfbc70L); /* 40 */
A = H(A, B, C, D, x[13], S31, 0x289b7ec6L); /* 41 */
D = H(D, A, B, C, x[ 0], S32, 0xeaa127faL); /* 42 */
C = H(C, D, A, B, x[ 3], S33, 0xd4ef3085L); /* 43 */
B = H(B, C, D, A, x[ 6], S34, 0x4881d05L); /* 44 */
A = H(A, B, C, D, x[ 9], S31, 0xd9d4d039L); /* 45 */
D = H(D, A, B, C, x[12], S32, 0xe6db99e5L); /* 46 */
C = H(C, D, A, B, x[15], S33, 0x1fa27cf8L); /* 47 */
B = H(B, C, D, A, x[ 2], S34, 0xc4ac5665L); /* 48 */
```

Etapa 3

```
A = I(A, B, C, D, x[ 0], S41, 0xf4292244L); /* 49 */
D = I(D, A, B, C, x[ 7], S42, 0x432aff97L); /* 50 */
C = I(C, D, A, B, x[14], S43, 0xab9423a7L); /* 51 */
B = I(B, C, D, A, x[ 5], S44, 0xfc93a039L); /* 52 */
A = I(A, B, C, D, x[12], S41, 0x655b59c3L); /* 53 */
D = I(D, A, B, C, x[ 3], S42, 0x8f0ccc92L); /* 54 */
C = I(C, D, A, B, x[10], S43, 0xffeff47dL); /* 55 */
B = I(B, C, D, A, x[ 1], S44, 0x85845dd1L); /* 56 */
A = I(A, B, C, D, x[ 8], S41, 0x6fa87e4fL); /* 57 */
D = I(D, A, B, C, x[15], S42, 0xfe2ce6e0L); /* 58 */
C = I(C, D, A, B, x[ 6], S43, 0xa3014314L); /* 59 */
B = I(B, C, D, A, x[13], S44, 0x4e0811a1L); /* 60 */
A = I(A, B, C, D, x[ 4], S41, 0xf7537e82L); /* 61 */
D = I(D, A, B, C, x[11], S42, 0xbd3af235L); /* 62 */
C = I(C, D, A, B, x[ 2], S43, 0x2ad7d2bbL); /* 63 */
B = I(B, C, D, A, x[ 9], S44, 0xeb86d391L); /* 64 */
```

Etapa 4

Al final de todos los ciclos, a, b, c y d son sumados a A, B, C y D respectivamente y el algoritmo continúa con el siguiente bloque de datos. La concatenación de A, B, C y D que produce un bloque de 128 bits.

Implementacion al programa

CLIENTE – SERVIDOR

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

CLIENTE

El cliente consta de una ventana con dos cuadros de texto, usuario y contraseña, dos opciones de comunicación con el servidor:

- Acceso
- Registro

Y por último un botón que lanza la acción para hacer la comunicación con el servidor.

Acceso cliente

En la opción del ingreso del sistema pedirá dos campos, los cuales son los siguientes

- Usuario
- Contraseña

La metodología de acceso es la siguiente:

1. Se manda la acción a realizar (“Login”).
2. Se manda el nombre de usuario.
3. Se espera la respuesta del servidor
    a. Si no es válido se manda un mensaje de que el acceso ha fallado.
    b. Si es valido
        i. Se espera la respuesta del servidor, la cual es el mensaje para generar la huella.
        ii. Se obtiene del campo contraseña su información, y concatenado con la huella se genera el “MD5”, la cual se manda al servidor para comprobación.
        iii. Se recibe la respuesta del servidor si el acceso ha sido exitoso o no.
4. Se manda el mensaje para finalizar la conexión.

Registro Cliente

En la opción del registro del sistema pedirá dos campos, los cuales son los siguientes

- Usuario
- Contraseña

La metodología de acceso es la siguiente:

1. Se manda la acción a realizar (“Registro”).
2. Se manda el nombre del usuario al servidor para comprobar su existencia
    a. Si existe usuario se recibe la respuesta
    b. Si no existe se obtiene lo que contiene el campo contraseña.
        i. Se sigue un pequeño algoritmo de encriptación y se manda al servidor.

ii. Se recibe respuesta del servidor

3. Se manda el mensaje de conexión finalizada.

SERVIDOR

El servidor solo es una aplicación de consola, que se muestra en la misma el estado de las conexiones, peticiones y huellas recibidas.

La metodología del servidor es la siguiente:

1. Recibir acción.
    a. Login

i. Recibe el usuario.

- Si existe Usuario en la base de datos. 13

a. Se obtiene la contraseña del usuario de la base de datos

b. Se obtiene un mensaje único que será la fecha en el que se intenta la conexión.

c. Se manda al cliente la fecha para que el cliente pueda generar su MD5.

d. Se genera el MD5 concatenando la contraseña con la fecha.

e. Se recibe el MD5 generado por el cliente.

f. Se comparan los MD5

- Si son iguales se manda una respuesta al cliente de acceso exitoso.
- Si no son iguales se manda respuesta al cliente de acceso fallido.

b. Registro

i. Recibe el usuario.

- Si no existe el usuario en la base de datos

a. Se manda la respuesta al cliente con el mensaje de que no existe el usuario

b. Se obtiene la contraseña que esta encriptado. Se desencripta con un simple algoritmo.

c. Se registra usuario y contraseña en la base de datos.

d. Se manda respuesta al cliente de que el registro ha sido exitoso.

- Si existe el usuario en la base de datos

a. Se manda la respuesta al cliente con el mensaje “El usuario existe”.

Capturas del programa

Ilustración 1 Cliente

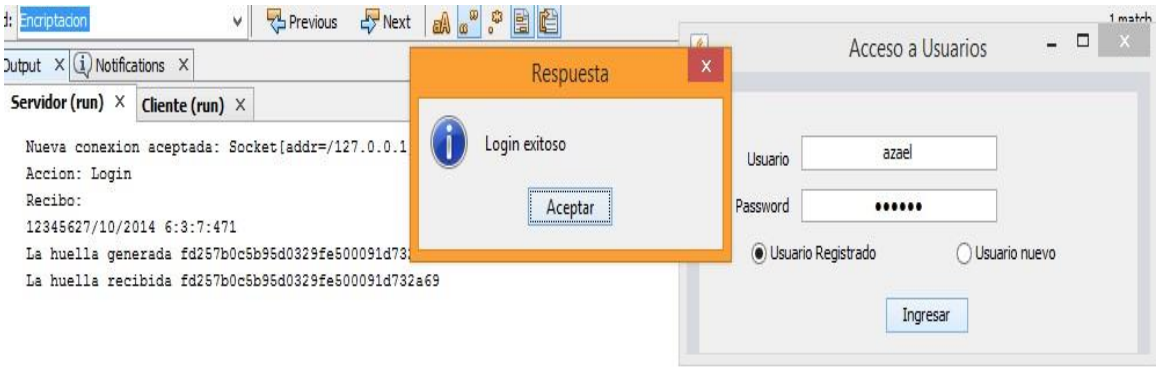


Ilustración 2 Acceso Exitoso

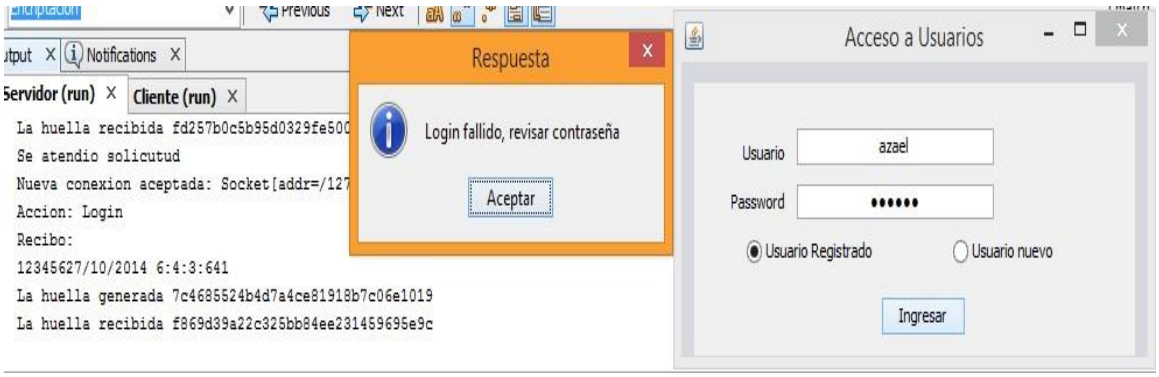


Ilustración 3 Acceso fallido

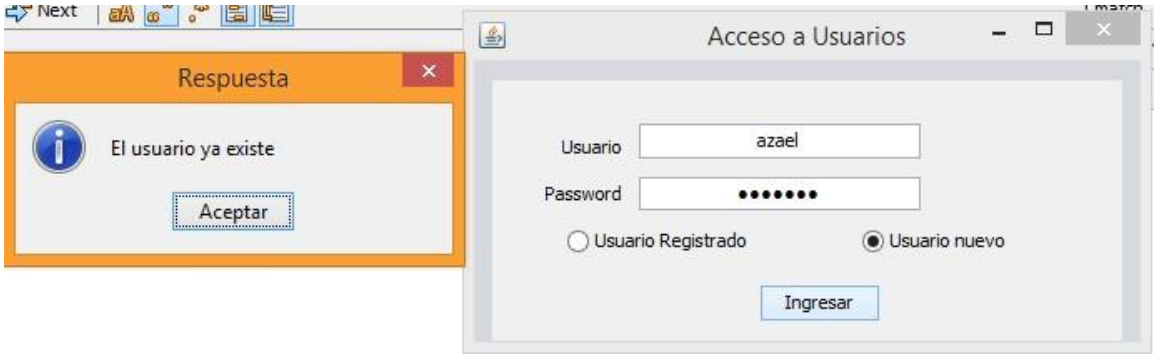


Ilustración 4 Registro de usuario fallido

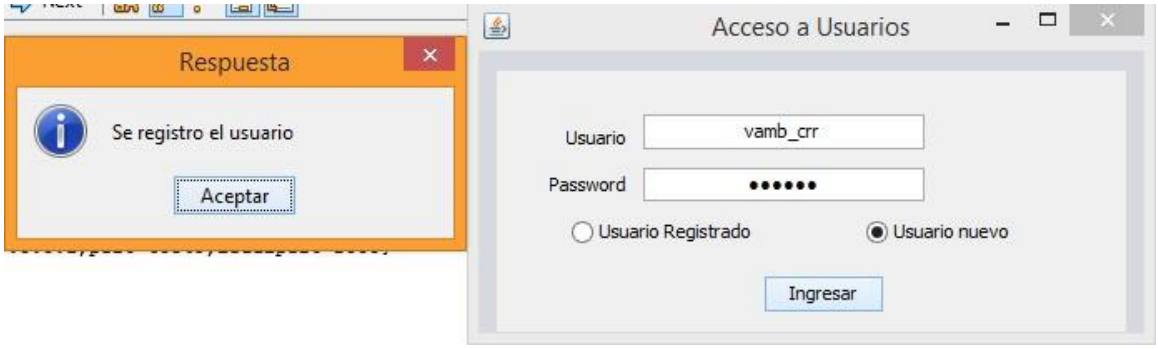


Ilustración 5 Registro de usuario exitoso

```
azael/123456
Pancho_Villa/pancho
liz_caca/castCam
lau_thal/123a56
vamb_crr/621960
```

Ilustración 6 Archivo donde se registran usuarios

Referencias

Paginas

- http://maytics.web44.net/web_documents/md5.pdf
- http://www.informatica-hoy.com.ar/seguridad-informatica/Criptografia.php
- http://www.cs.buap.mx/~iolmos/redes/9_Enlace_Datos.pdf
- http://redyseguridad.fi- p.unam.mx/proyectos/criptografia/criptografia/index.php/5-criptografia-

  asimetrica-o-de-clave-publica/55-funciones-hash/554-md5-message-digest-

  algorithm - Laboratorio de Redes y Seguridad. UNAM - Facultad de Ingeniería, División de Ingeniería Eléctrica.

Libros

- James Joshi – 2008 - Network Security: Know It All: Know It All

Reporte Tecnico

Sistema de validación de acceso a usuarios basado en MD5 para servicio de Chat

Ingeniería en Cs. De la Computación

Modelos de Redes

27/11/14

Integrantes:

Christopher Jesús Ruiz Escápita
201136965

Brianda Yareli Cruz Guerrero
201114701

Luis Conde Rodríguez 201124489

Alejandro Jiménez Ortega
201104393

Diana Barrientos Rodríguez
201103246

INTRODUCCIÓN

La seguridad para la comunicación en las redes es un factor importante hoy en día ya que a través de ella se realizan procesos en donde se involucra información importante como es envió de cuentas de banco, datos personales, entre otras cosas, así que para eso se han implementado algoritmos par el envió seguro de información de un lugar a otro.

Hoy en la actualidad existen muchos algoritmos para encriptar información pero muchos de hechos se pueden desencriptar, pero se ha creado un algoritmo el cual es desencriptable, y esto hace que sea uno de los mejores algoritmos que hay para el envió de información. Y nosotros lo aplicaremos en un chat para mostrar su funcionamiento.

ALGORITMOS

MD5

En criptografía, MD5 (Algoritmo de Resumen del Mensaje 5) es un algoritmo de reducción criptográfico de 128 bits ampliamente usado. El código MD5 fue diseñado por Ronald Rivest en 1991.

El algoritmo MD5 es una función de cifrado tipo hash que acepta una cadena de texto como entrada, y devuelve un número de 128 bits. Las ventajas de este tipo de algoritmos son la imposibilidad (computacional) de reconstruir la cadena original a partir del resultado, y también la imposibilidad de encontrar dos cadenas de texto que generen el mismo resultado.

Esto nos permite usar el algoritmo para transmitir contraseñas a través de un medio inseguro. Simplemente se cifra la contraseña, y se envía de forma cifrada. En el punto de destino, para comprobar si el password es correcto, se cifra de la misma manera y se comparan las formas cifradas.

CODIFICACION Y ALGORITMO DE MD5

La codificación del MD5 funciona d ela siguiente manera; su codificación de 128 bits es representada típicamente como un número de 32 dígitos hexadecimal.

Un ejemplo a continuación de una codificación de MD5:

En MD5 ("esto si es una prueba de MD5")

Ahora vemos su Hash de salida= e99008846853ff3b725c27315e469fbc

Pero veamos lo complejo que puede llegar a ser, con tan solo cambiar una letra del mensaje original arroja un Hash de salida muy diferente;

En MD5 ("esto no es una prueba de MD5")

Ahora vemos su Hash de salida= dd21d99a468f3bb52a136ef5beef5034

El simple hecho de codificar un espacio vacío tambien os da como resultado un Hash de salida complejo al igual que los antes vistos;

En MD5 (" ")

Ahora vemos su Hash de salida= d41d8cd98f00b204e9800998ecf8427e

Si en nuestra página web o sistema de envio de datos en red tenemos un sistema de usuarios y queremos proteger las contraseñas para prevenir posibles vulnerabilidddes en nuestro servidor, es una medida eficaz encriptar las contraseñas, como se hace con el MD5, de manera que si alguien puede acceder a ellas no pueda ver la contraseña.

APLICACIONES

Los resúmenes MD5 se utilizan extensamente en el mundo del software para proporcionar la seguridad de que un archivo descargado de Internet no se ha alterado.

Un usuario puede tener la confianza suficiente de que el archivo es igual que el publicado por los desarrolladores comparando una suma MD5 publicada con la suma de comprobación del archivo descargado. La comprobación de un archivo descargado contra su suma MD5 no detecta solamente los archivos alterados de una manera maliciosa, también reconoce una descarga corrupta o incompleta. Esto protege al usuario contra los 'Caballos de Troya' o 'Troyanos' y virus que algún otro usuario malicioso pudiera incluir en el software.

Para comprobar la integridad de un archivo descargado de Internet se puede utilizar una herramienta MD5 para comparar la suma MD5 de dicho archivo con un archivo MD5SUM con el resumen MD5 del primer archivo. 21

El MD5 también se puede usar para comprobar que los correos electrónicos no han sido alterados usando claves públicas y privadas.

ALGORITMO MD5

Se comienza suponiendo que se tiene un mensaje de b bits de longitud, escritos m0, m1,...m(b-1).

El algoritmo tiene cinco pasos.

1. Adición de bits de relleno.

El mensaje es rellenado con n bits, de tal manera que le falte a su longitud 64 bits para ser múltiplo de 512. El primer de los nbits es 1 y el resto son 0.

2. Adición de la longitud.

La nueva longitud es una representación de 64 bits y es añadida en forma de dos palabras de 32 bits, en primer lugar se muestran los bits menos significativos. Si la longitud del mensaje es mayor que 264, se usan los 64 bits menos significativos.

3. Inicializar lcuatro bufferes, A,B,C y D, que son registros de 32 bits.

Inicializados con los sigs. valores

A: 01 23 45 67

B: 89 ab cd ef

C: fe dc ba 98

D: 76 54 32 10

4. Procesar el mensaje en bloques de 16 bits (se tendra una entrada y salida de 32 bits).

F(X,Y,Z) = (X AND Y) OR ((NOT(X)) AND Z)

G(X,Y,Z) = (X AND Z) OR (Y AND (NOT(Z))

H(X,Y,Z) = X XOR Y XOR Z

I(X,Y,Z) = Y XOR (X OR (NOT(Z)))

Se usa una tabla de 64 elementos T[1 ... 64] construida con la función seno, siendo Ti la parte entera de 294967296 * abs(sen(i)) (i en radianes).

5. Salida.

Mensaje producido por A, B, C, D, empezando con los bits menos significativos de A y terminando con los más significativos de D. Independientemente de la longitud del mensaje, su tamaño será de 128 bits.

En la web se encontró el siguiente el algoritmo

```
public static String cryptMD5(String textoPlano)
{
    try
    {
       private static final char[] HEXADECIMALES = {
'0','1','2','3','4','5','6','7','8','9','a','b','c','d','e','f' };

       MessageDigest msgdgt = MessageDigest.getInstance("MD5");
       byte[] bytes = msgdgt.digest(textoPlano.getBytes());
       StringBuilder strCryptMD5 = new StringBuilder(2 * bytes.length);
       for (int i = 0; i < bytes.length; i++)
       {
           int low = (int)(bytes[i] & 0x0f);
           int high = (int)((bytes[i] & 0xf0) >> 4);
           strCryptMD5.append(HEXADECIMALES[high]);
           strCryptMD5.append(HEXADECIMALES[low]);
       }
       return strCryptMD5.toString();
    } catch (NoSuchAlgorithmException e) {
       return null;
    }
}
```

El algoritmo anterior utiliza las siguientes librerías

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
```


Algoritmo simple para encriptado

Para hacer la encriptacion de los datos que se envian en de un cliente a otro se aplicó un algoritmo no tan complicado, pero ese algoritmo no contiene un nombre, pero su función es que toma el mensaje a encriptar y lo pasa en un arreglo de caracteres y después de hacer esto a cada carácter le aplicar una suma de 5 para obtener otro carácter pero que este en un rango valido de caracteres. Para desencriptar solo se aplica el proceso inverso al anterior. Y es el siguiente:

```
public String encripta(String msj){
       char array[]=msj.toCharArray();

            for(int i=0;i<array.length;i++){
                array[i]=(char)(array[i]+(char)5);
            }
          String encriptado =String.valueOf(array);
        //System.out.println("Texto encriptado:");
        //System.out.println(encriptado+"\n");
        return encriptado;
   }

   public String desencripta(String msj){
       char arrayD[]=msj.toCharArray();

            for(int i=0;i<arrayD.length;i++){
                arrayD[i]=(char)(arrayD[i]-(char)5);
            }
          String desencriptado =String.valueOf(arrayD);
        //System.out.println("Texto desencriptado");
        //System.out.println(desencriptado);
        return desencriptado;}
```

FUNCIONAMIENTO

El funcionamiento consta básicamente de un servidor, el cual se activa para aceptar cliente, estos le envían de manera encriptada el Nickname del cliente, cuando el servidor la recibe verifica que el Nickname se encuentre en la base de datos y después de hacer esto hay 3 casos: 24

Caso 1.- Si el usuario existe, el servidor le permitirá al cliente poder iniciar y de esta manera puede empezar a realizar conversaciones con los demás clientes.

Caso 2.-Si el usuario existe pero su password es erróneo, el cliente enviar na notificación al cliente que su password no es correcto y esto provocara que se cierre el chat del cliente.

Caso 3.- Si el usuario no existe, si esto ocurre el servidor mandara un mensaje al cliente para notificarle que no se encuentra registrado, y este introduce sus datos y los manda al cliente para que este los coloque en la base de datos.

Ahora el funcionamiento entre el servidor y el cliente es el siguiente:

El cliente le manda su Nick al servidor este lo recibe y genera un mensaje Aleatorio y se lo manda al cliente, inmediatamente el servidor concatena el password y el mensaje aleatorio y aplicara el algoritmo de MD5, ahora en el cliente después de recibir el mensaje aleatorio de igual manera concatena el password y el mensaje aleatorio que recibió del servidor y aplica MD5 y la cadena obtenida de este proceso la envía al servidor.

Ahora, el servidor se encarga de validar la cadena que recibe del cliente, si es igual a la que genero el servido este permitirá el acceso al chat, si el password es erróneo notificara al cliente y si no existe el Nick lo pedirá que se registre.

CONCLUSION

En este proyecto pudimos verificar el funcionamiento para el envio del datos seguros, profundizamos muy a fondo la aplicación del algoritmo MD5. Asi que se considera un proyecto efectivo y con mucha enseñanza.

Referencias

http://donnierock.com/2012/03/29/encriptando-en-md5-con-java/

26

http://seguridadredesmedina.blogspot.mx/2009/10/md5-definicion-y-aplicaciones.html

http://pastebin.com/sM2dKDHW

FACULTAD de CIENCIAS de la COMPUTACIÓN

Benemérita Universidad
Autónoma de Puebla

Equipo:

López López Daniel

Sánchez Ramírez Carlos Alberto

Modelos de Redes

Dr. : Ivan Olmos Pineda

Examen 1: Manual Técnico

Fecha: 26-09-14

Manual técnico

Para resolver nuestro problema lo primero que se tuvo que realizar es el leer el archivo donde se almacenan los datos de dicho problema, la herramienta utilizada fue de la biblioteca de java *StringTokenizer* que compara si se encuentran comas en el archivo, conforme a la lectura se fueron guardando los datos en variables y en un arreglo.

Para poder manipular mejor los datos de cada arco se utilizo listas ligadas, en el cual el nodo contiene los siguientes datos:

```
int inicio;
int fin;
float distancia;
float velocidad;
float tampaquete;
float dc;
float du;
```

El siguiente problema a resolver fue el de hallar todos los caminos posibles del nodo inicial al nodo final, para esto se utilizo la clase *Graph.java*, la cual va guardando el listas ligadas las adyacencias de los nodos. El algoritmo que se implementa es BFS (Breadth First Search) Búsqueda en Anchura, función *breadthFirst()* en la clase *Ventana1.java*, el cual primero examina los nodos adyacentes, después el algoritmo trabaja recursivamente: coloca como visitado al último nodo y hace un llamado de sí misma con el nodo visitado, por final elimina al último nodo de los visitados, el caso base es que el nodo visitado sea el mismo de la llamada a la función o que el nodo sea el final.

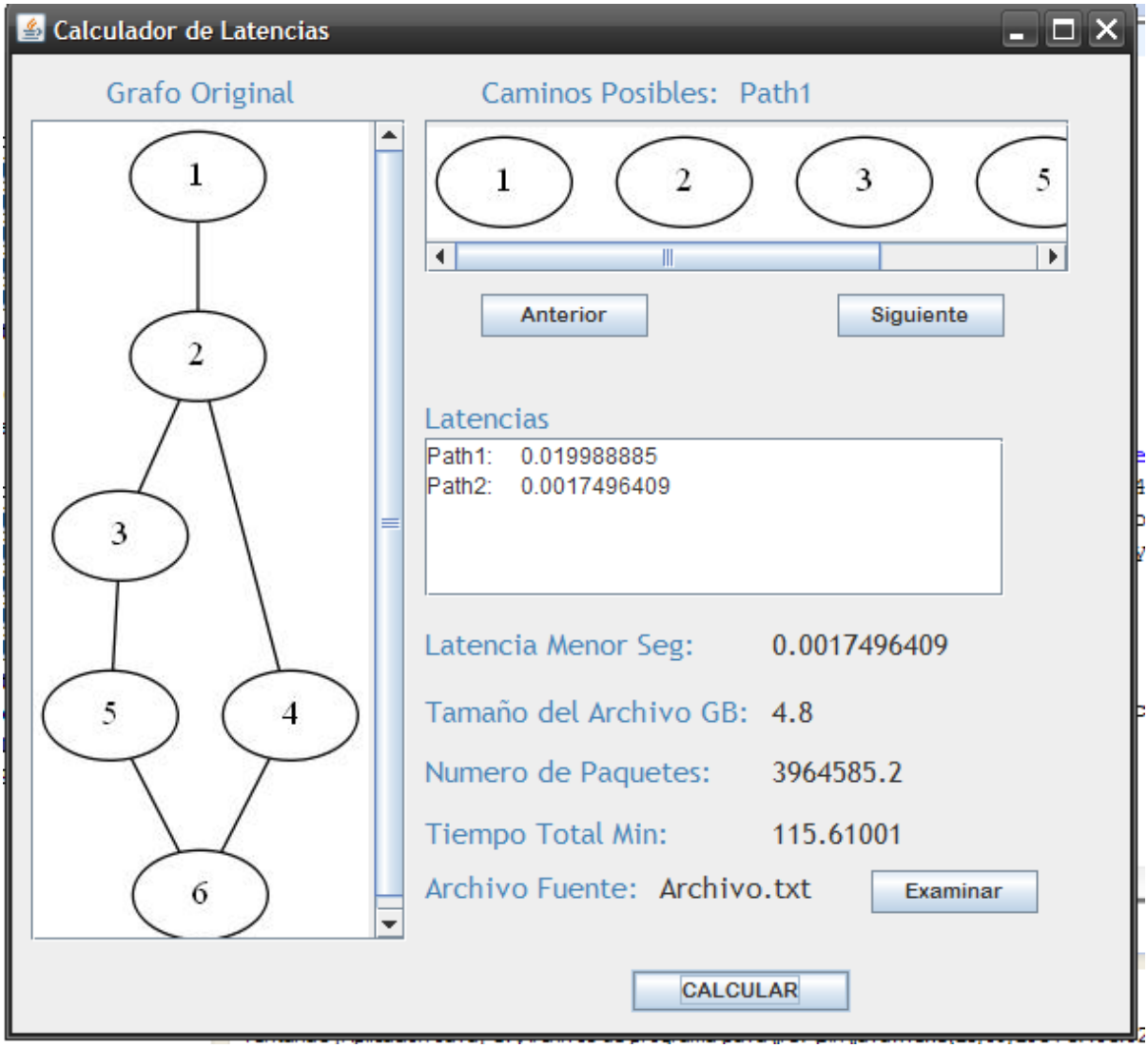
Para manipular todos los caminos posibles se almacenaron en un archivo de texto llamado *paths.txt* con la función *printPath()* en la clase *Ventana1.java.*

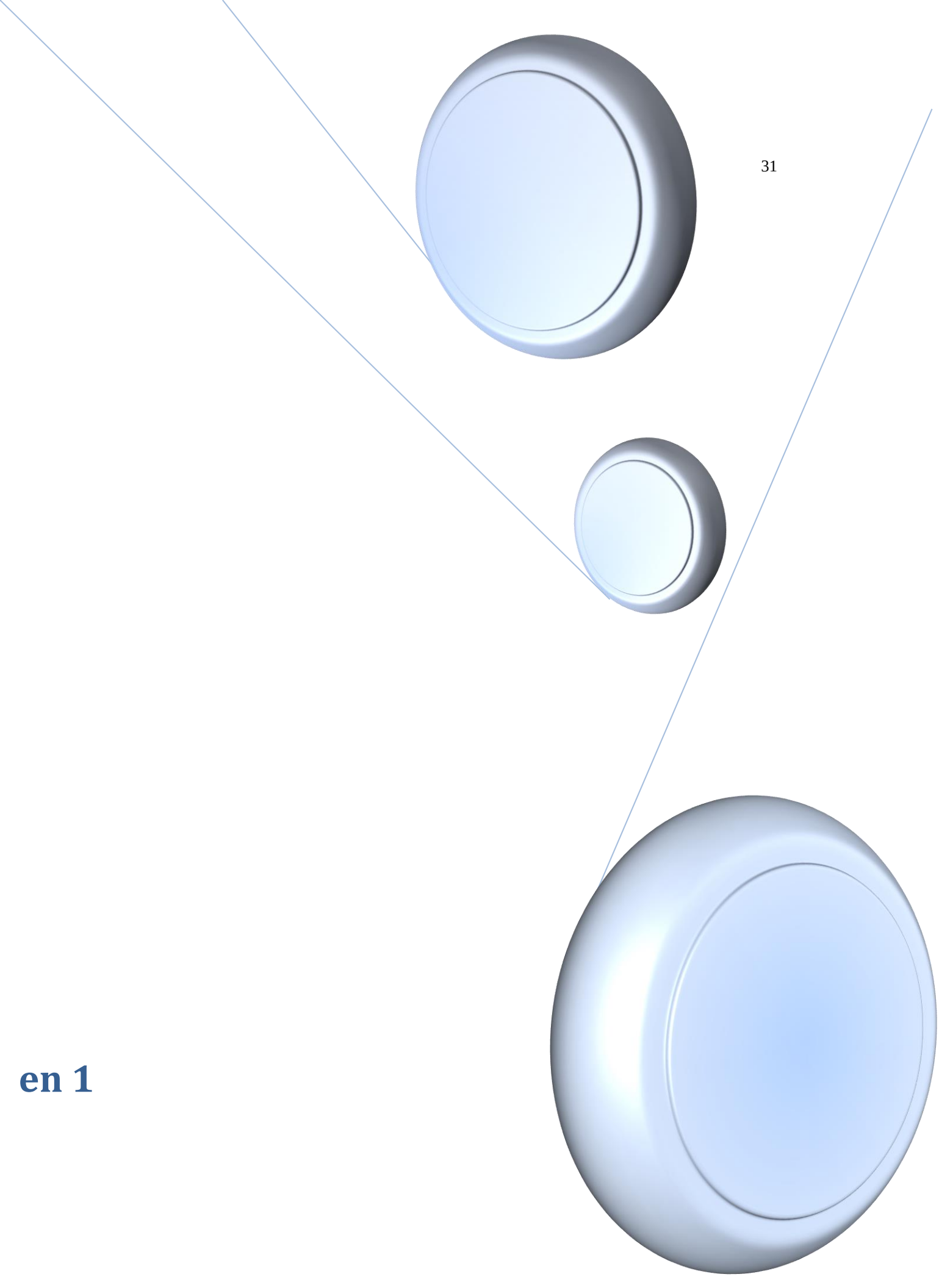
Para calcular las latencias se creó una función *Latencia()* la cual tiene como parámetros de entrada el nodo a, nodo b, y una lista ligada que se utiliza para el manejo de los paquetes. En esta función se calcula la latencia del nodo a al nodo b, así como su división de paquetes, cada paquete se almacena en la lista ligada *listapaq* con los datos *inicio,fin y tampaquete* y cada que se crea un paquete se aumenta un contador el cual después se multiplica con la fórmula para obtener latencia.

Cuando se crea un paquete se verifica si el paquete inicial es mayor que el du del arco correspondiente, si lo es se inserta en la lista un nuevo paquete con el tamaño de du y con inicio y fin de ese arco, además de eso se verifica que la resta de tamaño del paquete menos el du sea menor que el du del arco, si no lo es se crea nuevamente otro paquete asi hasta que la resta sea menor que el *du* del arco.

El programa contiene una clase llamada *GraphViz.java* la cual es utilizada para mostrar de forma grafica los caminos posibles y el grafo principal.

Para Utilizar el programa se debe cargar el archivo.txt a leer dando clic en Examinar y después dar clic en CALCULAR, se pueden ver el grafo principal y además de eso los grafos de los caminos posibles con los botones anterior y siguiente:


Calculador de Latencias
Grafo Original
Caminos Posibles: Path1
1
2
3
5
Anterior
Siguiente
Latencias
Path1: 0.019988885
Path2: 0.0017496409
Latencia Menor Seg: 0.0017496409
Tamaño del Archivo GB: 4.8
Numero de Paquetes: 3964585.2
Tiempo Total Min: 115.61001
Archivo Fuente: Archivo.txt
Examinar
CALCULAR
1
2
3
5
4
6



en 1

user

32

Introducción

Con este trabajo se podrá reconocer la importancia de la Latencia en las redes, tres de los factores más importantes son la propagación de la señal, el tiempo de transmisión de paquetes y el tiempo para el procesamiento de los paquetes, se mostrara cada uno en caso de la propagación de señal se entiende como el tiempo que tarda en viajar una señal de un punto A a un punto B, el tiempo de transmisión de paquetes es cuánto tarda en moverse a través de un enlace, aquí es donde se utilizan las conversiones más comunes y el tiempo de procesamiento es el cálculo de cuánto tarda un paquete en hacer el recorrido de un punto a otro.

Desarrollo del tema

Se realizara un programa que por medio de un archivo de el Path de menor Latencia+#paquetes +tiempo de transferencia

Se presentara un archivo.txt del cual se obtendrán los datos

1.-#V #E (vértices y aristas)

2.-#Vi, #vF, distancia (metros), velocidad (Mbps), Tamaño paquete y Datos de control

3.-Tiempo de cola

4.- #Vorigen #Vfinal

5.- Tamaño archivo

Lectura

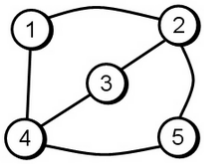
Se leerá cada una de las líneas y se asignaran las variables, vectores, matrices o listas que sean necesarias. Se utiliza el método Split (Sting) el cual nos permite dividir una cadena en base a las ocurrencias de una expresión regular la cual puede ser simplemente una letra, coma, signo o una construcción más elaborada.

Matriz Adyacencia

Sea G un grafo con n vértices {vi}i –> n = 1. Llamamos matriz de adyacencia a la matriz de orden n x n, de manera que cada elemento de la matriz es igual al número de aristas arcos del vértice i al j, es decir los caminos de longitud igual a la potencia de la matriz A = {aij}. Si A2, entonces los elementos de esta matriz contendrán el número de caminos de longitud 2 entre cada vértice.

Entonces podemos decir que se realizara una matriz de adyacencia y se indicara la relación que tiene un nodo con otro.

Ejemplo:



M	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	0	1
3	0	1	0	1	0
4	1	0	1	0	1
5	0	1	0	1	0

Latencia por arista

Se realizara el cálculo de cada arista con adyacencia y se guardara para que al tener los recorridos solo se utilizen en el cálculo de latencia del recorrido. Utilizando las formulas dadas en clase

Lat A-B=Tiempo propagación Tiempo Transmisión +Tiempo Cola

Tiempo propagación=Distancia a recorrer/velocidad de la luz

Además de la utilización de las formulas se realizaran cálculos y transformaciones para convertir por ejemplo de km a m o de Mbps a bytes.

Recorridos

Se utilizara el recorrido (o búsqueda) en profundidad (depth-first search):

La idea es alejarse lo más posible del nodo inicial (sin repetir nodos), luego devolverse un paso e intentar lo mismo por otro camino, así se podrá ir guardando cada ruta, este recorrido suele hacerse recursivo ya que de esa manera se podría visitar cada nodo faltante y checar sus adyacentes para un nuevo camino.

Latencia por recorrido

Se harán los cálculos de cada recorrido, utilizando las latencias ya obtenidas en una formula general la cual va sumando cada latencia de las aristas que corresponden a ese recorrido y se irán calculando los paquetes por los cuales se transmite la información.

Al terminar tendremos una latencia total para cada recorrido

Recorrido menos costoso

Se utilizara el Algoritmo Dijkstra o también llamado caminos mínimos es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista. En este caso los pesos serán las latencias, asi se podrá utilizar para comparar cada latencia e ir checando cual es el camino más corto, imprimir el path y la latencia menor.

Conclusión

Este programa ha mostrado como es posible diseñar e implementar el Calculo de Latencia del recorrido de un grafo, permite reconocer las distintas características y los factores importantes que implican una buena interconexión y transmisión de la información a través de equipos de enlace y se ha mostrado que el codigo fuente generado es util y valioso para producir mejoras apreciables y significativas al facilitar el cálculo de grandes recorridos.

BIBLIOGRAFIA

[1].-Estructura de datos y algoritmos – Alfred V. Aho, Jeffrey D. Ullman y John E. Hopcroft

[2].- http://users.dcc.uchile.cl/~bebustos/apuntes/cc30a/Grafos/

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Ingeniería en Ciencias de la Computación

Materia: Modelo de Redes

26/09/14

Primer examen

Calculo de Latencia

Profesor: Iván Olmos Pineda

Alumna:
Diana Barrientos Rodríguez 201103246

Introducción

Actualmente la cantidad de datos transmitida por una red aumenta debido al número de usuarios de las redes así como por la complejidad de los datos transmitidos. Existen dos variables para el cálculo de la transmisión:

- Tasa de Transferencia
- Latencia

En este proyecto nos enfocaremos en el cálculo de la latencia para obtener el tiempo que toma un bit para viajar de un extremo del medio a otro. Para realizar este cálculo debemos considerar los siguientes factores:

- Tiempo de propagación del bit por el medio: Depende del tiempo de propagación de la corriente o luz por el medio, además de la distancia recorrida.
- Máxima cantidad de datos: Son los datos que pueden ser transmitidos por la red sin ser segmentados. Al tiempo de propagación del bit en un paquete se le llama tiempo de transmisión.
- Tiempo de espera: Es el tiempo en el que se difunde un paquete a través de un conmutador (también se llama tiempo de cola).

La fórmula a ocupar será la siguiente:

Latencia = Tiempo de Propagación + Tiempo de Transmisión + Tiempo de Cola

Donde:

Tiempo de Propagación = $\frac{distancia\ a\ recorrer}{velocidad\ de\ la\ luz}$

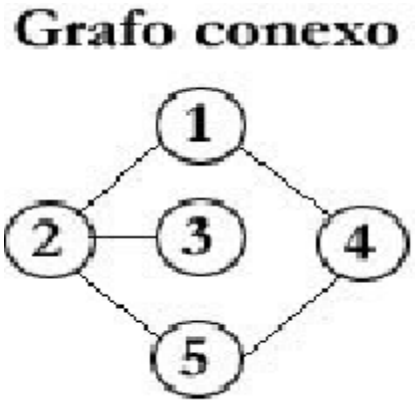
Tiempo de Transmisión = $\frac{tamaño\ del\ paquete}{tasa\ de\ transferencia\ teorica}$

Desarrollo

Ahora que sabemos que es la latencia y como se calcula desarrollaremos un programa en java que realice este cálculo a partir de la representación de la red en un grafo, primero realizaremos la lectura de los datos necesarios a través de un archivo con un formato especifico, posteriormente se realizara la división de los paquetes y finalmente teniendo todos los datos aplicaremos la formula antes mencionada.

Representación del grafo:

Este debe ser un grafo conexo, por ejemplo:



Archivo:

Los datos el archivo deberá contar con el formato siguiente:

1. # vértices , #enlaces
2. Para cada #enlace: #vertice_inicial , #vertice_final , distancia(metros) , velocidad(Mbps) , tamaño_paquete , datos_control
3. TC1 , TC2 , … , TCn
4. vértice_origen , vértice_destino
5. tamaño del archivo (GB)

Para realizar la lectura del archivo con el formato indicado usaremos la clase *StringTokenizer* de java la cual nos separa las líneas del archivo en subcadenas indicando un tipo de separador que en este caso será la “coma (,)”, una vez leídos los datos se almacenan en variables para que se puedan manipular.

Y así es como queda la lectura con el formato requerido:

```
46    public void LeeGrafo(String arch) //Lee archivo con los datos del grafo
47     {
48
49                    FileInputStream fp;
50                    DataInputStream f;
                      String linea = null;
52
                      int token1=0,token2=0,token3=0,token4=0,token5=0,token6=0,i,j,cont=1,nodos=0,aristas=0;
                      int ti=0,Ninicio=0,Nfinal=0,Tampaq=0;
55    try
56        {
57                    fp = new FileInputStream(arch);
58                    f = new DataInputStream(fp);
59                    linea=f.readLine();
```

```
Output - Grafos (run)
run:
 Numero de Vertices: 6
 Numero de Arcos: 6

1 2 98 1000 1500 200
2 3 105 500 1500 300
2 5 87 1000 1900 400
3 4 90 700 1100 400
5 6 130 350 1300 500
6 4 60 1000 1100 500
TiempoC1: 2.5E-4
TiempoC2: 0.001
TiempoC3: 7.0E-4
TiempoC4: 0.0093
Nodo inicial: 1
Nodo final: 4
Tamaño del paquete: 3
```

Recorrido del Grafo:

Para obtener todos los caminos del nodo inicial al nodo final se utilizo el siguiente algoritmo:

Algoritmo de Búsqueda: Depth First Search (Recorrido Profundo)

Una de las aplicaciones de este algoritmo es para problemas de conectividad, si un grafo es conexo, detecta los ciclos en un grafo.

Como trabaja:

DFS va formando un árbol al igual que BFS pero lo hace a profundidad. Existen dos formas de hacer el recorrido una es usando una Pila y otra de manera recursiva, siendo esta ultima la que se implementara ya que es más útil, ocupa menos recursos y es la forma más usada en la solución de problemas con este algoritmo.

Pseudocódigo:

método DFS(*origen*, *final*):

marcamos *origen* como visitado 41

recuperar el path si se llego a final

para cada vertice *v* adyacente a *origen* en el *Grafo*:

si *v* no ha sido visitado:

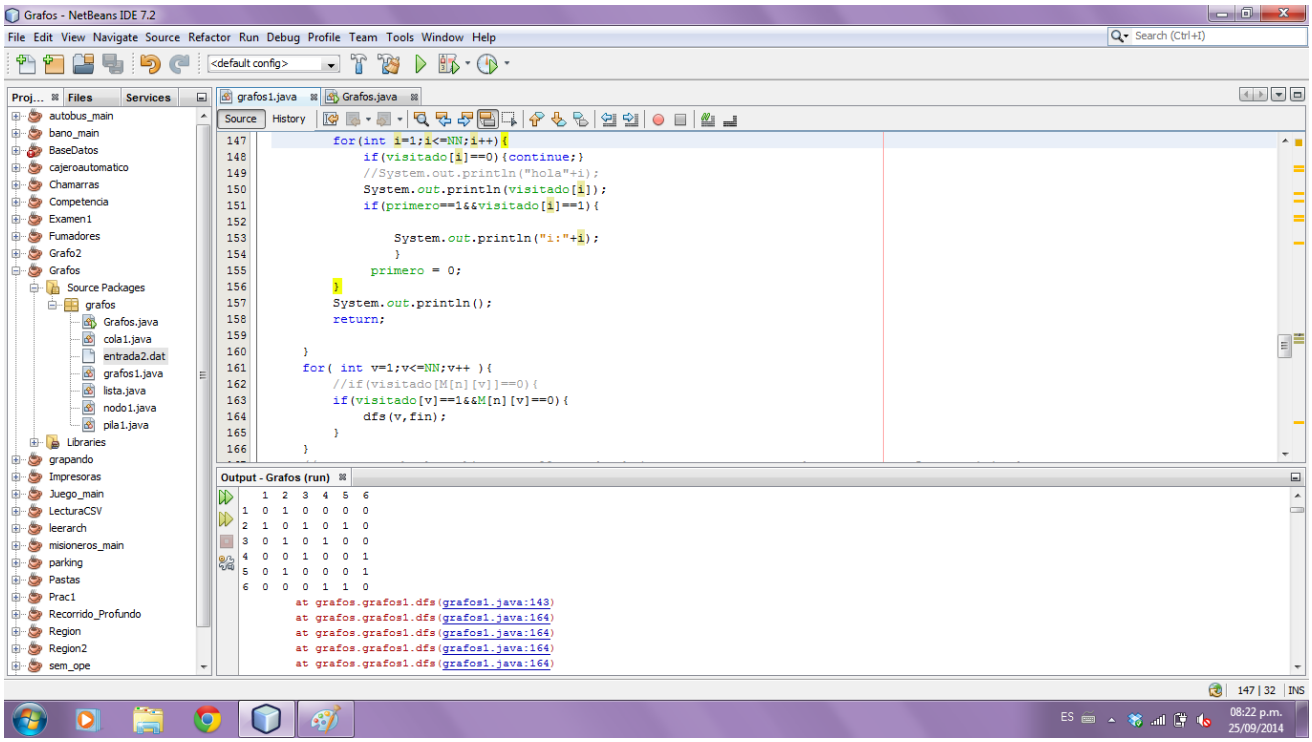
marcamos como visitado *v*

llamamos recursivamente DFS(*v*)

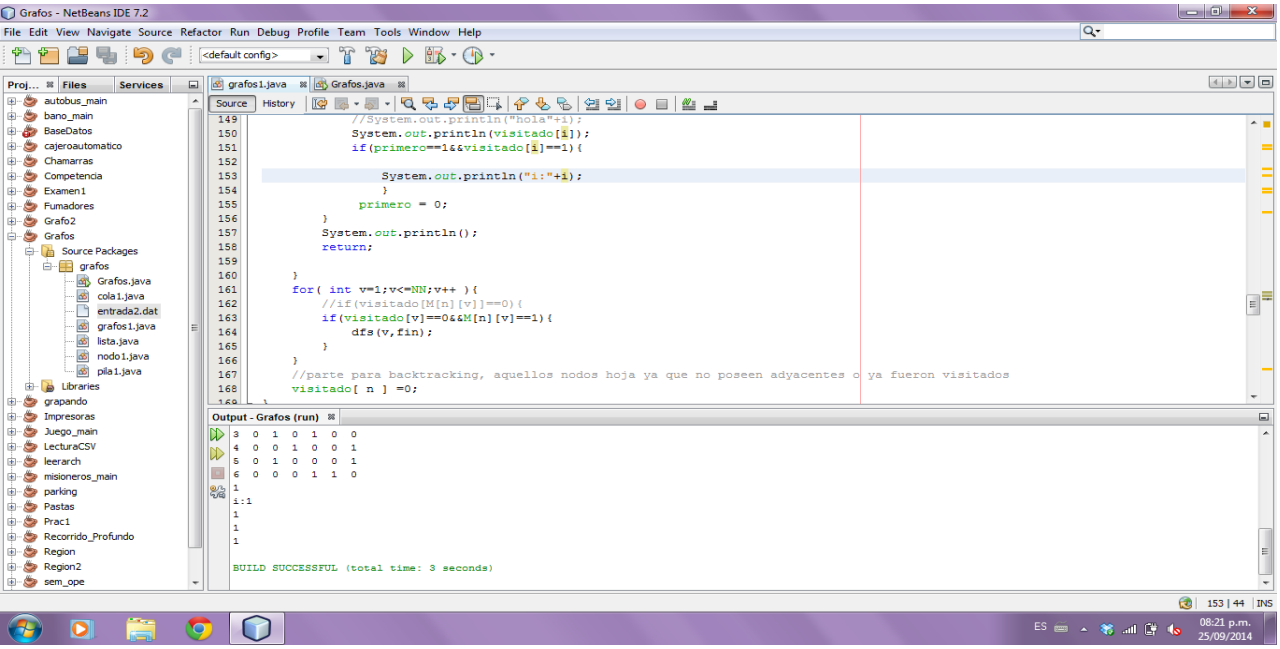
marcamos *origen* como no visitado

En cada llamada recursiva marcaremos el nodo actual como visitado, luego verificamos si es el nodo final o buscado para salir de la recursión, si no es el nodo que estamos buscando se hace la llamada recursiva con todos los nodos adyacentes al nodo actual, el algoritmo recorrerá en profundidad hasta llegar a un nodo extremo antes de regresar a la recursión en donde se encuentran los otros nodos.

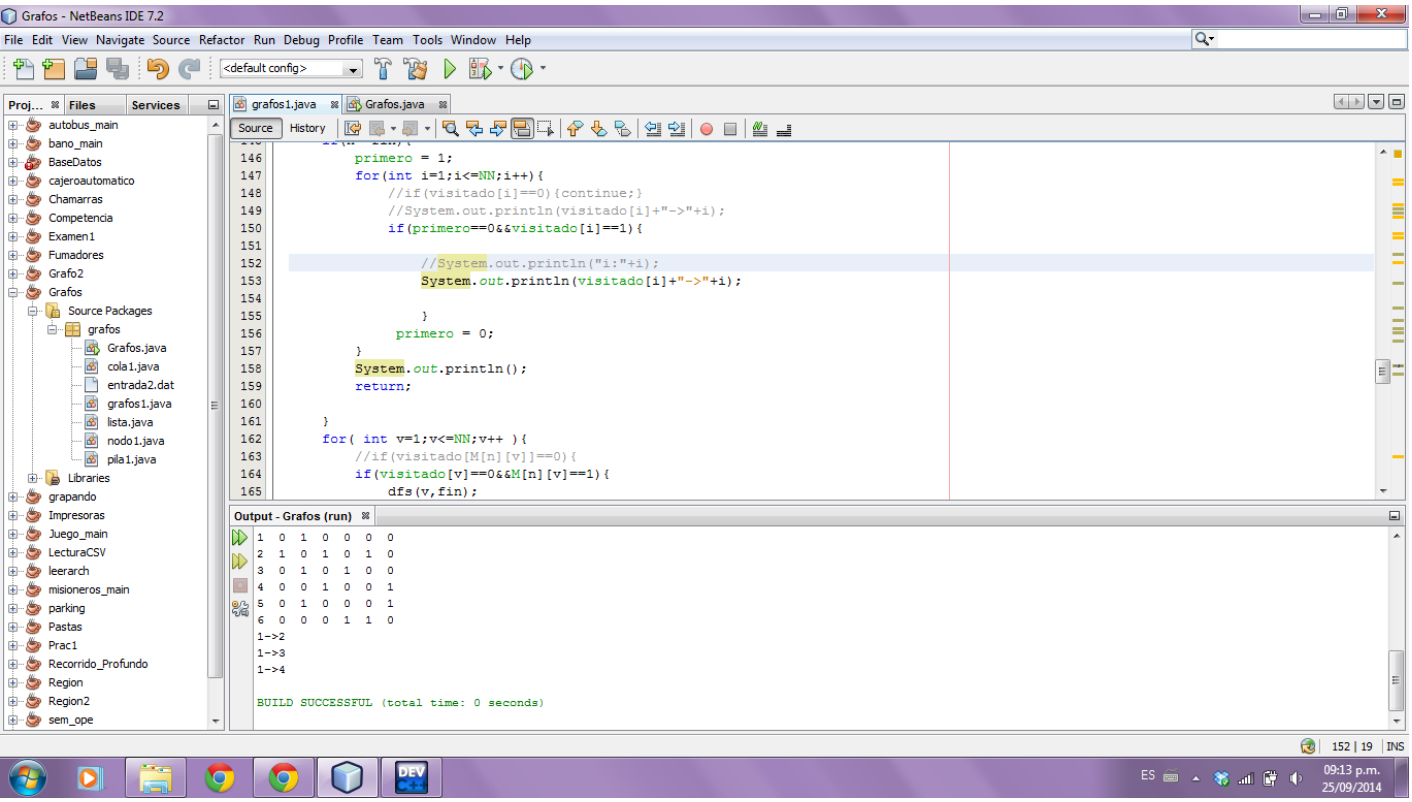
En esta parte del proyecto tuve que dedicar mucho tiempo ya que la salida no fue la esperada desde el principio de la implementación del algoritmo, marcaba errores solo al momento de compilar:



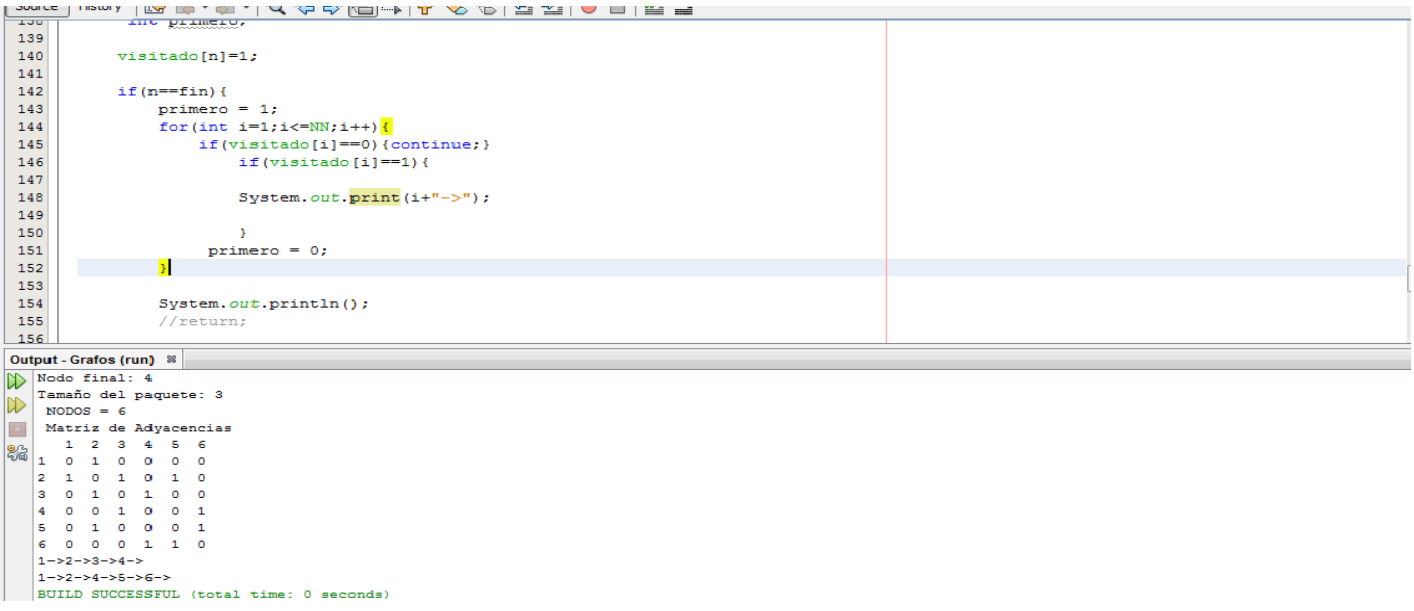
Cuando se resolvió esto al modificar las condiciones, nuevamente la salida no fue la esperada, ya que no se encontraban los caminos en forma adecuada:



Se logró resolver esto ya que solo estaba considerando a la variable primero con un uno, es decir verdadero, al modificar esta parte del algoritmo ya se pudo imprimir un camino posible:



Resolvemos el problema y ahora nos muestra el primer camino en orden y el segundo camino que encontró lo muestra en desorden:



Se puso una variable más para que imprimiera el camino de forma correcta, se le asignó el nodo fin para que saliera al final del recorrido.


```
142     if(n==fin){
143         primero = 1;
144         int temp = 0;
145         for(int i=1;i<=NN;i++){
146             if(visitado[i]==0){continue;}
147                 if(visitado[i]==1){
148                     if(fin==i){
149                         temp=i;
150                     }else{
151                         System.out.print(i+"->");
152                     }
153
154                 }
155              primero = 0;
156         }
157         System.out.println(temp);
158
```

```
Output - Grafos (run)
 Matriz de Adyacencias
   1  2  3  4  5  6
1  0  1  0  0  0  0
2  1  0  1  0  1  0
3  0  1  0  1  0  0
4  0  0  1  0  0  1
5  0  1  0  0  0  1
6  0  0  0  1  1  0
1->2->3->4

1->2->5->6->4

BUILD SUCCESSFUL (total time: 1 second)
```

Resultados:

Se obtuvieron los caminos de un nodo inicial a un nodo final, a través del algoritmo dfs recursivo.

Conclusión:

No se llegó a la terminación del proyecto, que era el cálculo de la latencia ya que me llevo mucho tiempo el análisis del algoritmo y la implementación para que lograra obtener todos los caminos posibles.

Bibliografía:

http://www.widget-101.com/codigo/algoritmos-de-busqueda-en-anchura-bfs-y-busqueda-en-profundidad-dfs/ consulta: 17 de Septiembre de 2014

http://codebreakerscorp.wordpress.com/2011/03/05/algoritmo-de-busqueda-depth-first-search/

consulta: 13 de Septiembre de 2014

http://blog.iweb.com/es/2014/02/entender-analizar-reducir-latencia/2463.html consulta: 10 de Septiembre de 2014

http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/colecciones/stringtokenizer.htm consulta: 10 de Septiembre de 2014

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Ingeniería en Ciencias de la Computación

Materia: Modelo de Redes

6/11/14

Segundo examen

Cableado Estructurado

Profesor: Iván Olmos Pineda

Alumnos:

Christopher Jesús Ruiz Escápita 201136965

Diana Barrientos Rodríguez 201103246

Luis Conde Rodríguez 201124489

Brianda Yareli Cruz Guerrero 201114701

Alejandro Jiménez Ortega 201104393

Introducción:

¿Qué es el Cableado Estructurado?

Es el conjunto de elementos pasivos, flexible, genérico e independiente, que sirve para interconectar equipos activos, de diferentes o igual tecnología permitiendo la integración de los diferentes sistemas de control, comunicación y manejo de la información, sean estos de voz, datos, video, así como equipos de conmutación y otros sistemas de administración.

En un sistema de cableado estructurado, cada estación de trabajo se conecta a un punto central, facilitando la interconexión y la administración del sistema, esta disposición permite la comunicación virtualmente con cualquier dispositivo, en cualquier lugar y en cualquier momento.

El Cableado Estructurado trata de especificar una “Estructura” o “Sistema” de cableado para empresas y edificios que sea:

– Común y a la vez independiente de las aplicaciones

– Documentada (Identificación adecuada)

– Proyectada a largo plazo (> 10 años)

Topología del Cableado Estructurado:

Los sistemas de Cableado Estructurado usan topología tipo estrella extendida donde todas las áreas de trabajo se enrutan hacia un armario en el CT.


ESTRELLA

Elementos de un Cableado Estructurado:

Área de Trabajo.- Los componentes del área de trabajo se extienden desde la terminación del cableado horizontal en la salida de información, hasta el equipo en el cual se está corriendo una aplicación sea de voz, datos, video o control.

Closet de comunicaciones.- Es el punto donde se concentran todas las conexiones que se necesitan en el área de trabajo.

Cableado Horizontal.- Es aquel que viaja desde el área de trabajo hasta el closet de comunicaciones.

Cuarto de equipo.- En este cuarto se concentran los servidores de la red, el conmutador telefónico, etc. Este puede ser el mismo espacio físico que el del closet de comunicaciones y de igual forma debe ser de acceso restringido.

Cuarto de entrada de servicios (Acometida).- Es el punto donde entran los servicios al edificio y se les realiza una adaptación para unirlos al edificio y hacerlos llegar a los diferentes lugares del edificio en su parte interior.

Cableado Vertebral (vertical o Back Bone).- Es el medio físico que une 2 redes entre sí.

Planteamiento del Problema:

Realizar el diseño de red para la facultad de ciencias de la computación, esta cuenta con 4 edificios (104A, 104B, 104C y 104D), de los cuales solo 3 recibirán

acceso a la red, tomando en cuenta que el servicio de red es proporcionado desde el SIU.

50

El edificio 104A brindara servicio a los cubos de los profesores, siendo un total de 23 cubos con 2 profesores cada uno. Este edificio cuenta con una línea de comunicación desde el SIU.

El edificio 104C cuenta con 4 laboratorios con servicio 50 – 50 para 35 computadoras.

El edificio 104D cuenta con 3 pisos y cada piso tiene 4 salones cada uno con 35 alumnos a los que se les proporcionara servicio.

El edificio 104B solo se tomara como punto de conexión hacia los edificios 104C y 104D ya que en ese edificio llega una línea de comunicación desde el SIU.

Para realizar el diseño se pide definir:

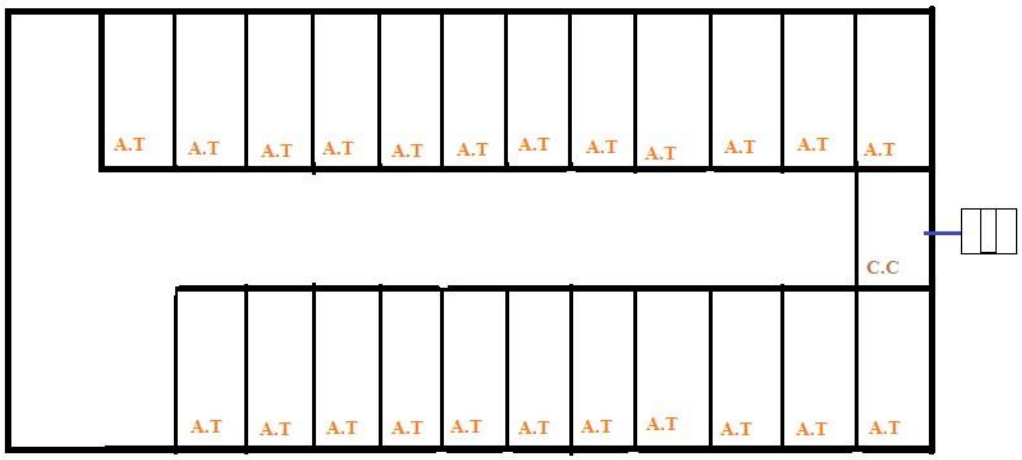
- Esquema general de instalación
- Especificar el equipo a utilizar
- Costos aproximados de instalación
- Propuesta de simulación lógica
- Balance de cargas equilibrado

Objetivo:

Con la elaboración de este proyecto se pretende desarrollar conocimientos necesarios que nos permitan en un futuro llevar a cabo el diseño e instalación de una red. Se pretende también adquirir nuevos conocimientos mediante la investigación así como la aplicación de todo lo aprendido hasta el momento.

Esquema General:

1) 104A



Este edificio cuenta con una línea de comunicación desde el SIU, la cual llega al closet de comunicaciones en el que se encontrara un rack de piso a una distancia de 100cm de la pared para que una persona pueda dar mantenimiento aquí se encontraran: un router balanceador de carga que se encargara de repartir los paquetes a cada uno de los cubos, dos switch troncales de 48 puertos, 2 patch panels y patch cords conectados del switch al patch panel.

Para la red Ethernet cada cubo contara con dos nodos en cada cubo a una distancia de 5.5 m del rack al primer cubo y asi sucesivamente para asegurar la eficiencia en la comunicación de datos.

Para la red inalámbrica se contara con 3 access point repartidos en el pasillo.

Presupuesto:

ALA "A"			
	METROS	CABLE UTP PRECIO POR METRO	PRECIO TOTAL
C1	5.5m	$9	$49.5
C2	8m		$72
C3	10.5m		$94.5
C4	13m		$117
C5	15.5m		$139.5
C6	18m		$162

1) 104A

C7	20.5m		$184.5
C8	23m		$207
C9	25.5m		$229.5

C10	28m		$252
C11	30.5m		$274.5
C12	33m		$297
			Total $2079

ALA "B"			
	METROS	CABLE UTP PRECIO POR METRO	PRECIO TOTAL
C1	7.5m	$9	$67.5
C2	10m		$90
C3	12.5m		$112.5
C4	15m		$135
C5	17.5m		$157.5
C6	20m		$180
C7	22.5m		$202.5
C8	25m		$225
C9	27.5m		$247.5
C10	30m		$270
C11	32.5m		$292.5
			Total $1980

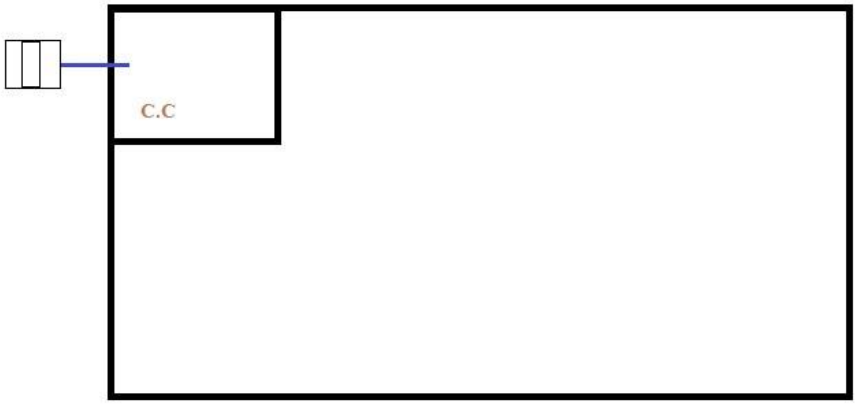
MATERIAL COMPLEMENTARIO			
Cantidad	Material	Precio unitario	Precio total
46	Plugs	$6	$276
23	Cajas de registro	$39	$897
114+111=225	Canaleta	$59	$13275

DISPOSITIVOS			
Cantidad	Material	Precio unitario	Precio total
2	Switch	$3446	$6892
1	Router	$2819	$2819
3	Access Point	$5468.56	$16405.68
2	Patch Panel	$685	$1370
24	Patch Cord	$389	$9336
2	Rack	$1990	$3980

Total final

$59309.68

2) 104B



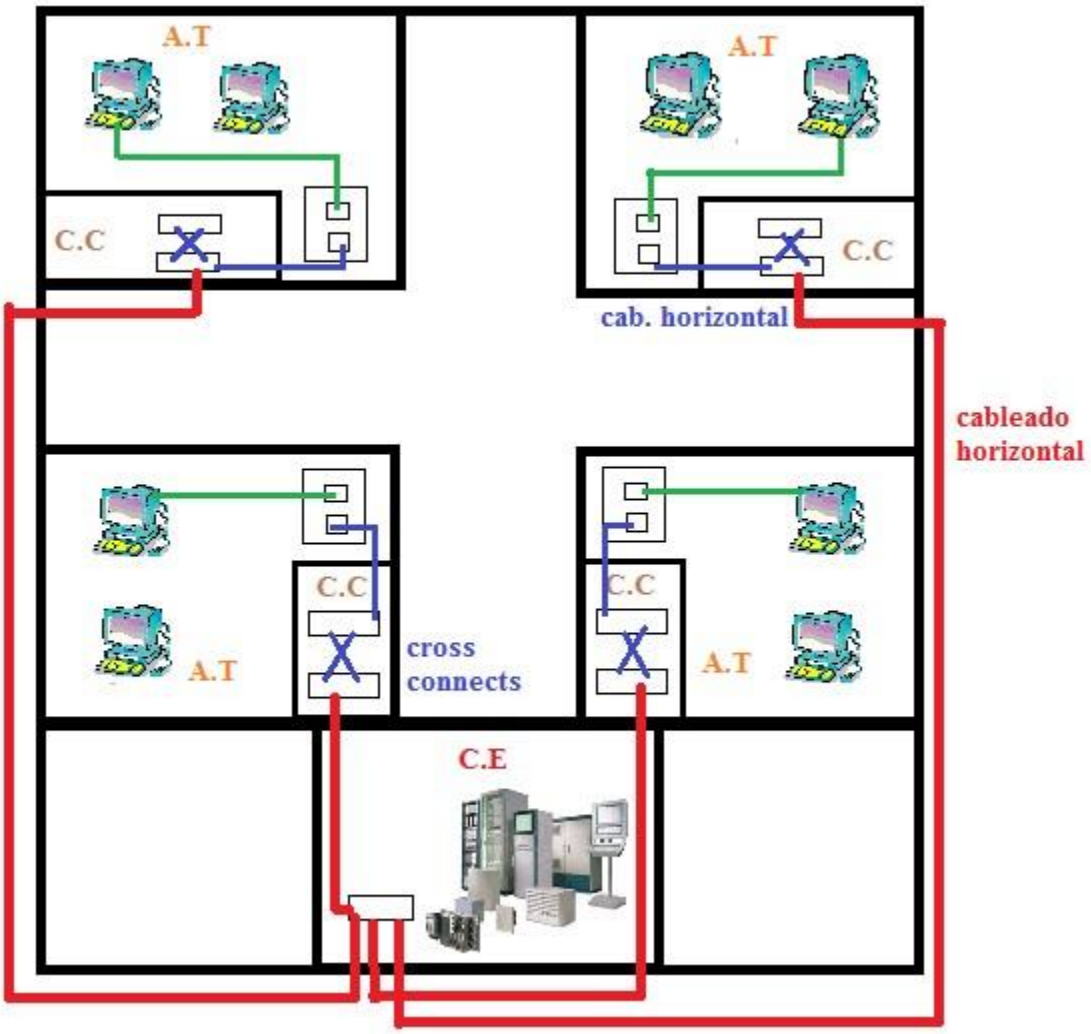
Este edificio al igual que el 104 A cuenta con una línea directa del SIU que llega al closet de comunicaciones, el cual se conectara al registro ubicado fuera del 104C.

Presupuesto:

DISPOSITIVOS			
Cantidad	Material	Precio unitario	Precio total
2	Switch	$3446	$6892
1	Router	$2819	$2819
4	Patch Panel	$685	$2740
12	Patch Cord	$389	$4668
2	Rack	$1990	$3980

Total final
$21099

3) 104C


A.T
A.T
C.C
C.C
cab. horizontal
cableado
horizontal
C.C
C.C
A.T
cross
connects
A.T
C.E

3) 104C

Este edificio contara con un cuarto de equipo que está conectado con el closet de comunicaciones del edificio 104B a través de un cableado backbone, cada laboratorio contara con su propio closet de comunicaciones los cuales estarán conectados desde el cuarto de equipo a través de un cableado horizontal usando cross connects , posteriormente estos cuartos se conectaran a las cajas de registro para dar servicio a 35 alumnos, este servicio será 50-50 , colocaremos 8 nodos para Ethernet y el resto para inalámbrica.

En cada fila se colocaran 4 cajas de registro para la red Ethernet a una distancia de 3.20m desde el closet hacia cada fila. Para la red inalámbrica se pondrá un Access point por cada laboratorio.

Presupuesto:

LAB 1, LAB2, LAB3, LAB4			
	METROS	CABLE UTP PRECIO POR METRO	PRECIO TOTAL
F1	1.60m	$9	$14.4
F2	3.20m		$28.8
F3	4.80m		$43.2
F4	6.40m		$57.6
F5	8.0m		$72
SubTotal $216*4			
Total $864			

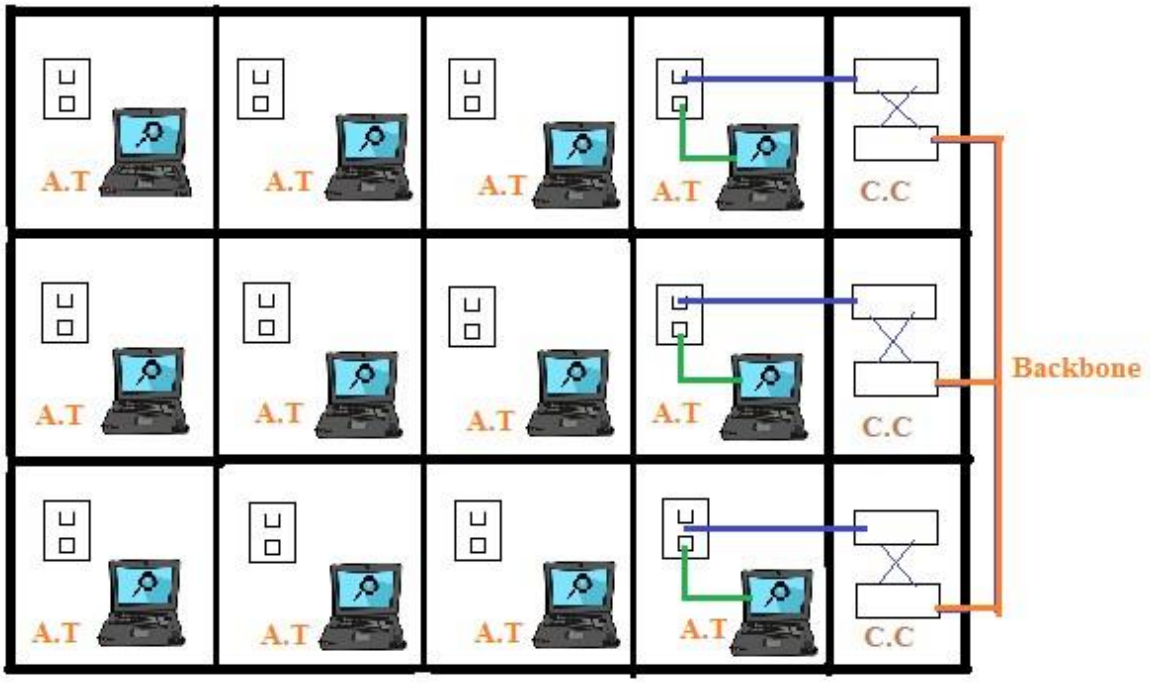
MATERIAL COMPLEMENTARIO			
Cantidad	Material	Precio unitario	Precio total
24	Plugs	$6	$114
80	Cajas de registro	$39	$119
56	Canaletas	$59	$3304

DISPOSITIVOS			
Cantidad	Material	Precio unitario	Precio total
8	Switch	$3446	$27568
4	Router	$2819	$11276
4	Access Point	$5468.56	$21874.24
4	Patch Panel	$685	$2740
12	Patch Cord	$389	$4668
8	Rack	$1990	$15920

Total final

$88447.24

4)104D



Este edificio se conectara a través del registro ubicado fuera del 104c por medio de un cableado backbone que llegara a otro registro que está ubicado fuera del 104D, este se conectara a cada closet de comunicaciones de cada piso hacia los cross-connects con una distancia de 3.40m, de ahí será conectado a cada salón por medio de un cableado horizontal a una distancia de 8.25m por cada salón.

Para la red Ethernet se contaran con 15 nodos en cada salón, colocando una caja de registro para el profesor, 3 cajas del lado derecho, 2 cajas en la parte posterior del salón y 2 más del lado izquierdo.

Para la red inalámbrica contaremos con un Access point dentro de cada salón.

Presupuesto:

Piso1,Piso2,Piso3			
	METROS	CABLE UTP PRECIO POR METRO	PRECIO TOTAL

P1	33m	$9	$297
P2	33m		$297
P3	33m		$297

Total $891

MATERIAL COMPLEMENTARIO			
Cantidad	Material	Precio unitario	Precio total
24	Plugs	$6	$144
96	Cajas de registro	$39	$3744
54	Canaletas	$59	$3186

DISPOSITIVOS			
Cantidad	Material	Precio unitario	Precio total
8	Switch	$3446	$27568
4	Router	$2819	$11276
4	Access Point	$5468.56	$21874.24
12	Patch Panel	$685	$8220
24	Patch Cord	$389	$9336
8	Rack	$1990	$15920

Total final
$102159.24

Presupuesto cableado exterior:

10B			
	METROS	CABLE UTP PRECIO POR METRO	PRECIO TOTAL
C.C-Reg	11m	$9	$99
		Total	$99

MATERIAL COMPLEMENTARIO			
cantidad	Material	Precio unitario	Precio total
2	Plugs	$6	$12
6	Canaletas	$59	$354

Total final

$81039

104C			
	METROS	CABLE UTP PRECIO POR METRO	PRECIO TOTAL
C.E-LAB1	15m	$9	$135
C.E-LAB2	18m		$162
C.E-LAB3	18m		$162
C.E-LAB4	18m		$162
C.E-R.C.	15m		$135
Reg-104D	70m		$630
			Total $1386

MATERIAL COMPLEMENTARIO			
cantidad	Material	Precio unitario	Precio total
15	Plugs	$6	$90
77	Canaletas	$59	$4543

Total final
$6019

104D			
	METROS	CABLE UTP PRECIO POR METRO	PRECIO TOTAL
Reg-P1	15.4m	$9	$138.6
Reg-P2	18.8m		$169.2
Reg-P3	22.2m		$199.8
			Total $507.6

MATERIAL COMPLEMENTARIO			
cantidad	Material	Precio unitario	Precio total

15	Plugs	$6	$90
30	Canaletas	$59	$1770

Total final
$2277.6

Presupuesto Final Total:

Edificio	Presupuesto total
104 A	$59309.68
104B	$21099
104C	$88447.24
104D	$102159.24
SubTotal	$271015.16

Edificio	Cableado Exterior
104B	$81039
104C	$6019
104D	$2277.6
SubTotal	$89335.6

Total
$360350.76

Especificaciones del Equipo:

Cisco Catalyst 3850-48T-L - switch - 48 ports - managed - desktop, rack – mount

Precio: $3,446.00

Caracteristicas:

De humedad de funcionamiento: 10-95% (sin condensación)

Temperatura máxima de funcionamiento: 113 ° F

Temperatura máxima de almacenamiento: 158 ° F

Temperatura mínima de funcionamiento: 23 ° F

Temperatura mínima de almacenamiento: -40 ° F

Marca: Cisco

Compatibilidad: Ordenador personal

Fabricante: Cisco Systems

Modelo: 3850-48T-L

Cantidad de Unidades: 1

Interfaz proporcionada

Tipo de conector: RJ-45

Interfaz: Ethernet 10Base-T / 100Base-TX / 1000BASE-T

Método de autenticación: Kerberos, RADIUS, Secure Shell (SSH), TACACS +

Cumplimiento de normas: BSMI CNS 13438 Clase A, CISPR 22 Class A, CISPR 24, CSA C22.2 No. 60950-1 segunda edición, EN 60950-1 segunda edición, EN 61000-3-2, EN 61000-3-3, EN55022 Clase A , EN55024, FCC Part 15 A, GOST, ICES-003 Clase A, IEC 60950-1 segunda edición, la norma ISO 7779, KCC, NOM, RoHS, UL 60950-1 segunda edición, VCCI Clase A

Redes

Cumplimiento de normas: IEEE 802.1D, IEEE 802.1p, IEEE 802.1Q, IEEE 802.1s, IEEE 802.1w, IEEE 802.1x, IEEE 802.3, IEEE 802.3ab, IEEE 802.3ad (LACP), IEEE 802.3u, IEEE 802.3x, IEEE 802.3z

Tecnología de conectividad: Wired

Velocidad de transferencia de datos: 1 Gbps

Características: Access Control List (ACL), soporte ARP, negociación automática, señal ascendente automática (MDI / MDI-X), la tecnología de Cisco StackPower, la tecnología de Cisco StackWise-480, snooping DHCP, Dynamic ARP Inspection (DAI), Dynamic Trunking Protocol (DTP) de apoyo, Energy Efficient Ethernet, NetFlow Flexible (FNF), snooping IGMP, soporte IPv4, soporte IPv6, balanceo de carga de capa 3, Link Aggregation Control Protocol (LACP), MLD, apoyo Multiple Spanning Tree Protocol (MSTP), Puerto Protocolo de agregación (PAgP), Quality of Service (QoS), soporte RADIUS, Rapid Per-VLAN Spanning Tree Plus (PVRST +), Protocolo Rapid Spanning Tree (RSTP) apoyo, interruptor remoto puerto Analyzer (RSPAN), en forma de Round Robin (SRR), Guardia de raíz STP, soporte para Syslog, Trivial File Transfer Protocol (TFTP) apoyo, Canalizaciones, Detección Enlace Unidireccional (UDLD), Virtual Ruta Forwarding-Lite (VRF-Lite), soporte VLAN

Tabla de direcciones MAC Tamaño: 32.000 entradas

Protocolo de gestión remota: CLI, RMON 1, RMON 2, SNMP 1, SNMP 2c, SNMP 3, SSH, Telnet

Indicadores de estado: Modo dúplex del puerto, velocidad de transmisión del puerto, Estado, Sistema

Dispositivo de alimentación

Frecuencia requerida: 50/60 Hz

Cantidad instalada: 1

Cantidad máxima soportada: 2

Voltaje nominal: CA 120/230 V

Potencia suministrada: 350 vatios

Tipo: Fuente de alimentación interna

Software

Tipo: Cisco IOS LAN Base

Dimensiones y peso

Profundidad: 17,7 in

Altura: 1.8 in

Peso: 17 libras

Ancho: 17.5 in

Servicio y soporte

Tipo: Garantía limitada de por vida

Capacidad

Tipo: Rutas IPv4

Valor: 24000

Tipo: Entradas NetFlow

Valor: 48000

Tipo: Interfaces virtuales conmutadas (SVI)

Valor: 1000

Memoria Flash

Tamaño instalado: 2 GB

Rendimiento

Tipo: Capacidad de conmutación

Valor: 176 Gbps

Tipo: Ancho de banda de apilamiento

Valor: 480 Gbps

Puertos

Cantidad: 48

Tipo: 10/100/1000

RAM

Tamaño instalado: 4 GB

TP-LINK®

ROUTER DE BANDA ANCHA GIGABIT CON BALANCE DE CARGA

TL-ER5120

1 Puerto fijo WAN gigabit, 1 puerto fijo LAN/DMZ gigabit y 3 puertos libremente intercambiables WAN/LAN gigabit

Abundantes características de seguridad entre las cuales se incluyen inspección de ARP, Defensa DoS, Filtro de dominios URL/palabras clave y control de acceso

Implementa la restricción en las aplicaciones IM/P2P para manejar de una manera más sencilla el comportamiento del equipo en línea

Soporte con el servidor PPPoE para un mejor control de la red interna

Balanceador de la carga, automáticamente elige la mejor ruta de acuerdo a la carga

Protección contra descargas de 4kV profesional que mantiene su inversión tan segura como sea posible.

Precio: $2819

CARACTERÍSTICAS DEL HARDWARE	
Estándares y Protocolos	IEEE 802.3, 802.3u, 802.3ab TCP/IP, DHCP, ICMP, NAT, PPPoE, SNTP, HTTP, DNS, IPsec, PPTP, L2TP
Interface	1 Puerto WAN de 10/100/1000Mbps 3 Puertos WAN/LAN Libres Intercambiables de 10/100/1000Mbps 1 Puerto LAN/DMZ Fijo de 10/100/1000Mbps 1 Puerto de Consola (RJ-45 en RS232)
Medios de Red	10BASE-T: cable UTP categoría 3, 4, 5 (Máximo 100m) EIA/TIA-568 100Ω STP (Máximo 100m) 100BASE-TX: cable UTP categoría 5, 5e (Máximo 100m) EIA/TIA-568 100Ω STP (Máximo 100m) 1000BASE-T: cable UTP categoría 5, 5e, 6 (Máximo 100m)
Flash	8MB
DRAM	DDRII 128MB
LED	PWR, SYS, Link/Act, Velocidad, WAN, DMZ
Botón	Botón de Reseteo
Dimensiones (Largo x Ancho x Alto)	17.3x8.7x1.7 pulgadas (440 x 220 x 44 mm) Anchura de la Base de Montaje Estándar de 19 pulgadas, 1U Altura
Suministro de Energía Eléctrica	Suministro de Energía Universal Interna AC100-240V~ 50/60Hz Input

RENDIMIENTO	
Sesión Concurrente	120000
Rendimiento de NAT	350Mbps

FUNCIONES BÁSICAS	
Clon MAC	Modifica la Dirección WAN/LAN/DMZ MAC
Configuración del Switch	Duplicación de Puertos (Port Mirror) Control de Velocidad Configuración del Puerto Puerto VLAN
DHCP	Servidor DHCP /Client Reservación DHCP
Tipo de Conexión WAN	IP Dinámico, IP Estático, PPPoE, PPTP, L2TP, Acceso Dual, BigPond

FUNCIONES DE SERVICIO	
Control de Tráfico	Control de banda ancha basado en IP Garantía y banda ancha limitada Política de tiempo programado Límite de sesión basado en IP
Enrutado	Enrutado Estático Enrutado Dinámico (RIP v1/v2)
Load Balance	Balanceador de Carga Inteligente Política de Enrutado Enlace de Protocolo Respaldo de enlaces (Tiempo de ejecución, Sistema de recuperación de fallos) Detección en Línea
Modo del Sistema	NAT, Non-NAT, Enrutado Clásico
NAT	NAT uno a uno Multi-nets NAT Servidor Virtual, DMZ Host, Activación de Puertos, UPnP FTP/H.323/SIP/IPsec/PPTP ALG

Cisco Aironet 1140 Series

Precio $5468.56

Descripción: dual - banda independiente 802.11a/g/n; mbps 300 de datos velocidad de transferencia.

la creación de redes	
factor de forma	externo
tecnología de conectividad	inalámbrico
los datos de tasa de transferencia	mbps 300

los datos de protocolo de enlace	802.11b ieee, 802.11a ieee, 802.11g ieee, 802.11n ieee
los indicadores de estado	Activo, el error, la condición de
Características	Power over ethernet (poe), la tecnología mimo, wi-fi multimedia (wmm) apoyo, cisco m - la unidad
el algoritmo de cifrado	Aes, tls, peap, ttls, tkip, wpa, wpa2
método de autenticación	Ms - el capítulo v. 2, eap - rápido
cumplimiento de normas	802.11b ieee, 802.11a ieee, 802.3af ieee, 802.11d ieee, 802.11g ieee, 802.1x ieee, 802.11i ieee, 80

garantía del fabricante	
Servicio de apoyo&	limitada garantía de por vida
Servicio de apoyo& detalles	Garantía limitada - de por vida
los parámetros del medio ambiente	
min de temperatura de funcionamiento	0& deg; c
max temperatura de funcionamiento	40& deg; c
rango de humedad de funcionamiento	- 10 90% (no - de condensación)

Protocolo 802.11

Redes inalámbricas de Cisco 802.11n se han mejorado con las siguientes características:

- La tecnología de Cisco Clean Air - Crear una auto-curación, red inalámbrica auto-optimización que mitiga el impacto de la interferencia inalámbrica
- Cisco VideoStream - Proporcionar, imágenes de vídeo claras y precisas en todas partes mediante la entrega eficiente de vídeo de multidifusión desde el cable a la red inalámbrica
- Cisco ClientLink - Permita que los puntos de acceso para rellenar los agujeros de cobertura, aumentar el acceso a ambos clientes 802.11a / g heredados y 802.11n, y proporcionan una mejor cobertura de la señal incluso en entornos de RF difíciles

Rack de piso

Se denominan así a las estructuras abiertas que son ancladas al piso, por eso es posible contar con bastidores de mayor tamaño, en este caso no es necesario considerar la profundidad a diferencia de los bastidores (brackets o racks de pared).

El Rack de piso es una solución económica de alta capacidad para colocar los equipos de red y sus cables, es posible incluir también el cableado de la telefonía.

Precio: $1990

Especificaciones técnicas del rack de piso:

- Normalizados en 19".
- Totalmente desarmables y de fácil acceso.

- Entrada de cables por base.
- Base tipo pedestal con agujeros para anclaje al piso.
- Fabricados 100% en acero laminado en frío.
- Agujeros laterales para conectar rack's en serie.
- Rieles de montaje perforados con forma redonda con rosca conforme a normas EIA patrón 12-24, paso estándar de agujeros de paso 5/8"- 5/8" - 1/2", por sus dos caras.
- Ensamblados con pernos y tuercas de acero inoxidable mejorando notablemente la apariencia.
- Carga máxima de soporte 500 Kg (depende del modelo elegido).
- Tornillos para montaje de equipos incluidos.
- Todas las piezas que lo requieran se someten a procesos de desengrasado, fosfatizado y zincado electrolítico asegurando el cumplimiento de las normas A.S.T.M. de impacto, flexibilidad y adherencia así como de resistencia al medio ambiente.
- Terminación en pintura electrostática (Electro Posición Catódica) en polvo Poliéster Epóxica de máxima adherencia alta resistencia mecánica y química. Ofrece una resistencia cinco veces mayor al óxido y Ralladuras que los sistemas tradicionales de pintura.
- Acabado en color Negro RAL 9005.
- Disponibles en gama liviana (económico) o pesada (premier).
- Se deben fijar al piso mediante 4 bujes de expansión.

Precio: $685

Principales características:

- Cumple con ANSI/TIA/EIA 568B.2, ISO/IEC 11801&EN50173
- 48 Jacks RJ-45 8P8C
- 1 Unidad de Rack, UTP Cat 5e, 19 pulgadas
- Estructura Metálica en color negro, montaje en Rack 19"

Garantía:

1 año de garantía contra defecto de fábrica.

Categoría 5e UTP Patch Cords

Precio: $389

ESPECIFICACIONES

- **RENDIMIENTO DE TRANSMISIÓN**

ANSI / TIA / EIA-568-C.2: categoría 5e (1-100 MHz) especificaciones
de categoría 5 (1-100 MHz) especificaciones IEC 11801 2ª Ed .: ISO /

- **MEDIOS DE TRANSMISIÓN**

4 pares trenzado par trenzado no apantallado (UTP)

- **ESQUEMA DE CABLEADO**

ANSI / TIA / EIA-568-C.2: T568A y T568B

ISO / IEC 11801 2ª Ed .: 8 posiciones conductor / asignación par (1-2 / 3-6 / 4-5 / 7-8)

- **Maromas**

Construcción: 4 pares, 7 hilos de 26 AWG, UTP,
aislamiento de polietileno de alta densidad,

resistente al fuego de la chaqueta de compuesto de PVC

- **TAPÓN**

Asamblea: FCC Parte 68 Subparte F

Vivienda: la lista de UL 94V-0 de policarbonato retardante de llama

IDC cuchillas: bronce de fósforo, min 50 micropulgadas de chapado en oro de más de 100 micropulgadas de níquel en área de contacto

- **MECÁNICA**

Retención: 50 N (11 lbf) de 60 ± 5 s

la vida ciclo de apareamiento: min 750 ciclos

Resistencia a la tracción: ≥ 20 N por cable

- **ELÉCTRICA**

Resistencia de aislamiento: min 500 MOhm @ 100 Vcc

Tensión de ruptura dieléctrica: 1.000 V CC / CA pico del contacto @ 60 Hz durante 1 min

Resistencia de los contactos: máx 20 MOhm

- **CONDICIONES AMBIENTALES**

Almacenamiento: -40 ° C + 70 ° C (-40 ° F + 158 ° F)

de la operación: -10 ° C + 60 ° C (14 ° F + 140 ° F)

de humedad relativa (en funcionamiento): máx sin condensación 93%

- **CUMPLIMIENTO**

ANSI / TIA / EIA-568-C.2, ISO / IEC 11801 2ª Ed., FCC Part 68, UL 1863, UL94V-0

- **APLICACIONES**

X.21, V.11, S0, RDSI, CSMA / CD 10BASE-T, 100BASE-TX, 100BASE-T4, 100BASE-T2, 1000BASE-T, TR 4/16/100, 100BASE-VG, ATM LAN 25 / 51/155, TP-PMD

- **GARANTÍA**

5 años de garantía limitada de componentes
de 10 años de garantía Signamax Link / Canal

15-años de garantía Componente extendido Signamax

25 años de garantía Signamax Sistema de Cableado

Caja de registro sencilla

Precio: $39

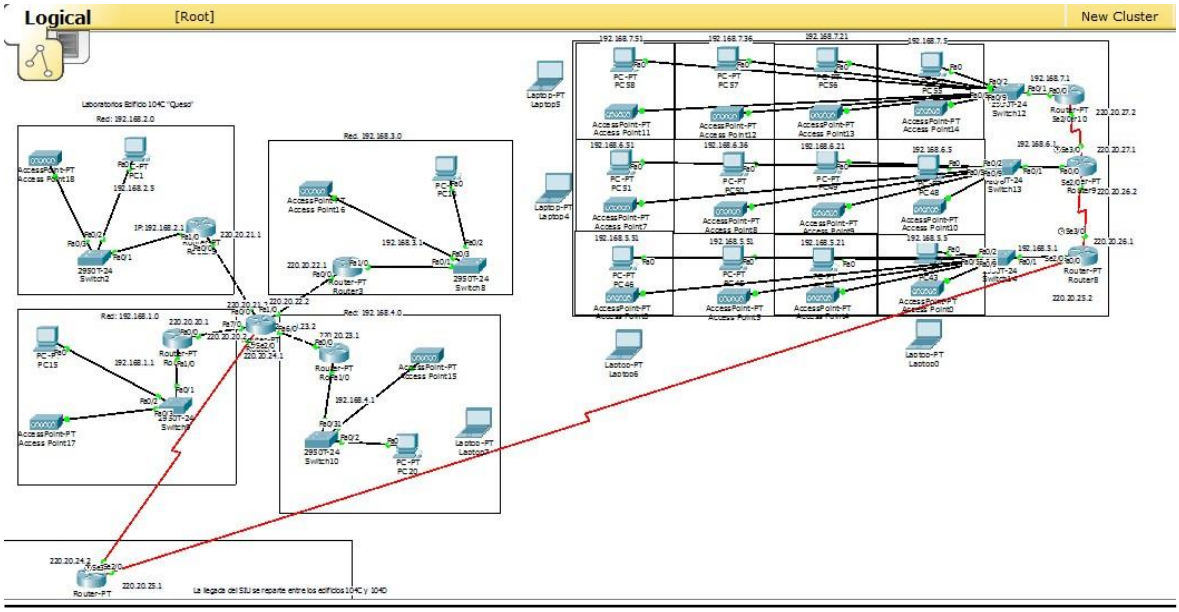
Canaleta autoadherible, de 2 m

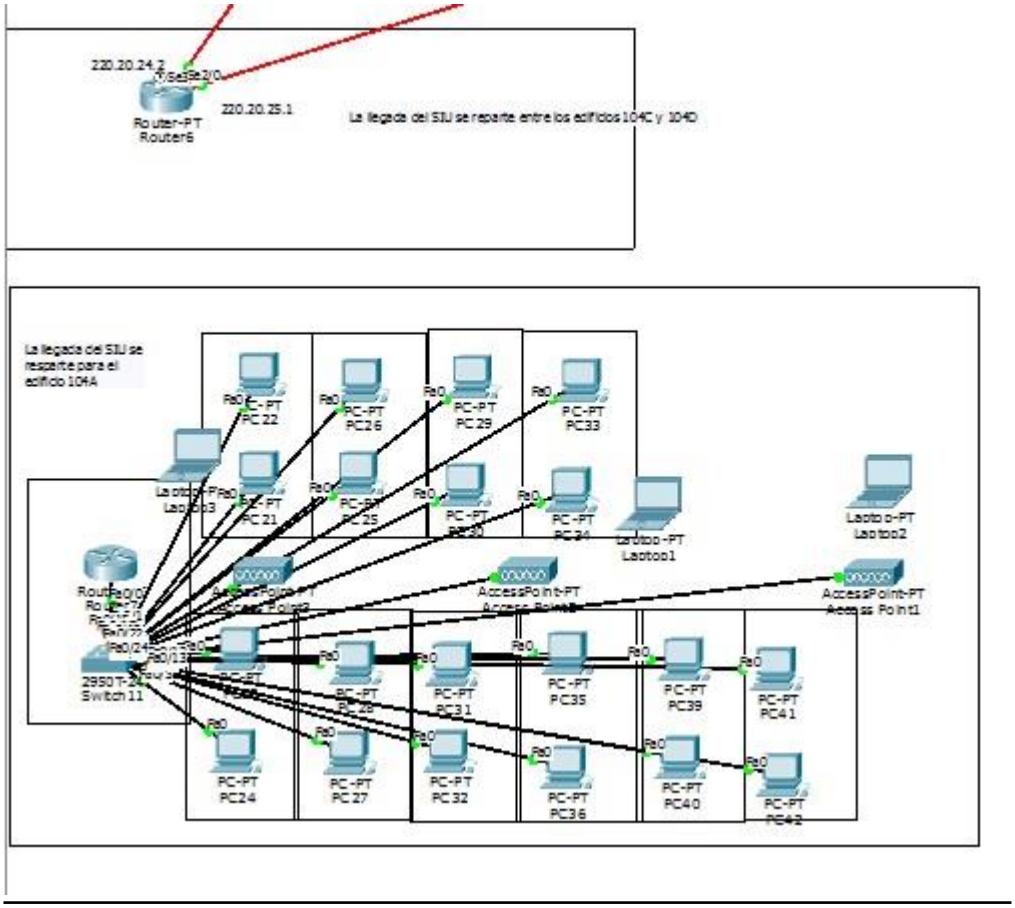
Precio: $59

Modelo: 370-401

Canaleta de PVC rígido antiflama, con adhesivo de alta calidad, de 2 m de largo, diseñada para proteger cableado o alambrado en instalaciones eléctricas, de voz o datos, en color gris.

Configuración Lógica:





Conclusiones:

El desarrollo de este proyecto nos permitió desarrollar habilidades y adquirir conocimientos que serán útiles para el diseño, análisis e implementación de una red. Nos permitió investigar los costos y características de los dispositivos disponibles en el mercado para así poder tomar una buena decisión sobre el material a ocupar ya que son factores fundamentales en la implementación de nuestra red para que esta pueda ser lo más eficaz posible.

Bibliografía:

http://redestelematicas.com/tipos-de-switches/

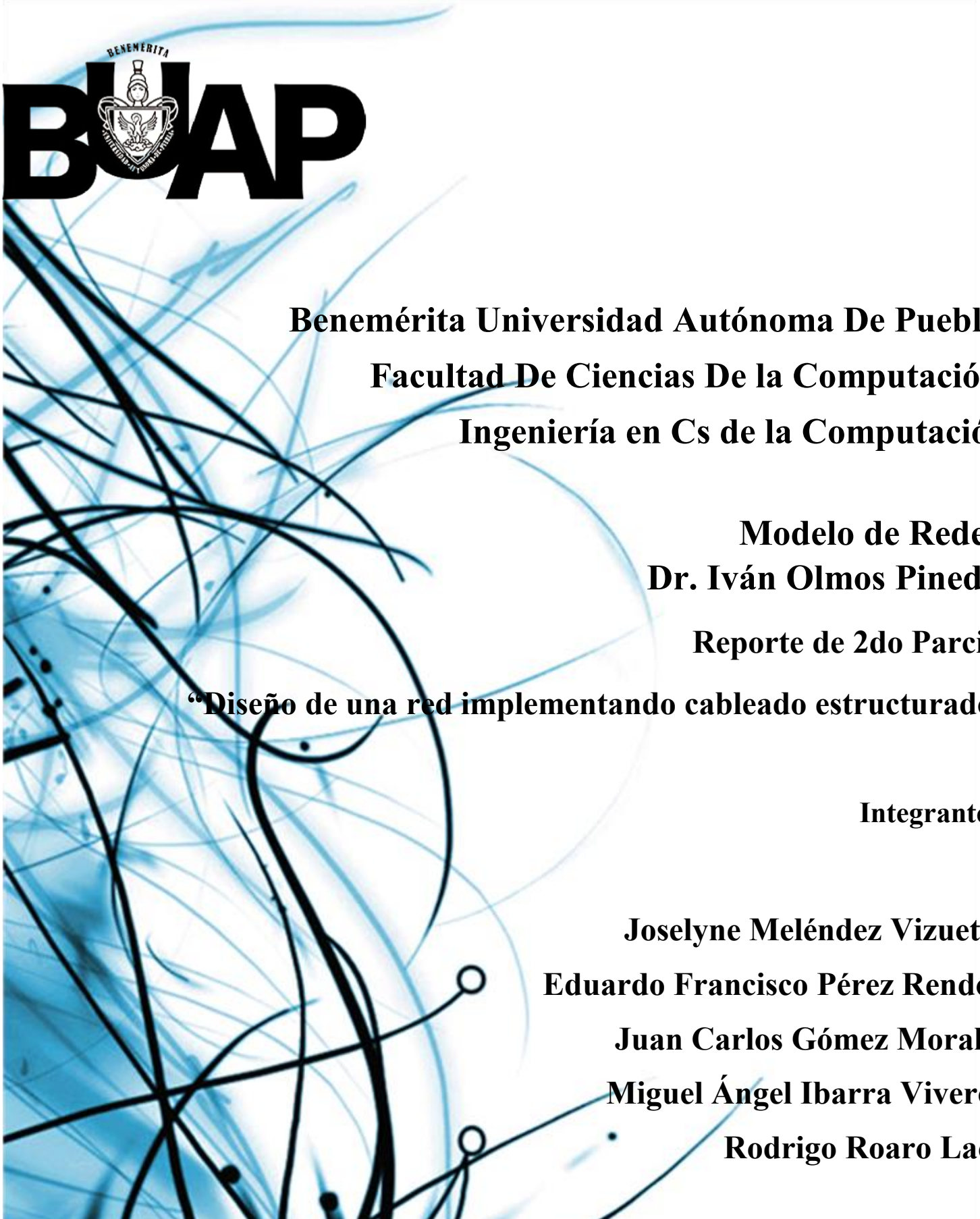
http://materias.fi.uba.ar/6679/apuntes/CABLEADO_ESTRUC.pdf

http://www.tp-link.es/products/details/?model=TL-R480T%2B

http://es.slideshare.net/eduardolov/estandares-568a-568b-569

http://www.signamax.com/patchcords/381?task=view

http://www.cisco.com/c/en/us/products/collateral/wireless/aironet-1130-ag-series/datasheet_c78-502793.html



Benemérita Universidad Autónoma De Puebla.

Facultad De Ciencias De la Computación.

Ingeniería en Cs de la Computación

Modelo de Redes.

Dr. Iván Olmos Pineda.

Reporte de 2do Parcial

“Diseño de una red implementando cableado estructurado”

Integrantes:

Joselyne Meléndez Vizuet

Eduardo Francisco Pérez Rendón

Juan Carlos Gómez Morales

Miguel Ángel Ibarra Viveros

Rodrigo Roaro Lack

ÍNDICE

INTRODUCCIÓN

Realizar el diseño e implementar una red en la Facultad de Ciencias de la Computación mediante cableado estructurado para proporcionar una alta fiabilidad al contar con diferentes alternativas de suministro y hacer que toda la infraestructura esté disponible sin importar la localización física del recurso y del usuario.

“De un buen cableado depende el buen desempeño de una red”.

MARCO TEÓRICO

- **¿Que es un cableado estructurado?**

Es el medio físico a través del cual se interconectan dispositivos de tecnologías de información para formar una red, y el concepto estructurado lo definen los siguientes puntos:

- ❑ Solución Segura: El cableado se encuentra instalado de tal manera que los usuarios del mismo tienen la facilidad de acceso a lo que deben de tener y el resto del cableado se encuentra perfectamente protegido.
- ❑ Solución Longeva: Cuando se instala un cableado estructurado se convierte en parte del edificio, así como lo es la instalación eléctrica, por tanto ese tiene que ser igual de funcional que los demás servicios del edificio. La gran mayoría de los cableados estructurados pueden dar servicio por un periodo de hasta 20 años, no importando los avances tecnológicos en las computadoras.
- ❑ Modularidad: Capacidad de integrar varias tecnologías sobre el mismo cableado voz, datos, video. Fácil Administración: El cableado estructurado se divide en partes manejables que permiten hacerlo confiable y perfectamente administrable, pudiendo así detectar fallas y repararlas fácilmente.

- **¿Qué tipos de cableado estructurado existen?**

Los cableados estructurados se dividen por categorías y por tipo de materiales que se utilizan. La categoría en la que se dio a conocer el cableado estructurado es 5, pero al día de hoy existen categorías superiores, Categoría 5 mejorada “5e” y categoría 6, estas se miden en función de su máxima capacidad de transmisión, a continuación se presenta una tabla con el detalle de las categorías disponibles, su velocidad de transmisión, las topologías que pueden soportar en esa velocidad de transmisión y el tipo de materiales que se requieren para integrarla.

Categoria Obtenida	Topologias soportadas	Velocidad Max. de Transferencia	Distancias Maximas entre Repetidores por norma. Ver Gráfica Anexa	Requerimeintos Mínimos de materiales Posibles a Utilizar	Status
Cat. 3	Voz (Telefonía) Arcnet - 2 Mbits. Ethernet - 10 Mbits.	10 Mbits.	100 Mts.	Cable y conectores Coaxiales o cable y conectores UTP de menos de 100 Mhz.	Obsoleto
Cat. 5	Inferiores y Fast Ethernet	100 Mbits.	90 Mts. + 10 mts. En Patch Cords	Cable UTP y conectores Categoría 5 de 100 - 150 Mhz.	Sujeta a Descontinuarse
Cat. 5e	Inferiores y ATM	165 Mbits.	90 Mts. + 10 mts. En Patch Cords	Cable UTP / FTP y conectores Categoría 5e de 150 - 350 Mhz.	Actual
Cat. 6	Inferiores y Gigabit Ethernet	1000 Mbits.	90 Mts. + 10 mts. En Patch Cords, Con cable de cobre Cat. 6. 1 Km. En Fibra Multimodo 2 Km. En Fibra Monomodo	Cable de cobre y conectores Categoría 6 y/o Fibra Optica.	Punta Técnológica

➢ ¿Cuáles son las partes que integran un cableado estructurado?

1. **Área de trabajo**: Su nombre lo dice todo, Es el lugar donde se encuentran el personal
2. trabajando con las computadoras, impresoras, etc. En este lugar se instalan los servicios (nodos de datos, telefonía, energía eléctrica, etc.)
3. **Closet de comunicaciones**: Es el punto donde se concentran todas las conexiones que se necesitan en el área de trabajo.
4. C**ableado Horizontal:** es aquel que viaja desde el área de trabajo hasta el closet de comunicaciones.
5. **Closet de Equipo**: En este cuarto se concentran los servidores de la red, el conmutador telefónico, etc. Este puede ser el mismo espacio físico que el del closet de comunicaciones y de igual forma debe ser de acceso restringido.
6. Instalaciones de Entrada (Acometida) – Es el punto donde entran los servicios al edificio y se les realiza una adaptación para unirlos al edificio y hacerlos llegar a los diferentes lugares del edificio en su parte interior. (no necesariamente tienen que ser datos pueden ser las líneas telefónicas, o Backbone que venga de otro edificio, etc.)
7. Cableado Vertebral (Backbone) – Es el medio físico que une 2 redes entre sí.

➢ ¿Cuáles son las ventajas de contar con un cableado estructurado debidamente instalado?

Un sistema de cableado estructurado tiene la ventaja de ser un diseño de arquitectura abierta, es decir, es independiente de la información que se trasmite a través de él. También es confiable porque está diseñado con una topología de estrella, la que en caso de un daño o desconexión, se limitan sólo a la parte o sección dañada, y no afecta al resto de la red. En los sistemas antiguos, basados en bus Ethernet, cuando se producía una caída, toda la red quedaba inoperante.
Otra ventaja es la facilidad de usar un solo tipo de cable para todos los servicios de telecomunicación actuales y futuros lo cual da como resultado que se gasten recursos en una sola estructura de cableado, y no en varias (como en los edificios con cableado convencional).

En casos de actualización o cambios en los sistemas empresariales, no se cambian todos los cables de la estructura del edificio. Se evita romper paredes para cambiar circuitos o cables, lo que además, provoca cierres temporales o incomodidades en el lugar de trabajo. Un sistema de cableado estructurado permite mover personal de un lugar a otro, o agregar servicios a ser transportados por la red sin la necesidad de incurrir en altos costos de re-cableado.

Un punto que al principio pareciera no favorecer el decidirse a instalar un sistema de cableado estructurado es el elevado costo de una instalación completa que permite evitar los cambios en la medida de lo posible. Pero a largo plazo la inversión da fruto al no tener que invertir en rediseño o re-cableado.

➢ **Un sistema de cableado estructurado debe caracterizarse por ser:**

- **Confiable**, es decir, no deben existir interrupciones o caídas continuas de la red que esté conectada a través de el, además de tener un mínimo de problemas de diafonía y atenuación de señal entre otros problemas, tendiendo a desaparecerlos.
- **Flexible**, para permitir la fácil reubicación de los servicios y de los mismos usuarios, así como de diversos servicios que en ocasiones son novedades tecnológicas.
- **Modular**, en el sentido de poder configurarse fácilmente según las necesidades de la empresa.
- **Integrador de sistemas**, puesto que en el mismo cableado se tienen diversos servicios de telecomunicación.
- **Sencillo de administrar**, para poder tener un control lógico y eficiente de los servicios de telecomunicación.

➢ ***Reglas de cableado estructurado***

Las tres reglas siguientes ayudan a asegurar que los proyectos de diseño de cableado estructurado sean a la vez efectivos y eficaces:

1. **Buscar una solución de conectividad completa**. Una solución óptima para la conectividad de red incluye todos los sistemas diseñados para conectar, enrutar administrar e identificar los sistemas de cableado estructurado. Una implementación basada en normas ayudará a asegurar que pueden soportarse tanto las tecnologías actuales como las futuras. Seguir normas asegura que el proyecto tenga rendimiento y fiabilidad a largo plazo.
2. **Plan para crecimiento a futuro**. El número de circuitos instalados debería cumplir también los requisitos futuros. Deberían considerarse cuando sean posibles las categorías 5e y 6, así como las soluciones de fibra óptica, para asegurarse de que se cumplan las necesidades futuras. Debe ser posible planificar una instalación de capa física que funcione diez años o más.
3. **Mantener la libertad de elección de los distribuidores**. Aún cuando un sistema patentado y cerrado puede ser menos caro inicialmente, puede terminar siendo mucho más costoso a largo
4. plazo. Un sistema no estándar a partir de un solo distribuidor puede hacer más difícil efectuar movimientos y cambios con posterioridad.

➢ ***Consideraciones de Seguridad***

La primera consideración para el diseño de las infraestructuras de cableado estructurado es relativa a la seguridad del personal y de los sistemas respecto de:

- ❖ El tendido eléctrico y el consiguiente peligro de descarga.
- ❖ Medidas de seguridad de las modificaciones que se puedan realizar en la estructura del edificio.

- ❖ Comportamiento del sistema de cableado en caso de incendio.
- ❖ Respecto a este punto hay que considerar que los cables empleados usan distintos tipos de plásticos en su construcción. Los materiales plásticos empleados deben generar poco humo en caso de incendios, no producir vapores tóxicos o corrosivos y no favorecer la propagación del fuego.

Por consiguiente los sistemas de cableado estructurado deben seguir las normas específicas en materia de seguridad.

➢ *Consideraciones técnicas*

El sistema de cableado deberá ser conforme con la normas dictadas por EIA/TIA. La aplicación de estas determina que el sistema de cableado ha de ser estructurado y emplear en cada uno de los subsistemas los tipos de cables autorizados por la norma.

La instalación se realizará de acuerdo a las especificaciones de un proyecto de cableado el cual contendrá: Memoria, Planos, Pliego de Prescripciones Técnicas y Presupuesto.
Así mismo dentro del proyecto se indicarán con claridad los siguientes aspectos:

- Número de puestos en cada área
- Número de tomas por puesto
- Posición y tipo de toma
- Detalle del tipo de cables y conectores utilizado en las tomas
- Espacios que hay que reservar para la instalación de los repartidores, incluyendo acceso y mantenimiento.
- Tipo de aplicaciones que puede soportar cada toma.

➢ ¿Qué es un Switch?

Un switch es un dispositivo de propósito especial diseñado para resolver problemas de rendimiento en la red, debido a anchos de banda pequeños y embotellamientos. El switch puede agregar mayor ancho de banda, acelerar la salida de paquetes, reducir tiempo de espera y bajar el costo por puerto.
Opera en la capa 2 del modelo OSI y reenvía los paquetes en base a la dirección MAC.
El switch segmenta económicamente la red dentro de pequeños dominios de colisiones, obteniendo un alto porcentaje de ancho de banda para cada estación final. No están diseñados con el propósito principal de un control íntimo sobre la red o como la fuente última de seguridad, redundancia o manejo.
Al segmentar la red en pequeños dominios de colisión, reduce o casi elimina que cada estación compita por el medio, dando a cada una de ellas un ancho de banda comparativamente mayor.

- **¿Qué es un Router?**

Un router es un dispositivo de red que como su propio nombre indica se encarga de llevar por la ruta adecuada el tráfico. En tu casa seguramente tendrás uno de estos que es el que te conecta con la red Internet.

Los routers funcionan utilizando direcciones IP para saber a donde tienen que ir los paquetes de datos no como ocurre en los switches. Gracias a estas direcciones, que son únicas para cada máquina, este dispositivo puede conocer por dónde debe enviar el paquete.

- **¿Qué es un Access Point?**

Un punto de acceso inalámbrico (WAP o AP por sus siglas en inglés: Wireless Access Point) enredes de computadoras es un dispositivo que interconecta dispositivos de comunicación inalámbrica para formar una red inalámbrica. Normalmente un WAP también puede conectarse a una red cableada, y puede transmitir datos entre los dispositivos conectados a la red cable y los dispositivos inalámbricos. Muchos WAPs pueden conectarse entre sí para formar una red aún mayor, permitiendo realizar "roaming". (Por otro lado, una red donde los dispositivos cliente se administran a sí mismos - sin la necesidad de un punto de acceso - se convierten en una red ad-hoc). Los puntos de acceso inalámbricos tienen direcciones IP

asignadas, para poder ser configurados.
Son los encargados de crear la red, están siempre a la espera de nuevos clientes a los que dar servicios. El punto de acceso recibe la información, la almacena y la transmite entre la WLAN(Wireless LAN) y la LAN cableada.
Un único punto de acceso puede soportar un pequeño grupo de usuarios y puede funcionar en un rango de al menos treinta metros y hasta varios cientos. Este o su antena son normalmente colocados en alto pero podría colocarse en cualquier lugar en que se obtenga la cobertura de radio deseada.
El usuario final accede a la red WLAN a través de adaptadores. Estos proporcionan una interfaz entre el sistema de operación de red del cliente (NOS: Network OperatingSystem) y las ondas, mediante una antena inalambrica.

- **¿Qué es la fibra óptica?**

La fibra óptica designa una nueva red fija que se apoya en un soporte físico muy delgado (fibra de vidrio o de

plástico) utilizado para la transmisión de datos IP (por internet) a alta velocidad.

La fibra óptica posee un núcleo de material transparente en el seno del cual la luz "rebota", quedando atrapada en el cable. Así los datos, que corresponden a impulsos luminosos muy cortos, viajan a la velocidad de la luz (o casi, porque la velocidad de la luz en la fibra óptica siempre será menos elevada que la verdadera velocidad de la luz que es medida en el vacío).

Actualmente, la fibra óptica asegura una velocidad (transmisión de datos por internet) que llega hasta los 100 MB/s y multiplica así por 10 las realizaciones de una red ADSL clásica. De ahora en adelante contemplamos velocidades que van hasta varios TB/s. Pero el problema vendrá de nuestros ordenadores que no sabrán tratar bastante rápido tal velocidad de datos.

Recordemos también que el Wi-Fi retiene estos rendimientos. Las normas Wi-Fi actuales (802.11a o 802.11b) permiten sólo una velocidad teórica máximo de 54 MB/s que es inferior a la velocidad de la fibra. La norma en curso de expansión (802.11n) permite velocidades mucho más elevadas (hasta 600 MB/s teóricos). Hasta entonces, si deseas explotar tu fibra como máximo, conéctate a Ethernet.

DESARROLLO

A).-Modelo a Implementar

Se reciben 2 cables de fibra optica con 1GB, uno en el edificio 104A y el otro se recibe en el edificio 104B. En el edificio 104A se pretende colocar un cluster en la parte trasera del edificio. El cluster del edificio 104A consta de lo siguiente:
- Un router de 8 pines que recibira el cable y proveera internet.
- 3 switches que tendran como ISP el router y otorgaran red a todo el edificio 104A por medio de conexiones de cable UTP, se otorgara internet a los 36 usuarios que se encuentran alli. Ademas se peretende poner un access point para poder otorgar internet a todo aquel que se encuentre cerca del edificio.

En el edificio 104B tambien se pretende colocar un cluster, que estara compuesto por lo siguiente:
- Un router de 8 pines que recibira el cable y sera el provedor de internet
- 1 switche de 16 pines que tendra como ISP el router ocupado en este mismo cluster, y este repartira fibra optica a los edificios 104D y 104C, ademas este switch se conectara al router del edificio 104A y viceversa (el switch del 104A se conectara al router del 104B), esto es para que se obtenga una division de cargas y no ocurra un embotellamiento de datos.

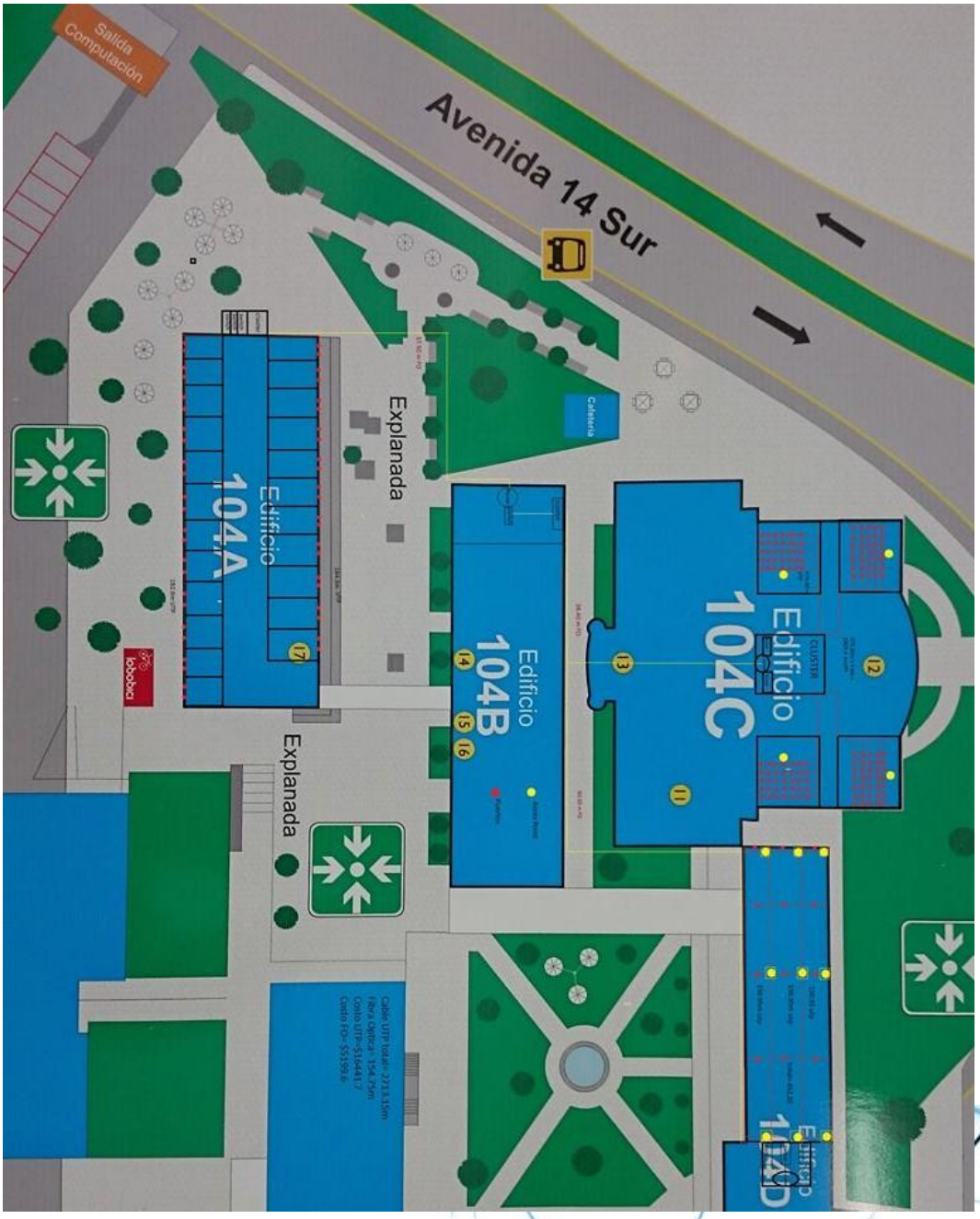
En el edificio 104C se colocara un cluster central con lo siguiente:
- 1 router de 8 pines
- 2 switches de 16 bits para la conexion de todos los laboratorios
- 4 access point (1 para cada laboratorio)

Ademas del cluster central, se colocara 1 cluster por cada laboratorio, para poder otorgar internet a cada computadora (35 computadoras por laboratorio) conformadas por lo siguiente:
- 2 switches de 24 pins
- Ademas de cable UTP para otorgar internet a todas las maquinas

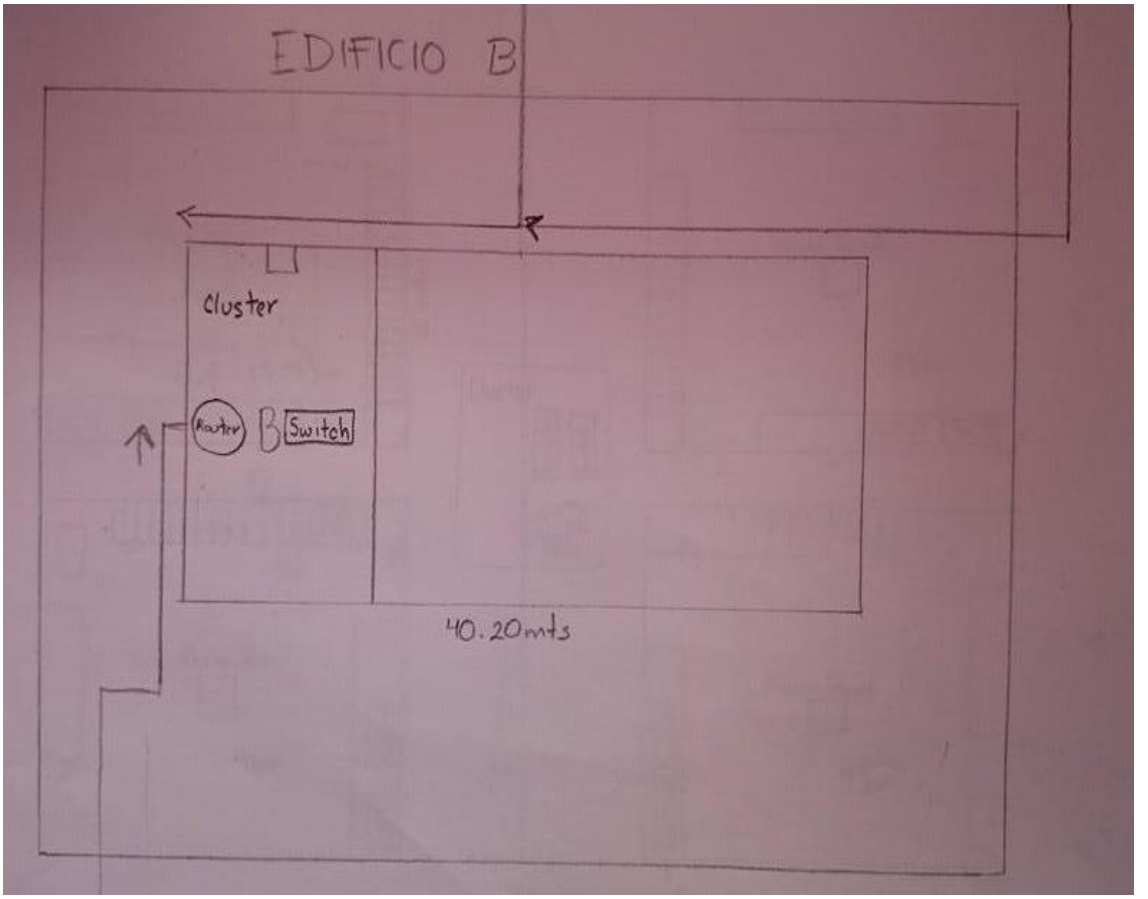
En el edificio 104D tambien se colocara un cluster enmedio del edificio conformado por lo siguiente:
- 1 router de 8 pines que recibira el cable que viene del edificio 104B
- 3 switches de 16 pins (1 por cada piso)
- 9 access point (3 para cada piso)
- Cable UTP para poder conectar el cluster con todos los access point

Ademas los edificios 104C y 104D tambien estaran interconectados mediante los routers, para poder dividir las cargas (ya que el edificio 104C tendra la mayor carga), ademas estos 2 estaran conectados al edificio 104B y este podra dividir cargas con el edificio 104A y asi evitar un embotellamiento de datos.

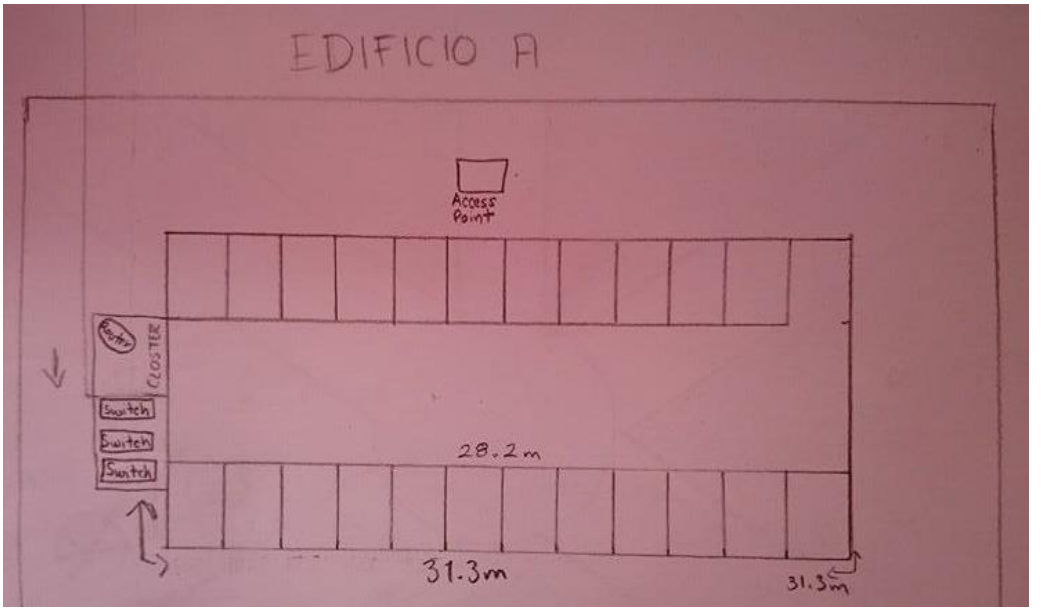
B).-Diseño de Planos


Salida Computación
Avenida 14 Sur
Cafetería
Explanada
Edificio 104A
Edificio 104B
Edificio 104C
Explanada
Edificio 104D
lobobici

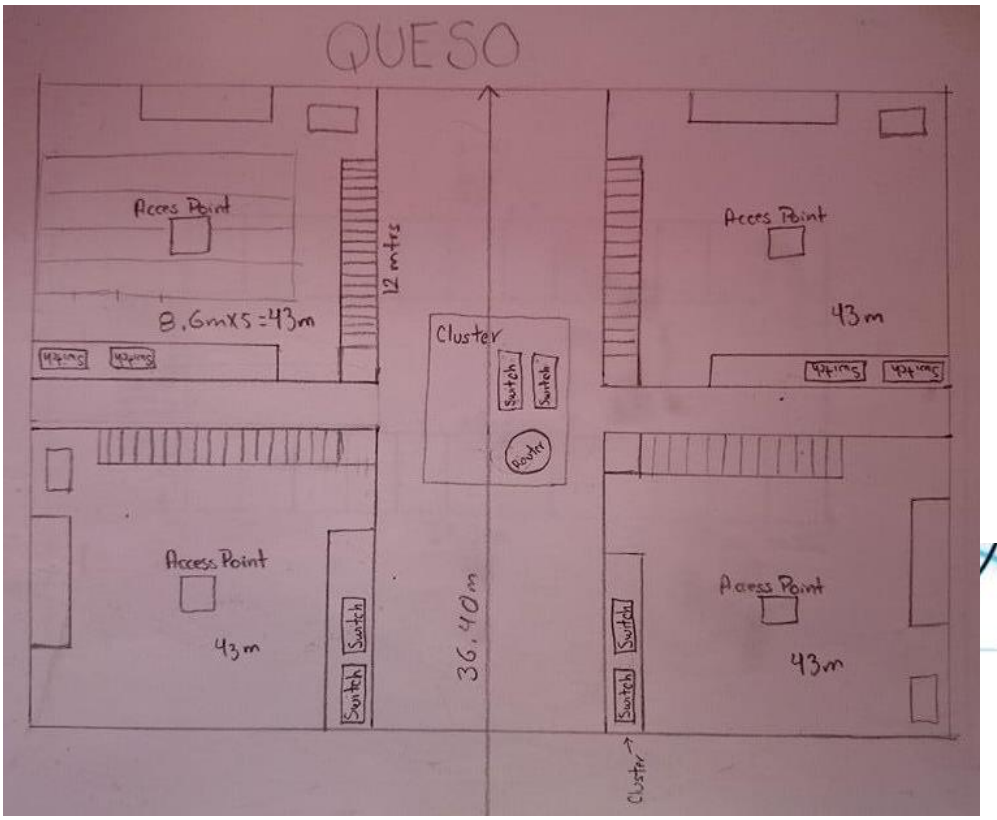
Edificio B


EDIFICIO B
Cluster
Router
B
Switch
40.20mts

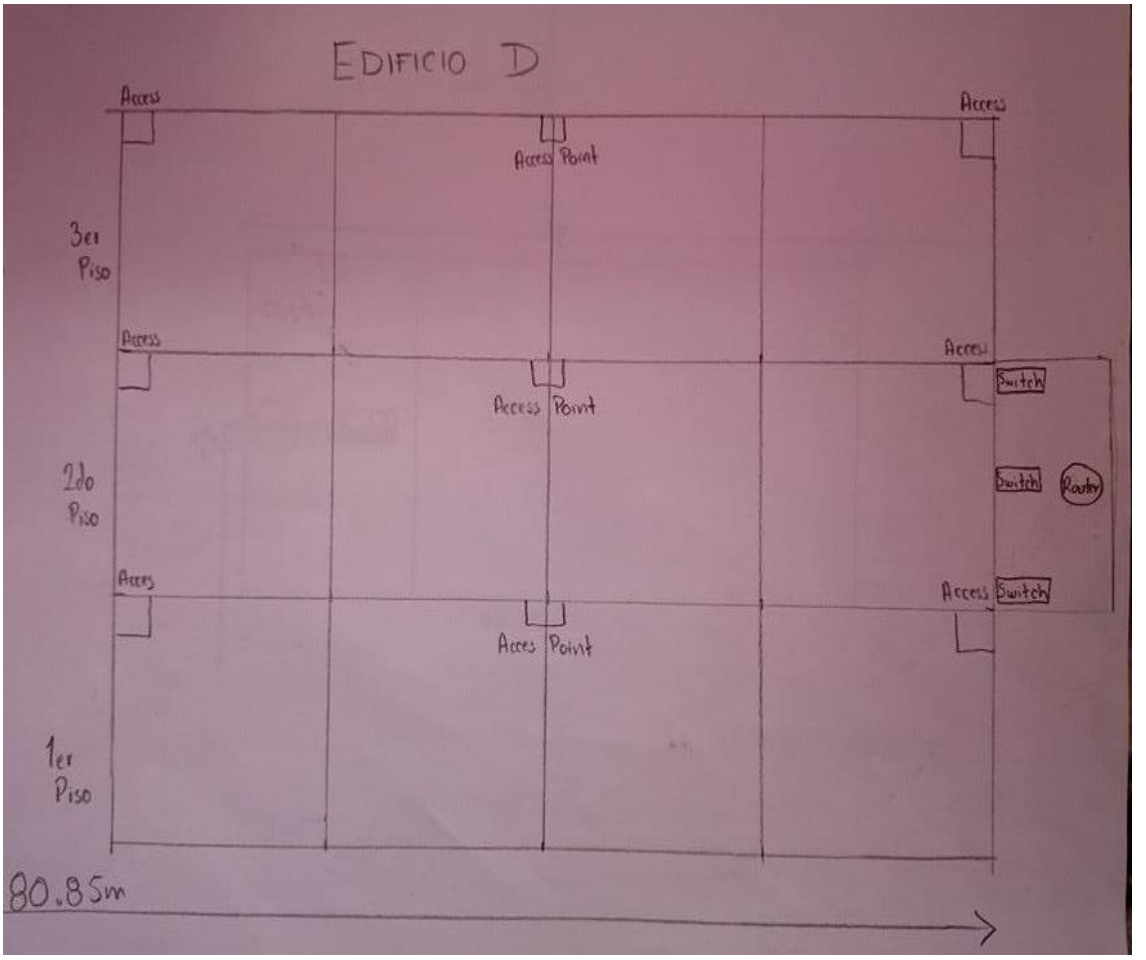
Edificio A



EDIFICIO C



EDIFICIO D



C) LISTA DE MATERIALES

Edificio A:

- ❑ **1 router 8 pines(cluster)**
- ❑ **1 access point**
- ❑ **2 switches 24 pines**
- ❑ **1 switch 16 pines**

Edificio B:

- ★ 1 router de 8 pines(cluster,cuarto equipo)
- ★ 1 switch de 16 pines
- ★ fibra óptica

Edificio C (Queso):

- 8 switches(2 cada Lab) de 24 pines
- 2 switches de 16 bits(cluster)
- 4 router de 8 pines(cluster)
- 4 accesspoint
- cable Utp

Edificio D:

- ❖ 1 router de 8 pines(cluster)
- ❖ 3 switches 16 pines(cluster)
- ❖ 9 access point
- ❖ Cable Utp

D) Cotización

Tipo de materiales:

- Switch Gigabit Cisco de 24 puertos c/100/1000 Mbps ,Qos=$2,989 c/uno

- Switch Tp Link 16 puertos 10/100/1000 Mbps= $1,229 c/uno

- **Ruteador SMC de 8 Puertos 10/100Mbps con Servidor de Impresión, Puertos RS232 ADSL y WAN Para compartir Internet. (Diseño para Rack)=$1,411.17 c/uno**

- Access Point LINKSYS WAP300N de doble banda 2.4 Ghz y 5 Ghz=$1,199 c/uno

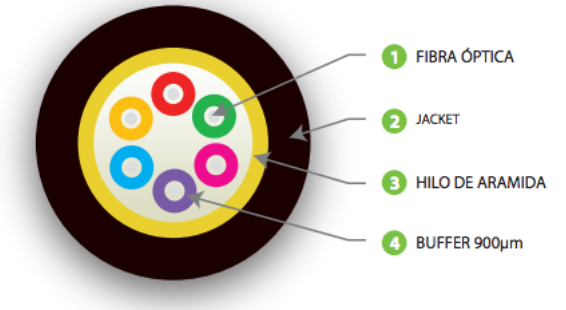
- Bobina de cable Catse(UTP) caja con 305m,gris,24 AWG,sólido,Reforzado(evita la deformación)=$1,848 c/una Precio p/metro=$6.06

- **Plugs RJ-45 Intellinet, Cat 5e, UTP, 2 Puntas para Cable Multifilar, bote con 100 piezas = $249.00 c/una**

Cable de Fibra Optica de 6 hilos Multimodo y Monomodo=

2.80 USD (6 fibras multimodo)

0.90 USD (6 fibras monomodo)



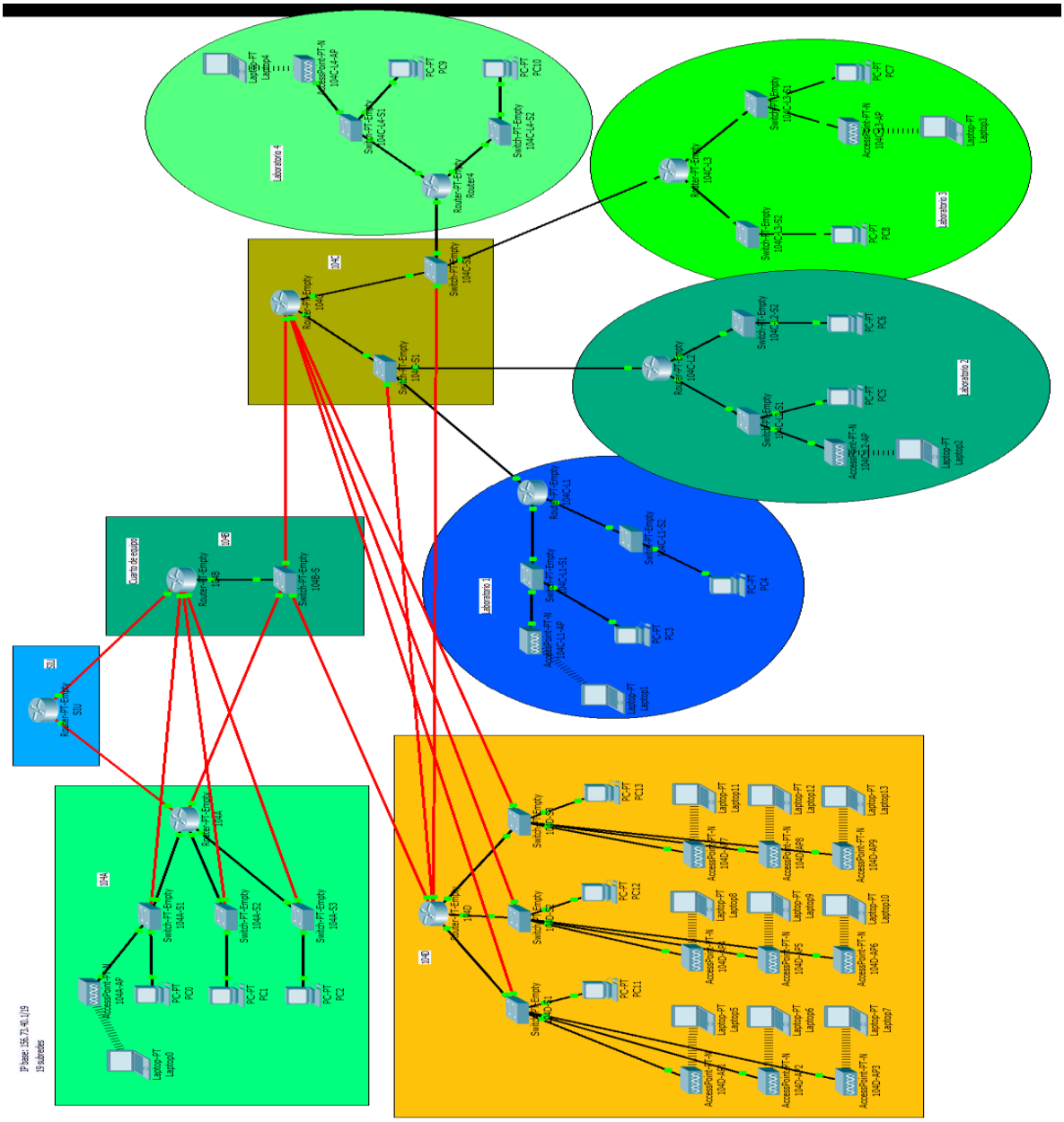
- **42U Standard 19" Vented Rear Door Glass Front Door = 1,999.00 USD c/una**

- Minisplit LG Jet Cool 22,000 BTU's = $ 10,990.00

EDIFICIO QUESO			
Cantidad	Descripcion	Precio Unita	Precio total
8	Switches de 24 pines(gigabit Cisco) 10/100/1000 Mbps	$2,989	$23,912
2	Switches de 16 bits(Cluster) TP_link 10/100/1000 Mbps	$1,229	$2,458
1	Router de 8 pines (Cluster)Ruteador SMC de 8 Puertos 10/100Mbps	$1,411.17	$1,411.17
4	Access Point LINKSYS WAP300N de doble banda 2.4 Ghz y 5 Ghz	$1,199	$4,796
EDIFICIO B			
Cantidad	Descripcion	Precio Unita	Precio total
1	Router de 8 pines (Cluster)Ruteador SMC de 8 Puertos 10/100Mbps	$1,411.17	$1,411.17
1	Switch de 16 bits(Cluster) TP_link 10/100/1000 Mbps	$1,229	$1,229
EDIFICIO D			
Cantidad	Descripcion	Precio Unita	Precio total
1	Router de 8 pines (Cluster)Ruteador SMC de 8 Puertos 10/100Mbps	$1,411.17	$1,411.17
3	Switches de 16 bits(Cluster) TP_link 10/100/1000 Mbps	$1,229	$3,687
9	Access Point LINKSYS WAP300N de doble banda 2.4 Ghz y 5 Ghz	$1,199	$10,791
EDIFICIO A			
Cantidad	Descripcion	Precio Unita	Precio total
1	Router de 8 pines (Cluster)Ruteador SMC de 8 Puertos 10/100Mbps	$1,411.17	$1,411.17
1	Access Point LINKSYS WAP300N de doble banda 2.4 Ghz y 5 Ghz	$1,199	$1,199
2	Switches de 24 pines(gigabit Cisco) 10/100/1000 Mbps	$2,989	$5,978
1	Switches de 16 bits(Cluster) TP_link 10/100/1000 Mbps	$1,229	$1,229

Cable utp	2713.15m
Fibra optica	154.75m
Costo UTP	$16,441.70
Fibra optica	$5,199.60

E) IMPLEMENTACIÓN VIRTUAL



CONCLUSIÓN

El desarrollo del anterior proyecto ha permitido adquirir conocimientos de vital importancia, que mas tarde serán utiles cuando se requiera, analizar, diseñar, administrar e implementar una red mediante un diseño de cableado estructurado.

Los costos de equipos y partes, la disponibilidad de instalaciones, el uso que se pretenda dar a la red en cuanto a grado de eficiencia, son factores fundamentales que deben de considerarse al momento de diseñar la implementación de una red determinada.

Cabe recordar que siempre hay ciertos limitantes en la infraestructura donde se requiere implementar una red. Es necesario conocer con precisión la reglamentación existente, por lo cual se tiene que realizar un diseño en base a los planos teniendo en cuenta las situaciones que se presenten y puedan perjudicar el desarrollo del mismo.

BIBLIOGRAFÍA

• [Torres, 2001] F. Torres, F.A. Candelas, S.T. Puente, "*Sistemas para la Transmisión de Datos*". 2º Edición. Publicaciones Universidad de Alicante, Alicante, 2001.

• [Berná, 2002] J.A. Berná, M. Pérez, L.M. Cr3espo, "*Redes de Computadores para Ingenieros en Informática*". Publicaciones Universidad de Alicante, Alicante, 2002.

- **Comer, Douglas. TCP/IP: Redes globales de información con Internet y TCP/IP. Prentice Hall. México. 1996.**

- **"Interconnections: Bridges, Routers, Switches and Internetworking Protocols" – Perlman, Radia.– Addison-Wesley Professional Computing Series – 1991.**

"Switching Technology in the Local Network" – Hein M.And Griffiths D. – International Thomson Publishing Inc –1997.

Brianda Yareli Cruz Guerrero

201114701

Primer examen

Modelos de Redes

25 de Septiembre 2014

Introducción

En el siguiente documento se hace una breve explicación de cómo se realizó el software para calcular la Latencia. Pero antes daremos un breve resumen de que es la latencia.

Latencia:

La latencia depende de 3 factores:

- Tiempo de propagación.
- Máxima cantidad de datos que pueden ser transmitidos
- Tiempos de espera

Para calcular la latencia tenemos una formula.

Latencia=Tiempo de propagación+ Tiempo de transmisión +Tiempo de cola

Estos datos se calculan de la siguiente forma:

Tiempo de Propagación = Distancia a recorrer/velocidad de la luz

Velocidad de la luz= 3x10^8

Tiempo de transmisión=Tamaño de paquete/Tasa de transferencia técnica

Ocupamos la latencia para calcular cuánto tarde en llegar un paquete a su destino.

La realización del software tenía tres problemas principales

1.- lectura del archivo con un formato especifico

2.- Recorrido del grafo encontrando todos los posibles paths

3.- la división de paquetes y cálculo de latencias.

El punto número uno no fue el punto más complicado simplemente se implementó mediante Tokens separando el carácter que no se quería leer que en este caso fue la coma (,), se leyó la primera línea del archivo la cual tenía como datos dos de los más importantes en primero lugar el número de nodos y en segundo lugar el número de aristas, son dos de los datos más importantes porque la tener el número de aristas puedes implementar un ciclo de repetición para leer las conexiones que tiene el grafo dado.

Después de implementar el for (sentencia de repetición)y leer las conexiones se lee, el tiempo de cola el grafo inicial y final otros datos importantes pues se usan para encontrar los paths y se lee el tamaño del archivo, por lo cual pienso que este punto fue el menos complicado.

Pasando al paso numero dos se incrementó el nivel de complejidad, primero se intentó realizar un algoritmo de búsqueda en profundidad pero se realizó de manera iterativa, si bien era una solución , no era una solución funcional pues, crecía exponencialmente el número de líneas del código, se intentó después de leer implementar el algoritmo de Dijktra, como es bien sabido el algoritmo de Dijktra Encuentra el camino menos costoso, en este caso en particular no teníamos peso por lo cual le mandamos peso “1” a todas las artistas para así garantizar que encontrara todos los posibles paths, nos dimos cuenta que el código se había que modificar quitándole la condición de paro, pero al momento en el que se hizo el programa entro en un bucle y no hubo manera de pararlo, por éste motivo se tomó la decisión de implementa de nuevo el algoritmo de

búsqueda en profundidad pero esta vez utilizando la técnica de Back-tracking para optimizar la aplicación y hacer menos líneas de código, se consiguió un algoritmo en pseudo código el cual muestro a continuación.

método DFS(origen,final):

marcamos origen como visitado

recuperar el path si se llego a final

para cada vertice v adyacente a origen en el Grafo:

si v no ah sido visitado:

marcamos como visitado v

llamamos recursivamente DFS(v)

marcamos origen como no visitado

Se tomó el algoritmo y se modificó para satisfacer las exigencias del programa, fue así como se resolvió el problema los paths

El tercer punto y el mas difícil bajo un punto de vista personal fue el de dividir los paquetes, el principal problema al que me enfrente fue el de manipular correctamente los datos como tiempo de cola tamaño total datos de control, se tenía también que sacar los datos d usuario lo cual fue fácil pues del archivo lees el total y los datos de control así los datos de usuario se obtiene de una manera muy fácil con una operación básica, DU=TT-DC donde DU= datos de usuario TT= tamaño total y DC= datos de control,

Para sacar el número de paquetes se intentó primero declarar un arreglo donde se fueran guardando los datos de usuario para después poder comparar las posiciones y hacer uso de una lista ligadas de lista ligadas (Lista de listas), se hizo una función para calcular los datos de usuario y se guardaron en el arreglo, seguido se hizo una función para calcular el nuero de paquetes, al hacer pruebas de escritorio, el algoritmo parecía funcionar pero al llegar a la implementación, no funciono como se esperaba.

Los siguientes puntos, calcular la latencia no los realice.

CONCLUSION

No se termino el software, pues hubo problemas desde el inicio con tiempo y procedimientos, se realizo un 30% del trabajo solicitado con satisfacción.

Fuentes

https://jariasf.wordpress.com/tag/backtracking/

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

MODELO DE REDES

DR. IVÁN OLMOS PINEDA

BRIAN MENESES CASARRUBIAS

“DOCUMENTO TÉCNICO
PRIMER EXAMEN PARCIAL-LATENCIA.”

26 DE SEPTIEMBRE DEL 2014

INTRODUCCIÓN

El trabajo es acerca del Primer Examen Parcial, éste consistió en que dado una topología de red (grafo), tengo que calcular el tiempo total de transmisión de un Emisor hasta un Receptor, considerando un paquete de tantos GB.

Para esto primero tuve que haber calculado todos los caminos posibles donde puede viajar ese paquete, teniendo todos los caminos, calcular la latencia de cada camino. Luego calcular cuál es la menor latencia, ya teniendo cual es el camino con menor latencia, calcular el tiempo total de transmisión.

CONCEPTOS DESARRROLLADOS

Latencia:

TP (Tiempo de Propagación) + TT (Tiempo de Transmisión) + TC(Tiempo de Cola).

Tiempo de Propagación: TP=distancia a recorrer / velocidad de la luz (300, 000,000 m/s).

Tiempo de Transmisión: TT= tamaño del paquete en bytes / tasa de transferencia teórica.

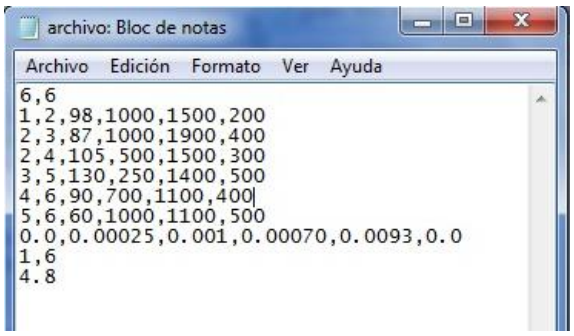
Tiempo de Cola: TC=es el tiempo que se tarda un switch en procesar ese paquete y enviarlo a otro destino, ese depende del tipo de aparato que utilicemos.

DESARROLLO

El proyecto consistía en implementar en algún lenguaje de programación (C, C++, Java) el proceso que implica que dado un grafo de red, obtener la latencia mínima de un camino de todos los posibles, y con esa latencia calcular el tiempo de transmisión total.

Yo lo implemente en Java. Mi entrada de datos era un archivo con extensión .txt en donde contenían todos los datos de la red por renglones, cada dato separado por comas; primero iban el número de vértices, número de enlaces, en los siguientes renglones iban todas las adyacencias de cada nodo con sus respectivos datos como: distancia, velocidad, tamaño del paquete, datos de control, así hasta acabar con todas las adyacencias, luego los tiempos de cola de cada nodo, enseguida vértice inicial y vértice final, y por último el tamaño del archivo que se quiere transmitir por la red.

Como salida de datos eran: el camino con menor latencia, la latencia de ese camino, total de paquetes y el tiempo total de transmisión.



Primero comencé con leer el archivo con todos los datos de la red, para esto utilice la clase StringTokenizer para poder leer dato por dato sin contar la coma que los separaba, todos estos datos los guarde en un objeto Grafo que implemente, éste objeto estaba formado por una lista cabecera en donde estaban todos los nodos cabeceras, cada cabecera tenía como atributo los tiempos de cola de cada nodo así como una lista ligada en donde estaban almacenados los nodos adyacentes a cada nodo con sus respectivos datos (distancia, velocidad, tamaño del paquete, datos de control, datos de usuario, etc.)

Ya teniendo almacenados los datos en el Grafo, implemente una función recursiva con el concepto de búsqueda a lo profundo para encontrar todos los caminos posibles del nodo inicial al nodo final. La función consistía en que, recibía como parámetros el nodo inicial y el nodo final; primero buscaba en el grafo el nodo inicial, y empezaba a analizar su lista de adyacencia, en una lista iba insertando las adyacencias de ese nodo, mi condición de paro era que una vez llegando al nodo final se detuviera y teníamos un camino e insertábamos ese camino en una lista donde iba a ir insertando todos los caminos posibles, si aún no llegábamos al nodo final hacia una llamada recursiva con el nodo actual y el nodo final, y hacia lo mismo buscaba los adyacentes del nodo actual los insertaba en la lista y así sucesivamente hasta obtener una lista con todos los caminos posibles.

Ya teniendo la lista con todos los caminos, tenía que ver en cuantos paquetes de cada enlace se tenían que dividir para transportar todo el paquete completo de un punto a otro. Para eso tuve que analizar cada camino posible, enlace a enlace fui viendo cuantos paquetes cabían dependiendo de los datos de usuario de cada enlace y fui arrastrando una variable llamada acarreo donde me decía el tamaño del paquete que iba circulando por la red.

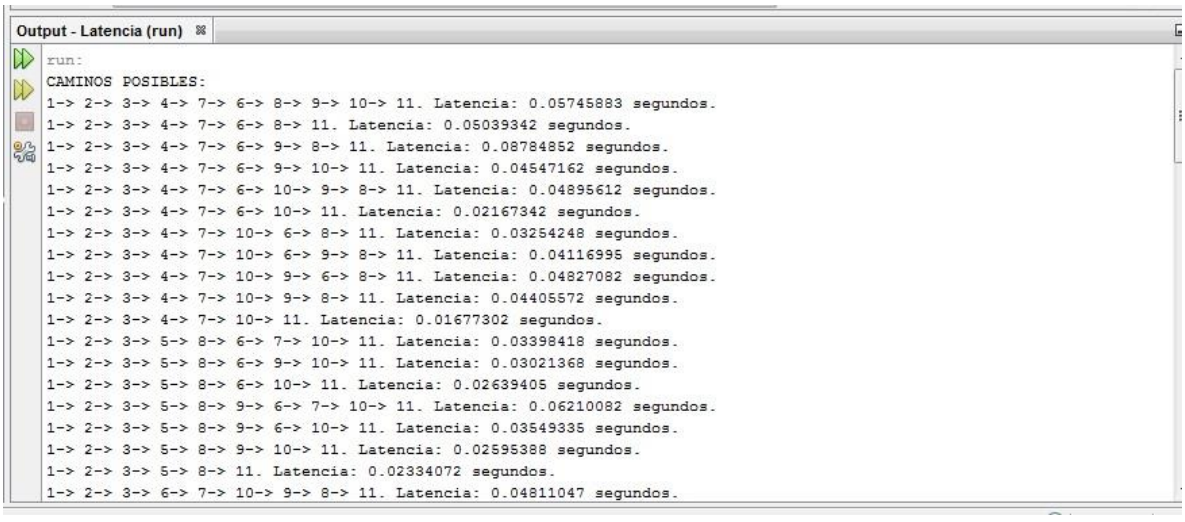
Después de tener la división de paquetes, procedía a obtener la latencia de cada camino y calcular la menor de todas. Para esto, recorrí la lista de todos los caminos posibles y por cada

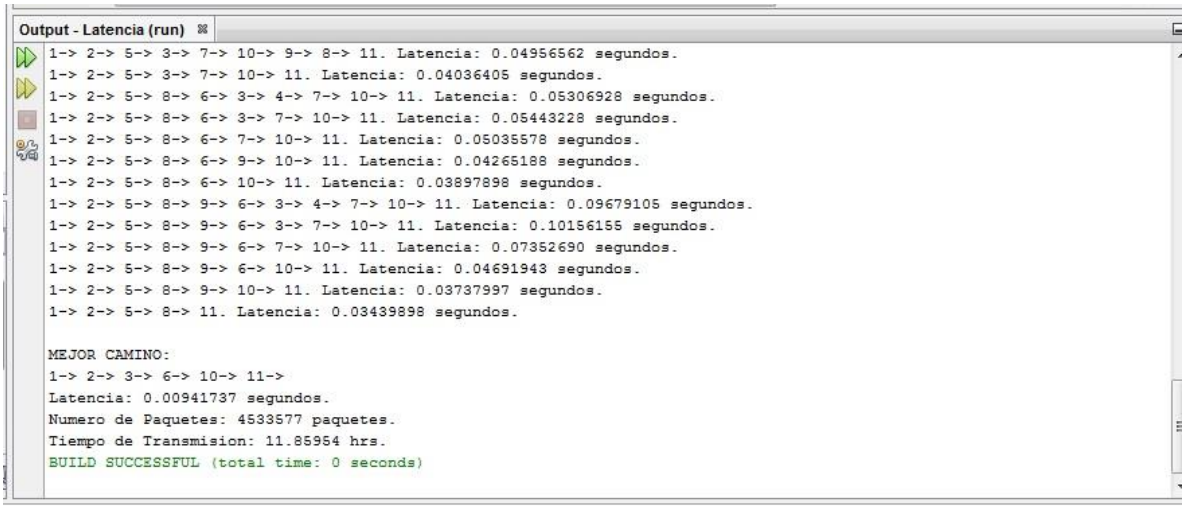
enlace aplicaba las formulas: Latencia = TP(Tiempo de Propagación) + TT (Tiempo de Transmisión) + TC(Tiempo de Cola), TP=distancia a recorrer / velocidad de la luz (300,000,000 m/s), TT= tamaño del paquete en bytes / tasa de transferencia teórica, TC=es el tiempo que se tarda un switch en procesar ese paquete y enviarlo a otro destino, al final de un camino guardaba la latencia en el último campo de cada camino, ya obtenidas todas las latencias, con una función recorría de nuevo la lista y obtenía la menor de ellas y ese camino era el que elegía como el de menor latencia.

Por ultimo obtenía el tiempo total de transmisión, para esto necesitaba el número de paquetes, lo obtuve con la fórmula: #paquetes = tamaño total del paquete / datos de Usuario, teniendo el #paquetes calculaba el tiempo total de transmisión con la formula TTT=#paquetes * Latencia, para estos cálculos utilice los datos de mi lista con menor latencia.

Al final, en la salida de datos, imprimo todos los caminos posibles, el camino con menor latencia, su latencia, el número de paquetes totales y el Tiempo Total de Transmisión del Paquete del Origen al Destino.

RESULTADOS





CONSLUSIONES

La finalidad del proyecto si se cumplió que ere obtener el camino más corto en cuestión de la menor latencia de ese camino para poder transmitir un archivo de un nodo origen a un nodo destino, después de esto obtener el número de paquetes en que se divide el archivo y por último el tiempo de transmisión que tardaría en transmitir ese paquete por el camino más corto. Es muy importante saber cómo obtener la latencia, ya que esta nos dice el tiempo que tarda en transmitir un paquete de un origen a un destino.

El proyecto no fue nada fácil, pero investigando y poniendo en práctica lo visto en cursos anteriores como programación II, estructura de datos, etc., se me facilitaron más las cosas, el asunto era analizar bien el problema y ver la manera de atacarlo para poder resolverlo.

BIBLIOGRAFIA

-A.S. Tanenbaum, "*Redes de Computadoras*", 4° Edición, Pearson Education, México, 2003.

-W. Stallings, "*Comunicaciones y Redes de Computadoras*", 7° Edición, Pearson Education, Madrid, 2004.

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

IVÁN LOZANO SÁNCHEZ

MATRICULA: 201114047

E-mail: i.va.n589@hotmail.com

PROFESOR: IVAN OLMOS PINEDA

MATERIA: MODELOS DE REDES

TEMA: PRIMER PARCIAL (PROGRAMA QUE CALCULA LATENCIAS).

26 DE SEPTIEMBRE DE 2014

1.- INTRODUCCION:

Hoy en día las redes de computadoras juegan un papel muy importante dentro de la informática ya que mediante ellas tenemos comunicación con cualquier parte del mundo, la latencia es importante para las redes ya que la latencia, o retardo, es el tiempo que una trama o paquete tarda en hacer el recorrido desde la estación origen hasta su destino final. Es importante determinar con exactitud la cantidad de latencia que existe en la ruta entre el origen y el destino para las LAN y las WAN. En el caso específico de una LAN Ethernet, un buen entendimiento de la latencia y de su efecto en la temporización de la red es de importancia fundamental para determinar si CSMA/CD podrá funcionar correctamente.

Factores que influyen en la latencia de una red son:

- El tamaño de los paquetes transmitidos.
- El tamaño de los buffers dentro de los equipos de conectividad. Ellos pueden producir un Retardo Medio de Encolado.

2.- DESARROLLO DEL TEMA:

Como ya definimos anteriormente la latencia es el tiempo en que tarda un paquete en transmitirse desde un origen hacia otro punto por lo que necesitamos desarrollar un programa que nos realice este cálculo; El programa será desarrollado en el lenguaje de programación java los puntos a tomar en cuenta son:

- Los datos serán introducidos mediante un archivo txt el cual debemos de analizar para que nuestros datos sean usados correctamente. Los datos son: -Numero de vértices,-Numero de enlaces,-Vértice inicial, -Vértice final, -Distancia (metros),-Datos de usuario,-Datos de control,-Tiempos de cola,-Vértice origen,-Vértice destino y por último el Tamaño del archivo).
- Analizar y obtener todos los caminos o paths posibles que halla en el grafo mediante un algoritmo de búsqueda en profundidad.
- Para cada path que obtengamos en nuestra búsqueda debemos de calcular su latencia ya que solo así analizaremos cual es la ruta que más nos conviene tomar y esto nos brindara la menor latencia de todos los paths.

- Debemos de implementar el cálculo final utilizando la fórmula para calcular la latencia desde un vértice origen hasta un vértice final.
- Por ultimo debemos de validar nuestros resultados probando con diferentes grafos para verificar si lo que implementamos realmente funciona. De lo contrario debemos de regresar al programa para localizar los posibles errores y hacer que los cálculos sean lo más confiable posible.

3.- RESULTADOS

De los puntos que se debían de tomar en cuenta mi proyecto para el primer parcial solo cumple con los dos primeros pasos.

- Primer punto Analizar el archivo: Lo primero que realice fue empezar a hacer pruebas para leer el archivo y separar cada línea en subcadenas para posteriormente guardar los datos en una estructura de datos. En este caso para hacer el recorrido en el grafo los vértices y enlaces los guarde en un Hashmap para evitar que existan adyacencias repetidas y tener un mejor manejo a la hora de realizar el recorrido ya que el hashmap guarda sus elementos como llaves únicas y es más fácil sacar la información de ellos.

  Los datos restantes los guarde en una LinkedList de java que es el equivalente a una lista ligada de igual manera para obtener de una manera más rápida y práctica los datos guardados en dicha estructura.
- Segundo punto Obtener todos los caminos posibles: Analizar el algoritmo no fue una manera sencilla ya que había que modificar un algoritmo en profundidad para poder obtener todos los paths. Lo que realice fue primero analizar el algoritmo en profundidad y ver qué es lo que me servía de él, después hacer que lo hiciera recursivamente para obtener resultados más confiables al momento de analizar el grafo.

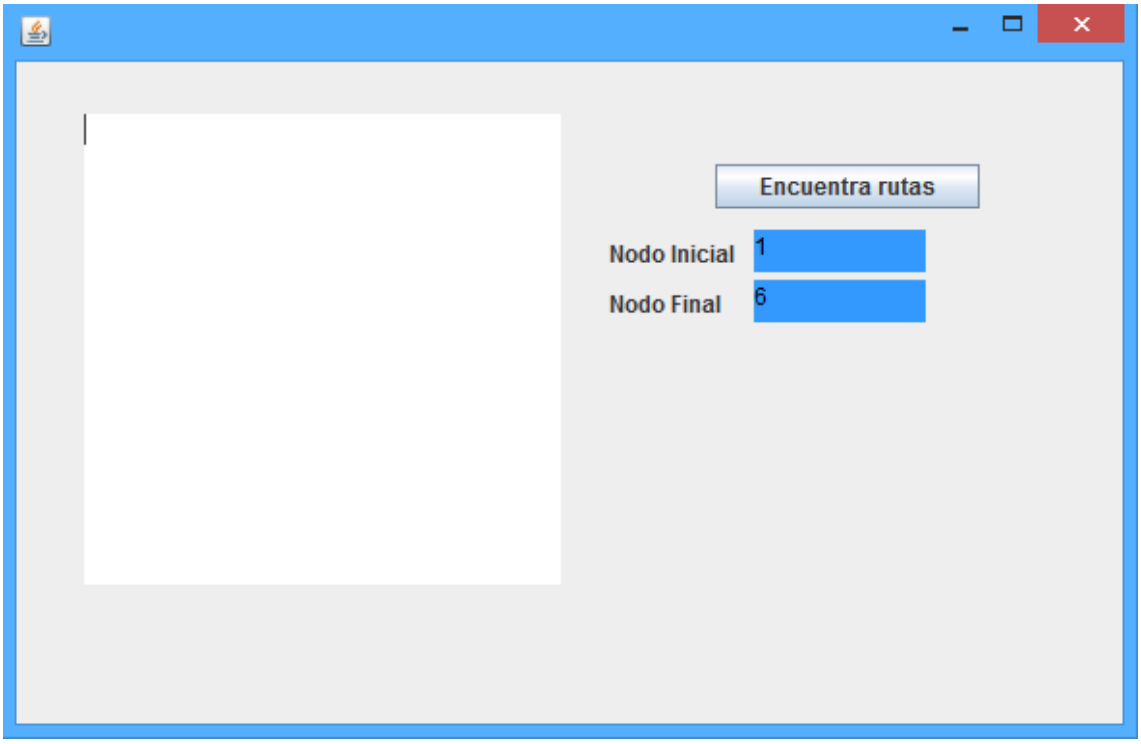
4.- CONCLUSIONES:

El programa se me complico en la parte de los cálculos de las latencias para cada path localizado ya que no supe como asociar cada ruta encontrada con su respectiva latencia.

La realización de este proyecto necesitaba de una buena planeación para poder darle determinado tiempo a cada uno de los puntos a realizar.

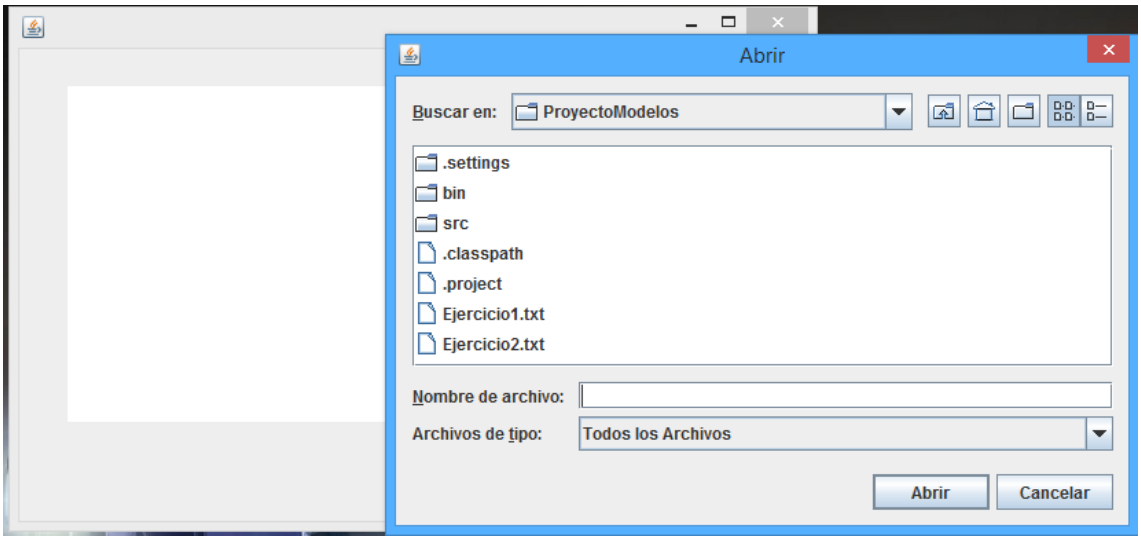
Uso del programa:

1.- En la primera parte nos sale la interfaz que usa el programa:

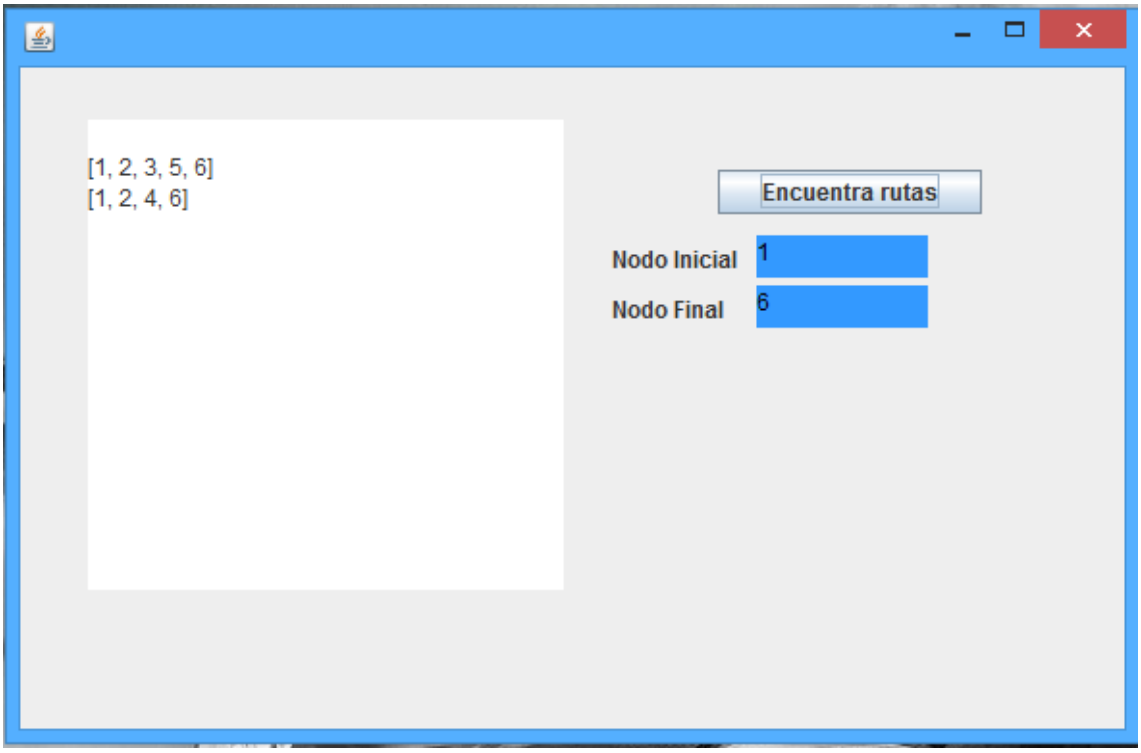


2.- En esa parte en los JtexField azules debemos de colocar primero de que nodo a que nodo queremos encontrar el camino.

3.- Después debemos de darle clic al botón encuentra rutas, se abrirá una nueva ventana para seleccionar el archivo.



4.-Por ultimo después de seleccionar el archivo el programa va a hacer la búsqueda de todos los caminos posibles y los mostrara en el jtextArea.



Nota: Los grafos que utiliza son los mismos que se usaron en clase para calcular la latencia.

5.- BIBLIOGRAFIA

Estructuras de **datos** en **Java** – Luis Joyanes Aguilar.

Estructura de datos en java - Mark Allen.

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Modelo de Redes

Roberto Jimenez Coronado

Examen 1

Otoño 2014

INTRODUCCION

Una topología de red es la forma en la que los routers estan interconectados siendo esta representada por un grafo conexo.

El cálculo de latencias es una de las tareas importantes en la implementación de topología de redes ya que al encontrar el camino más eficiente se ahorra mucho tiempo de trasnferencia.

Cada topología cuenta con cierto número de caminos entre los puntos de interes A-B para los cuales existe una latencia diferente.

DESARROLLO

Para poder calcular la latencia menor en una topología es necesario conocer todos los caminos posibles del punto A al punto B, es necesario conocer la distancia, el tamaño de los paquetes y la velocidad de transmision entre los routers que conforman el path o camino.

El desarrollo de esta practica se realizo en varias partes.

La primera parte consiste en leer el archivo de especificaciones. Dicho archivo debe llevar el formato siguiente:

<#Vertices>,<#Aristas>

<#Vinicial>,<#Vfinal>,<dist(m)><Vel(Mbps)>,<TamañoPaquete>,<DatosControl>

<#Vinicial>,<#Vfinal>,<dist(m)><Vel(Mbps)>,<TamañoPaquete>,<DatosControl>

... /*Se repite tantas aristas se tenga*/

<Tcola1>,<Tcola2>...<Tcola<#Vertices>>

<Vinicial>,<Vfinal>

<TamArchivo>

Se leen las especificaciones del archivo en una matriz de adyacencias y se guarda el grafo en un arreglo de listas para poder obtener los caminos mas facilmente.

Para buscar los caminos se utilizo un algoritmo DFS recursivo donde los nodos visitados se van marcando para ir construyendo el primer camino, una vez que se obtiene, se desmarcan los nodos para poder buscar un camino diferente. Se guardan cada uno de los caminos en un arreglo de listas.

Se procede a calcular el número de paquetes que viajara por cada uno de los path previamente obtenidos para poder calcular la latencia correctamente.

Una vez que se obtienen todos los datos se puede aplicar las formulas vistas en clase para calcular la latencia:

Lat = <#Paquetes>*(TT + TP) + <#Paquetes>*TC

Donde:

TT = <TamañoPaquete>/<Vel(Byte)>

/*Para tener la Vel en Byte se tiene que convertir ((Vel*1000*1000)/8)*/
TP = <dist>/<3*10^8>

TC = <Tcola<del router destino>>

<#Paquetes> Cuantos paquetes se envian desde el punto X1->X2

EJEMPLO

Tenemos el archivo de especificaciones en formato txt almacenado preferentemente en la carpeta donde se encuentra el programa.

8,9

1,2,85,1000,1500,200

2,3,90,800,1500,200

2,4,63,900,1500,200

3,4,98,1500,1500,200

3,5,105,1000,1500,200

4,7,87,700,1500,200

1,6,130,1200,1500,200

6,7,60,1300,1500,200

7,8,90,900,1500,200

0.000032,0.0011,0.00025,0.001,0.0007,0.0093,0.0004,0.0001,0.0006

1,8

4

Los caminos encontrados por el algoritmos propuesto en esta

práctica son: Caminos:

1->6->7->8

1->2->4->7->8

1->2->3->4->7->8

CONCLUSIONES

El cálculo de Latencias se puede hacer con varios algoritmos(Dijkstra,Dfs,etc.). Para obtener la latencia correcta es necesario tratar adecuadamente cada uno de los datos proporcionados en archivo.

Para una topología de red con un grafo conexo la complejidad del calculo de la mejor latencia esta dada, principalmente, por el número de nodos que posee el grafo, ya que, entre más grande el grafo, la busqueda de caminos asi como el calculo de sus paquetes y la aplicación de la fórmula en papel se vuelve cada vez más complicada, y más suseptible a errores.

Sin embargo, el programador debe cuidar cada uno de los datos utilizados para que su algoritmo sea eficiente y confiable.

Proyecto Examen 1

Azael Ortega Valdovinos | Modelos de Redes | 26 de septiembre de 2014

Tabla de contenido

Introducción

Una red de computadoras, también llamada red de ordenadores, red de comunicaciones de datos o red informática, es un conjunto de equipos informáticos y software conectados entre sí por medio de dispositivos físicos que envían y reciben impulsos eléctricos, ondas electromagnéticas o cualquier otro medio para el transporte de datos, con la finalidad de compartir información, recursos y ofrecer servicios.

Como en todo proceso de comunicación se requiere de un emisor, un mensaje, un medio y un receptor. La finalidad principal para la creación de una red de computadoras es compartir los recursos y la información en la distancia, asegurar la confiabilidad y la disponibilidad de la información, aumentar la velocidad de transmisión de los datos y reducir el costo general de estas acciones. Un ejemplo es Internet, la cual es una gran red de millones de computadoras ubicadas en distintos puntos del planeta interconectadas básicamente para compartir información y recursos.

La “Latencia informática”.es el tiempo que tarda un dato en estar disponible desde que se realiza su petición. Se puede comparar con el tiempo de reacción. Se mide en nanosegundos (ns) o en milisegundos (ms). Cuanta menos latencia, mejor.

En redes informáticas de datos se denomina latencia a la suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red.

Otros factores que influyen en la latencia de una red son:

- El tamaño de los paquetes transmitidos.
- El tamaño de los buffers dentro de los equipos de conectividad. Ellos pueden producir un Retardo Medio de Encolado.

La latencia es igual al “Tiempo de propagación” más el “Tiempo de transmisión” más el tiempo de cola.

$$\text{Latencia} = \text{Tiempo de propagación} + \frac{\text{Tamaño de paquete}}{\text{Velocidad de transmisión}} + \text{Tiempo de cola}$$

Como trabajo para la evaluación del 1er parcial es Implementar un programa que dada la metodología de red(Velocidad de trasmisión, distancia de segmentos y tiempos de cola) poder determinar el camino más corto entre dos nodos del sistema así como calcular el tiempo total de trasferencia.

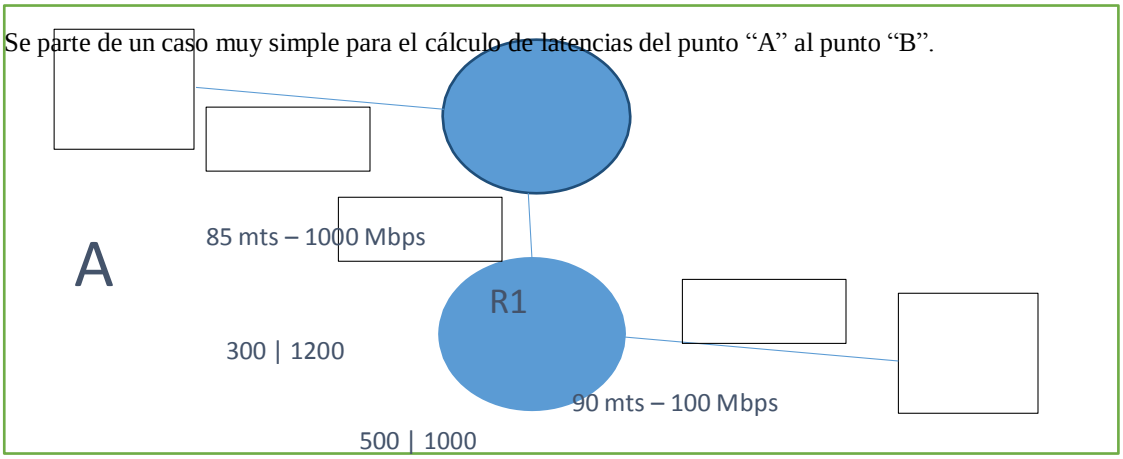
Se debe determinar:

- Todos los caminos posibles
- Reportar todas las latencias
- Reportar el mejor camino y el tiempo total de trasferencia para el archivo especificado.

Desarrollo

EJEMPLO:

Se parte de un caso muy simple para el cálculo de latencias del punto “A” al punto “B”.



El cálculo de la latencia de “A” a “B” es el siguiente:

$$\text{[illegible]} = \text{[illegible]} + \text{[illegible]} + \text{[illegible]} + T\text{[illegible]}$$

$$+ T\text{[illegible]}$$

En este ejemplo ya que el paquete del segmento “A R1” que los datos de usuario es más grande que los datos de usuario del segmento “R1 R2” el paquete se parten, formando uno con 1000 y otro con 200, de la misma manera de “R2 B” el paquete de 1000 se vuelve a partir uno en 800 y otro 200, quedando en total 3 paquetes y la formula será la siguiente.

$$\text{[illegible]} = \text{[illegible]} + 2(\text{[illegible]} + T\text{[illegible]}) + 3(\text{[illegible]}$$

$$+ T\text{[illegible]})$$

$$\text{}_{\text{}1} = \frac{85 \text{ }}{\frac{\text{}}{300000 \frac{\text{}}{\text{}}}} + \frac{1500 \text{}}{1000 \frac{\text{}}{\text{}}}$$

$$\text{}_{\text{}1} = 1.2283 \times 10^{-5}$$

$$\text{}_{1\ 2} = \frac{90 \text{ }}{\frac{\text{}}{300000 \frac{\text{}}{\text{}}}} + \frac{1500 \text{}}{100 \frac{\text{}}{\text{}}}$$

$$\text{}_{1\ 2} = 1.203 \times 10^{-4}$$

$$\text{}_{2\ } = \frac{95 \text{ }}{\frac{\text{}}{300000 \frac{\text{}}{\text{}}}} + \frac{1500 \text{}}{1000 \frac{\text{}}{\text{}}}$$

$$\text{}_{2\ } = 1.2316 \times 10^{-5}$$

$$\text{} = 1.2283 \times 10^{-5} \text{} + 2(1.203 \times 10^{-4} \text{}) + 3(1.2316 \times 10^{-5} \text{})$$

$$\text{} = 0.000899831$$

Con esta serie de fórmulas solo se ha obtenido la latencia, para poder calcular el tiempo total de transferencia se debe aplicar la siguiente:

$$\#\text{paquetes} = \frac{\text{tamaño de archivo}}{\text{tamaño de datos enviados}}$$

Supongamos que el tamaño de archivo es de 2.2 Gb, y el tamaño de datos enviados corresponde a los datos de usuario del primer paquete enviado. Al ser 2.2 Gb y los datos de usuarios están en bytes, se debe hacer una transformación sobe la escala de bytes.

1024 byte	1 Kbyte
1024^2 bytes	1 Mbyte
1024^3 bytes	1 Gbyte

Usando la tabla queda nuestro archivo a bytes si el resultado de número de paquetes tiene parte decimal se redondea al tope ya que no se pueden mandar paquetes fraccionados quedando así de la siguiente manera:

$$\#\text{paquetes} = \frac{2{,}362{,}232{,}012}{\text{tamaño de datos enviados}}$$

$$\#\text{paquetes} = \frac{2{,}362{,}232{,}012}{1200}$$

$$\#\text{paquetes} = 1968526.67 \cong$$

$$1{,}968{,}527$$

$$\text{tiempo de transmisión} = \#\text{paquetes} \times$$

$$\text{tiempo de un paquete}$$

$$\text{tiempo de transmisión} = 1{,}968{,}527 \times$$

$$0.000899831\ \text{seg}$$

$$\text{tiempo de transmisión} =$$

$$1771.34\ \text{seg}$$

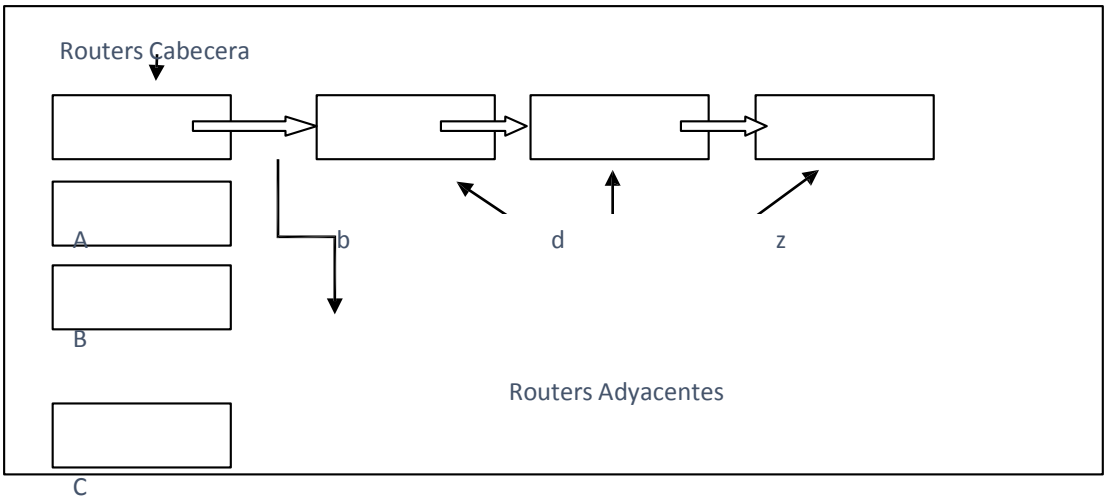
Partiendo de este caso se podrán programar las formulas y estructuras de datos correspondiente para llegar a su solución.

IMPLEMENTACION AL PROGRAMA

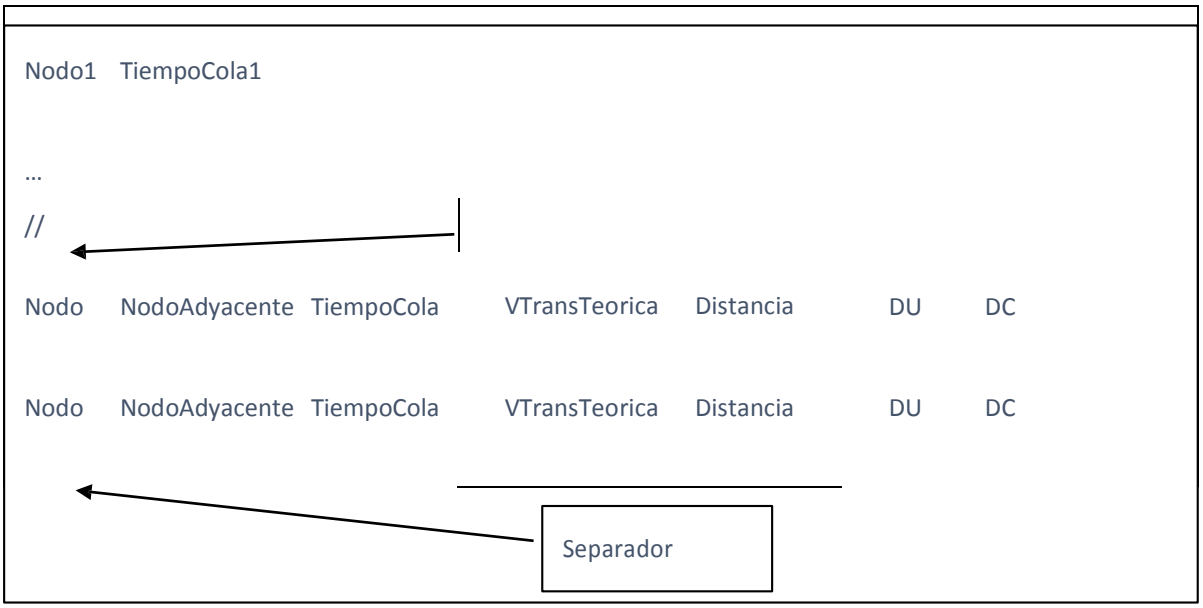
Representacion en grafo

La estructura de la red se puede representar con un grafo no dirigido, siendo los routers nodos, y los segmentos como aristas.

Se implementaron listas de adyacencias en java y quedan de la siguiente forma



En el programa se hace la lectura desde un archivo de texto que lleva el siguiente patrón:



En la primera parte se colocan el total de nodos que existen en el grado, en la segunda parte se coloca en el siguiente orden , nodo en el cual se le colocara la adyacencia, el nodo adyacente, el tiempo de cola del nodo adyacente, la velocidad de transferencia teórica (Mbps), distancia(mts), datos de usuario y datos de control

(ambos en bytes), en los campos de tiempo de cola, transferencia teórica, distancia, datos de usuario y datos de control solo se colocan números.

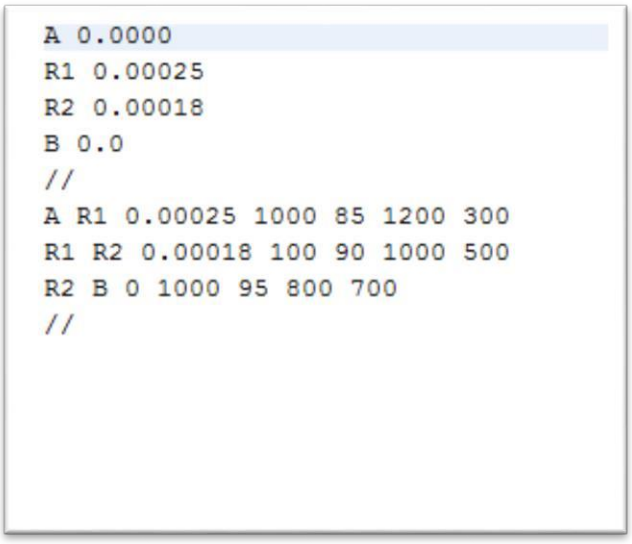


Ilustración 1 Ejemplo de archivo para lectura

Determinacion de todos los caminos posibles

Se deben de calcular todos los caminos posibles en el grafo, este se calcula con una pequeña variante del algoritmo de Dijkstra el cual es el siguiente:

Se inserta Router Inicial a la cola de prioridades

Mientras la cola no este vacía

Saca elemento de la cola

Si el último elemento del dato extraído no es el Router Final

Se buscan adyacencias

Si no se repite algún router en el camino

Se agrega nuevo camino a la cola

Siguiente adyacencia

Se registra en el archivo el camino

Fin Mientras

Con este algoritmo se eliminan los posibles ciclos.

Determinación de latencias

Ya que se tienen todos los caminos registrados en un archivo se calculan las latencias de todos los segmentos más su tiempo de cola multiplicados por los paquetes que son transferido. En la lista de adyacencia se guardan las latencias de cada segmento y su tiempo de cola.

Existe un conflicto en el momento de dividir los paquetes, ya que se si se manda un paquete de 1100 bytes en un segmento y pasar por otro se tendrán que dividir dependiendo del siguiente segmento en el que se mande, esto manejando un límite.

Esta solución se llevó acabo implementando un lista ligada en el cual tiene un elemento llamado datos y un elemento limite, cada vez que se inserte un elemento se compara el dato con el límite, si el dato es mayor se debe reacomodar la lista de la siguiente manera:

Si dato es mayor limite

Aux = Dato – limite

Insertar al final de la lista el "Aux"

Dato = Dato – Aux

//fin SI

//Se sigue recorriendo la lista ligada

Con esto se puede determinar el número de paquetes que se tienen en la lista ligada y este se multiplicara por las latencias del segmento correspondiente.

Se guardan las latencias totales con su camino en un archivo y en un algoritmo simple de determinacion de cual numero es menor sacamos el camino con la latencia menor.

Ejecución del programa

Se muestra la ejecución del programa con otra configuración en la red. A continuación las capturas de pantalla de los resultados.

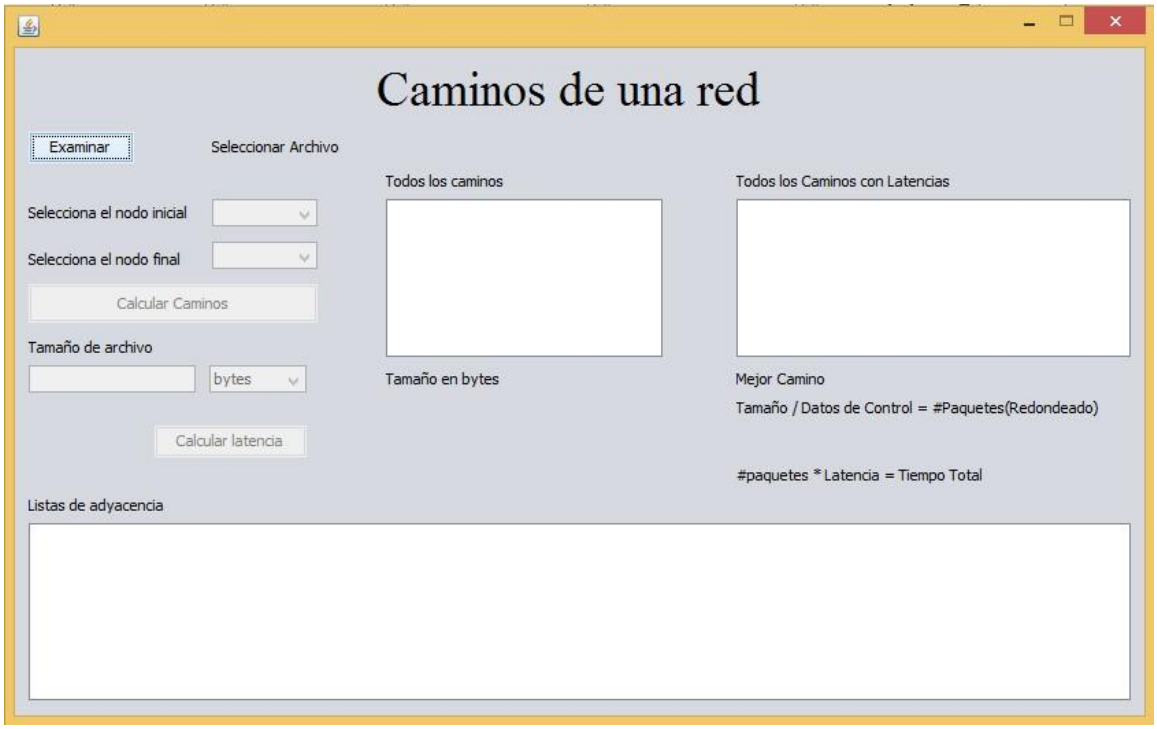


Ilustración 2: Como primer paso se debe de presionar el boton examinar

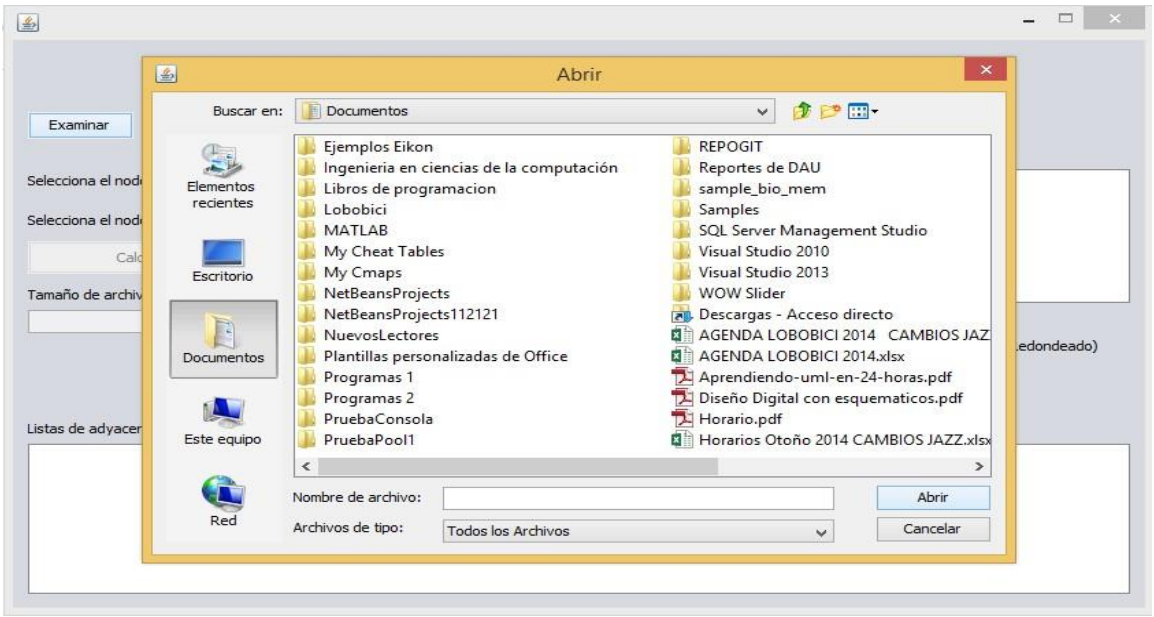


Ilustración 3: Se busca un archivo valido,

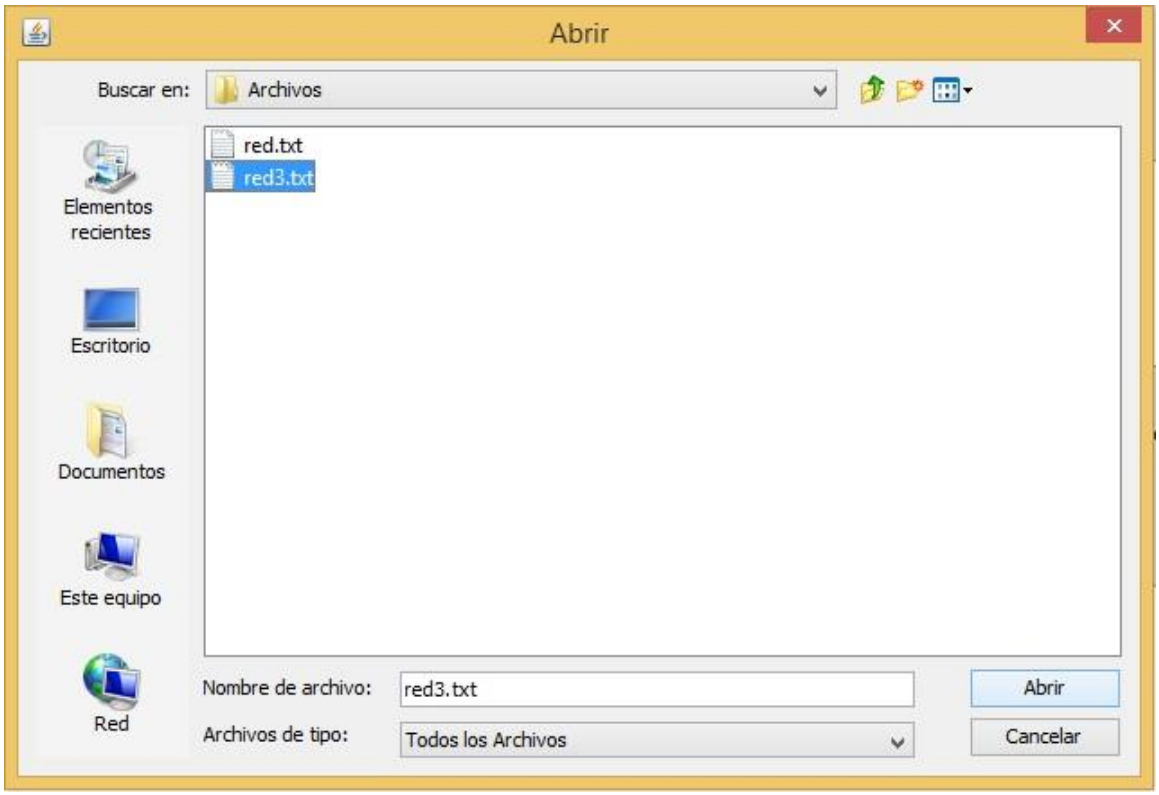


Ilustración 4: Se selecciona un archivo valido, si no es un archivo valido se repetira la accion

```
A 0.0000
R1 0.00018
R4 0.00036
R5 0.00012
R2 0.00025
B 0.0
//
A R1 0.00018 100 80 1200 300
A R4 0.00036 100 90 1100 400
A R5 0.00012 1000 120 1200 300
R1 R2 0.00025 1000 70 800 700
R1 A 0.00018 100 80 1200 300
R2 R4 0.00036 1000 90 700 800
R2 R1 0.00025 1000 70 800 700
R4 B 0 100 85 600 900
R4 R2 0.00036 1000 90 700 800
R4 A 0.00036 100 90 1100 400
R5 B 0 1000 110 1000 500
R5 A 0.00012 1000 120 1200 300
B R4 0 100 85 600 900
B R5 0 1000 110 1000 500
//
```

Ilustración 5: Muestra del archivo que se cargo

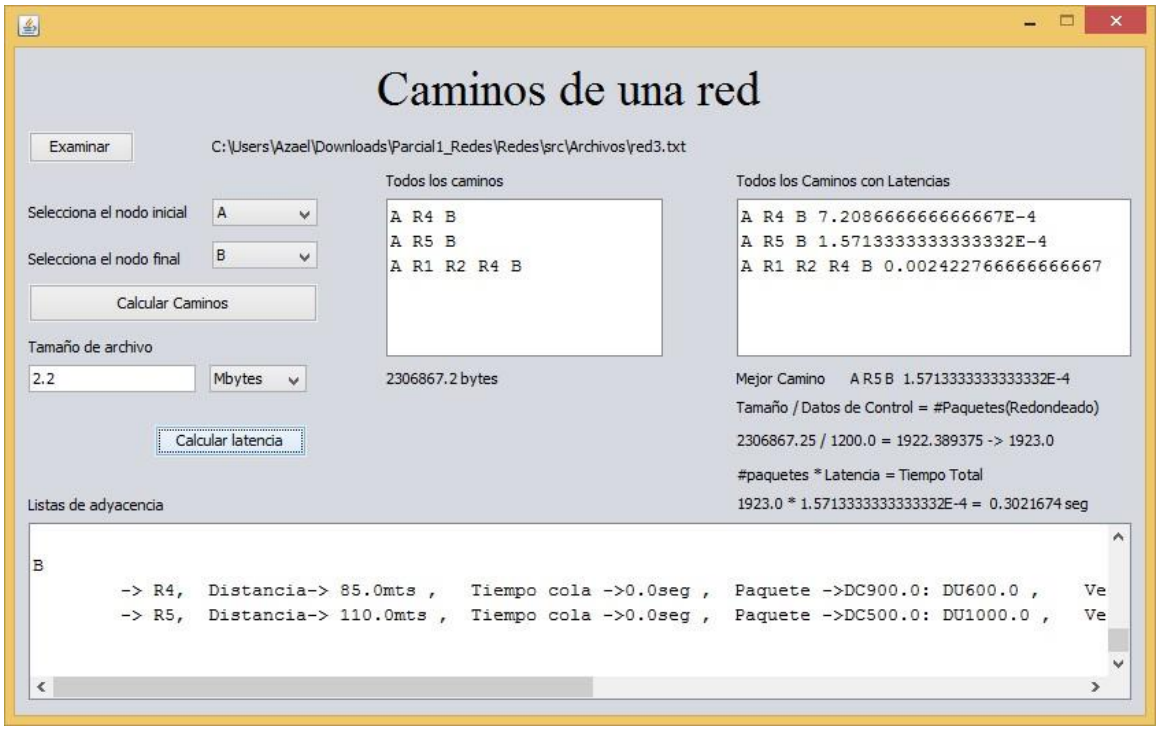


Ilustración 6: Se selecciona el nodo de cual se parte ,cual es nodo al que se quiere llegar y por ultimo oprimir el boton "Calcular cambios".Esto calculara todos los caminos, se llena la casilla y se oprime el boton "Calcular latencia"

Nota: El archivo .jar se localiza en la carpeta dist del proyecto de netbeans.

Referencias

- http://es.wikipedia.org/wiki/Latencia
- http://www.ecured.cu/index.php/Latencia_inform%C3%A1tica
- https://www.citycloud.com.ar/city-cloud-es/algunos-datos-interesantes-sobre- latencia/#.Ukj1pH9TXM0

FACULTAD *de* CIENCIAS
de la COMPUTACIÓN

Benemérita Universidad Autónoma de Puebla

Equipo:

López López Daniel

Delgadillo Arroyo Diana

Sánchez Ramírez Carlos Alberto

Alvarado Pineda Esperanza

Walberth Hernández Ramírez

Modelos de Redes

Dr.: Iván Olmos Pineda

Documento Técnico: Cableado Estructurado FCC

Fecha: 6-11-14

Introducción

Actualmente la red de la Facultad de Ciencias de la Computación no tiene implementado un cableado estructurado y no cubre la necesidad de los usuarios.

El objetivo principal consiste en implementar una red de cableado estructurado en la FCC para una mejor administración y satisfacer las necesidades de los usuarios, además que futuros administradores puedan hacer un uso correcto de la red sin conocerla de antemano y así mismo evitar fallas en dicha red.

Este trabajo se estructura en 6 secciones:

1. Puntos de Acceso
2. Número de Conexiones
3. Especificación Cableado Horizontal
4. Especificación por Salón
5. Especificación por Edificio
6. Equipo y Costos

Problemas

- Mala infraestructura y conectividad dentro de la FCC.
- Pocos accesos a Ethernet y puntos de accesos a internet.
- Falta de organización en el cableado.
- Interferencia electromagnética.
- Equipo inadecuado y por lo tanto ineficiente.
- Inconformidad de los usuarios.
- Balance de cargas.

- Falta de administración de la red.

Técnicas de Resolución

Implementar un cableado estructurado:

El cableado estructurado consiste en la implementación de una red aun edificio aplicando normas y estándares que permita dar una infraestructura flexible de cables que pueda aceptar y soportar múltiples sistemas de computación y de comunicación.

Desarrollo

1 Puntos de Acceso

La FCC tiene dos puntos de acceso de fibra óptica proveniente del edificio SIU de la BUAP, como se muestra en la Fig. 1.1:

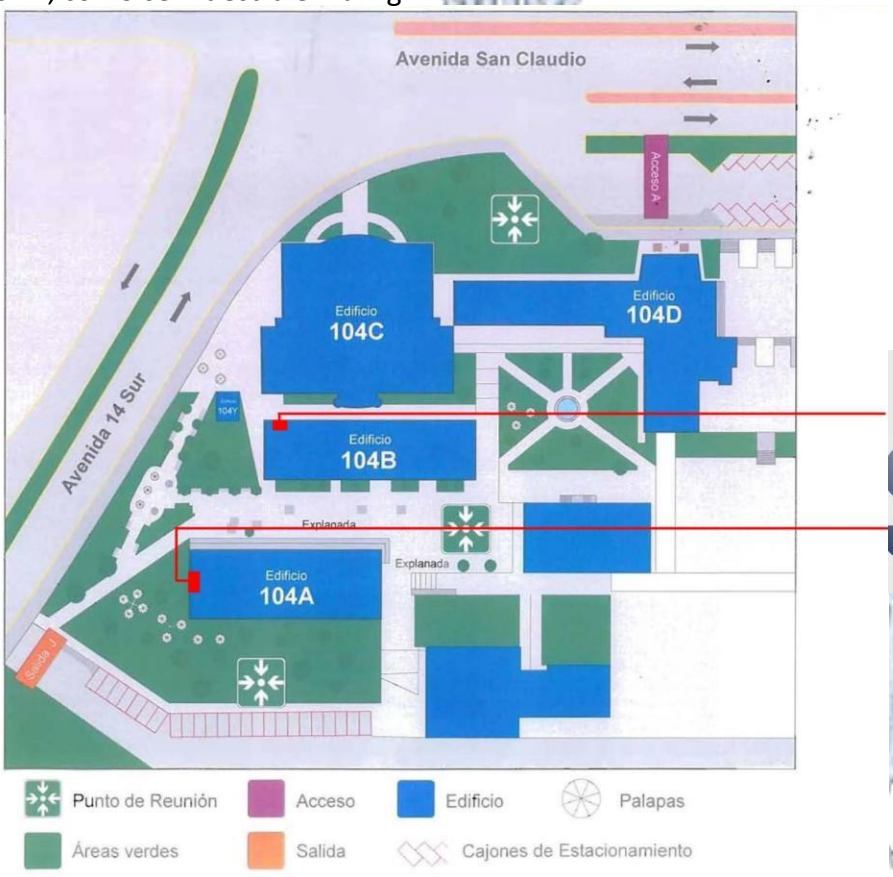


Fig. 1.1

Cada punto de acceso tiene las siguientes caracteristicas:

- 3 pares de fibras monomodo TeraSPEED.
- Velocidad de transmisión de hasta 10 Gbps.
- Actualmente solo se usa a 1 Gbps.
- Las tres fibras estarán trabajando en estándar LC.

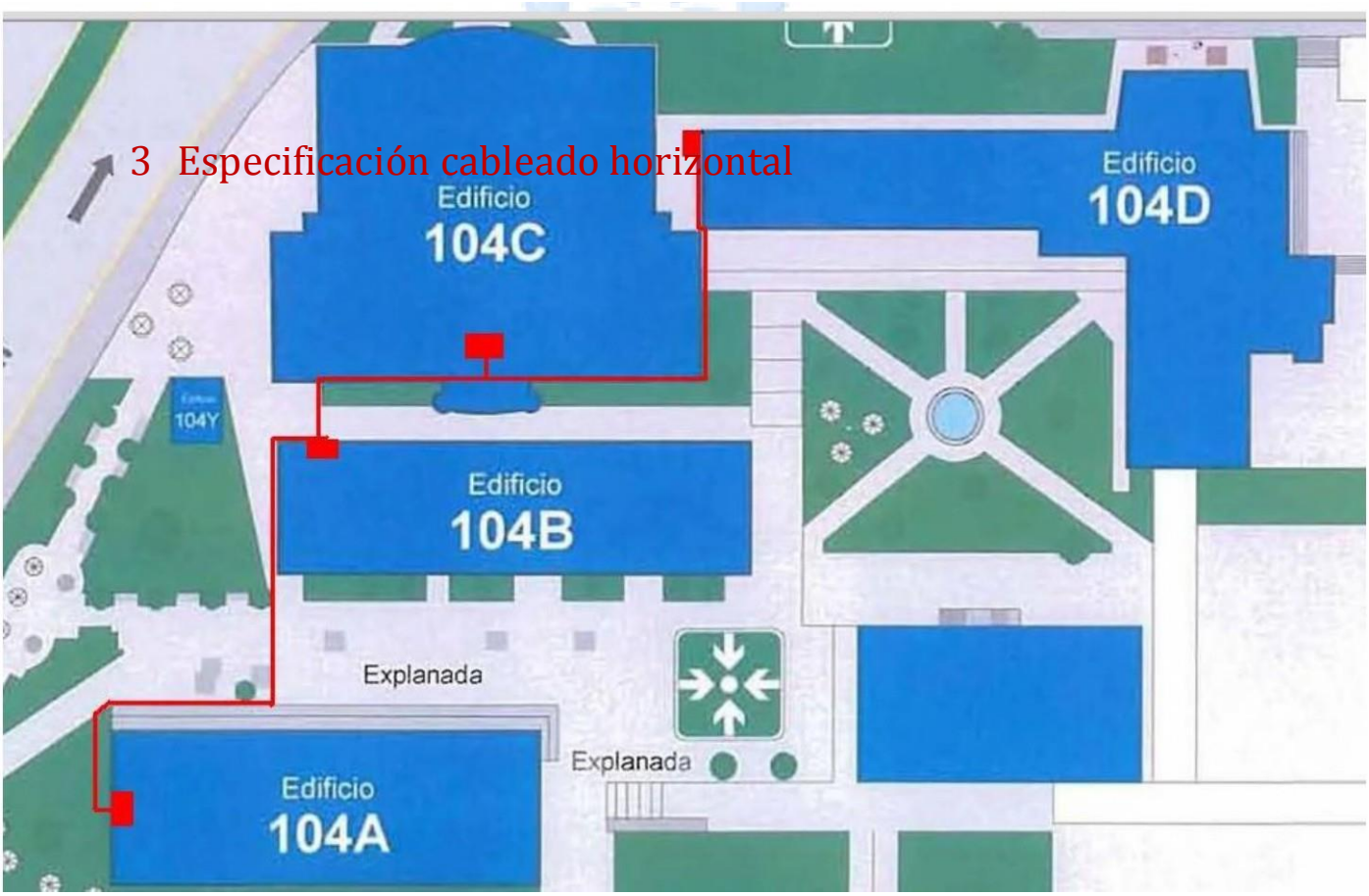
2 Número de Conexiones

La cantidad de conexiones por edificio se muestra en la Tabla 2.1:

Edificio	Número de Pisos	Numero de Salones piso	Conexiones por salón	Redundancia 30%	TOTAL conexiones
104A	**1**	**23**	**2**	**1**	**69**
104B	**1**	**4**	**0**	**0**	**0**
104C	**1**	**4**	**30**	**9**	**156**
104D	**3**	**4**	**35**	**11**	**552**

Tabla 2.1

3 Especificación cableado horizontal



Cuarto de equipo

Cableado

- Cada cableado estará cubierto por un tubo galvanizado de 2 pulgadas.
- Todos los tubos estarán dentro de la pared y en el subsuelo como se muestra en la Imagen 3.1.

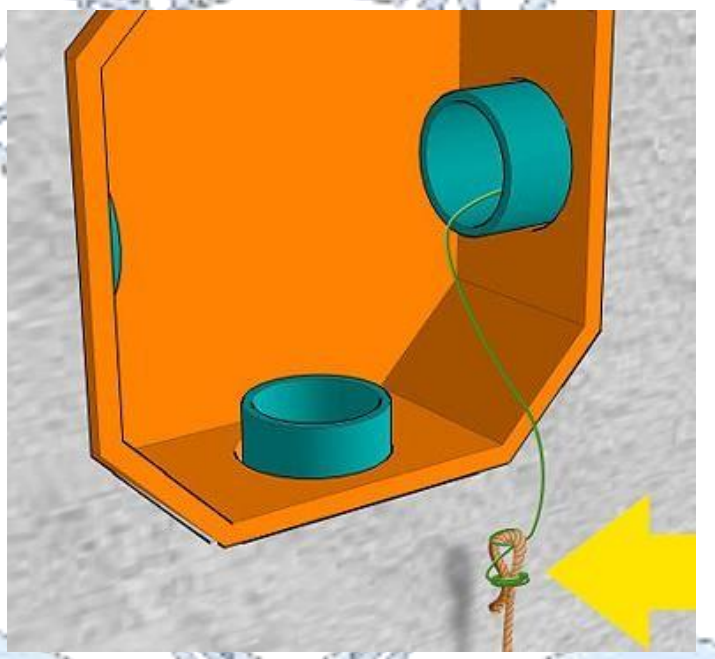


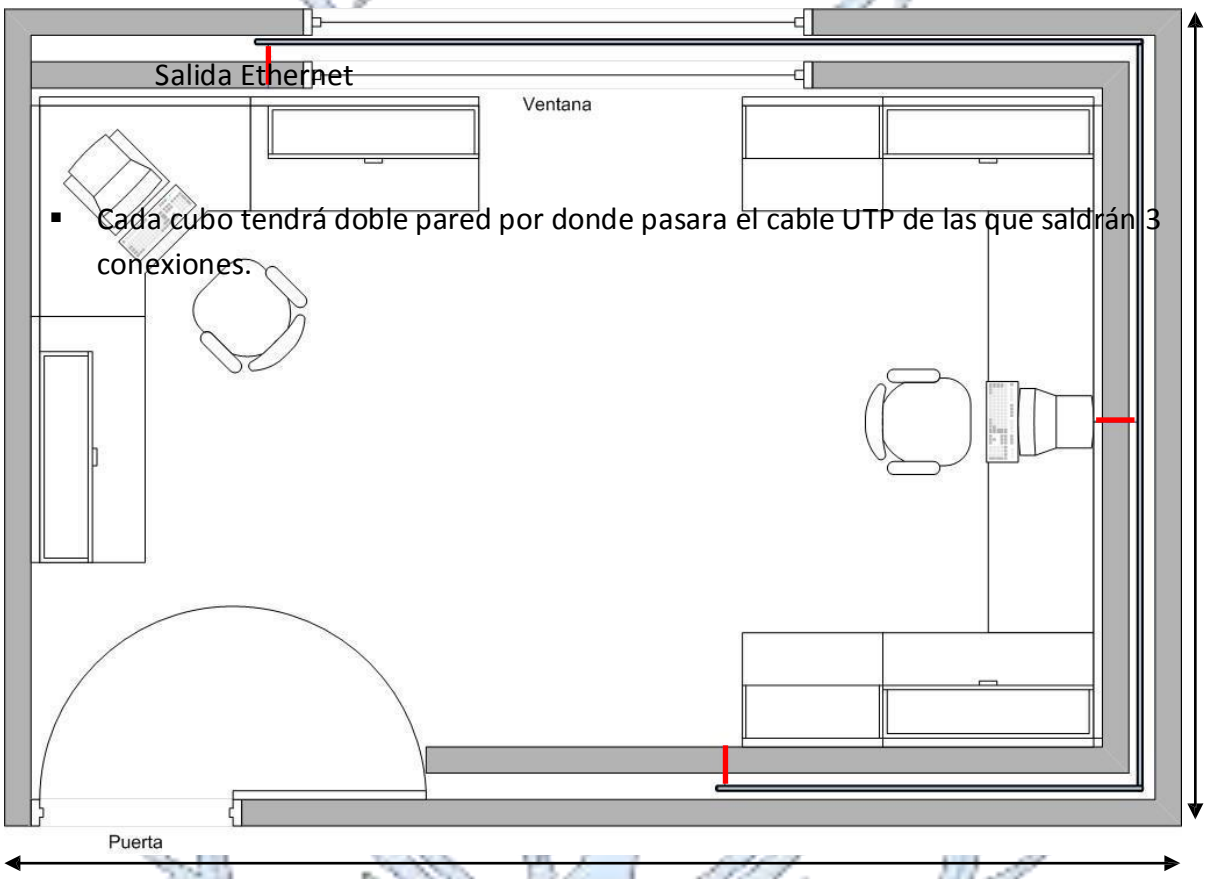
Imagen 3.1

- Cada tubo estará separado mínimo 13 cm para cables 2KVA o menos y mínimo 30 cm para cables 2KVA a 5 KVA.
- El cable será UTP Categoría 5e.

4 Especificación por salón

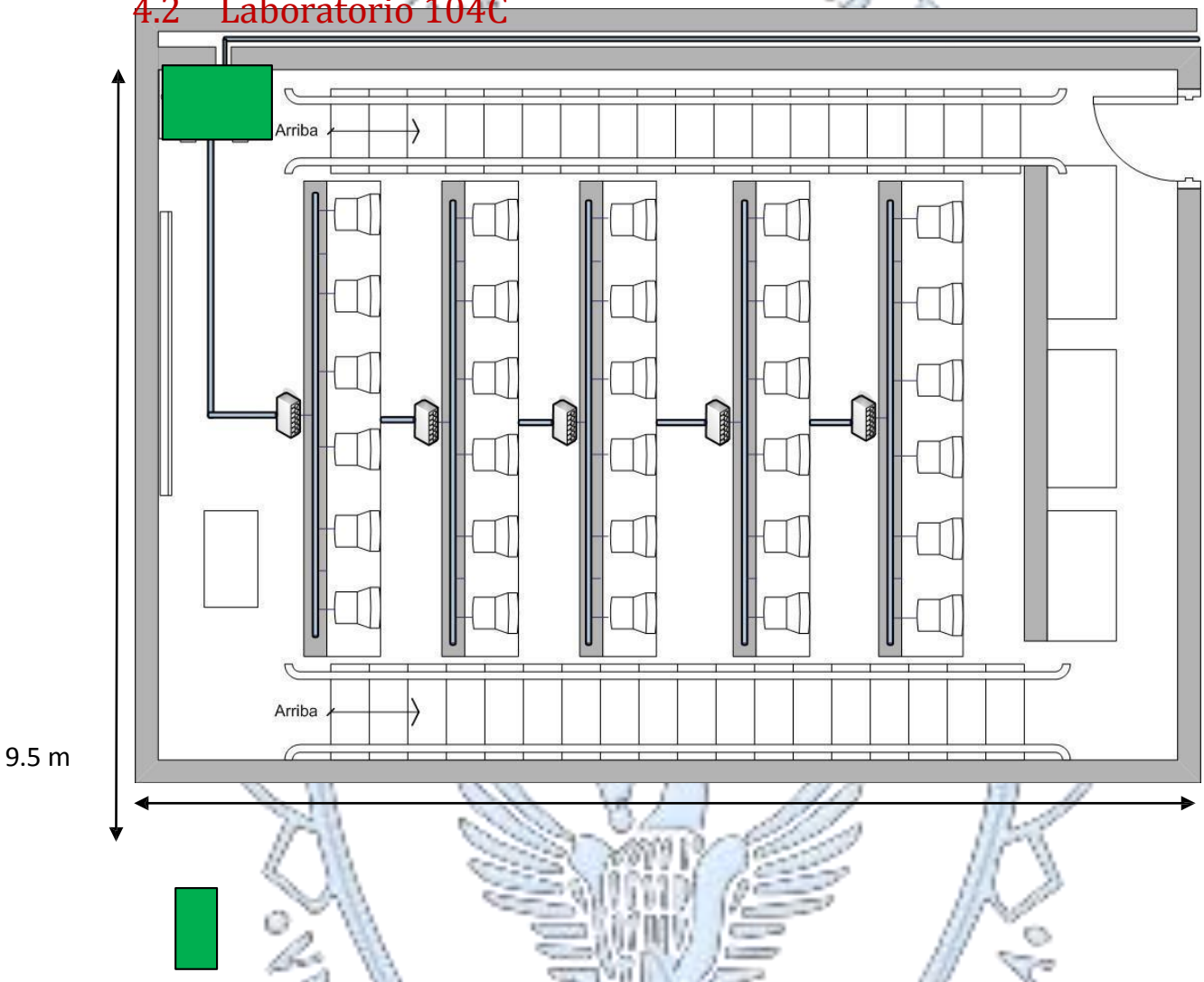
4.1 Cubo 104A

2.26 m



- Cada cubo tendrá doble pared por donde pasara el cable UTP de las que saldrán 3 conexiones.

4.2 Laboratorio 104C

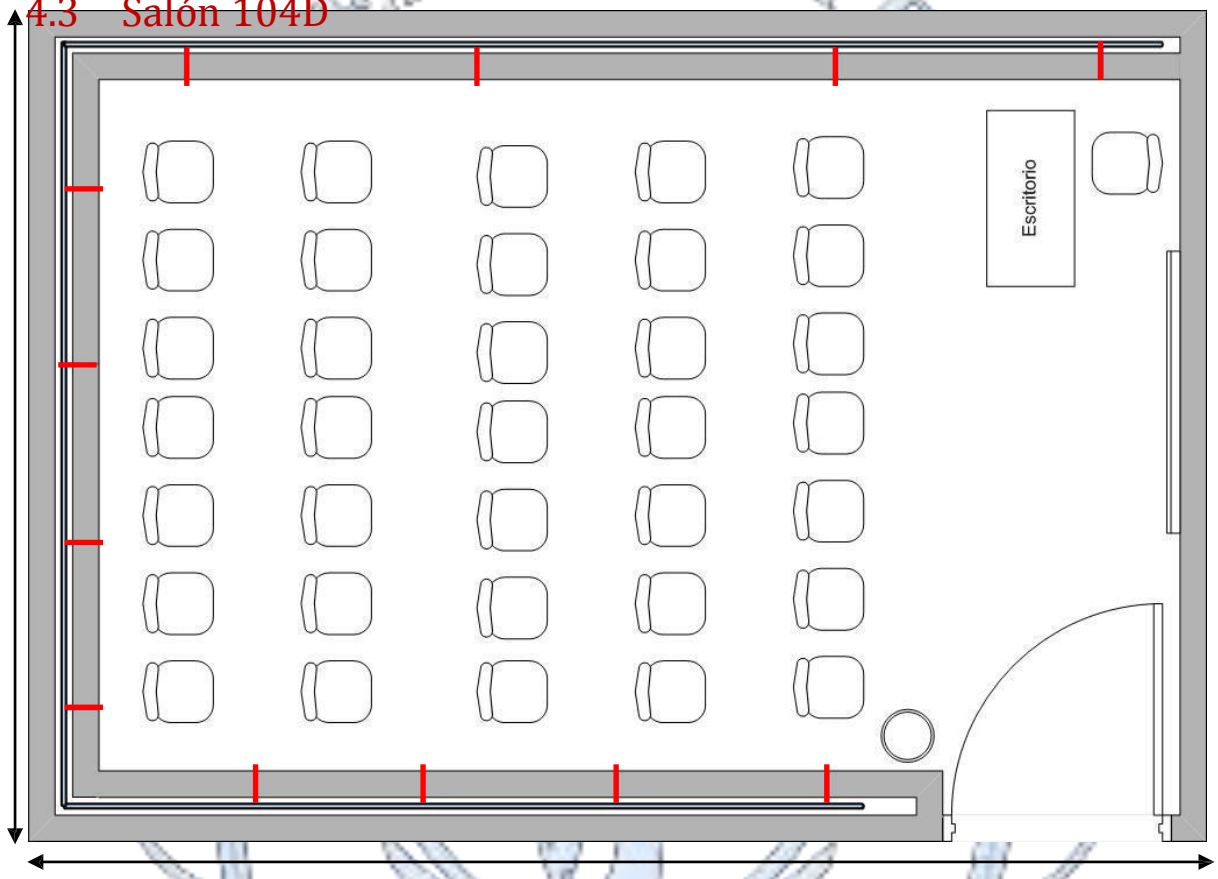


14.32 m

Closet de Comunicaciones

- Cada laboratorio tendrá doble pared por donde entrara el cable que viene del cuarto de equipo, el cable llegara a un closet de comunicaciones en donde repartirá la conexión a las 30 maquinas, cada 6 maquinas se conectara un switch. Todo el cableado estará oculto por la pared hueca. Y solo un tramo del closet de comunicaciones a las primeras computadoras será cubierto por canaleta.

4.3 Salón 104D



8.04 m

9 m

Salida Ethernet

- En este salón también se usa doble pared, el cable pasa por 3 paredes y en total reparte 12 conexiones a Ethernet.

5 Especificación por edificio

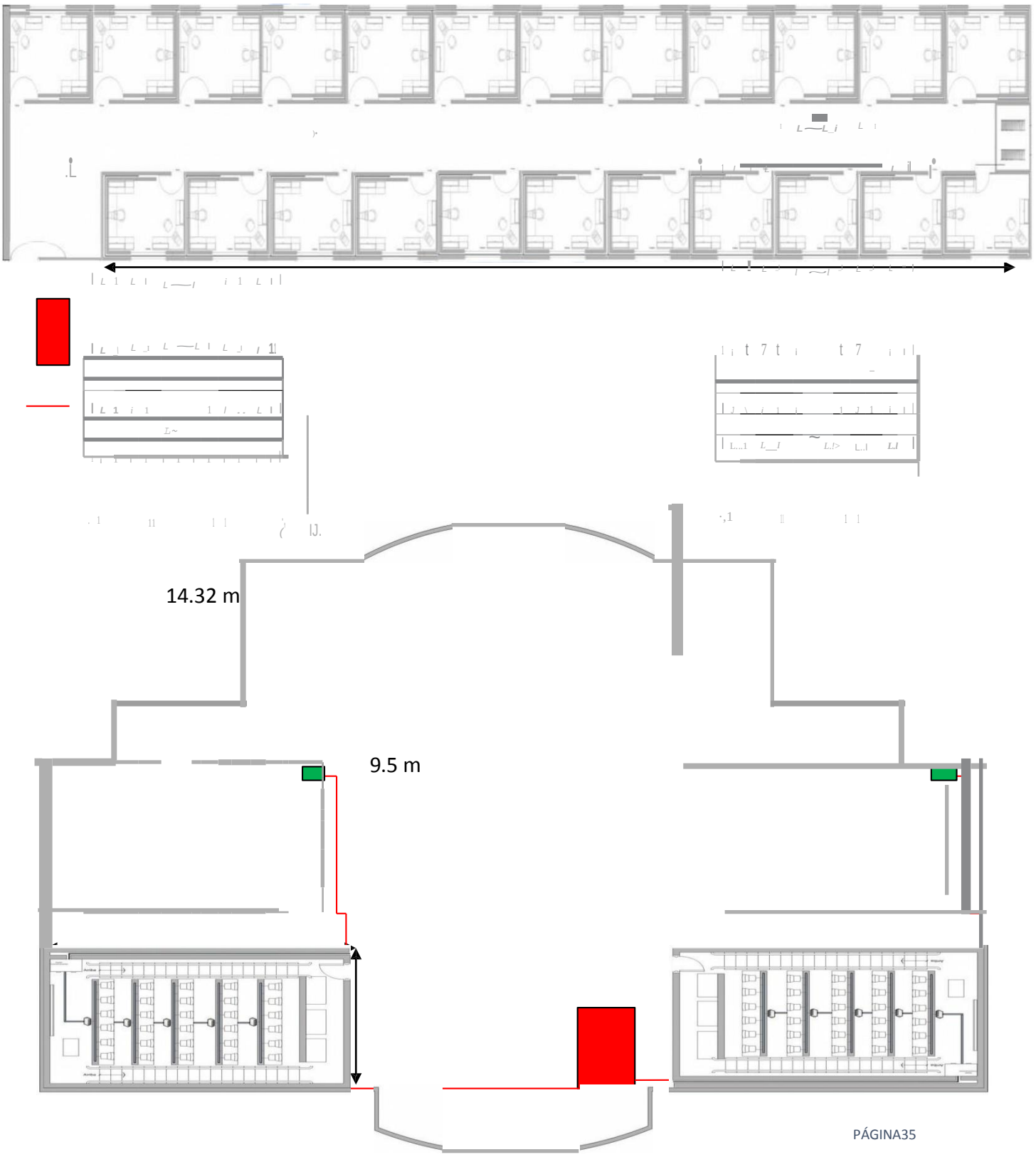
5.1 Edificio 104A

24.9 m

Cuarto de Equipo

Cable UTP Cat. 5e

5.2 Edificio 104C


14.32 m
9.5 m

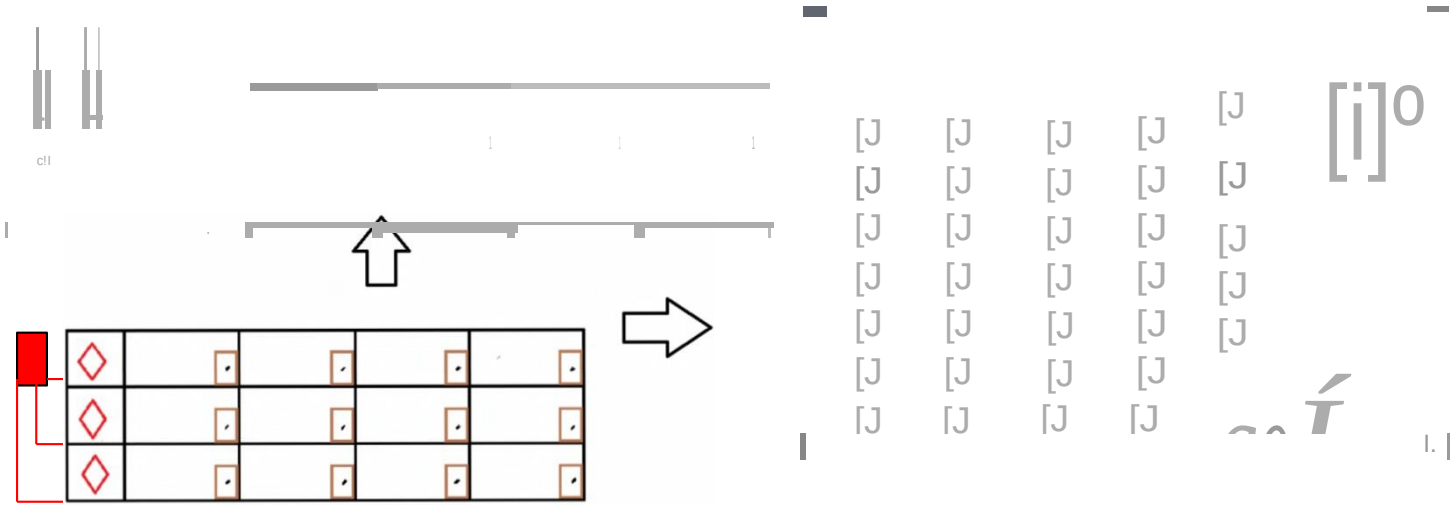
Cuarto de Equipo

Closet de Comunicaciones

Cable UTO Cat. 5e

En este edificio se colocara un closet de comunicaciones por cada laboratorio, todos ellos estarán conectados a un closet de comunicaciones, el cableado UTP pasara por doble pared y dentro de cada laboratorio los cables estarán dentro del muro.

5.3 Edificio 104D cableado vertical



Cuarto de Equipo

Closet de Comunicaciones

Cable UTO Cat. 5e

En este edificio se colocara un closet de comunicaciones por piso, los cuales irán conectados a un closet de comunicaciones el cual se instalara debajo en el primer piso a un costado del edificio, se construirá ese espacio. Los salones tendrán doble pared por donde pasaran los cables UTP. Además abra un punto de acceso por cada salón disponible para 20 alumnos.

6 Equipo y Costos

ACCESS POINT CISCO WAP121	**$ 1,968.41**	$ 23,620.92
PATCH PANEL INTELLINET CAT5E 2U	**$ 1,012.69**	$ 9,114.21
SWITCH CISCO CATALYST WS-C2960X-48TD-L	**$ 83,268.51**	$ 166,739.02
SWITCH CISCO CATALYST 3750V2-48PS	**$ 123,404.58**	$ 370,213.74
Cisco Catalyst 3750V2-48TS	**$ 100,012**	$ 700,014
	Sub-Total	$ 1,269,701.89
Cable UTP Categoría 5E		$ 7,200
RJ45	**$ 6.00**	$ 2,208
Rack vertical de 47,5 cm de ancho x 210 cm de altura, de aluminio	**$ 1,990**	$ 11,940
Panel porta cables para instalación en rack	**$480**	$ 2,880
Caja de registro sencilla	**$39**	$ 12,987
Conector hembra (jack) RJ45 Keystone negro Categoría 5e	**$24**	$ 7,992
Aire acondicionado	**$ 6,699.00**	$ 40,014
Tl-sf1016d Switch Tp-link 16Pts 10/100 Mbps	**$ 605.00**	$ 12,100
	Sub-Total	$ 97,321
	TOTAL:	$ 1.367.022.89

FACULTAD DE CIENCIAS DE LA COMPUTACION

Modelo de Redes

Examen Parcial 2

Reporte Técnico de Cableado Estructurado

ALUMNOS:

Sergio López Hernández 200914368

Israel Cruz Guzmán 200711817

Miguel Angel Cuaya Coyotl 200922387

Elizeth Escudero Álvarez 200909143

Ruy Baruc Vera Ramos 200931481

Cableado estructurado

Objetivos

Proporcionar una conexión a internet que sea segura, eficaz y robusta, para que los estudiantes y profesores de la Facultad de Ciencias de la Computación puedan comunicarse, enviar tareas, investigar, etc., en todo momento sin lugar a fallas con el fin de impulsar el aprendizaje.

Introducción

Los constantes avances y cambios tecnológicos obligan a la integración de la informática y de las telecomunicaciones, es por eso que nace el concepto de redes de computadores y de telecomunicaciones que no es más que la integración de dos o más unidades de procesamiento de información.
Es una disposición física de equipos de comunicaciones que permitan compartir el uso de recursos.

Apareció la necesidad de uniformizar los sistemas a través de los estándares que permitan la compatibilidad entre productos ofrecidos por diferentes fabricantes.
En 1985 se organizan comités técnicos para desarrollar estándares para cableado de telecomunicaciones, cuyo trabajo final se presentó el 9 de julio de 1991.

Se consideraron los siguientes conceptos:
• Los edificios son dinámicos
Durante la existencia de un edificio, las remodelaciones son comunes y deben ser tenidas en cuenta desde el momento del diseño.
• Los sistemas de telecomunicaciones son dinámicos.
Durante la existencia de un edificio, las tecnologías y los equipos de telecomunicaciones pueden cambiar drásticamente.
• Telecomunicaciones es más que "Voz y Datos"
El concepto de Telecomunicaciones también incorpora otros sistemas tales como control ambiental, seguridad, audio, televisión, alarmas y sonido.

Que es el Cableado Estructurado

Es el conjunto de elementos pasivos, flexible, genérico e independiente, que sirve para interconectar equipos activos, de diferentes o igual tecnología permitiendo la integración de los diferentes sistemas de control, comunicación y manejo de la información, sean estos de voz, datos, video, así como equipos de conmutación y otros sistemas de administración.
En un sistema de cableado estructurado, cada estación de trabajo se conecta a un punto central, facilitando la interconexión y la administración del sistema, esta disposición permite la comunicación virtualmente con cualquier dispositivo, en cualquier lugar y en cualquier momento.

El cableado estructurado trata de especificar una "Estructura" o "Sistema" de cableado para empresas y edificios que sea:
– Común y a la vez independiente de las aplicaciones
– Documentada (Identificación adecuada)
– Proyectada a largo plazo (> 10 años)

¿Por qué Cableado Estructurado?

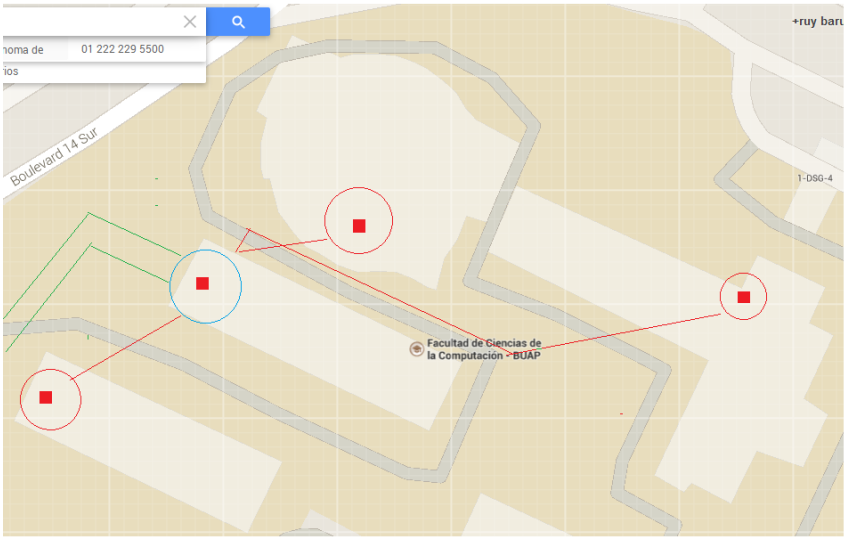
• Menores fallas en la red respecto a un sistema convencional, por lo tanto se tiene menos tiempos improductivos.
• El 40% de empleados que trabajan en un edificio se mudan cada año por lo que un sistema de cableado estructurado ofrece la simplicidad de la interconexión temporal para realizar estas tareas rápidamente, en vez de necesitar la instalación de cables adicionales.
• El costo inicial de un sistema de cableado estructurado puede resultar alto, pero este hará ahorrar dinero durante la vida útil del sistema.
• La administración y gestión de la red es sencilla.

Organismos y normas que rigen para el cableado estructurado

• ANSI: American National Standards Institute.
Organización Privada sin fines de lucro fundada en 1918, la cual administra y coordina el sistema de estandarización voluntaria del sector privado de los Estados Unidos.
• EIA: Electronics Industry Association.
Fundada en 1924. Desarrolla normas y publicaciones sobre las principales áreas técnicas: los componentes electrónicos, electrónica del consumidor, información electrónica, y telecomunicaciones.
TIA: Telecommunications Industry Association.
Fundada en 1985 después del rompimiento del monopolio de AT&T. Desarrolla normas de cableado industrial voluntario para muchos productos de las telecomunicaciones y tiene más de 70 normas preestablecidas.
• ISO: International Standards Organization.
Organización no gubernamental creada en 1947 a nivel Mundial, de cuerpos de normas nacionales, con más de 140 países.
• IEEE: Instituto de Ingenieros Eléctricos y de Electrónica.
Principalmente responsable por las especificaciones de redes de área local como 802.3 Ethernet, 802.5 Token Ring, ATM y las normas de Gigabit Ethernet.

Propuesta de Cableado Estructurado para la Facultad de Ciencias de la Computación

A continuación se muestra la vista aérea de la facultad que muestra la interconexión de edificios o cableado Backbone.



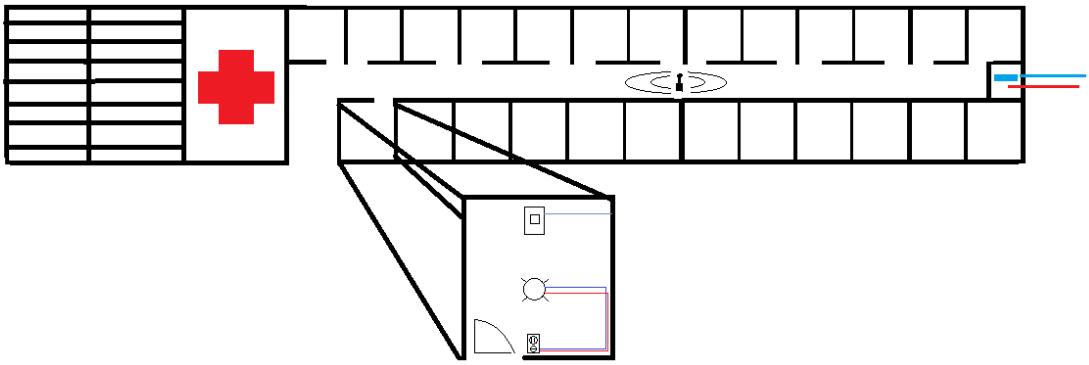
Edificio 104A

Diseño del Cableado Horizontal

Para el diseño del cableado horizontal en el edificio 104A de la facultad se considerara que este edificio tiene tres plantas, pero solo se requiere proporcionar red al primer piso, donde se encuentran los cubículos de profesores. En cada cubículo se necesita conexión cableada para dos usuarios usuarios que se encuentren en el pasillo puedan conectarse a la red inalámbrica.

y

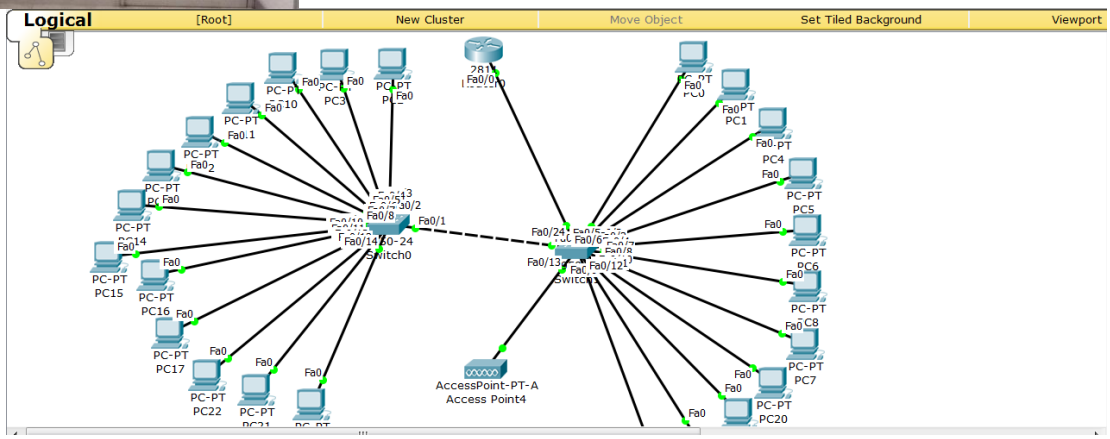
Distribución por cubículo



El cuarto de equipo se encontrara al fondo de los cubículos, como actualmente

El cable de red estará completamente alejado de la electricidad. La distancia será de una pared de los cubículos, por lo tanto no será afectado ningún cable.

Las ip del edificio 104 A serán estáticas, para el uso de los profesores.



Presupuesto

Para el Área de Trabajo por cada cubículo se utilizaran 2 Faceplate Dobles de color Blanco IFP 12W de la marca Hubbel, ya que contienen etiquetas de papel que cumplen con la norma ANSI/TIA-606-B para etiquetado de estación.

- Cisco Gigabit Ethernet Switch SF302-08P
- 10/100/1000Mbps
- 8 Puertos

- Tabla de direcciones MAC, 16.000 entradas
- $3851

Cisco Ethernet Router 1905/K9
2x RJ-45, 1x USB
Velocidad de transferencia de datos10, 100, 1000 Mbit/s
$11,208

Intellinet Panel de Parcheo
Categoria 5e
12 Puertos
$ 412

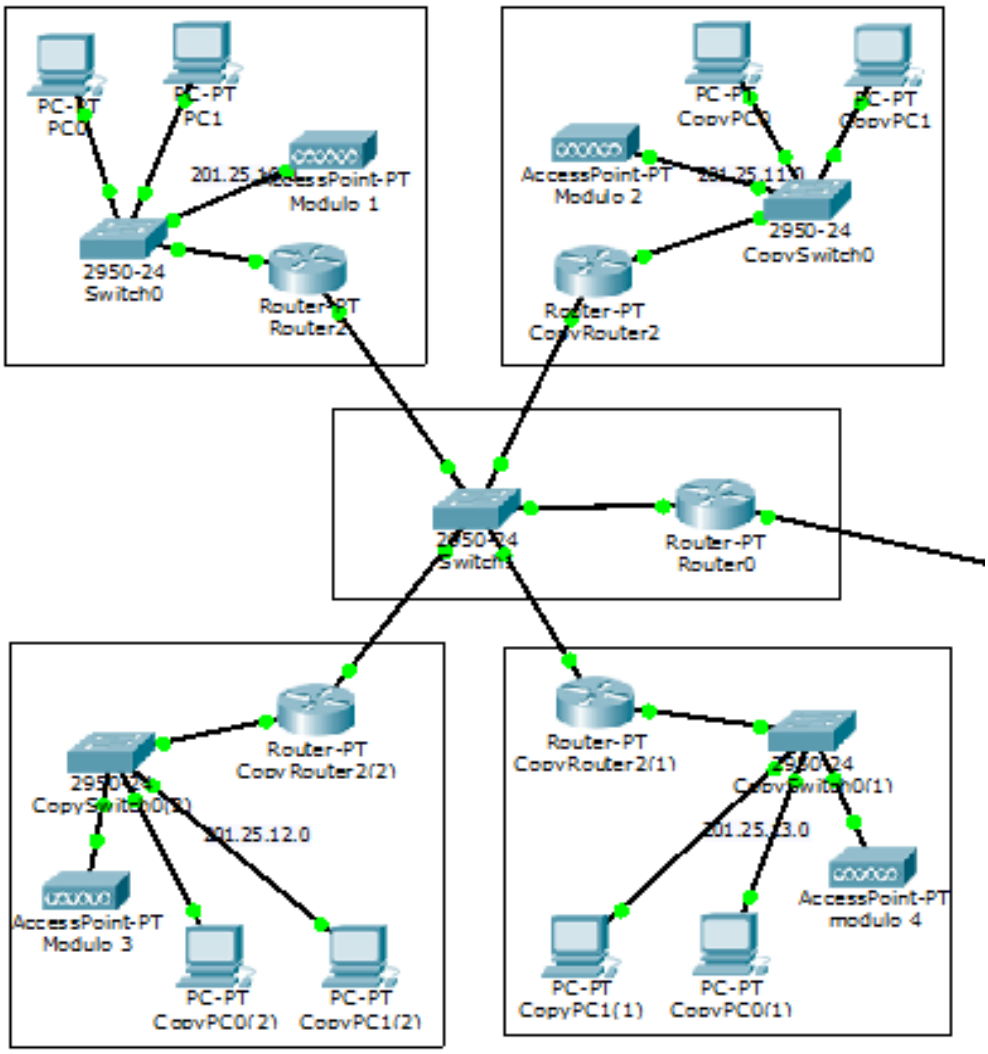
Rack de 4pies
Uso de 50 kg de capacidad de carga
$ 800

Esto equivale a un total de 46 faceplates
Además 2 Módulos RJ 45 Categoria 5e por cada Faceplate, es decir un total de 92 Jacks.
425 metros de cable UTP de 4 pares categoría 5e $765.5
1 acces point con soporte a 30 equipos. con un costo de $569.95 c/u.
80 metros de Canaleta lineal 20x10 marca Zoloda = $1600

- Total estimado = 19 206.45

Edificio 104 C

Esquema general



Cuarto de equipo

- En el cuarto de equipo llegaran las 4 señales de los 4 laboratorios, pasaran a travez de un switch y se conectara a un router el cual sera el encargado de distribuir la señal de internet a los 4 laboratorios
- Cisco Gigabit Ethernet Switch SF302-08P
- 10/100/1000Mbps
- 8 Puertos
- Tabla de direcciones MAC, 16.000 entradas
- $3851

- Cisco Ethernet Router 1905/K9
- 2x RJ-45, 1x USB
- Velocidad de transferencia de datos10, 100, 1000 Mbit/s
- $11,208

- Intellinet Panel de Parcheo
- Categoria 5e
- 12 Puertos
- $ 412
- Rack de 4pies
- Uso de 50 kg de capacidad de carga
- $ 800

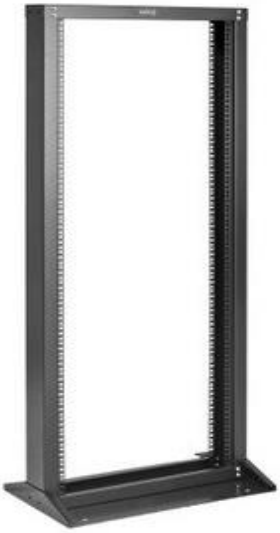
Closet de comunicaciones

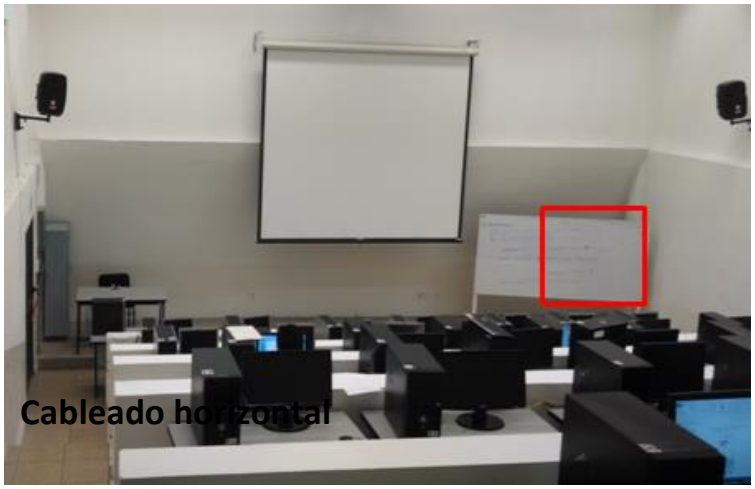
- Cisco Gigabit Ethernet Router RV180W
- Wifi 2.4GHz, 2 Antenas de 1.8dBi
- Puertos Ethernet 4
- $ 2686

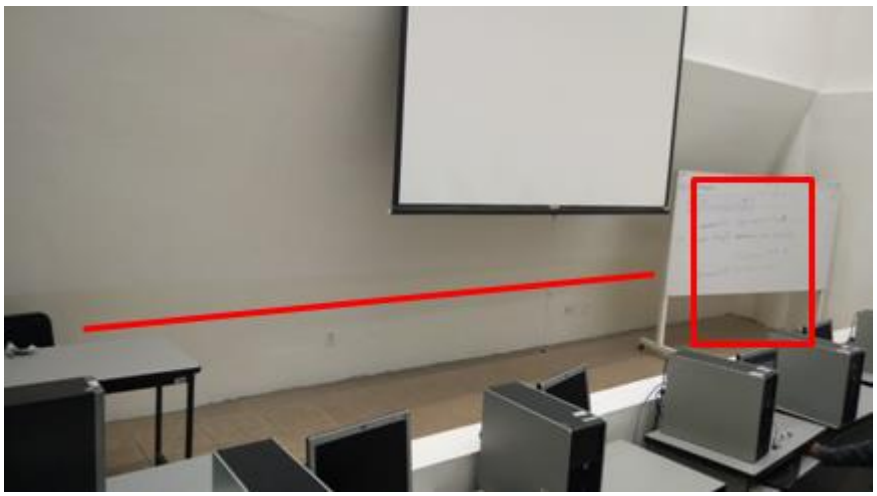
- TP-LINK Gigabit Ethernet Switch TL-SF1048
- 10/100Mbps
- 48 Puertos
- $ 1507

- Belkin Panel de Parcheo
- Categoria 5e
- 48 Puertos
- $ 1636

- Rack de 4pies
- Uso de 50 kg de capacidad de carga
- $ 800







Cable categoria 5e

5 filas * 8 salidas * 8 metros = 320 metros

5 salidas administradores * 8 metros = 40 metros
2 salidas profesores * 8 metros = 16 metros
6 m * 8 s + 8 m * 8 s + 10 m * 8 s + 12m * 8 s + 14m * 8 s + 16 m * 5 s = 400
776 metros => 800 metros
800 * 4 modulos = 3200 metros
$ 2 * 3200 metros = $ 6400

Cableado vertical
Cable categoria 6
200 metros * $ 15 = $ 3000

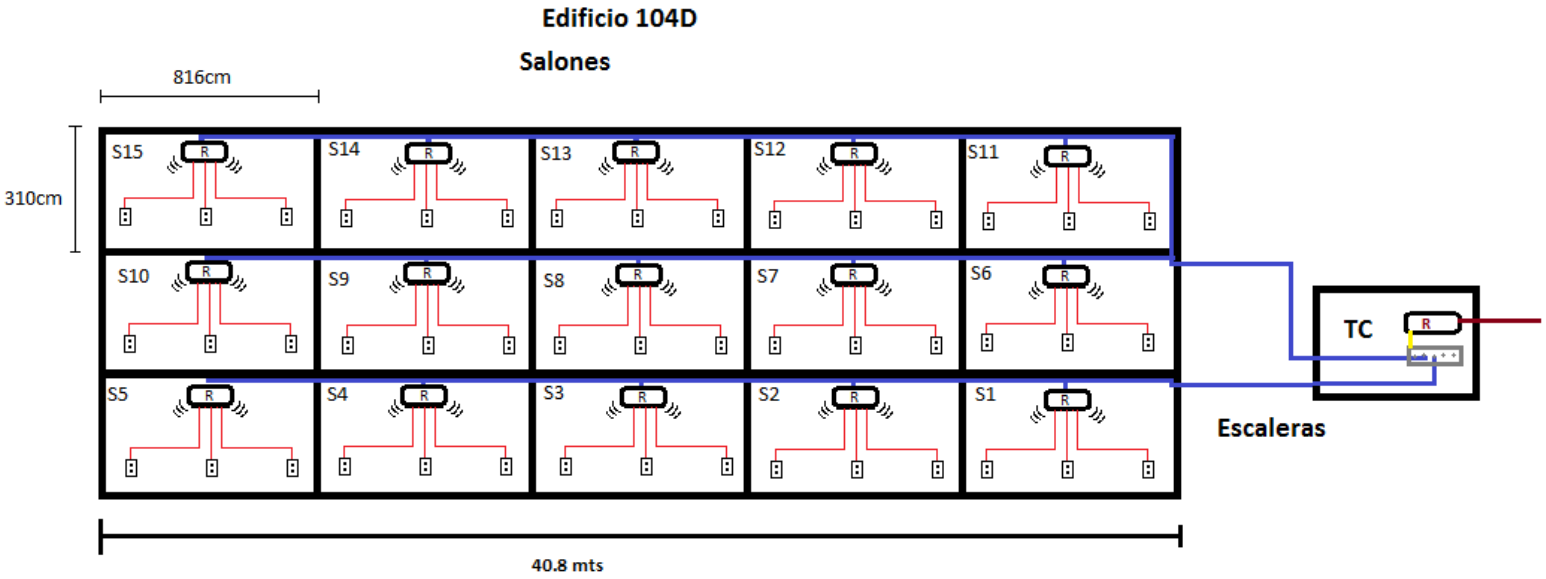
Presupuesto

Cuarto de equipo = $ 16 271
Closet de comunicaciones = 4 modulos *$ 6 629 = $ 26 516
Cableado horizontal = $ 6 400
Cableado vertical = $ 3000
Total estimado = $ 52 187

Edificio 104D

Diseño del Cableado Horizontal

Para el diseño del cableado horizontal en el edificio 104D de la facultad se considerara que este tiene tres plantas y en cada planta hay 5 salones. En cada salón se contemplan 35 usuarios; 32 de estos usuarios van a tener disponibilidad de conexión a internet mediante red inalámbrica WiFi, y los 3 restantes conectividad por red alámbrica.



Presupuesto

Para el Área de Trabajo por cada salón se utilizaran 3 Faceplate de color Blanco IFP 12W de la marca Hubbel, ya que contienen etiquetas de papel que cumplen con la norma ANSI/TIA-606-B para etiquetado de estación. Además 45 jacks Rj45 categoría 5e

- 45 faceplates. $30 c/u
- 45 Módulos RJ45 Categoría 5e. $115 x 5
- 38 conectores RJ45 $45.00
- 483 metros de cable UTP de 4 pares categoría 5e $180 x 100mts
- 290 metros de Canaleta lineal $ 50 x2.5 mts
- 15 Router Wifi Inalambrico Tp-link Wr741nd Clase N 150 Mbps $300 c/u
- 1 Router Balanceador De Carga Tp-link Tl-r470t+ $650 + $120 envío
- Rack Para Cableado Estructurado De 2.10m De Alto X 74 $ 1,000.00

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

NOMBRE: MANUEL ALEJANDRO JIMÉNEZ ORTEGA

PROYECTO: PRIMER PARCIAL

DOCENTE: IVAN OLMOS PINEDA

Introducción:

El proyecto trata sobre realizar un programa que calcule las latencias de un determinado grafo mediante un formato de entrada especificado (un archivo de tipo .txt) en el que recibirá como parámetros:

-1° Número de vértices y número de enlaces (arcos).

-2° Número de vértice inicial, número de vértice final, distancia (metros), velocidad (mbps)

-2.1 Este paso se repetirá dependiendo del número de enlaces en el archivo .txt

-3° TC1, TC2, TC3, … , TCN (donde N es el número de vértices)

-4° Vértice origen, vértice destino.

-5° Tamaño del archivo (GB)

Una vez capturados los datos antes mencionados, el programa debe arrojar como salida:

-La lista de todos los paths posibles.

-Tiempo de latencia calculada.

También deberá mostrar el path con menos latencia y el número de paquetes y el Tiempo Total de Transferencia.

Además el profesor especificó que el proyecto podía entregarse en el lenguaje Java o lenguaje C/C++.

Desarrollo del tema:

Lo primero que hice para mi programa fue crear una clase Pila llamada: StackX en donde coloqué métodos tradicionales aprendidos en el curso de estructuras de datos, como son: push, pop, tope y vacía. Posteriormente realicé una clase Vértice llamada Vertex, en donde solamente inicialicé una etiqueta para los nodos y una variable para saber cuándo ha sido visitado un nodo y cuándo no.

Después, escribí una clase Grafo llamada Graph, donde declaro las variables que utilizaré para los grafos, como el número máximo de vértices, la lista de vértices, la matriz de adyacencia, una variable para el número actual de vértices recorridos, una pila y una variable para imprimir el número de nodos.

En el constructor creo la matriz de adyacencia con los parámetros del máximo de vértices ([MAX_VERTS][MAX_VERTS]) y después inicializo la matriz de adyacencia en ceros.

Posteriormente creo un método donde leo un archivo para el archivo de entrada .txt y después un par de métodos para añadir vértice y añadir arista. (addVertex y addEdge).

Al final y como lo último que pude realizar, fue crear mi método del algoritmo de búsqueda (DFS) que es un recorrido a lo profundo.

Conclusiones del trabajo:

Como conclusiones, puedo decir que mi proyecto no alcanzó el objetivo especificado, ya que mis clases aún no lograban leer el grafo desde un archivo, sino que yo le introducía los vértices y las adyacencias que tiene cada nodo del grafo desde la clase Main implementada. Mostrando en pantalla los nodos

visitados en el grafo. Además tampoco pude implementar el método donde se deberían calcular las latencias del grafo, así como la división de los paquetes, los tiempos de cola y el tamaño del archivo en GB.

Bibliografía:

Me apoyé en las siguientes páginas para la lectura de ficheros para el archivo .txt de entrada y también para el algoritmo de búsqueda para los paths del grafo.

http://www.lawebdelprogramador.com/codigo/Java/2315-Leer_una_linea_de_un_archivo_y_separar_sus_palabras.html

http://chuwiki.chuidiang.org/index.php?title=Lectura_y_Escritura_de_Ficheros_en_Java

http://codebreakerscorp.wordpress.com/2011/03/05/algoritmo-de-busqueda-depth-first-search/

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Materia: Modelos de Redes

Integrantes:

Sarai Vanessa Santiago Inés	201023521
Daniel de San Gines Garcia Cortez	201228619
Héctor Ramírez Ruíz	201005044
Cinthia Flores Mendez	200937657
José Luis Quiroz Camargo	201023205

2do Examen

Profesor: Ivan Olmos Pineda

Fecha de entrega: 6 de Noviembre de 2014

Introducción

En este segundo proyecto de la materia de Modelo de Redes tenemos la tarea de presentar una propuesta para poder tener una cobertura de internet en la facultad de Ciencias de la Computación, de tal forma que los alumnos y docente puedan acceder de forma rápida a este medio.

Para poder realizar esta propuesta se tomará ayuda de:

- Un esquema general de la instalación.
- Equipos que se van a utilizar.
- Costos aproximados.
- Propuesta de configuración lógica (simulación en Packet Tracer).

Para realizar este proyecto usaremos una norma para una buena infraestructura de comunicación llamada ***"Cableado Estructurado"***, posteriormente se presentarán con detalle qué y cómo fue lo que se realizó en esta propuesta.

DESARROLLO

Esquema General de la Instalación

Para empezar a realizar este proyecto primero debemos saber con qué recursos se cuenta y a que destinos los tenemos que llevar. En la facultad de Ciencias de la Computación se cuenta con 4 edificios (104A, 104B, 104C y 104D) y cada uno de estos tiene sus propias características:

- 104A: Cuenta 23 Cubículos (donde se encuentran situados los profesores) y un *Closet de Comunicaciones* que se encuentra
- 104B: Cuenta con un *Closet de Comunicaciones.*
- 104C: Cuenta con 4 salones/laboratorios
- 104D: Cuenta con 3 pisos/plantas donde cada piso tiene cuatros salones.

Tanto como en el edifico A y B llegan dos puntas de *Fibra Óptica* que vienen de la *Biblioteca Central* de la Universidad, de las cuales dependemos para lograr los objetivos de este proyecto.

Empezamos con el edifico ***104A***, este edificio cuenta con 23 cubículos, en cada cubículo hay 2 profesores, en consecuencia se deben conectar 2 máquinas de manera física, pero para este proyecto en todas las conexiones se debe tomar al menos en cuenta un 30% de redundancia (salidas extra), entonces por cubículo deber contar con al menos 3 salidas, haciendo uso de la norma de cableado estructurado y recursos que ya tenemos, nosotros decidimos que reutilizaremos una *canaleta* que ya está ubicada en el edificio y que está dedicada solo para cables de red, y ahí colocar todos nuestros cables a utilizar, solo habría tres cambios:

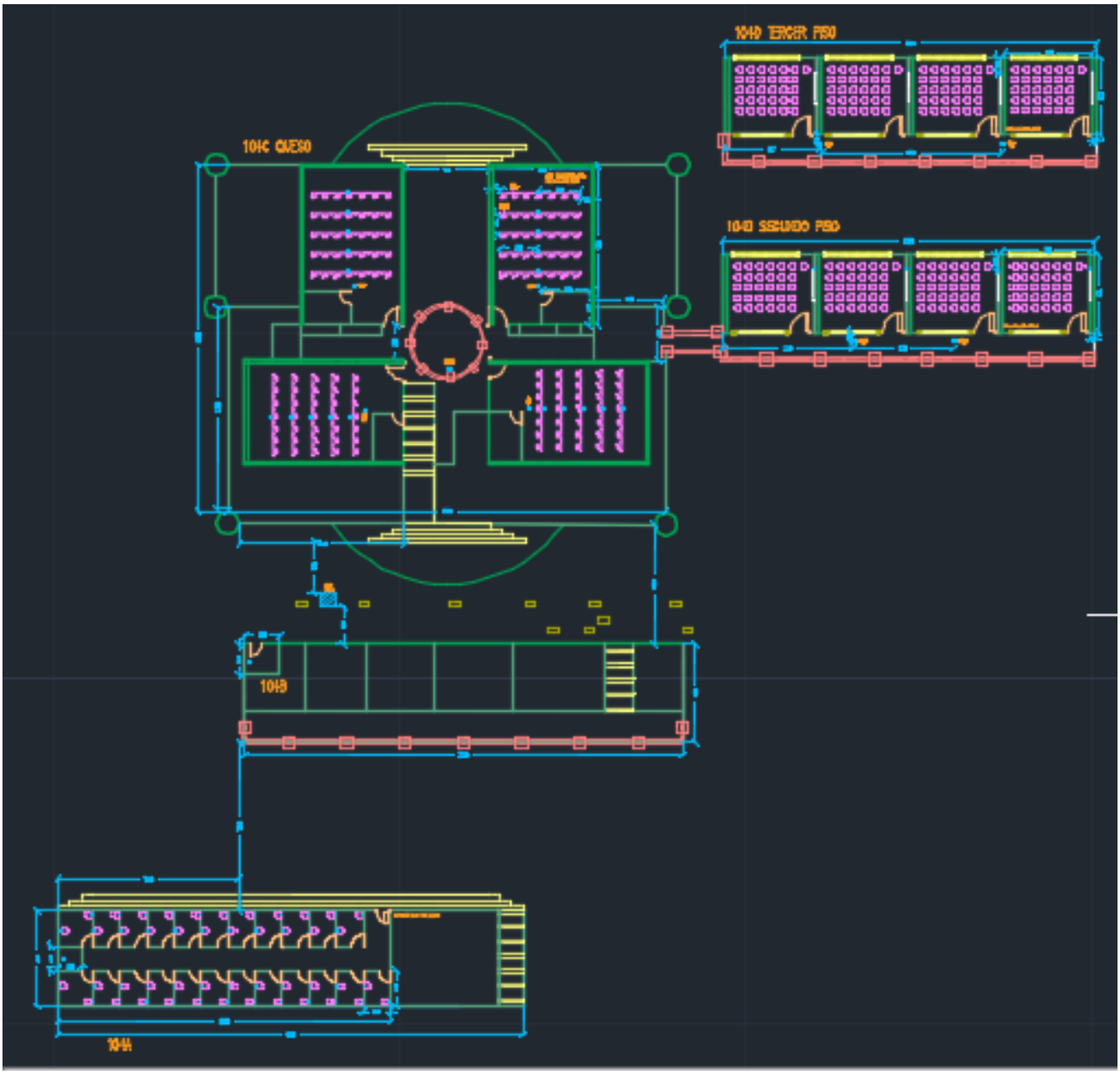
1. Cambio de lugar de las placas con servicios, las cuales se encuentran muy cerca de conexiones eléctricas, por lo tanto violarían las normas del cableado estructurado.
2. Quitar una impresora multifuncional que se encuentra dentro del *Closet de Comunicaciones*.
3. Remover y minimizar el uso de multicontactos en el *Closet de Comunicaciones* ya que también violaría una de las normas del cableado estructurado.

Continuamos con el edifico ***104B***, este edifico solo cuenta con un *Closet de Comunicaciones* que está totalmente ordenado y cumple con todas las especificaciones, para este edificio no habría ningún cambio.

Seguimos con el edifico ***104C***, este edifico cuenta con 4 salones/laboratorios, donde casa salón tiene 5 columnas donde se ponen las computadoras, cada columna debe tener 5 máquinas + 30% de redundancias, es decir, 7 máquinas por columna, en total serían 35 salidas físicas por cada laboratorio, además de que cada salón debe contar con un *Access point* para el uso del internet inalámbrico, también este edificio tiene un sótano, donde podremos nuestro *Closet de Comunicaciones* y de ahí enviar los cables a cada uno de los salones.

Terminamos con el edifico ***104D***, este edifico cuenta con 3 pisos, donde casa piso tiene 4 salones, la idea principal para este edificio es que debemos colocar en cada uno de los pisos *Access point* y conexiones físicas para computadoras (al menos 2 salidas), para cada salón se tomó en cuenta la conexión de 35 alumnos por internet inalámbrico + 30% de redundancia, es decir, 46 alumnos por salón, 184 alumnos por piso y un total de 552 personas por el edificio.

Se muestra el boceto de un croquis representando a los edificios de la facultad.



Para la conexión de los edificios C y D se tomó en cuenta la siguiente propuesta:

- Del *Rack* que se encuentra dentro del *Closet* del Edificio A (donde llega una de las puntas de la *Fibra Óptica*), vaya una conexión al edificio C.
- Del *Rack* del edificio B (donde llega la otra punta de la *Fibra Óptica*), se vaya al edifico D.

Especificación de equipos a utilizar y costos

Después de haber realizado una investigación en diferentes tiendas y páginas web, llegamos a la conclusión de utilizar los siguientes equipos y materiales, las cuales tienen sus propias características:

Gabinete/Rack:

Rack Tripp Lite de 2 postes de 19"

$2,509.00 (IVA incluido)

Fabricante: Tripp-Lite
Modelo: SR2POST25

Características principales:

- Soporta 800 libras de equipo de montaje en rack
- Con textura de pintura en polvo acabado
- Resistente de aluminio de la construcción
- Numeradas las posiciones de montaje
- Bolt se establecen las disposiciones
- Alquiler de diseño cuadratura
- Acabado en negro
- Fácil montaje
- Cinco años de garantía de reparación o reemplazo

Charola:

Charola Fija StarTech.com 1U

$489.00 (IVA incluido)

Fabricante: StarTech.com
Modelo: CABSHELF1U

Características principales:

- Soporta 15 kilos de equipo de montaje en rack
- Con textura de pintura en polvo acabado
- 7" de Profundidad

Panel de Parcheo:

Panel de parcheo de 24 puertos Categoría 6, para montaje en rack

$720.00 (IVA incluido)

Fabricante: Steren
Modelo: Modelo: 360-324

Características principales:

- Protocolo IEEE 802.3
- 1000BASE-T Gigabit Ethernet
- 100BASE-T
- 10BASE-T
- 24 puertos

- Cumple con los estándares ANSI/EIA/TIA
- Compatible con sistemas de cableado de Categoría 3, 5 y 5e
- Retardante al fuego UL 94 V-0
- Conductores sólidos terminados en 22-26 AWG
- Fabricado en acero de bajo carbón calibre 16 AWG
- **Consumo nominal:** (no aplica)
- **Consumo en espera:** (no aplica)
- **Color:** negro
- **Peso:** 553grm

RJ45 Hembra:

Charola Fija StarTech.com 1U

$35.00 x2 (IVA incluido)

Placa Keystone de dos cavidades, para conector jack RJ45, color blanco

$12.00 (IVA incluido)

Caja de registro sencilla

$39.00 (IVA incluido)

Switch:

Switch Cisco SF100 24 puertos

$1,579.00 (IVA incluido)

Fabricante: Cisco
Modelo: SF100-24

Características principales:

- Modelo: SF100-24-NA
- 24 puertos 10/100
- Garantía de por vida

Switch Cisco SF100D-08 puertos

$659.00 (IVA incluido)

Fabricante: Cisco
Modelo: SF100D-08-NA

Los switches Cisco de la serie 100 brindan:

- Capacidades de alto rendimiento: con un potente rendimiento de red a un precio accesible, puede aumentar la velocidad y capacidad de su red para brindar soporte a aplicaciones de alto consumo de ancho de banda.
- Compatibilidad con tecnologías avanzadas: inteligencia de calidad de servicio (QoS, quality of service) integrada en todos los modelos que ayuda a un rendimiento uniforme de la red y mantiene la eficiencia de las aplicaciones.
- Una solución ecológica: optimiza el uso de energía para ser más eficiente energéticamente sin afectar el rendimiento.
- Asequibilidad: diseñada especialmente para empresas en crecimiento que necesitan una red básica con características automatizadas para estar en funcionamiento en minutos.
- Tranquilidad: todos los switches Cisco de la serie 100 cuentan con una garantía de hardware limitada de por vida de Cisco, que los protege durante toda su vida útil.
- Facilidad de uso: viene listo para funcionar, sin software que instalar ni nada que configurar.

Router:

Router Cisco 16 puertos

$6,149.00 (IVA incluido)

Fabricante: Cisco
Modelo: RV016

Características

- Firewall SPI para máxima seguridad
- Switch 10/100 de 16 puertos que admite interfaz dependiente del medio (MDI) e interfaz cruzada dependiente del medio (MDI-X), y una capacidad de transferencia de hasta 200 Mbps por puerto
- 5 de los 16 puertos se pueden configurar como puertos WAN/LAN
- Dos puertos WAN dedicados para conectividad a Internet de carga equilibrada
- Análisis del correo electrónico y URL dinámicas a través del Servicio de Seguridad Trend Micro ProtectLink Gateway (opcional)
- Capacidad de VPN IPsec (IP security, seguridad de IP) completa mediante el cifrado DES (Data Encryption Standard, norma de cifrado de datos) y 3DES (triple DES)
- Compatibilidad con los algoritmos de autenticación MD5 y SHA.
- Hasta 100 túneles de IPsec VPN simultáneos permitidos
- Gestión a través de Internet, protocolo SNMP (Simple Network Management Protocol, protocolo de gestión de red simple) y asistente de configuración para facilitar la conexión a los administadores
- Capacidades de gestión del ancho de banda para ofrecer mejor QoS (Quality of Service, calidad de servicio)
- Hasta 50 usuarios de QuickVPN admitidos

Puntos de Acceso:

Punto de Acceso con PoE y Banda Seleccionable Cisco WAP321 Wireless-N

$2,779.00 (IVA incluido)

Fabricante: Cisco
Modelo: WAP321 Wireless-N

Características

- Ofrece conectividad inalámbrica 802.11n de gran ancho de banda de banda seleccionable para un máximo
- Admite conexiones de alta velocidad con interfaz de LAN Gigabit Ethernet para aplicaciones exigentes
- La configuración de un solo punto simplifica la implementación de varios puntos de acceso inalámbricos
- Conecta las redes LAN cableadas de manera inalámbrica para reducir los costos de cableado e instalación
- Fácil de configurar y administrar mediante el uso del asistente de configuración
- Protege la información comercial con seguridad mejorada, incluidos el cifrado avanzado, la autenticación Segura y la detección de puntos de acceso dudosos.

Cable:

Cable UTP Categoría 6

$139.00 USD

Fabricante: Optronics
Modelo: OPCABOCAT6S

Características

- Bobina 305m color gris, azul o negro

Canaleta:

Canaleta auto adherible, de 2 m, para cable de 20 mm

$59.00 (IVA incluido)

Fabricante: Steren
Modelo: Modelo: 370-401

Características

- **Longitud:**2 m
- **Consumo nominal:** (no aplica)
- **Consumo en espera:** (no aplica)
- **Color:** gris
- **Peso:** 341.1 grm

Canaleta auto adherible, de 2 m, para cable de 40 vías

$140.00 (IVA incluido)

Fabricante: Steren
Modelo: 3 70-501

- **Longitud:**2 m
- **Consumo nominal:** (no aplica)
- **Consumo en espera:** (no aplica)
- **Color:** gris
- **Peso:** 912 grm

Para calcular el costo total aproximado de todo la instalación se debe tomar en cuenta:

- Total de metros de cable utp cat. 6 que se utilizará.

- Número total de switch’s a utilizar.
- Número total de Rj45 Hembra y de Placas con servicios.
- Total de Canaleta.

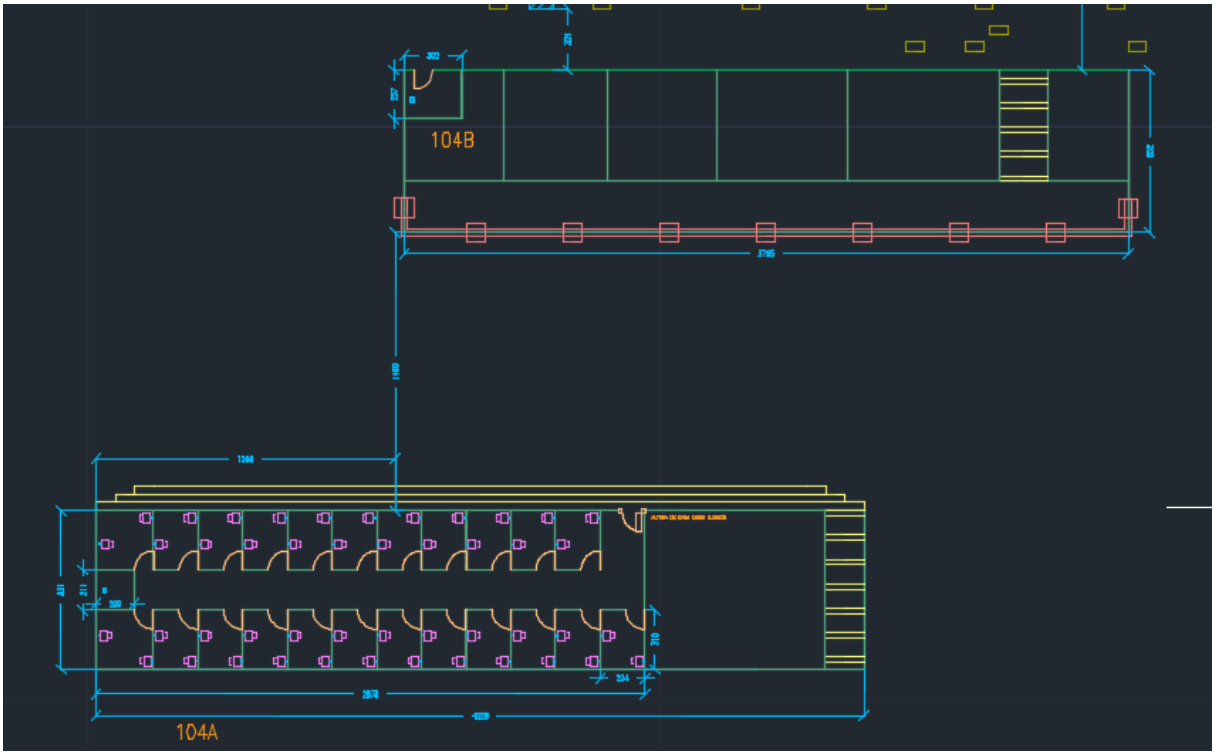
Material	Cantidad	Subtotal	Total
Cisco SF100 24 puertos	11	$1,579.00	$17,369.00

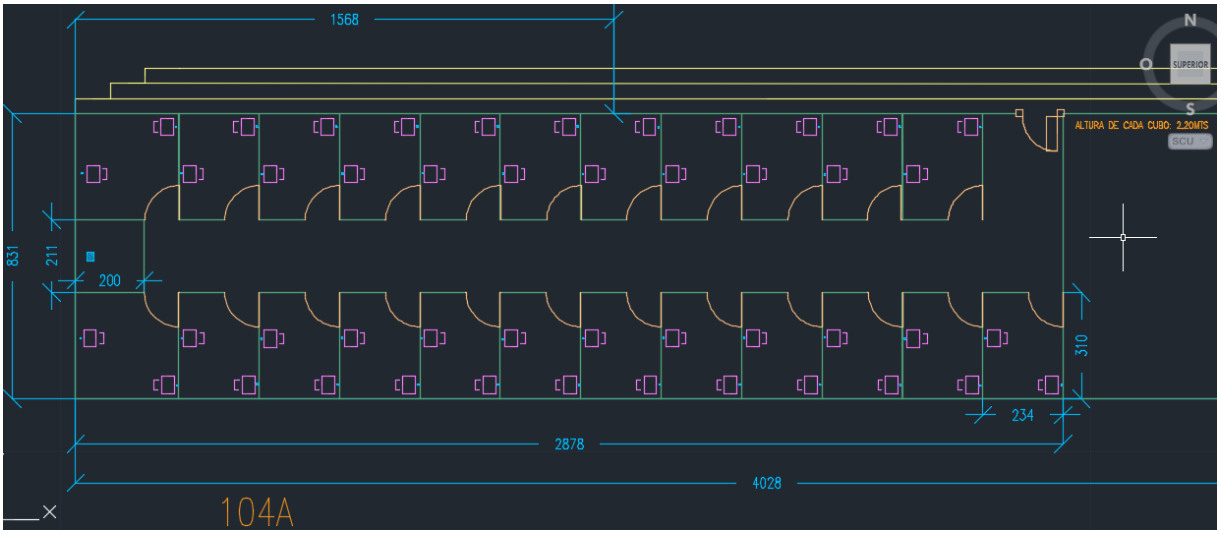
Switch Cisco SF100D-08 puertos	20	$659.00	$13,180.00
Router Cisco 16 puertos	5	$6,149.00	$30,745.00
Rack Tripp Lite de 2 postes	5	$2,509.00	$12,545.00
Charola Fija StarTech	20	$489.00	$9,780.00
Panel de parcheo de 24 puertos Categoría 6	14	$720.00	$10,080.00
Cable UTP Categoría 6	10	$1,946.00	$19,460.00
Canaleta de 20mm (2mts)	10	$59.00	$590.00
Canaleta para cable de 40 vías (2mts)	12	$140.00	$1,680.00
RJ45 Hembra de 2 salidas	210	$70.00	$14,700.00
Cajas y Tapas	210	$51.00	$10,710.00
Punto de Acceso con PoE y Banda Seleccionable Cisco	11	$2,779.00	$30,569.00

Costo total aproximado: $171,408.00

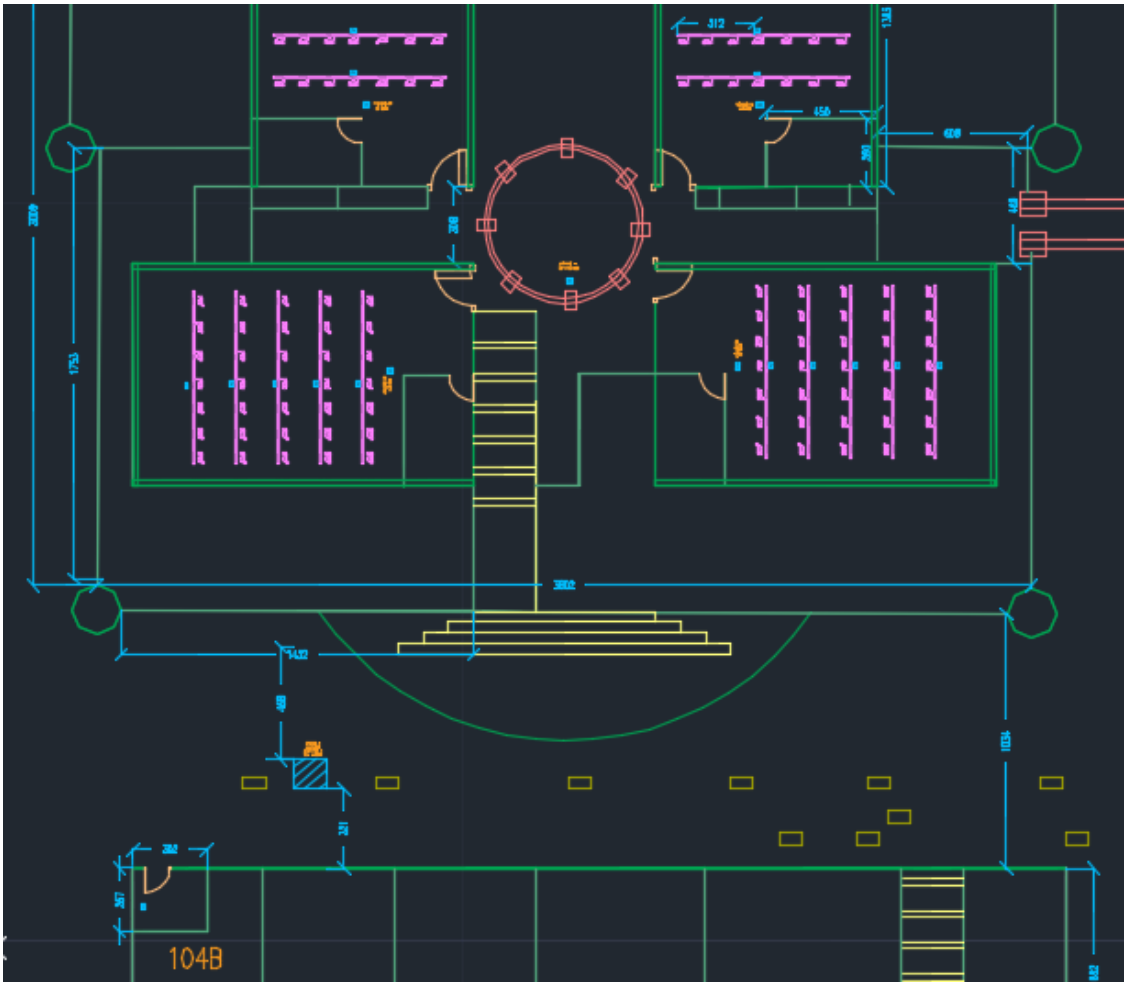
A continuación se mostrarán algunos planos de los edificios con medidas, las cuales fueron tomadas en cuenta para el presupuesto.

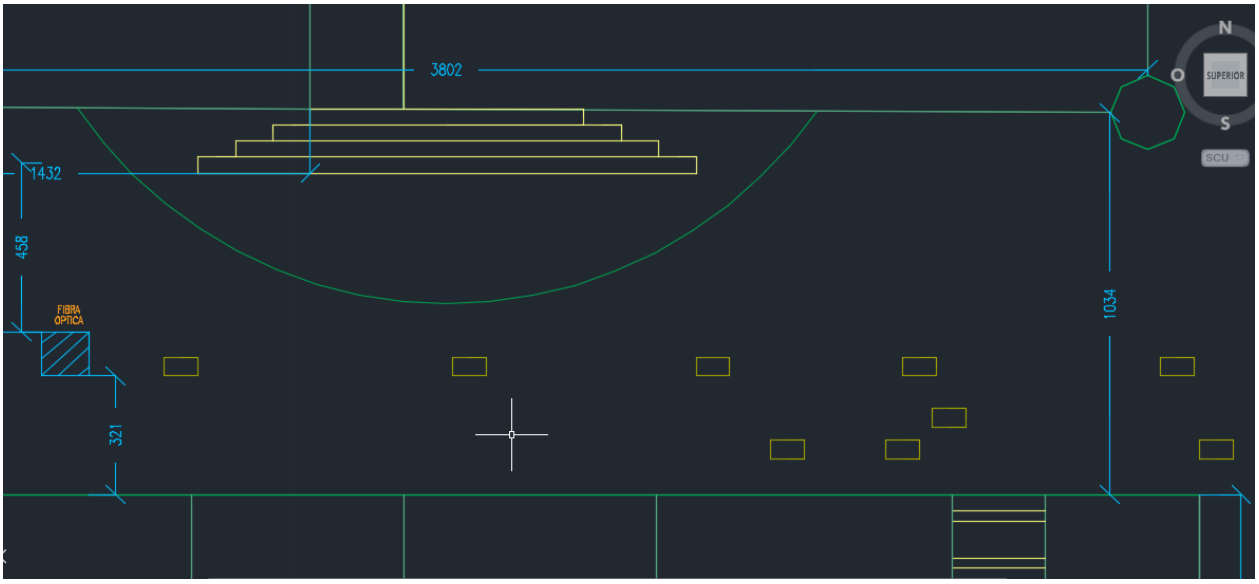
Edificio 104A

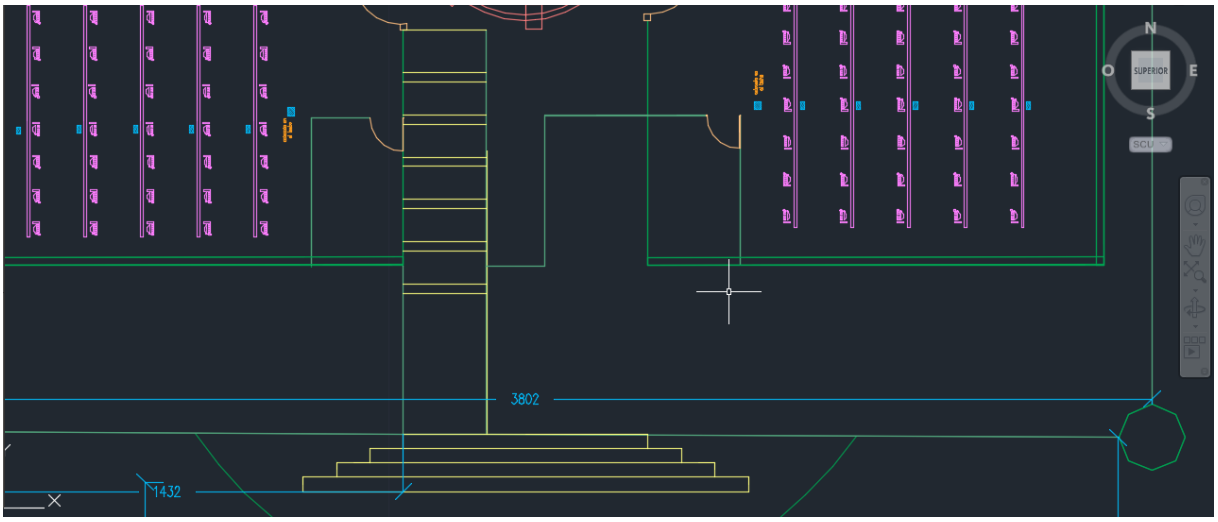


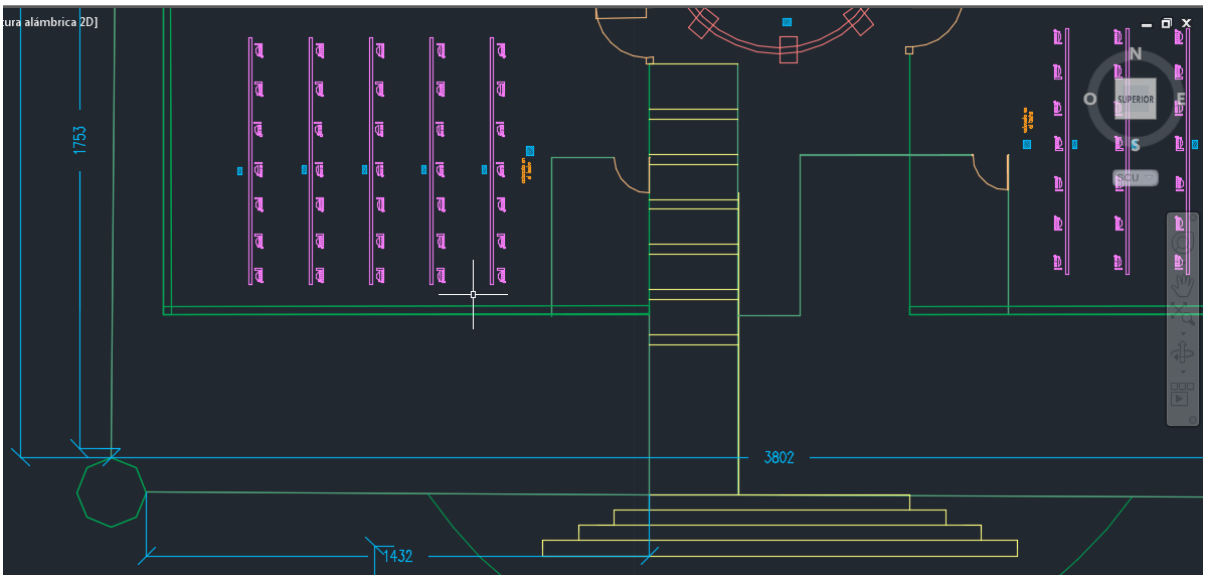


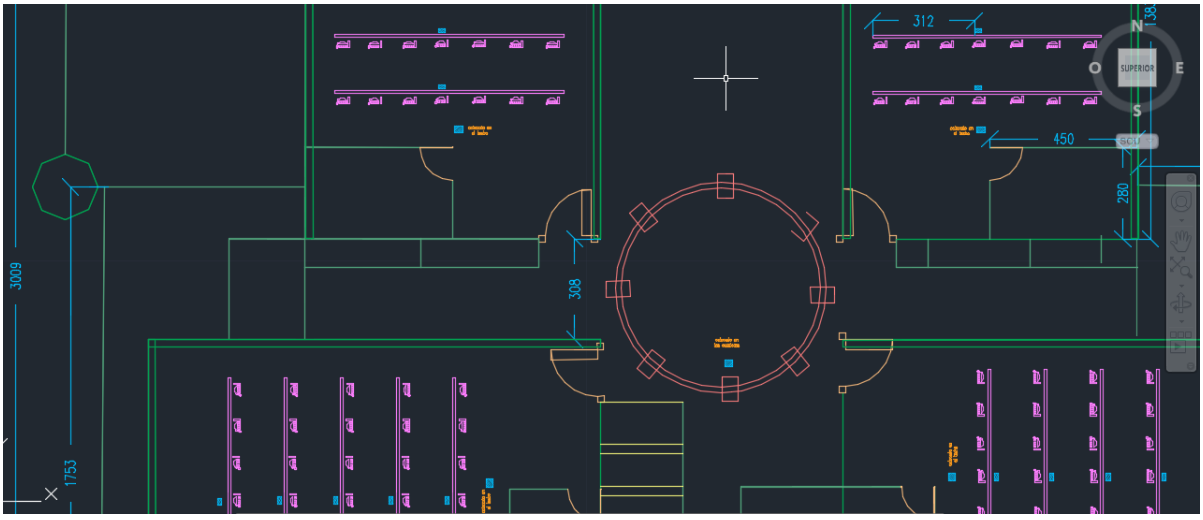
Edificio 104B y Edificio 104C

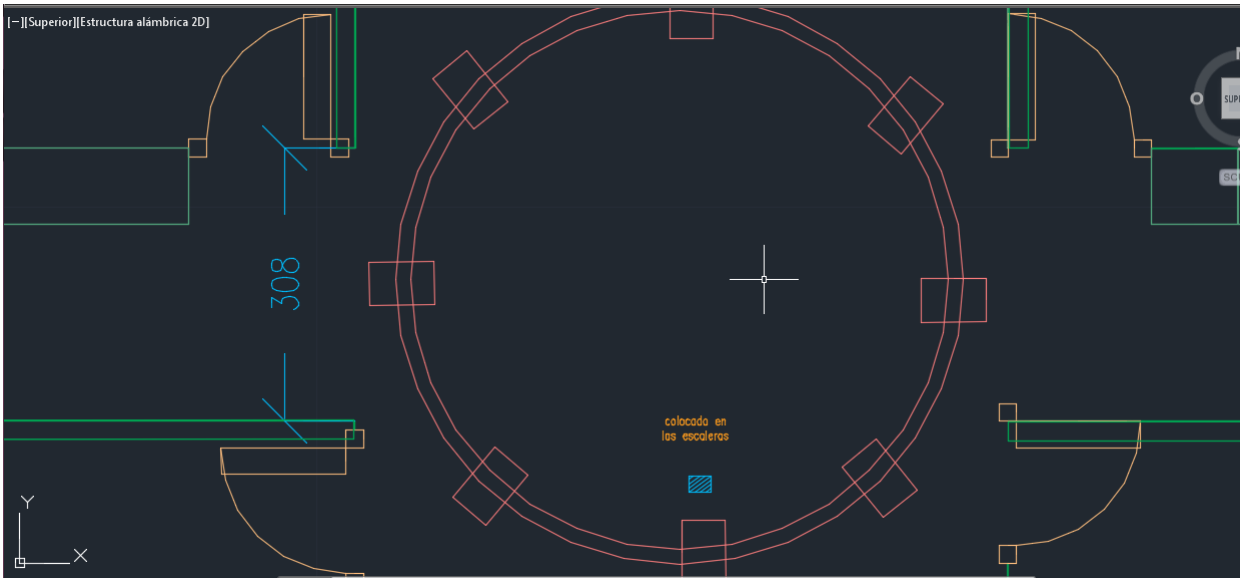

104B

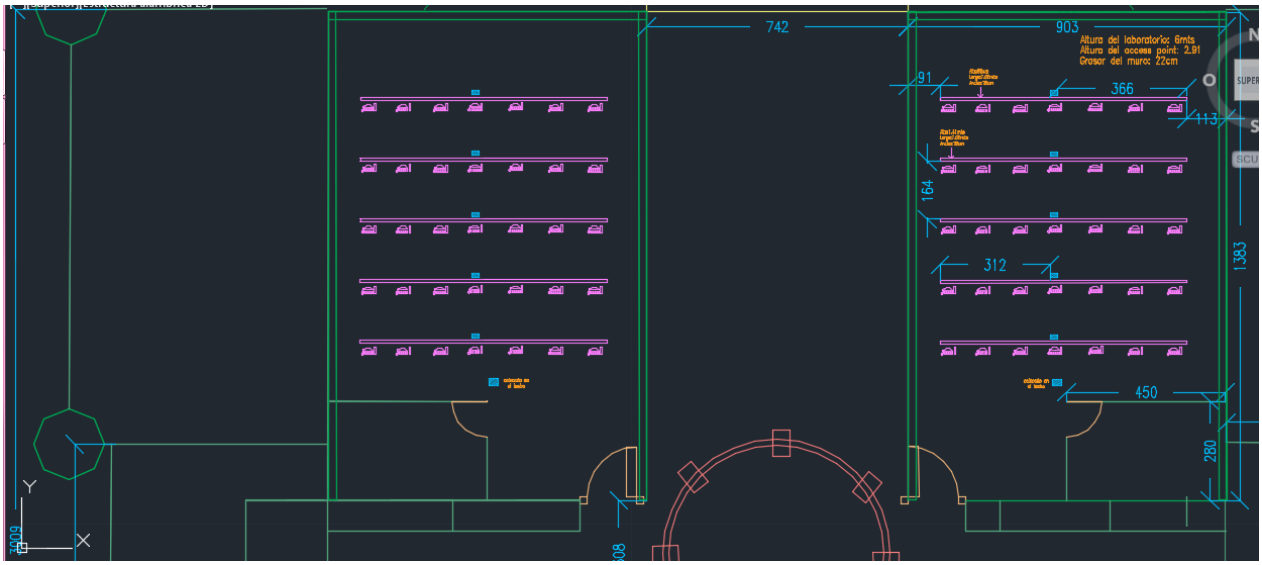

3802
1432
458
FIBRA OPTICA
321
1034
N
O
SUPERIOR
S


N
O
SUPERIOR
E
S
3802
1432


ura alámbrica 2D]
1753
N
O
SUPERIOR
E
S
3802
1432

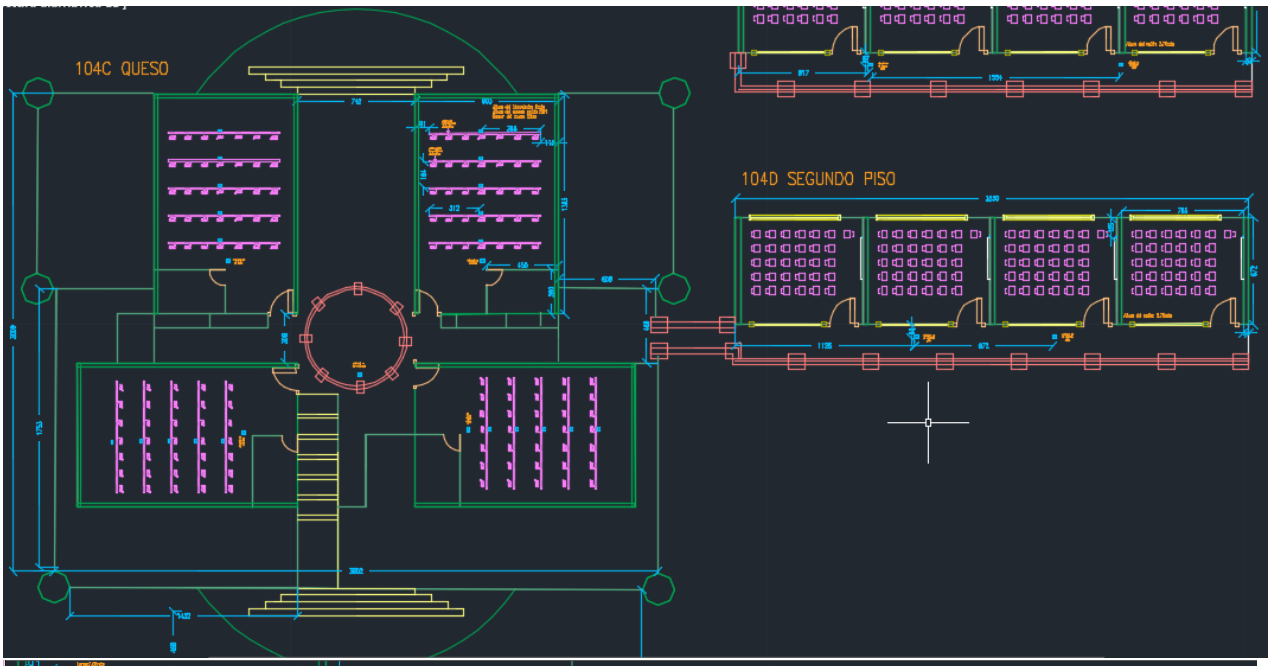

312
N
O
SUPERIOR
E
S
450
280
3009
308
1753

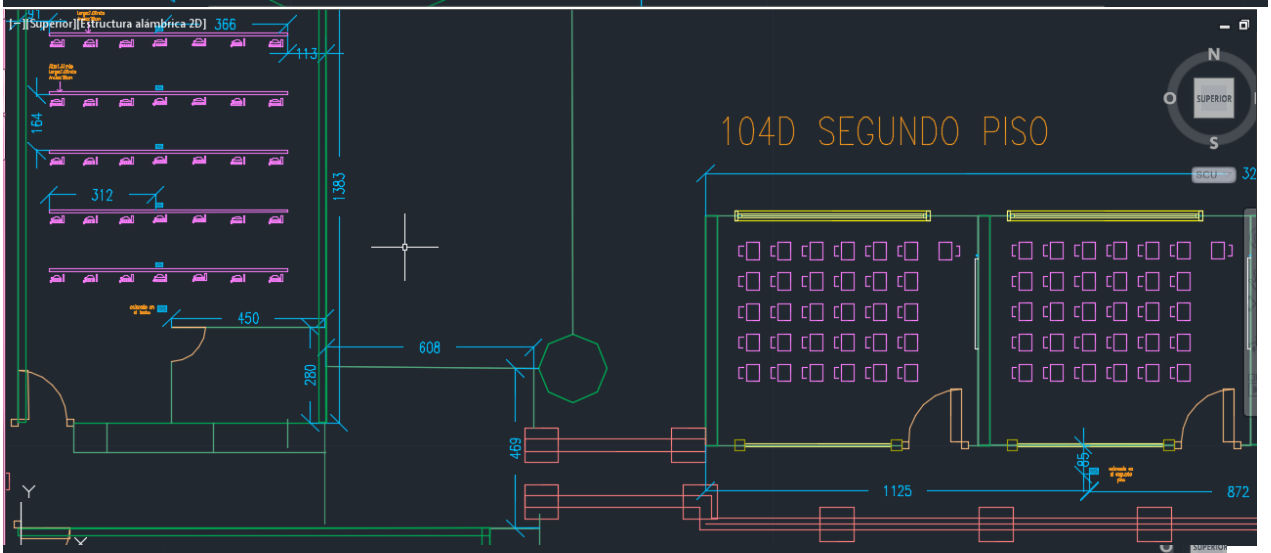

[−][Superior][Estructura alámbrica 2D]
308
colocado en
las escaleras
Y
X

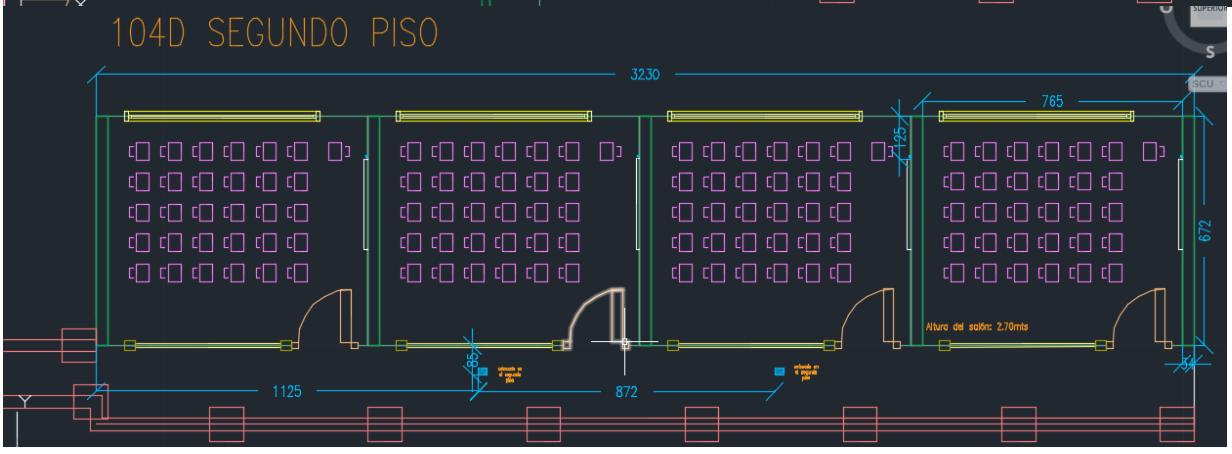

742
903
Altura del laboratorio: 6mts
Altura del access point: 2.91
Grosor del muro: 22cm
91
366
113
164
312
1383
450
280
Y
X

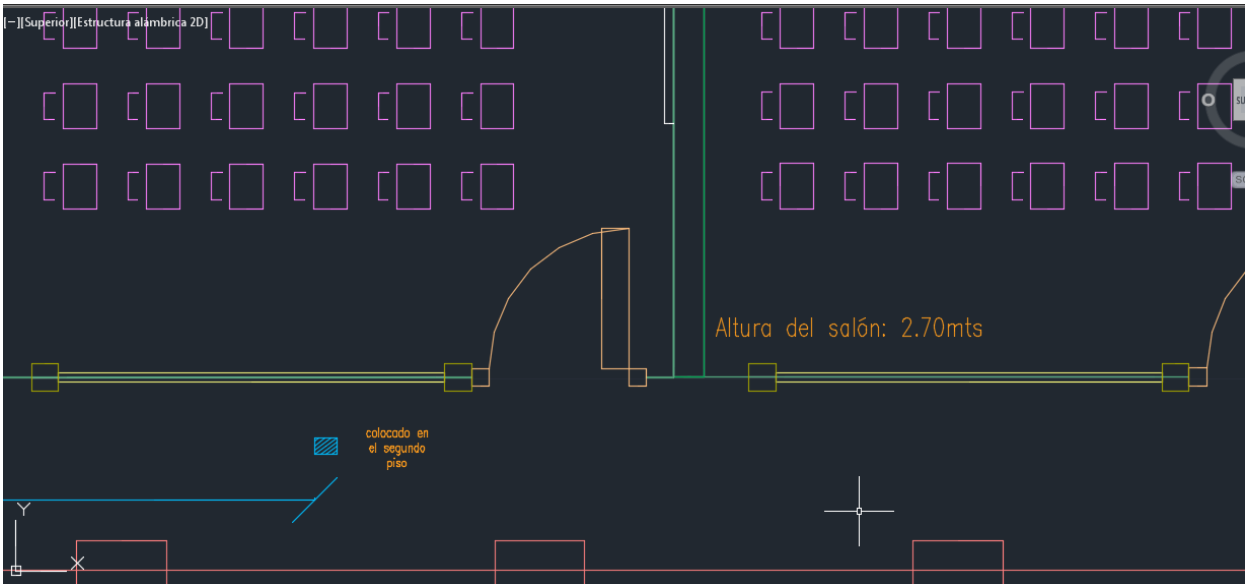

903
Altura del laboratorio: 6mts
Altura del access point: 2.91
Grosor del muro: 22cm
91
Alto:80cm
Largo:7.00mts
Ancho:18cm
366
113
Alto:1.41mts
Largo:7.00mts
Ancho:18cm
164
Terminado.

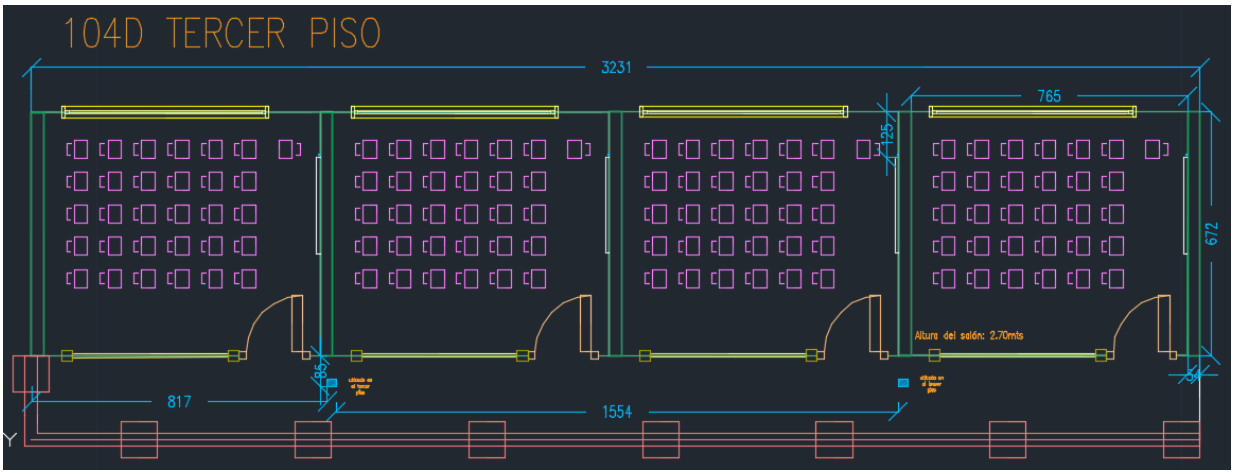
Edificio 104D

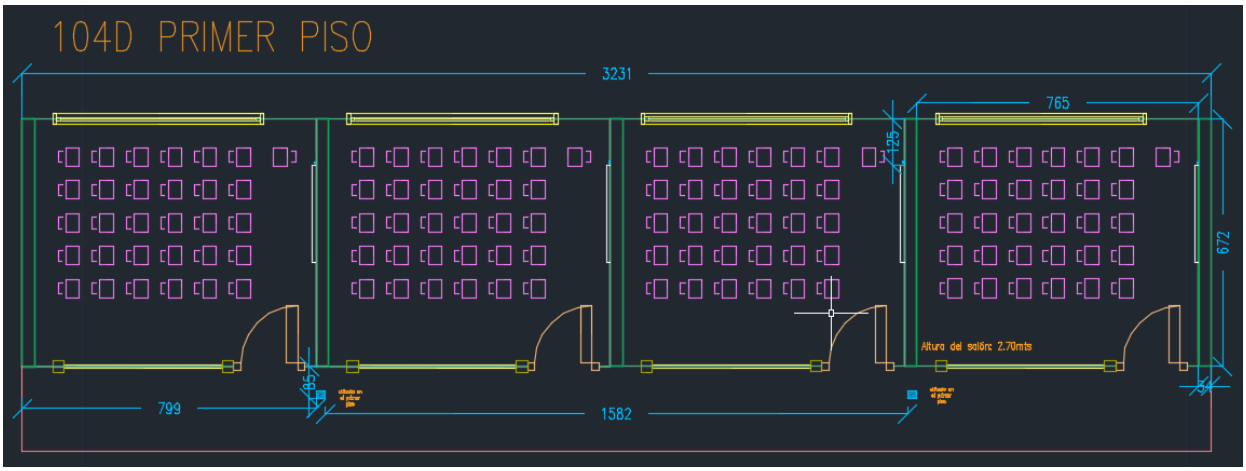






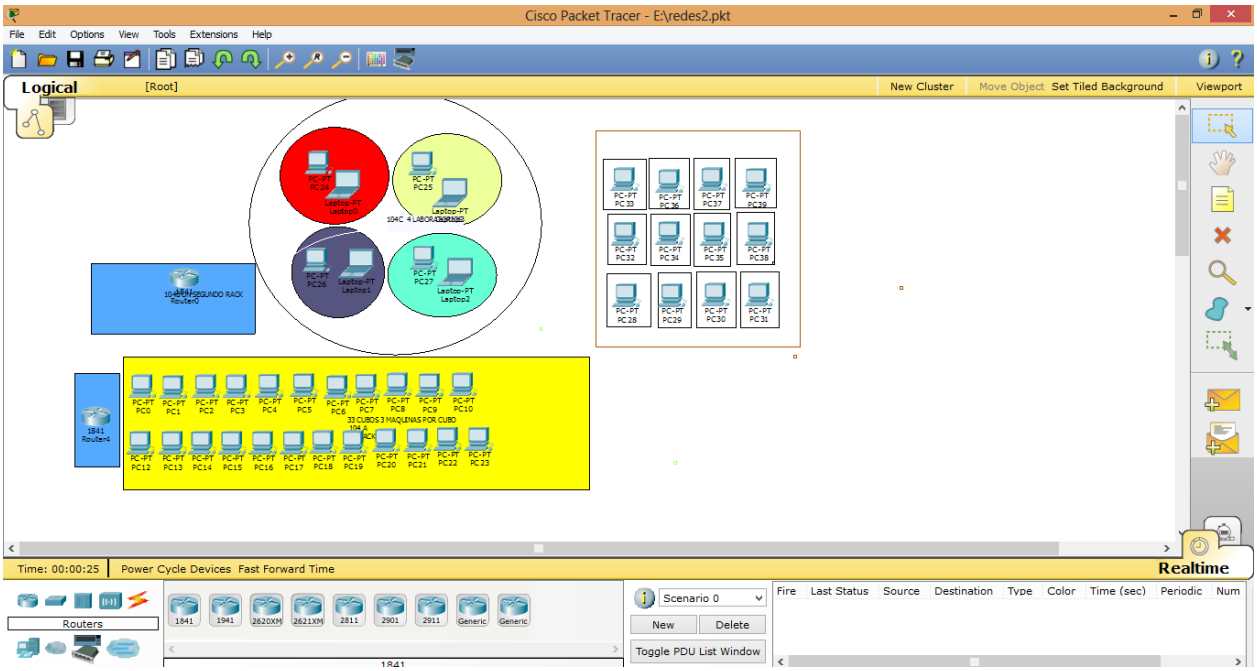






Propuesta de Configuración Logica

Después de haber realizado un esquema general de instalación y la especificación de materiales y costos, nosotros realizamos una simulación de asignación de IP’s en el programa Packet Traser, el cual se presentaran capturas de pantalla de la simulación.



CONCLUSIONES

Después de haber realizado un esquema general, investigación y especificación de costos y simulación de una configuración lógica terminamos este proyecto, concluyendo que debido a toda la indagación que se realizó, pudimos saber más sobre muchas cosas relacionadas al área de redes, como los diferentes usos de routers y switch's, diferencia entre algunos cables (UTP cat.5e, UTP cat.6 y Fibra Óptica) y una mejor retroalimentación del uso de Packet Traser.

También recalcamos las definiciones y uso de las normas del cableado estructurado.

°

REDES PRIMAVERA 2015

BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACION

PRIMAVERA 2015

MODELO DE REDES

REPORTE DE MEJORA EN LA RED DE LA FACULTAD DE CIENCIAS DE LA COMPUTACION.

Cableado Estructurado

Un cableado estructurado es un sistema de red de cables, conectores y demás dispositivos de infraestructura flexible con los cuales podemos unir dos o más puntos de un sistema de computación en red de una forma universal dentro de un edificio para diferentes tipos de comunicaciones (de red) como de voz, datos o imágenes, al igual que soportar implementaciones y mejoras de tecnologías (hubs o concentradores, switches o conmutadores, routers o enrutadores, etc).

Ventajas y beneficios del cableado estructurado
–Facilita un mantenimiento económico, sencillo y confiable.
–Proporciona seguridad de acceso para la administración del sistema.
–Soporta todas las tecnologías actuales y futuras por al menos 10-15 años.
–Existen normas e instrumentos que garantizan la calidad de la red instalada.
–Es de fácil administración.
–Permite cambios rápidos y sencillos.
–Posibilita ampliaciones económicas.
–Permite convivir muchos servicios en red (voz, datos, vídeo, etc.) con la misma instalación,
–Se facilita y agiliza mucho las labores de mantenimiento.

Switch

¿Qué es un Switch?

Son los encargados de la **interconexión de equipos dentro de una misma red**, o lo que es lo mismo, son los dispositivos que, junto al cableado, constituyen las redes de área local o LAN.

¿Para qué sirve un switch?

La función básica de un switch es la de **unir o conectar dispositivos en red**. Es importante tener claro que un switch **NO** proporciona por si solo conectividad con otras redes, y obviamente, **TAMPOCO** proporciona conectividad con Internet. Para ello es necesario un router.

Clasificacion de Switches

Switch troncal / switch perimetral

El término **switch troncal** se refiere a los que se utilizan en el núcleo central (core) de las grandes redes. Es decir, a estos switches están conectados otros de jerarquía inferior, además de servidores, routers WAN, etc. Por otro lado el término **switch perimetral** se refiere a los utilizados en el nivel jerárquico inferior en una red local y a los que están conectados los equipos de los usuarios finales.

Switch gestionable (managed) / switch no gestionable (unmanaged)

El término **gestionable (managed)** se refiere a los switches que ofrecen una serie de características adicionales que requieren de configuración y gestión. Por el contario los switches **no gestionables (unmanaged)** suelen ser los que ofrecen funcionalidades básicas que no requieren procedimiento de configuración o gestión.

Tipos de switches

- **Switches desktop**

Este es el tipo de switch más básico que ofrece la función de conmutación básica sin ninguna característica adicional. Su uso más habitual es en redes de ámbito doméstico o en pequeñas empresas para la interconexión de unos pocos equipos, por lo que no están preparados para su

montaje en rack 19''. Estos switches no requieren ningún tipo de configuración, ya que utilizan el modo de ***autoconfiguración*** de Ethernet para configurar los parámetros de cada puerto.

Las principales características son:

El numero de puertos es de 4 a 8 RJ-45

Configuración de los puertos: normalmente admiten 10BASE-T y 100BASE-TX tanto en modo half-dúplex como full-dúplex. Su configuración se lleva a cabo por negociación mediante la característica de ***autonegociación*** que proporciona el estándar **IEEE 802.3**.

Switches perimetrales no gestionables

Este tipo de switches se utilizan habitualmente para constituir redes de pequeño tamaño de prestaciones medias. No admiten opciones de configuración y suelen tener características similares a los switches desktop pero incrementando el número de puertos y ofreciendo la posibilidad de montaje en rack 19''.

Algunas características son:

El número de puertos de este tipo de switch puede ser típicamente de 4, 8, 16 o 24 puertos

Suelen ser puertos 10/100 RJ-45 que admiten ***autonegociación*** y ***Auto MDI/MDI-X***. Existen algunos modelos con puertos 10/100/1000

En algunos casos pueden presentar puertos adicionales de rendimiento superior al resto de puertos.

Switches perimetrales gestionables

Este tipo se utiliza para la conexión de los equipos de los usuarios en redes de tamaño medio y grande, y se localizan en el nivel jerárquico inferior. Es necesario que estos switches ofrezcan características avanzadas de configuración y gestión. Sus características más habituales son:

- EL número de puertos fijos que ofrecen oscila entre 16 y 48 puertos.
- Existen modelos con puertos 10/100 y otros con puertos 10/100/1000, todos con soporte *Auto MDI/MDI-X*.
- Incluyen puertos adicionales de mayores prestaciones o puertos modulares (**GBIC** o **SFP**) para la conexión con un switch troncal.
- Características avanzadas de gestión por **SNMP**, puerto de consola, navegador web, ssh, monitorización *Port Mirroring*.
- Características avanzadas de configuración en el nivel 2 como *Port Trunking*, *Spanning Tree*, *IEEE 802.1x*, *QoS, VLAN*, soporte de tramas *Jumbo*, etc.
- Algunos modelos pueden ofrecer *Power Over Ethernet* en todos los puertos.

Switches troncales de prestaciones medias

Este tipo de switches están diseñados para formar el núcleo o troncal de una red de tamaño medio. Proporcionan altas prestaciones y funcionalidades avanzadas. Una de las principales diferencias con los switches perimetrales es que ofrecen características de nivel 3 como enrutamiento IP. A continuación se exponen sus características más representativas:

- Características avanzadas de configuración de nivel 2 similares a los switches perimetrales gestionables.
- Habitualmente ofrecen entre 24 y 48 puertos fijos 10/100 con conector RJ-45 con algunos puertos modulares adicionales para Gigabit Ethernet y 10GbE para cable y fibra. Existen también modelos con puertos de altas prestaciones 10/100/1000 o incluso puertos 10GbE.
- Permiten expandir sus capacidades mediante la apilación de switches.
- Niveles 2/3. Además de cubrir funciones de conmutación avanzadas del nivel 2 también proporcionan funciones de enrutamiento y gestión en el nivel 3.

Switches troncales de altas prestaciones

La principal característica de este tipo, además de su alto rendimiento, es su alta modularidad. El formato habitual es de tipo *chasis* donde se instalan los módulos que se necesitan. Se utilizan en grandes redes corporativas o de campus, e incluso se utilizan por los operadores para constituir sus redes metropolitanas. Sus principales características son:

- Altamente modulares mediante un chasis con un número variable de slots donde se insertan módulos con los elementos requeridos. Normalmente suelen admitir la inserción de módulos "en caliente" (*hot swappable*) de forma que no hay que desconectar el switch para realizar dicha operación, garantizando así una alta disponibilidad.
- Niveles 2/3/4. Además de cubrir funciones de conmutación avanzadas del nivel 2 también proporcionan funciones de enrutamiento y gestión en los niveles 3 y 4.
- Fuentes de alimentación redundantes.
- Admiten módulos con todos los tipos de puertos, tanto de cobre como de fibra con velocidades 10/100/1000 Mbps hasta 10Gbps.
- Alta densidad de puertos. Pueden llegar a más de 500 puertos 10/100, hasta 200 puertos Gigabit o sobre unos 25 puertos 10GbE.
- Características avanzadas de configuración y gestión en el nivel 2.
- Enrutamiento en el nivel 3 (IPv4 e IPv6).

Router o encaminador

El término router se podría traducir como **enrutador o encaminador**. Desde el punto de vista de la telemática, un router es un dispositivo de red utilizado para unir redes y encaminar datos entre ellas. Así de simple.

Unir redes es la función básica asociada a un router. Sin embargo la evolución de las redes y de Internet ha hecho evolucionar también a los routers añadiendo cada vez más funcionalidades a los mismos. En la actualidad podemos clasificar los routers en dos grandes grupos:

Routers de acceso. Son routers utilizados para unir dos redes, normalmente la red de un operador de telecomunicaciones con la red de su cliente, ya sea residencial o corporativo, y ya sea para proporcionar acceso a Internet o proporcionar acceso a otras redes de datos. En este tipo de routers la función de "enrutamiento" es más o menos simple porque solo tienen que intercambiar datos entre dos redes. Por el

contrario, suelen incorporar otras funciones adicionales como cortafuegos, NAT, proxy, balanceo de carga, Wi-Fi.

- **Routers de distribución**. Son routers que, a diferencia de los anteriores, están conectados a más de dos redes. Este tipo de routers sí mantiene como principal función la de "enrutar" datos entre las diferentes redes a las que están conectados y deben estar preparados para procesar una gran cantidad de información. Utilizan algoritmos de enrutamiento para optimizar la búsqueda de las rutas más óptimas para los datos que manejan.

PROBLEMA

Presentar una propuesta para mejorar el rendimiento de la red en la Facultad de Ciencias de la Computacion abarcando únicamente partes especificas.

Debido a la gran demanda con la que se cuenta en las instalaciones, el servicio es insuficiente y en ocasiones nulo, es por eso que se pide realizar una propuesta para ayudar a solucionar la conectividad en los siguientes puntos:

- En el edificio 104 A los cubos de Profesores
- Laboratorios del edificio 104 C de el primer nivel.
- Salones del edificio 104 D.
- Punto de acceso inalámbrico.

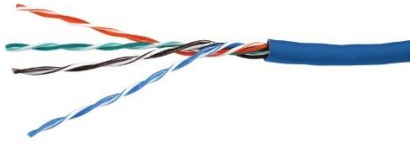
INFRAESTRUCTURA

El servicio es sumistrado desde el SIU en Ciudad Universitaria cruzando toda la universidad hasta llegar al rack de la Facultad de Ciencias de la Computacion en los edificios 104 A y 104 B.

Se planea unir ambos nodos colocándolos en el la parte baja del edificio donde se encuentra secretaria academica y a partir de ahí conectar los nodos ubicados en cada edificio.

Se requiere el siguiente material:

Cable UTP para crear los nodos de laboratorios, salones y cubículos e interconectarlos.



Cisco Gigabit Ethernet Switch Catalyst 2960-X, 10/100/1000Mbps, 216 Gbit/s, 48 Puertos - Gestionado

http://www.cyberpuerta.mx/Computo-Hardware/Redes/Switches/Cisco-Gigabit-Ethernet-Switch-Catalyst-2960-X-

10-100-1000Mbps-216-Gbit-s-48-Puertos-Gestionado-cp2-1.htmlr

Fibra Optica

Rack

Clima

Placa de 2 cavidades

Conector Jack hembra

CONCLUSION

El acceso a internet en la facultad es muy complejo y requiere de una gran inversión, en este documento se trata de hacer una mejora de forma simbolica.

Los factores que se toman en cuenta son: la flexibilidad, la vida útil de los elementos utilizados, el tamaño y ventilación del sitio, cantidad de usuarios conectados, etc.

Instalacion

* En el edificio 104 B junto al auditorio Albert Einstein se instalara el centro de red, ahí se combinaran los enlaces de fibra óptica mediante un router, asu vez a este se conectara un firewall físico para la administración y bloqueo de sitios y puertos en especifico.

Un Switch que admininstrara la conexión a los cubículos, laboratorios y salones de los edificios 104 A, 104 C y 104 D respectivamente.

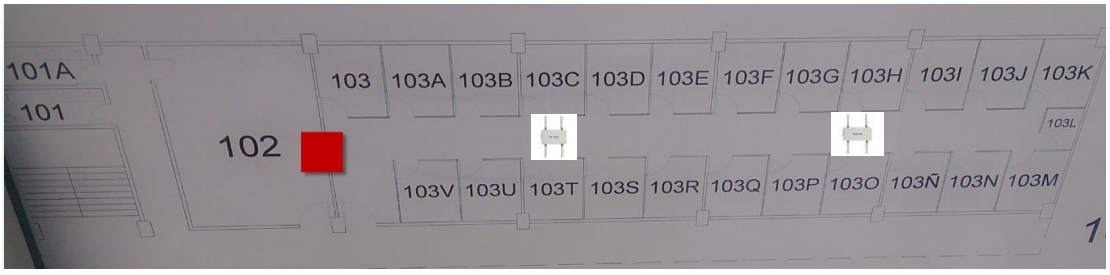
Cuatro enlaces de fibra óptica con salida hacia:

- Edificio 104 A hacia los switches del área de cubículos.
- Edificio 104 C para el área de laboratorios del primer nivel.
- Edificio 104 D para los salones de clase.

Sistema de Aire acondicionado para mantener el lugar fresco y libre de humedad.

Area de inst.	Auditorio Albert Einstein				

- En el edificio 1104 A en el área de cubículos donde hay 23, se instalaran 2 switches de 48 puertos, donde se instalaran 2 conexiones por cubículo para los profesores.



* En el edificio 104 C se instalaran 2

switches por laboratorio para tener un margen de maniobra de hasta 96 equipos de computo por laboratorio.

En el sótano de este se instalara un switch para recibir la conexión de fibra óptica y administrar la conexión con cada laboratorio.

* En el edificio 104 D se instalaran 4 switches de 48 puertos en cada nivel del edificio, en cada salón se instalaran conectores para 30 computadoras en las bodegas que hay en cada nivel al lado de las escaleras.

En la plnata baja es donde llegara el enlace de fibra óptica, este se conectara a un switch que administrara el enlace y se conectara con los demás switches del edificio.

Se instalaran antenas wi-fi en el edificio siendo 1 en la planta baja a la mitad del pasillo, en el primer nivel se colocaran 2, una al incio y otra al final de corredor y en el segundo nivel se colocara una antena al centro del corredor para complementar la conexión en caso de no ser suficiente la cableada, cada antena se conectara al switch de su nivel para ser administrada.

Asi mismo se instalara un sistema de aire acondicionado para mantener fresco el equipo.

CANTIDAD	DESCRIPCION	COSTO POR UNIDAD	TOTAL
25	Cisco Gigabit Ethernet Switch Catalyst 2960-X, 10/100/1000Mbps, 216 Gbit/s, 48 Puertos - Gestionado	$50,519.00	$1,212,456.00
415	Placa de 2 cavidades	$27.00	$11,205.00
830	Conector Jack hembra	$26.00	$21,580.00
3	Tripp-Lite Aire Acondicionado SRCOOL12K, 12000BTU/h, 1250W, Negro	$11,410.00	$34,230.00
200	Fibra Optica en mts.	$180.00	$36,000.00

19	Leviton Bobina de Cable Cat5e UTP, 305 Metros, Gris	$1,718.00	$32,642.00
415	Cabezal RJ-45	$6.00	$2,490.00
1	Cisco Ethernet Router 2911, Alámbrico, 3x RJ-45, 2x USB, 1 Gbit/s	$30,049.00	$30,049.00
10	EnGenius Access Point para Exteriores AC1750, Inalámbrico, 1300 Mbit/s, 2.4/5GHz	$19,848.00	$198,480.00
		Sub Total	$1,585,663.00
		IVA	$252,661.12
		TOTAL	$1,831,793.12

Benemérita Universidad Autónoma

De Puebla

“Facultad de Ciencias de la Computación”

MODELO DE REDES

“Primer examen”

Alumno

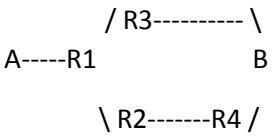
Rafael Ferrusca Marin

Fecha

9 - Febrero - 2015

Diseñar un sistema que calcule la latencia de cualquier topología que reciba, recibe un archivo de entrada y la respuesta que debe dar es la siguiente:

1.-imprimir en pantalla el número de caminos posibles



Path: A-R1-R3-B | .0021 seg

A-R1-R2-R4-B | .0201 seg

- Para este primer punto tenemos la opción de crear un arreglo que guarde la información como una cadena y que detecte cada dato separado por una coma “, ”, después lo convertiremos a numérico para poder manejar cada línea.

- Después por al leer la primera línea y al ver el dato por ejemplo 2,2 sabrá que es un árbol con dos nodos y dos aristas , entonces crearemos una lista ligada de sus nodos adyacentes , también se guardara el nodo inicial ,nodo final, distancia(m) ,velocidad de enlace , etc de cada arista.

2.- Por cada camino imprimir su latencia.

Para calcular la latencia se requiere la distancia, tamaño del archivo, velocidad de transferencia.

La fórmula es:

Latencia= (Tiempo de propagación + Tiempo de transmisión + Tiempo de cola)

3.- Imprimir el tiempo de transmisión considerando un archivo de x tamaño de origen a destino

-Por ejemplo cuánto tarda un archivo de 3.8 GB en cruzar del punto A al punto B

Para poder obtener el tiempo de propagación = **Distancia a recorrer**

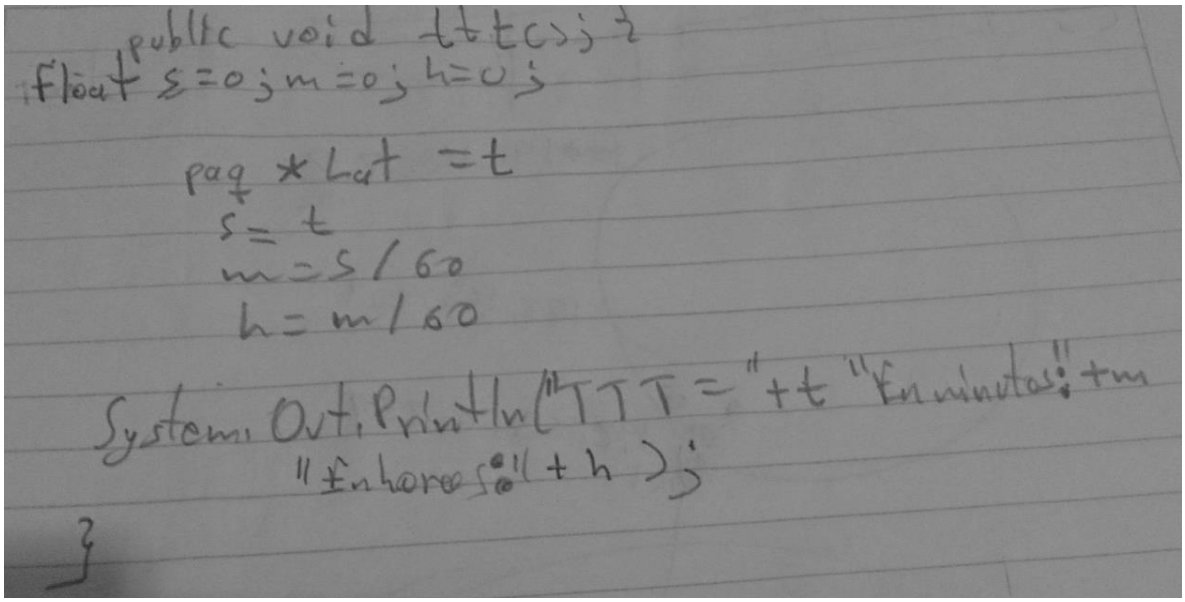
Velocidad de la luz

El tiempo de transmisión: **Tamaño de paquete**

Tasa de trasferencia.

- Al final se nos da un nodo inicial y un nodo final por ejemplo (1,3)en donde realizaremos la prueba de conseguir el tiempo total de transmisión y recibiremos la cantidad en gigabytes por ejemplo 28.5 GB

Donde se propone el siguiente seudocódigo.



CODIGO FUENTE

```
package leerficherotexto;

import java.io.*;
import java.util.*;

public class LeerFicheroTexto {

public void Leer(){
                                try {
```

```
                    FileReader fr = new FileReader("fichero.txt");
                    BufferedReader bf = new BufferedReader(fr);
                    String sCadena;

                    while ((sCadena = bf.readLine())!=null) {
                 System.out.println(sCadena);}

                    }

                 catch (FileNotFoundException fnfe){

                    fnfe.printStackTrace();}

               catch (IOException ioe){

                    ioe.printStackTrace();}


}


public void lligada(){




}


   public static void main(String[] args) {

      // TODO code application logic here

      LeerFicheroTexto  lee1;


                    lee1 = new       LeerFicheroTexto();
                    System.out.println("Los datos leidos fueron");
                    lee1.Leer();

                    System.out.println("La lista de adyacencias es:");


   }}
```

Benemérita Universidad Autónoma
De Puebla
“Facultad de Ciencias de la Computación”

MODELO DE REDES

“Documento Tecnico”

Alumnos
Silvestre Mora Tobón
Fernando Joel Vázquez Díaz
Rafael Ferrusca Marin
Enrique González Martínez

Fecha
6 - Abril - 2015

DESARROLLO

Un cableado estructurado es un sistema de red de cables, conectores y demás dispositivos de infraestructura flexible con los cuales podemos unir dos o más puntos de un sistema de computación en red de una forma universal dentro de un edificio para diferentes tipos de comunicaciones (de red) como de voz, datos o imágenes, al igual que soportar implementaciones y mejoras de tecnologías

Ventajas y beneficios del cableado estructurado

–Facilita un mantenimiento económico, sencillo y confiable.
–Proporciona seguridad de acceso para la administración del sistema.
–Soporta todas las tecnologías actuales y futuras por al menos 10-15 años.
–Existen normas e instrumentos que garantizan la calidad de la red instalada.
–Es de fácil administración.
–Permite cambios rápidos y sencillos.
–Posibilita ampliaciones económicas.
–Permite convivir muchos servicios en red (voz, datos, vídeo, etc.) con la misma instalación,
–Se facilita y agiliza mucho las labores de mantenimiento.

PLANTEAMIENTO DEL PROBLEMA

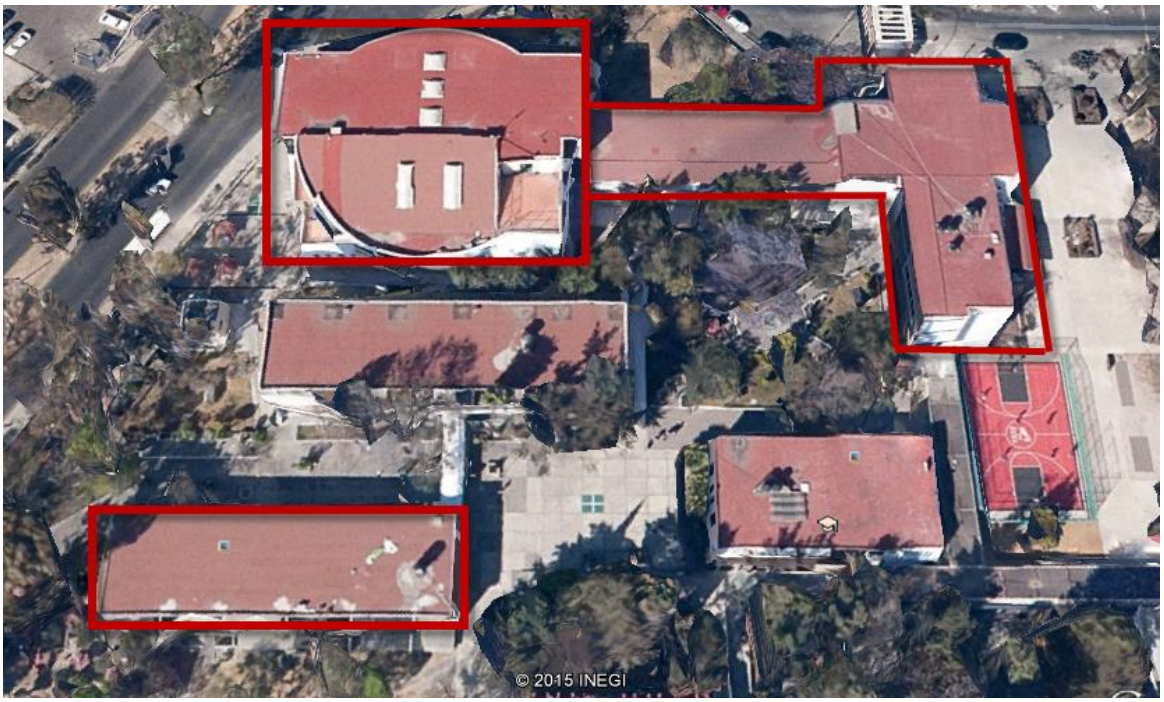
La Facultad de Ciencias de la Computación desea que se le presente un proyecto de diseño sobre una nueva red en sus instalaciones. El proyecto solo comprenderá tres edificios que deberán ser conectados al cuarto principal.

Debido a la gran cantidad de alumnos con que cuenta la facultad, la red es insuficiente para abastecer una señal de internet funcional en todo momento. Es por eso que la facultad desea cambiar parte de su infraestructura de red de tal manera que cubra la demanda de del alumnado que se encuentren en estos edificios.

Se pretende cubrir los siguientes puntos:

- Abastecer los 4 laboratorios del edificio 104C con red.
- Abastecer de red a los salones del edifico 104D.
- Abastecer de red a los Cubículos del edifico 104A.

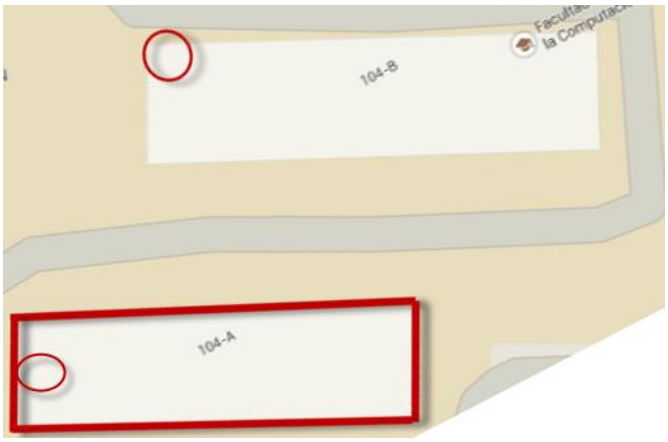
- Satisfacer las necesidades de red del personal docente y del alumnado.
- Permitir la conexión de todos los alumnos que se encuentren en las aulas de dichos edificios.



ESTRUCTURA FISICA

La sede principal que abastece a toda la universidad de internet es el SIU (Sistema de Información Universitaria) que se encuentra en el extremo izquierdo de CU.

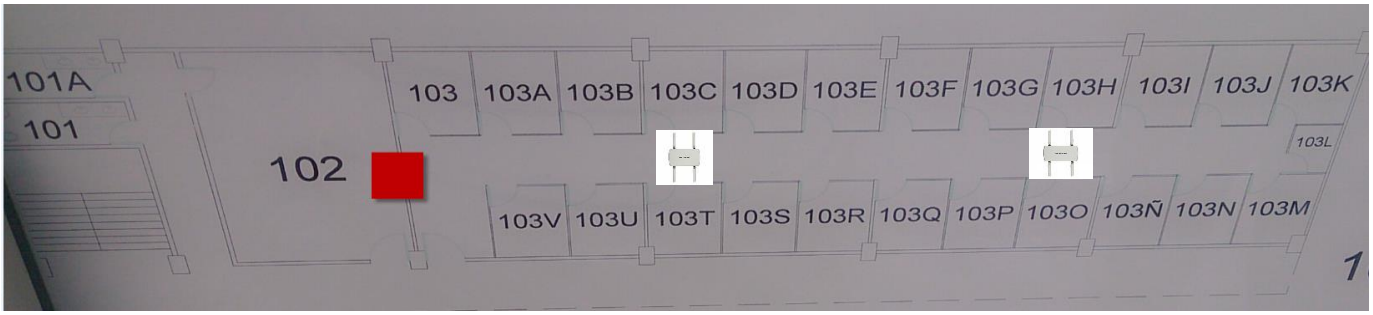
El SIU abastece de internet a la Facultad de Ciencias de Computación a través de un BackBone que atraviesa todo CU y llega a los rack de la facultad localizada en el edificio 104B y en el edificio 104A.



Rack principal de la facultad a donde llega la señal del SIU y que abastece a todos los edificios, se encuentra localizado en el edificio 104B (encerrado con un círculo)

Nuestra propuesta es unir los dos nodos en el laboratorio que actualmente ocupa química y después distribuirlos a sus correspondientes cuartos de trabajo.

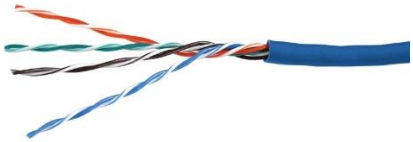
Edificio 104A



23 placas de 2 cavidades

23 de 1 cavidad

1569 metros de cable UTP $



69 Conectores hembra (jack) RJ45 cat 6 $

160 metros de canaleta autoadherible, de 2 m, para cable de 40 vías

2 Acces point NETGEAR WND930 Outdoor Dual Band Wireless-N

El punto de acceso WND930 combina un diseño industrial y robusto con opciones de antena flexible y funcionamiento simultáneo de doble banda (2,4 y 5 GHz) para una óptima implementación en exteriores. El punto de acceso para exteriores WND930 recibe alimentación mediante puertos gigabit de doble alimentación a través de Ethernet (PoE). Ofrece también la capacidad de alimentación para otros dispositivos PoE. El punto de acceso para exteriores WND930 funciona en entornos difíciles gracias a su carcasa con clasificación industrial IP67.

• Doble alimentación a través de Ethernet Gigabit para una conectividad sencilla

• Hasta 128 usuarios conectados

• Funcionamiento de doble barra simultánea (2,4 y 5 GHz) para una máxima flexibilidad

- Encriptación Wired Equivalent Privacy (WEP)
- Acceso Protejido WiFi Pre-shared Key (WPA-PSK, WPA2-PSK)
- Filtrado por dirección MAC
- 1 BSSID

11 106 $

Cuarto de comunicaciones 104A

1 Rack s

47,5 cm (19 pulgadas) de ancho por 120 cm (48 pulgadas) de altura.

2 Cisco Gigabit Ethernet Switch SG500-52P, 10/100/1000Mbps, 120Gbit/s, 48 Puertos, 16.000 Entradas.

Puertos RJ-45 Ethernet 52, 48, 52

Capacidad de conmutación	120 Gbit/s
Tasa de transferencia (máx)	1 Gbit/s
Tabla de direcciones MAC	16000 entradas
Tipo de interruptor	Gestionado
Montaje en rack	✔

2 Panel de parcheo de 48 puertos Categoría 6, para montaje en rack $1,290x7= $9030

2 LG CLIMA VENTANA 12000 BTUS 1 TON 110 V $4799 x 2= $9598

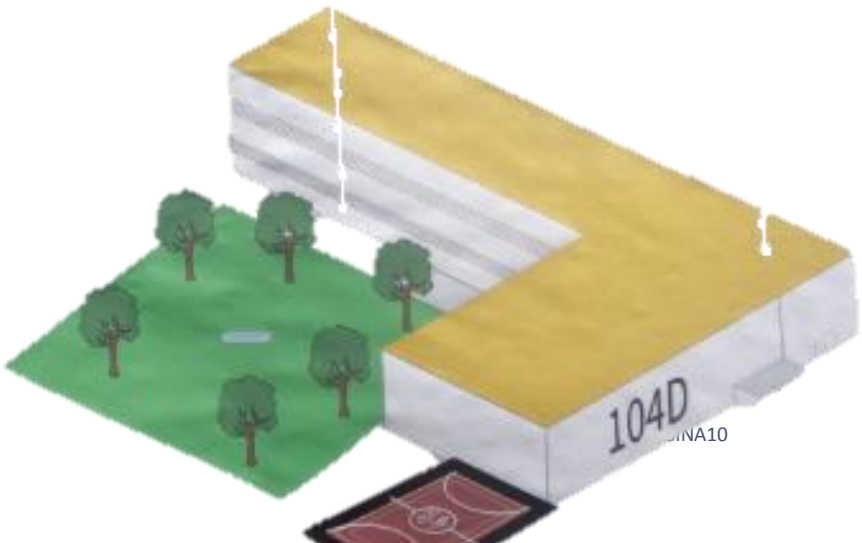
Edificio 104A

PROVEDOR	MATERIAL	COSTO POR UNIDAD	UNIDADES REQUERIDAS	TOTAL
Steren	Placa de 2 cavidades	$ 11.00	23	$ 253.00
Steren	Placa de 1 cavidad	$ 13.00	23	$ 299.00
Steren	Cable UTP	$ 6.40	1569	$ 10,041.60
Steren	Conectores hembra (jack) RJ45 cat 6	$ 24.50	69	$ 1,690.50
Steren	Canaleta autoadherible, de 2 m, para cable de 40 vías	$ 91.50	160	$ 14,640.00
Amazon	Acces point NETGEAR WND930 Outdoor Dual Band Wireless-N	$ 749.57	2	$ 1,499.14
			TOTAL	$ 28,423.24

Cuarto de comunicación 104 A

PROVEDOR	MATERIAL	COSTO POR UNIDAD	UNIDADES REQUERIDAS	TOTAL
Steren	Rack vertical	$ 2,890.00	1	$ 2,890.00
Cisco	Cisco Gigabit Ethernet Switch SG500-52P	$ 35,470.00	2	$ 70,940.00
Steren	Panel de parcheo de 48 puertos Categoría 6	$ 1,290.00	7	$ 9,030.00
Homedepot	LG CLIMA VENTANA 12000 BTUS 1 TON 110 V	$ 4,799.00	2	$ 9,598.00
			TOTAL	$ 92,458.00

Edificio 104D



Primer piso.

76 placas de 1 cavidad 494$

2920 metros de cable UTP 14 600$

76 Conectores hembra (jack) RJ45 cat 6 2090$

70 metros de canaleta autoadherible, de 2 m, para cable de 40 vías 3022.5 $

2 Acces point NETGEAR WND930 Outdoor Dual Band Wireless-N

El punto de acceso WND930 combina un diseño industrial y robusto con opciones de antena flexible y funcionamiento simultáneo de doble banda (2,4 y 5 GHz) para una óptima implementación en exteriores. El punto de acceso para exteriores WND930 recibe alimentación mediante puertos gigabit de doble alimentación a través de Ethernet (PoE). Ofrece también la capacidad de alimentación para otros dispositivos PoE. El punto de acceso para exteriores WND930 funciona en entornos difíciles gracias a su carcasa con clasificación industrial IP67.

• Doble alimentación a través de Ethernet Gigabit para una conectividad sencilla

• Hasta 128 usuarios conectados

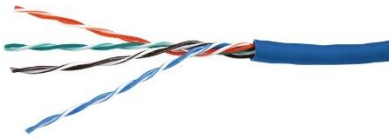
• Funcionamiento de doble barra simultánea (2,4 y 5 GHz) para una máxima flexibilidad

- Encriptación Wired Equivalent Privacy (WEP)
- Acceso Protejido WiFi Pre-shared Key (WPA-PSK, WPA2-PSK)
- Filtrado por dirección MAC
- 1 BSSID

Segundo piso.

133 placas de 1 cavidad

4620 metros de cable UTP



133 Conectores hembra (jack) RJ45 cat 6

123 metros de canaleta autoadherible, de 2 m, para cable de 40 vías

4 Acces point NETGEAR WND930 Outdoor Dual Band Wireless-N

El punto de acceso WND930 combina un diseño industrial y robusto con opciones de antena flexible y funcionamiento simultáneo de doble banda (2,4 y 5 GHz) para una óptima implementación en exteriores. El punto de acceso para exteriores WND930 recibe alimentación mediante puertos gigabit de doble alimentación a través de Ethernet (PoE). Ofrece también la capacidad de alimentación para otros dispositivos PoE. El punto de acceso para exteriores WND930 funciona en entornos difíciles gracias a su carcasa con clasificación industrial IP67.

• Doble alimentación a través de Ethernet Gigabit para una conectividad sencilla

• Hasta 128 usuarios conectados

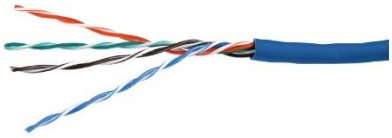
• Funcionamiento de doble barra simultánea (2,4 y 5 GHz) para una máxima flexibilidad

- Encriptación Wired Equivalent Privacy (WEP)
- Acceso Protejido WiFi Pre-shared Key (WPA-PSK, WPA2-PSK)
- Filtrado por dirección MAC
- 1 BSSID

Tercer piso.

133 placas de 1 cavidad

4847 metros de cable UTP



133 Conectores hembra (jack) RJ45 cat 6

123 metros de canaleta autoadherible, de 2 m, para cable de 40 vías

4 Acces point NETGEAR WND930 Outdoor Dual Band Wireless-N

El punto de acceso WND930 combina un diseño industrial y robusto con opciones de antena flexible y funcionamiento simultáneo de doble banda (2,4 y 5 GHz) para una óptima implementación en exteriores. El punto de acceso para exteriores WND930 recibe alimentación mediante puertos gigabit de doble alimentación a través de Ethernet (PoE). Ofrece también la capacidad de alimentación para otros dispositivos PoE. El punto de acceso para exteriores WND930 funciona en entornos difíciles gracias a su carcasa con clasificación industrial IP67.

• Doble alimentación a través de Ethernet Gigabit para una conectividad sencilla

• Hasta 128 usuarios conectados

• Funcionamiento de doble barra simultánea (2,4 y 5 GHz) para una máxima flexibilidad

- Encriptación Wired Equivalent Privacy (WEP)
- Acceso Protejido WiFi Pre-shared Key (WPA-PSK, WPA2-PSK)
- Filtrado por dirección MAC
- 1 BSSID

Cuarto de comunicaciones

4 Rack s $8,400

47,5 cm (19 pulgadas) de ancho por 120 cm (48 pulgadas) de altura.

7 HP Gigabit Ethernet Switch V1910, 10/100/1000Mbps, 48 Puertos - Gestionado

Descripción breve

Puertos RJ-45 Ethernet	48
Capacidad de conmutación	104 Gbit/s
Tasa de transferencia (máx)	1 Gbit/s
Tabla de direcciones MAC	8192 entradas
Tipo de interruptor	Gestionado
Montaje en rack	✔

$ 9 680.00

1 Cisco Gigabit Ethernet Switch SF300, 10/100Mbps, 12.8Gbit/s, 24 Puertos, 16.000 Entradas – Gestionado $ 3,954.00

7 Panel de parcheo de 48 puertos Categoría 6, para montaje en rack $1,290x7= $9030

1 Panel de parcheo de 24 puertos Categoría 6, para montaje en rack $660

2 LG CLIMA VENTANA 12000 BTU 1 TON 110 V $4799 x 2= $9598

Edificio 104D Primer piso

PROVEDOR	MATERIAL
Steren	Placa de 1 cavidad
Steren	Cable UTP
Steren	Conectores hembra (jack) RJ45 cat 6
Steren	Canaleta autoadherible, de 2 m, para cable de 40 vías
Amazon	Acces point NETGEAR WND930 Outdoor Dual Band Wireless-N

Edificio 104D Segundo piso

PROVEDOR	MATERIAL
Steren	Placa de 1 cavidad
Steren	Cable UTP
Steren	Conectores hembra (jack) RJ45 cat 6
Steren	Canaleta autoadherible, de 2 m, para cable de 40 vías
Amazon	Acces point NETGEAR WND930 Outdoor Dual Band Wireless-N

Edificio 104D Tercer piso

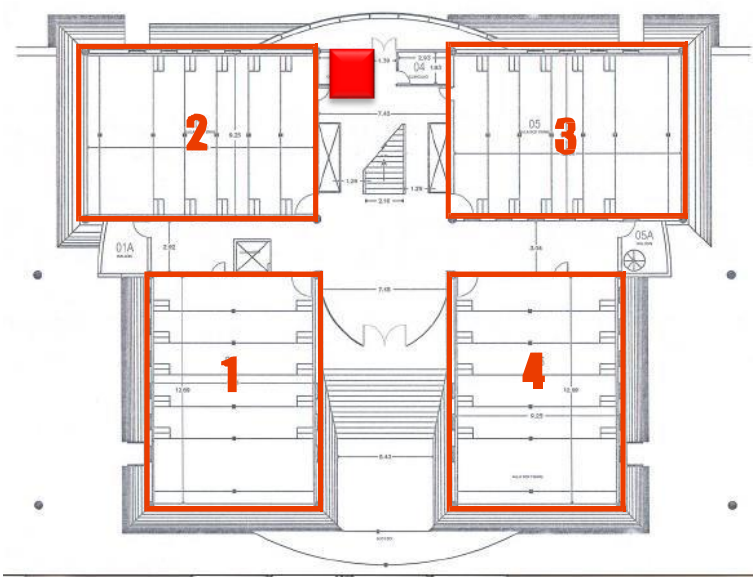
PROVEDOR	MATERIAL
Steren	Placa de 1 cavidad
Steren	Cable UTP
Steren	Conectores hembra (jack) RJ45 cat 6
Steren	Canaleta autoadherible, de 2 m, para cable de 40 vías
Amazon	Acces point NETGEAR WND930 Outdoor Dual Band Wireless-N

Cuarto de comunicaciónes

PROVEDOR	MATERIAL
Steren	Rack vertical
CyberPuerta	HP Gigabit Ethernet Switch V1910
Cisco	Cisco Switch Cisco SF300-24P
Steren	Panel de parcheo de 48 puertos Categoría 6
Steren	Panel de parcheo de 24 puertos Categoría 6
Homedepot	LG CLIMA VENTANA 12000 BTUS 1 TON 110 V

Edificio 104C

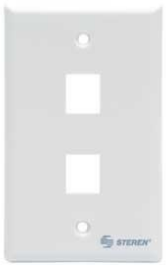
El edificio 104C consta de 3 pisos y una planta baja; en su mayoría es utilizado para laboratorios y la biblioteca de la facultad. En el proyecto el objetivo es abastecer de red los cuatro laboratorios localizados en el primer piso.



En el plano anterior se observa la localización de los 4 laboratorios que se encuentran en el primer piso del edificio 104C.

Laboratorio 1

25 Placas de 2 cavidades y 3 de una cavidad



53 conectores jack hembra.

1,415 metros Cable UTP cat 6.

- 59 metros canaleta.

Laboratorio 2

25 Placas de 2 cavidades y 3 de una cavidad

54 conectores jack hembra.

480 metros Cable UTP cat 6.

- 59 metros canaleta.

Laboratorio 3

25 Placas de 2 cavidades y 3 de una cavidad

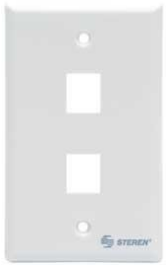
55 conectores jack hembra.

740 metros Cable UTP cat 6.

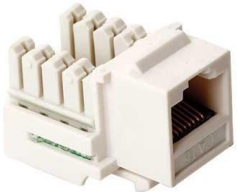
- 59 metros canaleta.

Laboratorio 4

25 Placas de 2 cavidades y 3 de una cavidad



56 conectores jack hembra.



1,200 metros Cable UTP cat 6.

- 59 metros canaleta.

Cuarto de comunicaciones 104C

4 Rack s $8,400

47,5 cm (19 pulgadas) de ancho por 120 cm (48 pulgadas) de altur

4 HP Gigabit Ethernet Switch V1910, 10/100/1000Mb

Descripción breve

Puertos RJ-45 Ethernet	48
Capacidad de conmutación	104 Gbit/s
Tasa de transferencia (máx)	1 Gbit/s
Tabla de direcciones MAC	8192 entradas
Tipo de interruptor	Gestionado
Montaje en rack	✔

$ 9 680.00

1 Cisco Gigabit Ethernet Switch SF300, 10/100Mbps, 12.8Gbit/s, 24 Puertos, 16.000 Entradas – Gestionado $ 3,954.00

4 Panel de parcheo de 48 puertos Categoría 6, para montaje en rack $1,290x7= $9030

1 Panel de parcheo de 24 puertos Categoría 6, para montaje en rack $660

2 LG CLIMA VENTANA 12000 BTUS 1 TON 110 V $4799 x 2= $9598

Laboratorio 1

PROVEDOR	MATERIAL	POR UNIDAD	CANTIDAD	TOTAL
Steren	Placa de 2 cavidades	$ 11.00	25	$ 275.00
Steren	Placa de 1 cavidad	$ 13.00	3	$ 39.00
Steren	Conectores hembra (jack) RJ45 cat 6	$ 24.50	53	$ 1,298.50
Steren	Cable UTP	$ 6.40	1415	$ 9,056.00
Steren	Canaleta autoadherible, de 2 m, para cable de 40 vías	$ 91.50	59	$ 5,398.50
			TOTAL	$ 16,067.00

Laboratorio 2

PROVEDOR	MATERIAL	POR UNIDAD	CANTIDAD	TOTAL
Steren	Placa de 2 cavidades	$ 11.00	25	$ 275.00
Steren	Placa de 1 cavidad	$ 13.00	3	$ 39.00
Steren	Conectores hembra (jack) RJ45 cat 6	$ 24.50	54	$ 1,323.00
Steren	Cable UTP	$ 6.40	480	$ 3,072.00
Steren	Canaleta autoadherible, de 2 m, para cable de 40 vías	$ 91.50	59	$ 5,398.50
			TOTAL	$ 10,107.50

Laboratorio 3

PROVEDOR	MATERIAL	POR UNIDAD	CANTIDAD	TOTAL
Steren	Placa de 2 cavidades	$ 11.00	25	$ 275.00
Steren	Placa de 1 cavidad	$ 13.00	3	$ 39.00
Steren	Conectores hembra (jack) RJ45 cat 6	$ 24.50	55	$ 1,347.50
Steren	Cable UTP	$ 6.40	740	$ 4,736.00
Steren	Canaleta autoadherible, de 2 m, para cable de 40 vías	$ 91.50	59	$ 5,398.50
			TOTAL	$ 11,796.00

Laboratorio 4

PROVEDOR	MATERIAL	POR UNIDAD	CANTIDAD	TOTAL
Steren	Placa de 2 cavidades	$ 11.00	25	$ 275.00
Steren	Placa de 1 cavidad	$ 13.00	3	$ 39.00
Steren	Conectores hembra (jack) RJ45 cat 6	$ 24.50	56	$ 1,372.00
Steren	Cable UTP	$ 6.40	1200	$ 7,680.00
Steren	Canaleta autoadherible, de 2 m, para cable de 40 vías	$ 91.50	59	$ 5,398.50
			TOTAL	$ 14,764.50

Cuarto de comunicaciónes 104C

PROVEDOR	MATERIAL	POR UNIDAD	CANTIDAD	TOTAL
Steren	Rack vertical	$ 2,890.00	4	$ 11,560.00
CyberPuerta	HP Gigabit Ethernet Switch V1910	$ 9,680.00	4	$ 38,720.00
Cisco	Cisco Switch Cisco SF300-24P	$ 8,809.00	1	$ 8,809.00
Steren	Panel de parcheo de 48 puertos Categoría 6	$ 1,290.00	4	$ 5,160.00
Steren	Panel de parcheo de 24 puertos Categoría 6	$ 660.00	1	$ 660.00
Homedepot	LG CLIMA VENTANA 12000 BTUS 1 TON 110 V	$ 4,799.00	2	$ 9,598.00
			TOTAL	$ 74,507.00

2 Convertidor de Medios Gigabit Ethernet RJ45 a Fibra Óptica SC Multimodo - 550m
1558$

200 metros Fibra óptica $ 22000

Router de banda ancha de Balance de carga 3200$
TL-R470T+

- IP dinámica, IP estática, PPPoE, L2TP, PPTP y BigPond las opciones de Internet por cable de conexión se conectan fácilmente a Internet
- Basada en el tiempo de control de acceso permite a los padres o los administradores establecer políticas de acceso restringido para los niños o el personal
- IP basada en el control de ancho de banda permite a los administradores determinar la cantidad de ancho de banda es asignado a cada PC que garantice el rendimiento de las aplicaciones de VoIP o de vídeo

Conexiones entre edificios

	PROVEDOR	MATERIAL	COSTO POR UNIDAD	UNIDADES REQUERIDAS	TOTAL
	StarTech	Convertidor de Medios Gigabit Ethernet RJ45 a Fibra Óptica	$ 1,558.31	2	$ 3,116.62
	Tp-Link	Router de banda ancha de Balance de carga	$ 3,200.00	1	$ 3,200.00
				TOTAL	$ 6,316.62

CONCLUCION

Entre los factores que influyen para lograr un buen diseño se deben citar: la flexibilidad con respecto a los servicios soportados, la vida útil requerida, el tamaño del sitio y la cantidad de usuarios que estarán conectados y los costos, entre otros.

Teniendo en cuenta estos factores no se debe dudar en utilizar el mecanismo que provea las facilidades de estandarización, orden, rendimiento, durabilidad, integridad y facilidad de expansión como el cableado estructurado provee.

Con esto se cumple el objetivo establecido desde el principio, de tal manera que los edificios contarían con un servicio de red confiable, veloz y seguro.

Benemérita Universidad Autónoma de Puebla

Facultad en Ciencias de la Computación

Alumno

Mora Tobón Silvestre

Proyecto

Implementación de MD5

Fecha de entrega

28/04/2015

Introducción

En la comunicación actual entre los distintos medios que se tiene han pasado muchas y grandes cosas las cuales han llevado a que la personas se puedan comunicar de forma muy rápida y sencilla sin que la mayoría sepa como es el funcionamiento de este, dentro de las clases vimos la importancia de tener seguridad en los medios a los cuales nos conectamos, por lo cual en clase hemos visto una forma sencilla de cómo se puede generar un chat con seguridad para que los usuarios puedan acceder de forma segura.

Desarrollo

Dentro de lo que se ha visto en clase y para el desarrollo del proyecto se usaran sockets para usar un cliente y un servidor que nos servirán para la emulación de cómo se conectaría un cliente con el servidor y este a su vez le daría acceso al cliente pero de forma segura resguardando sus datos, para que esto suceda se ha tenido que investigar el uso de la librería MD5 el cual es un algoritmo que ayuda a encriptar información que no se desea mostrar esto para ocultar las contraseñas.

En que consiste MD5

La codificación del MD5 de 128 bits es representada típicamente como un número de 32 dígitos hexadecimal. El siguiente código de 28 bytes ASCII será tratado con MD5 y veremos su correspondiente *hash* de salida:

MD5 ("Generando un MD5 de un texto") =
5df9f63916ebf8528697b629022993e8

Un pequeño cambio en el texto (cambiar '5' por 'S') produce una salida completamente diferente.

MD5 ("Generando un MDS de un texto") =
e14a3ff5b5e67ede599cac94358e1028

Otro ejemplo sería la codificación de un campo vacío:

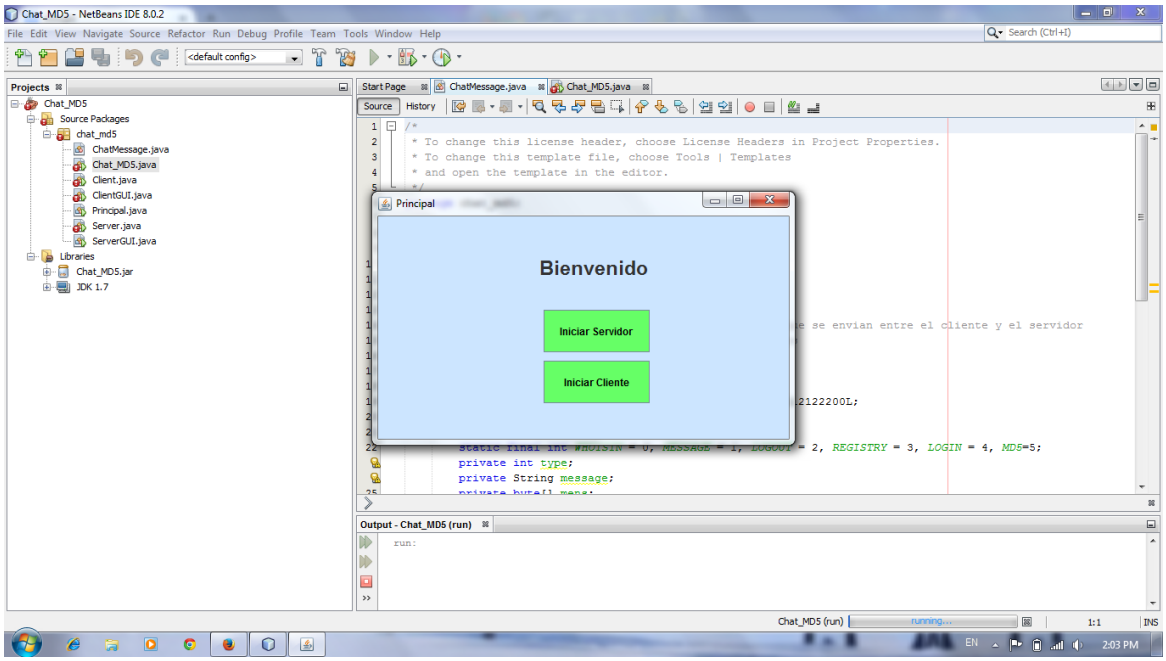
MD5 ("") = d41d8cd98f00b204e9800998ecf8427e

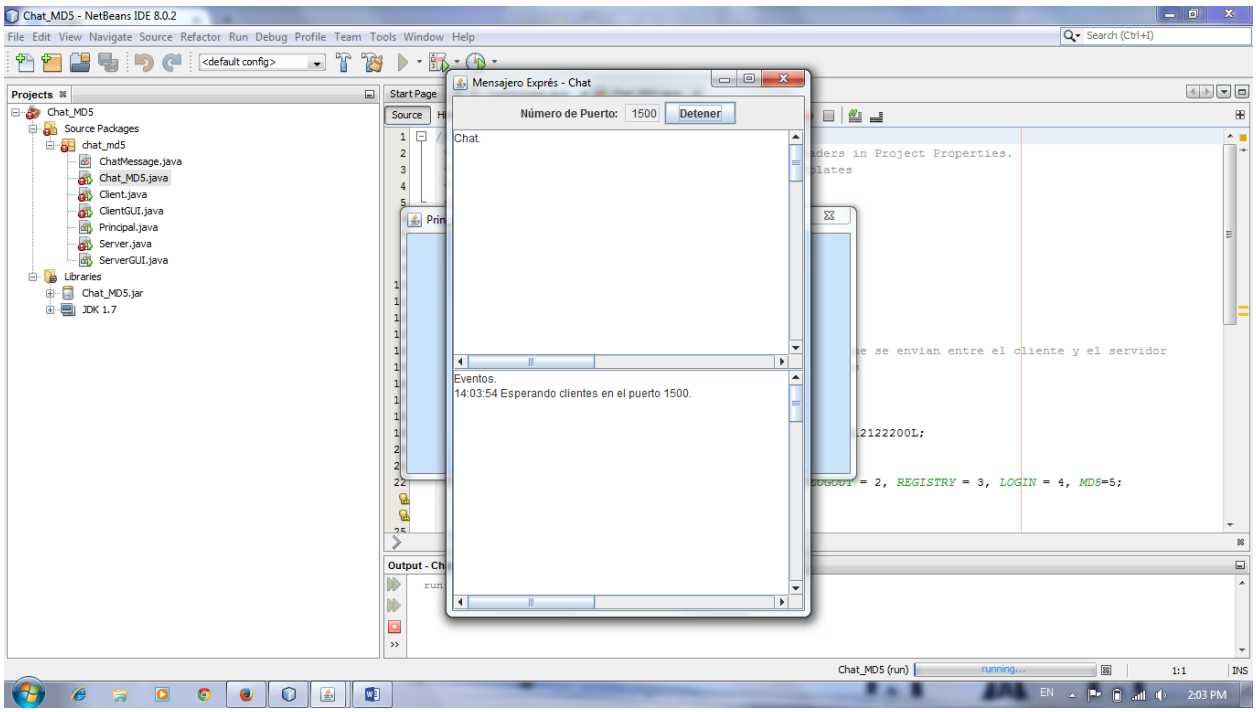
Este es de forma muy general de cómo funciona.

Para la implementación del programa se realizó en netbeans y se trabajó bajo la plataforma de java.

Se han creado ventanas y botones para un entorno amigable en los cuales podremos hacer uso de este pequeño chat que tiene MD5.

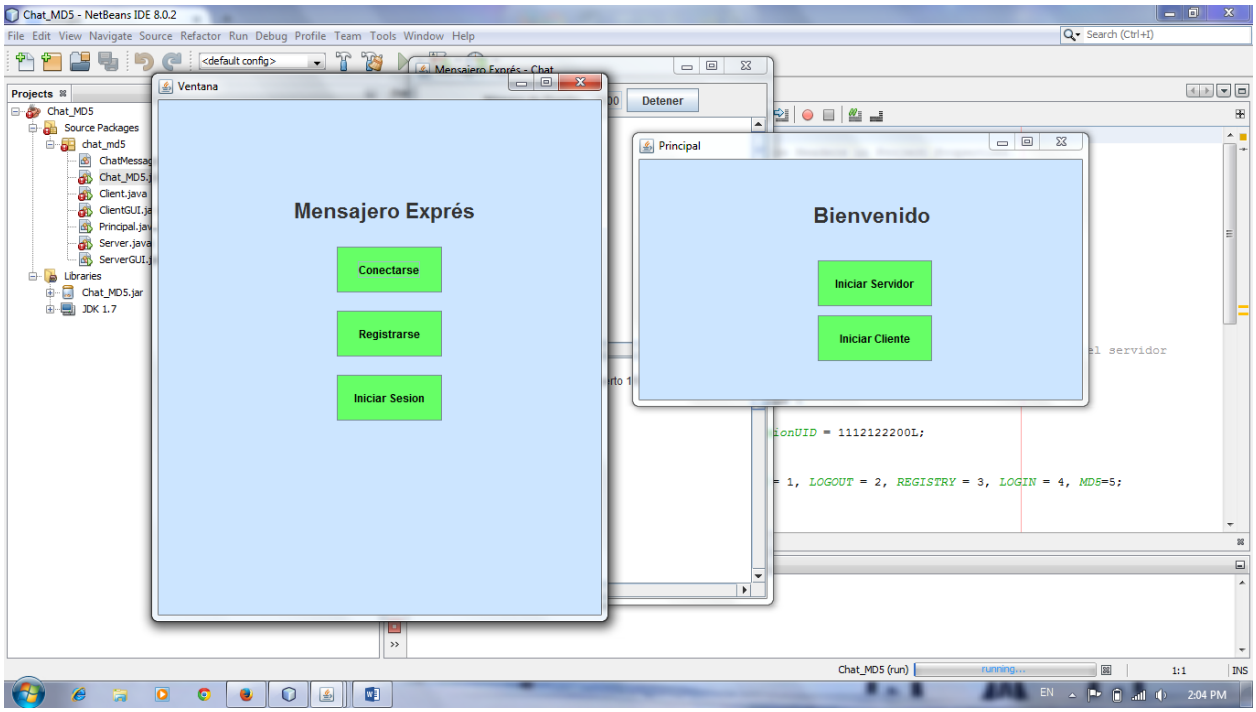
Para poder ingresar primero tendremos que iniciar el servidor


Chat_MD5 - NetBeans IDE 8.0.2
Principal
Bienvenido
Iniciar Servidor
Iniciar Cliente

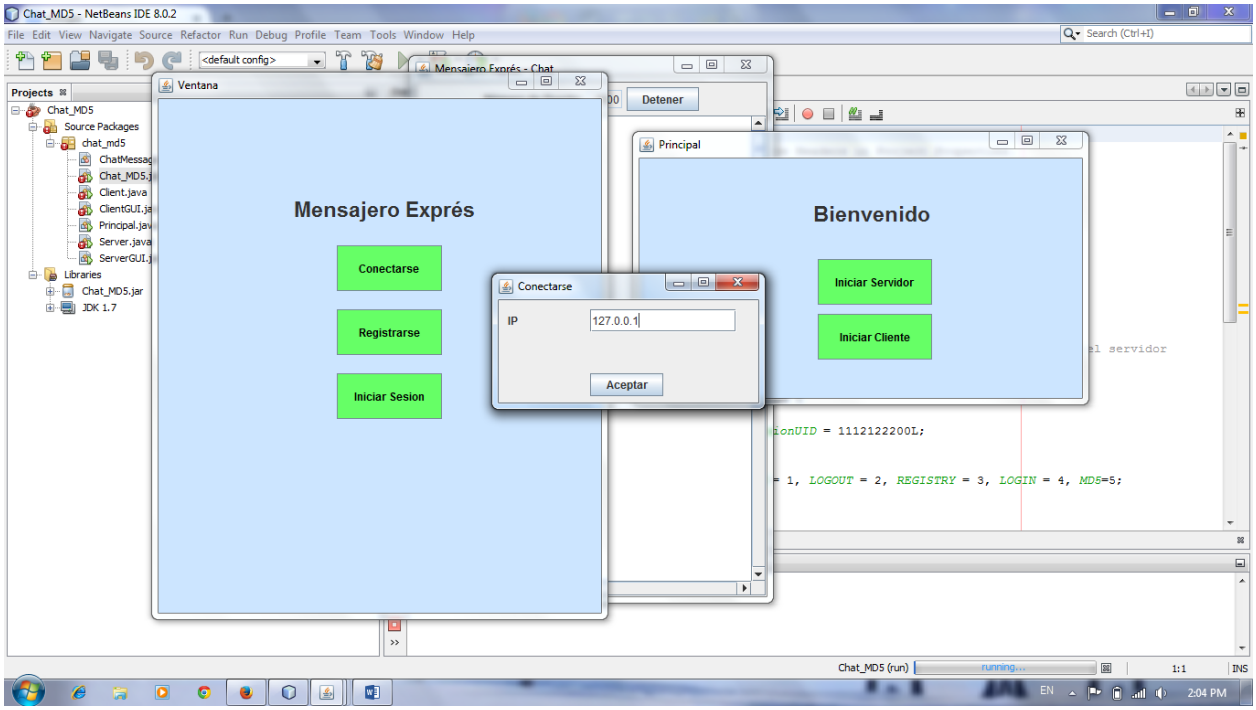

Chat_MD5 - NetBeans IDE 8.0.2
Mensajero Exprés - Chat
Número de Puerto:
1500
Detener
Chat.
Eventos.
14:03:54 Esperando clientes en el puerto 1500.

Una vez iniciado el servidor habrá que poner el puerto ocuparemos en este caso usamos el 1500

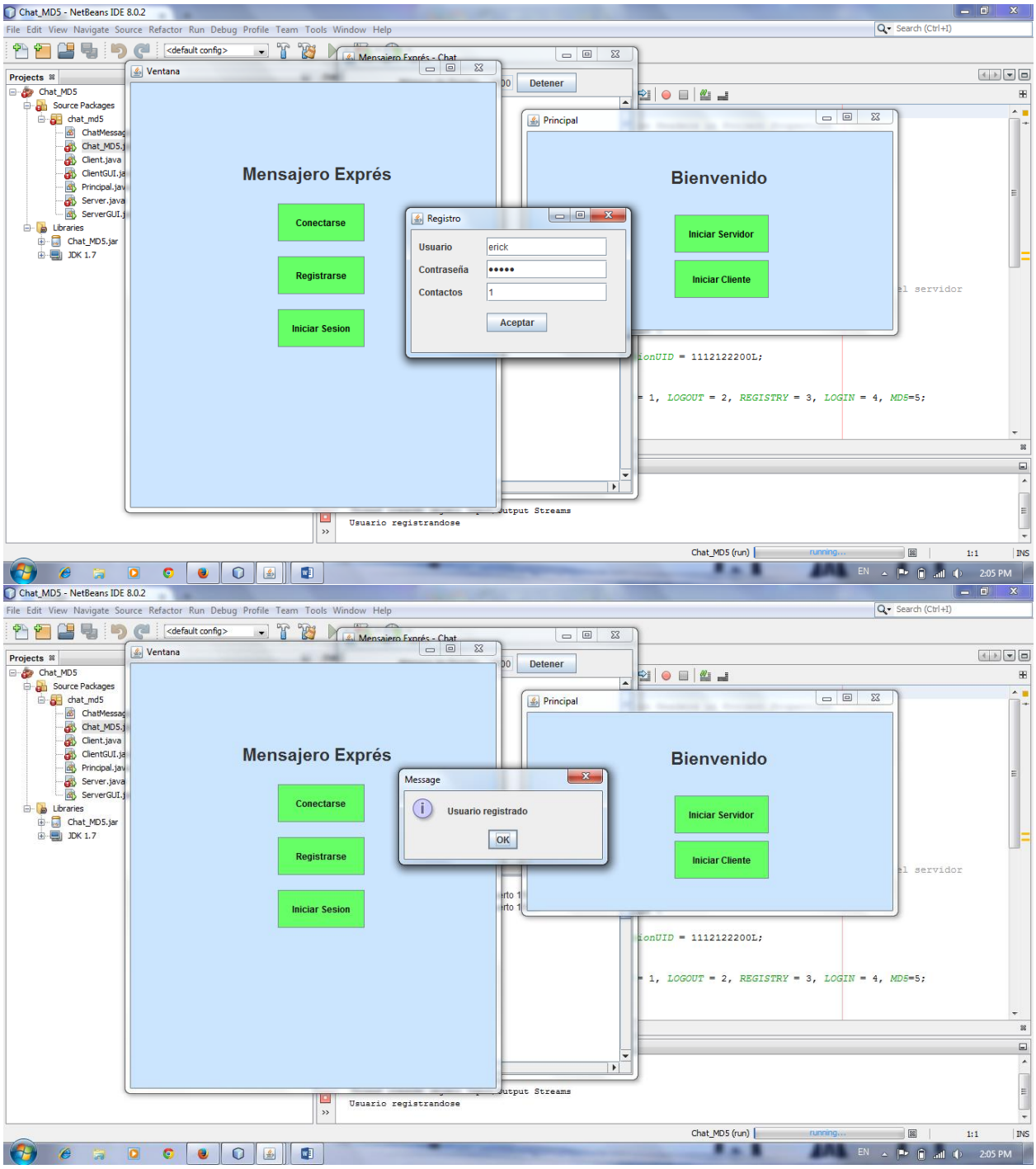
Una vez iniciado nos vamos al botón de iniciar cliente

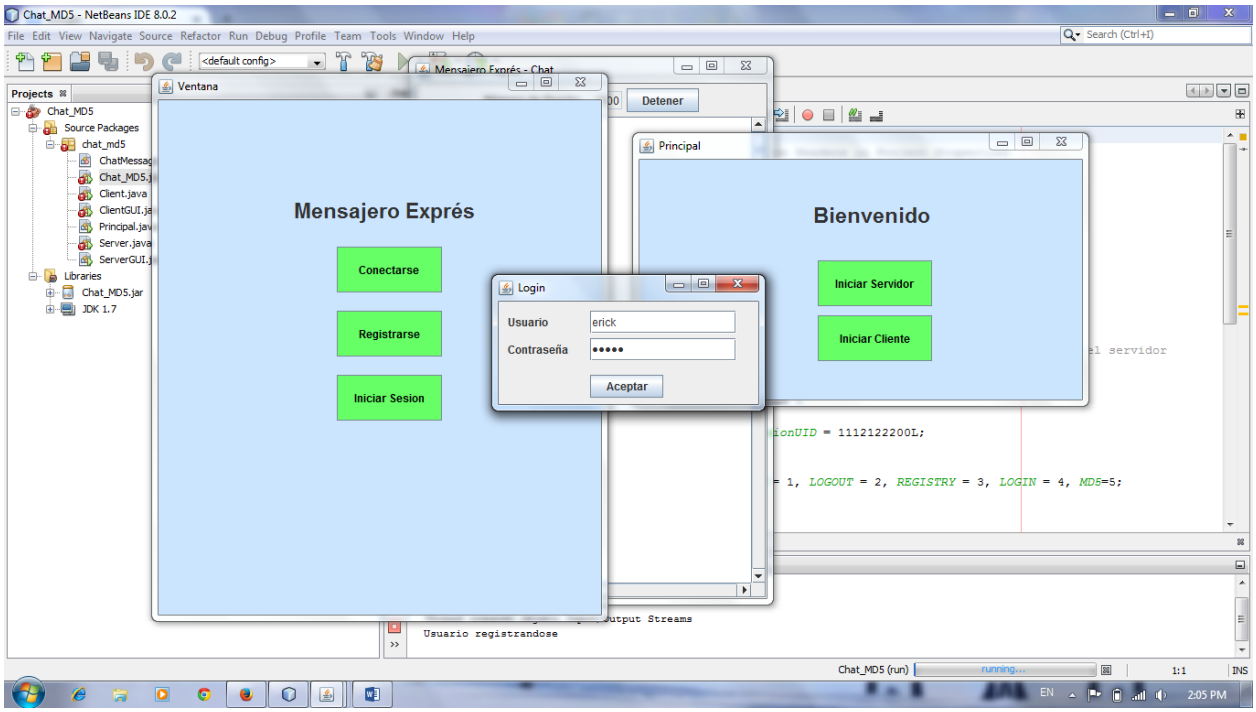


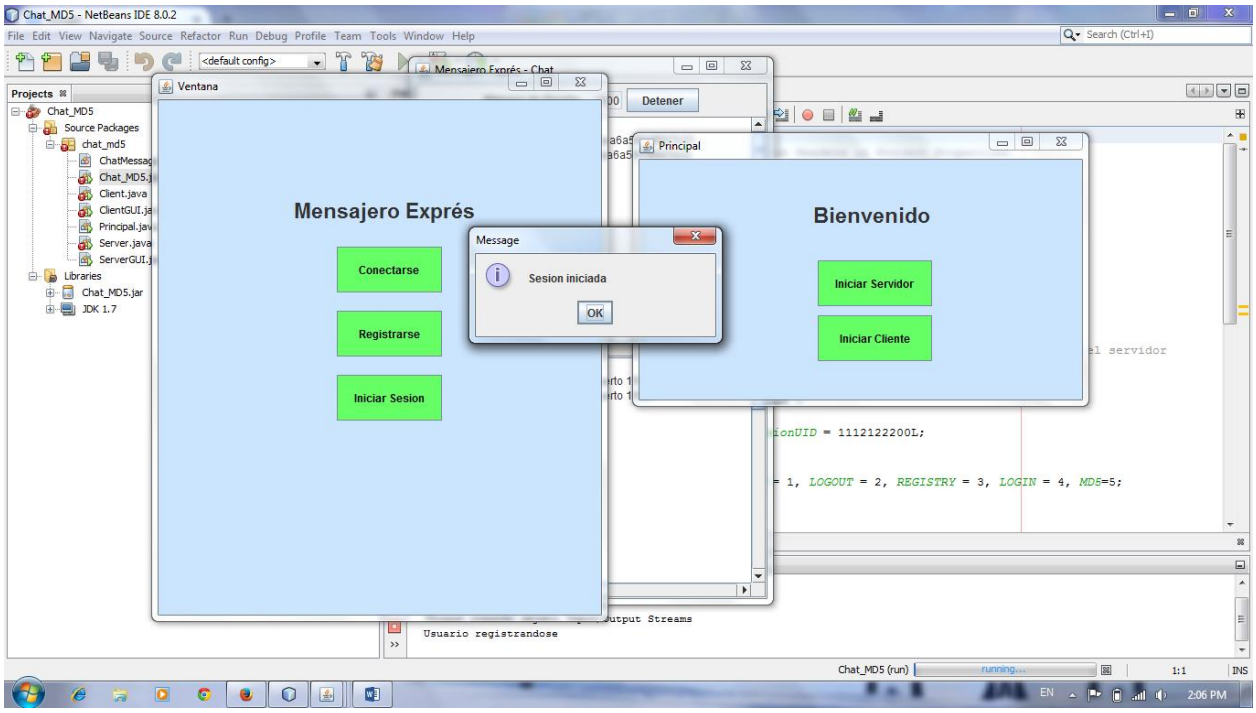
Nos vamos a la opción de conectarse hay tendremos que poner la ip que tengamos en este caso usamos al ip 127.0.0.1



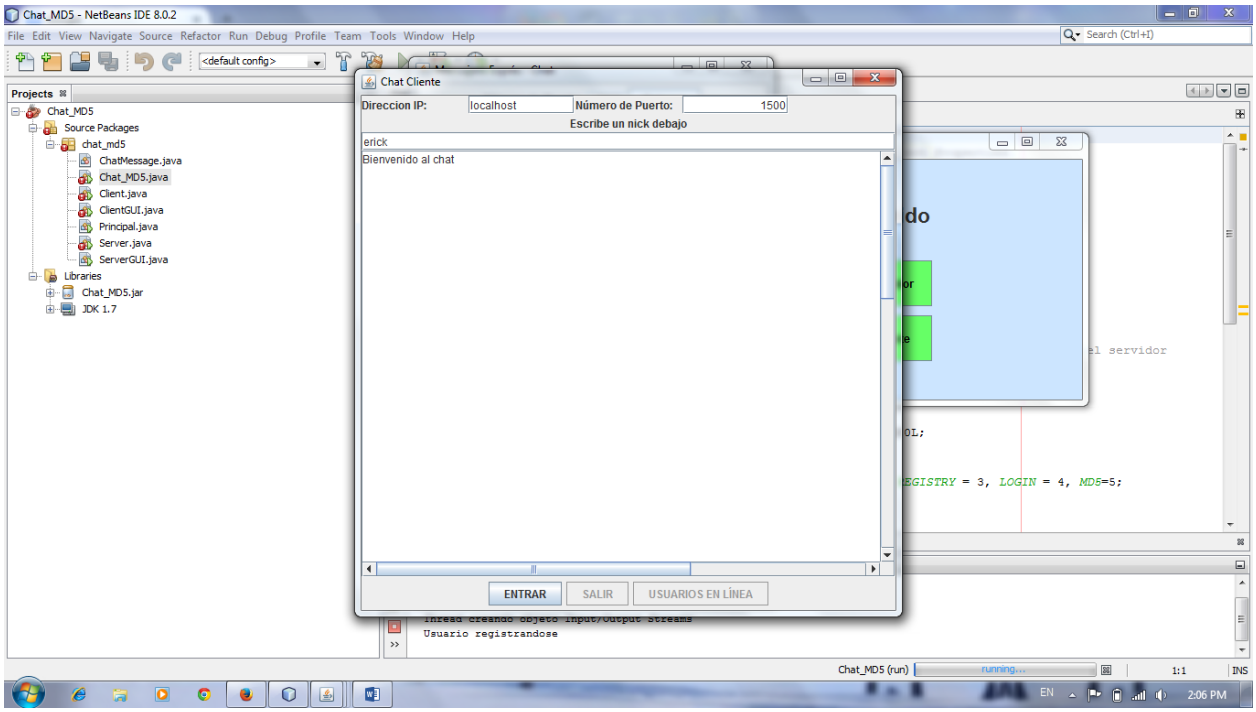
Después tendremos que registrarnos e inmediatamente después iniciaremos sesión si los datos concuerdan con los dados al registrarse accederemos al chat de lo contrario nos saldrá un mensaje que nos dirá que el usuario y/o contraseña son incorrectos

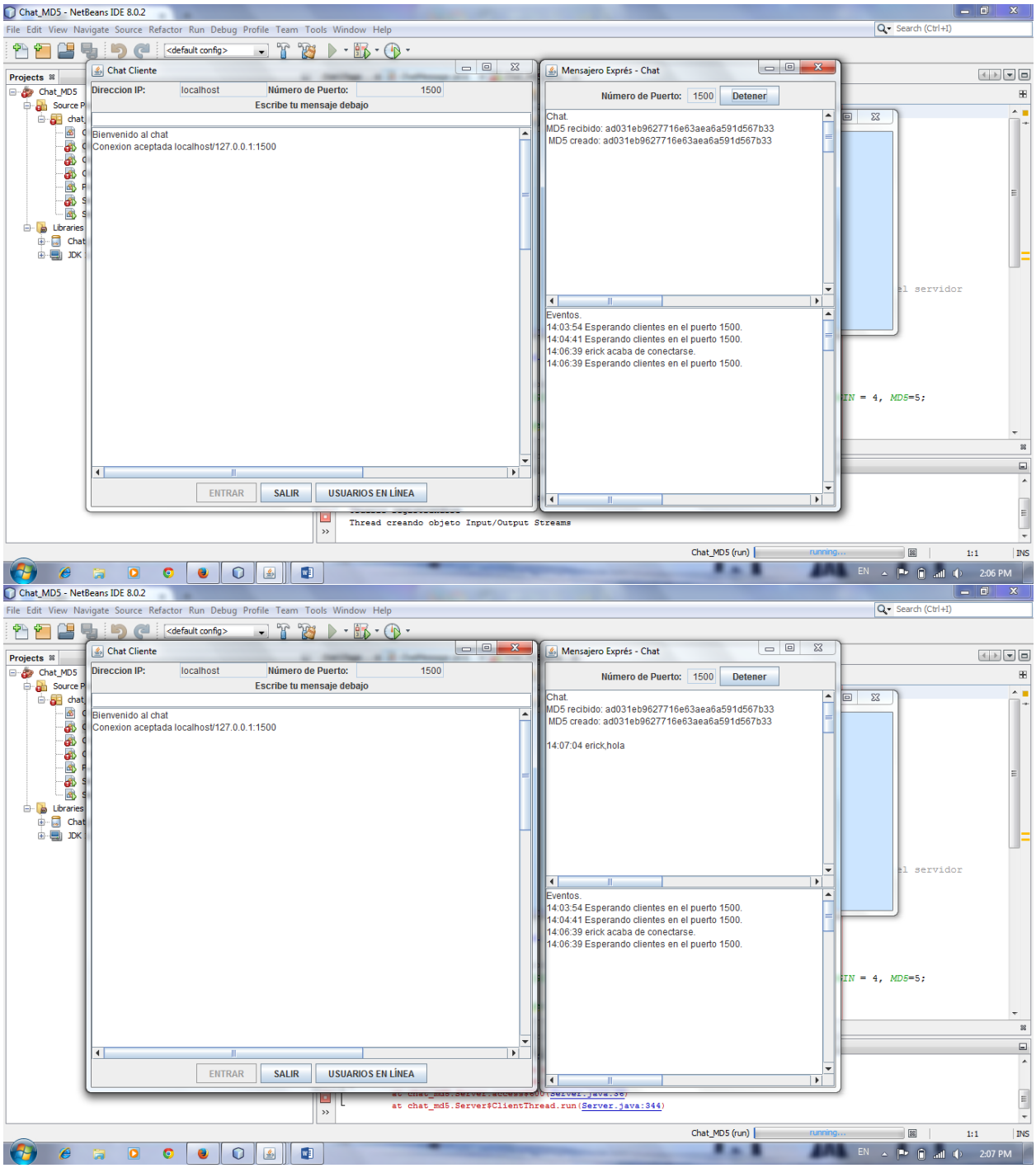

Chat_MD5 - NetBeans IDE 8.0.2
Ventana
Mensajero Exprés
Conectarse
Registrarse
Iniciar Sesion
Registro
Usuario
erick
Contraseña
Contactos
1
Aceptar
Principal
Bienvenido
Iniciar Servidor
Iniciar Cliente
Detener
Usuario registrandose
Message
Usuario registrado
OK





Una vez dentro del chat elegiremos el nombre con el cual nos identificaran y podremos chatear


Chat_MD5 - NetBeans IDE 8.0.2
Chat Cliente
Direccion IP:
localhost
Número de Puerto:
1500
Escribe un nick debajo
erick
Bienvenido al chat
ENTRAR
SALIR
USUARIOS EN LÍNEA
Usuario registrandose



A continuación se presentan los códigos del servidor y el cliente además del MD5

Código del servidor

```
package chat_md5;

import com.sun.xml.messaging.saaj.packaging.mime.util.BASE64DecoderStream;

import com.sun.xml.messaging.saaj.packaging.mime.util.BASE64EncoderStream;

import java.io.*;

import java.math.BigInteger;

import java.net.*;

import java.security.InvalidAlgorithmParameterException;

import java.security.InvalidKeyException;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.security.NoSuchProviderException;

import java.text.SimpleDateFormat;

import java.util.*;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.SecretKey;

import javax.crypto.spec.IvParameterSpec;

import javax.crypto.spec.SecretKeySpec;


public class Server {

//**************************************************************************
**********

   // Base de datos
```

```
    private Map<Integer,String> usuarios  = new HashMap<>();
    private Map<String,String> passwords  = new HashMap<>();
    private Map<String,ArrayList<String>> contactos  = new HashMap<>();
    private ArrayList<String> lista = new ArrayList<>();
    private String pass;
    private String mensajeAleatorio;



//*********************************************************************************
**********



//*********************************************************************************
**********
    //Desencriptación
    private static Cipher dcipher, ecipher;
    private static SecretKey key;

//*********************************************************************************
**********
    // un ID único para cada conexión
    private static int uniqueId;
    // ArrayList para guardar a los clientes
    private ArrayList<ClientThread> al;
    // GUI
    private ServerGUI sg;
    // para mostrar tiempo
    private SimpleDateFormat sdf;
    // puerto para escuchar conexiones
    private int port;
    // para ver si debemos parar el servidor
```

```
private boolean keepGoing;

private byte[] keyBytes;
private byte[] iviBytes;
private boolean existe1;

public Server(int port) {
        this(port, null);
   this.existe1 = false;
   keyBytes = "12345678".getBytes();
   iviBytes = "input123".getBytes();
}

public Server(int port, ServerGUI sg) {
   this.existe1 = false;
   keyBytes = "12345678".getBytes();
   iviBytes = "input123".getBytes();
        // GUI or not
        this.sg = sg;
        // the port
        this.port = port;
        // to display hh:mm:ss
        sdf = new SimpleDateFormat("HH:mm:ss");
        // ArrayList for the Client list
        al = new ArrayList<ClientThread>();
}

public void start() {
        keepGoing = true;
```

```
 /* iniciar servidor */
 try
 {
// socket usado por el servidor
ServerSocket serverSocket = new ServerSocket(port);
// ciclo infinito para esperar conexiones
while(keepGoing)
{
		display("Esperando clientes en el puerto " + port + ".");

		Socket socket = serverSocket.accept();  // aceptar conexión
		// si se detuvo el servidor
		if(!keepGoing)
   break;
		ClientThread t = new ClientThread(socket);  // hilos para el cliente
		al.add(t);                                  // se guarda en el ArrayList
		t.start();
}
// Detener el sevidor
try {
		serverSocket.close();
		for(int i = 0; i < al.size(); ++i) {
	ClientThread tc = al.get(i);
	try {
			tc.sInput.close();
			tc.sOutput.close();
			tc.socket.close();
	}
	catch(IOException ioE) {
```

```
                display(ioE + "");
            }
                }
        }
        catch(Exception e) {
           display("Exception closing the server and clients: " + e);
        }
         }
         // si hay errores
         catch (IOException e) {
        String msg = sdf.format(new Date()) + " Exception on new ServerSocket: " + e + "\n";
        display(msg);
         }
}
/*
 * Detener el servidor desde la interfaz
 */
protected void stop() {
   keepGoing = false;
   // conectarme como cliente
   // Socket socket = serverSocket.accept();
   try {
      new Socket("localhost", port);
   }
   catch(Exception e) {
   }
}
/*
 * Mostrar evento
```

```
 */
private void display(String msg) {
        String time = sdf.format(new Date()) + " " + msg;
        if(sg == null)
      System.out.println(time);
        else
      sg.appendEvent(time + "\n");
}
/*
 *  broadcast mensajes
 */
private synchronized void broadcast(String message) {
   // agregar HH:mm:ss y \n al mensaje
   String [] datos = message.split(",");
   String time = sdf.format(new Date());
   String messageLf = time + " " + message + "\n";
        // mostrar mensaje en consolo o GUI
        if(sg == null)
      System.out.print(messageLf);
        else
      sg.appendRoom(messageLf);     //mostrar en la ventana del servidor

   // Ciclo en reversa en caso de que se necesite remover un cliente
   // porque se desconectó
   for(int i = al.size(); --i >= 0;) {
      ClientThread ct = al.get(i);
      // Escribimos al cliente, si no se puede, se elimina de la lista
      if(!ct.writeMsg("")) {
         al.remove(i);
```

```
   display("Cliente desconectado " + ct.username + " no existe en la lista.");
}else{
   if(ct.username.equals((datos[2]))){
      for(int p = al.size(); --p >= 0;) {
            ClientThread ct2 = al.get(p);
            if(ct2.username.equals((datos[0]))){
               try {
                  String time2 = sdf.format(new Date());
                  String message2 = time2 + " " + datos[0] + ": " + datos[1] + "\n";
                  SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
                  IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
                  ecipher = Cipher.getInstance("DES/CTR/NoPadding");
                  // inicializar los ciphers con la llave
                  ecipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
                  String encrypted = encrypt(message2);
                  ct2.writeMsg(encrypted);
               } catch (NoSuchAlgorithmException ex) {
                  Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
               } catch (NoSuchPaddingException ex) {
                  Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
               } catch (InvalidKeyException ex) {
                  Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
               } catch (InvalidAlgorithmParameterException ex) {
                  Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
               }
            }
      }
      try {
         String time2 = sdf.format(new Date());
```

```
                String message2 = time2 + " " + datos[0] + ": " + datos[1] + "\n";
                SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
                IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
                ecipher = Cipher.getInstance("DES/CTR/NoPadding");
                // inicializar los ciphers con la llave
                ecipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
                String encrypted = encrypt(message2);
                ct.writeMsg(encrypted);
            } catch (NoSuchAlgorithmException ex) {
                Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
            } catch (NoSuchPaddingException ex) {
                Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
            } catch (InvalidKeyException ex) {
                Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
            } catch (InvalidAlgorithmParameterException ex) {
                Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
}

// cerrar sesion con mensaje de LOGOUT
synchronized void remove(int id) {
   // buscamos el id
        for(int i = 0; i < al.size(); ++i) {
      ClientThread ct = al.get(i);
      // lo encontramos
      if(ct.id == id) {
```

```
                al.remove(i);
                return;
        }
         }
}

public static void main(String[] args) {
         int portNumber = 1500;
         switch(args.length) {
        case 1:
                try {
             portNumber = Integer.parseInt(args[0]);
                }
                catch(Exception e) {
             System.out.println("Invalid port number.");
             System.out.println("Usage is: > java Server [portNumber]");
             return;
                }
        case 0:
        break;
        default:
                System.out.println("Usage is: > java Server [portNumber]");
                return;

         }
         // crea objeto servidor y se inicia
         Server server = new Server(portNumber);
         server.start();
}
```

```
/** Hilo para cada cliente */
class ClientThread extends Thread {
        // socket para la conexión
        Socket socket;
        ObjectInputStream sInput;
        ObjectOutputStream sOutput;
        // id unico
        int id;
        String username;
        ChatMessage cm;
   //Fecha en la que se conecta
        String date;
        ClientThread(Socket socket) {
      // unico id
      id = ++uniqueId;
      this.socket = socket;
      System.out.println("Thread creando objeto Input/Output Streams");
      try
      {
         //creamos input y output
                  sOutput = new ObjectOutputStream(socket.getOutputStream());
                  sInput  = new ObjectInputStream(socket.getInputStream());
                  username = (String) sInput.readObject();
         if(username.equals("SYSTEM")){
            System.out.println("Usuario registrandose");
         }else{
            display(username + " acaba de conectarse.");
         }
```

```
}
catch (IOException e) {
   display("Exception creating new Input/output Streams: " + e);
   return;
}
catch (ClassNotFoundException e) {
}
date = new Date().toString() + "\n";
  }

  // corre por siempre
  public void run() {
boolean keepGoing = true;
while(keepGoing) {
            // leer cadena como objeto
            try {
      cm = (ChatMessage) sInput.readObject();
            }
            catch (IOException e) {
      display(username + " Exception reading Streams: " + e);
      break;
            }
            catch(ClassNotFoundException e2) {
      break;
            }
            String message = cm.getMessage();
   //System.out.println(message);
            // switch para los tipos de mensaje
            switch(cm.getType()) {
```

```
case ChatMessage.MESSAGE:
   try {
      // generar llave secreta
      //System.out.println(message+" key: "+key);
      String msg = message;
      SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
      IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
      //key = KeyGenerator.getInstance("DES").generateKey();
      dcipher = Cipher.getInstance("DES/CTR/NoPadding");
      dcipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
      String decrypted = decrypt(msg);
      broadcast(username + "," + decrypted);
   }
   catch (NoSuchAlgorithmException e) {
      System.out.println("No Such Algorithm:" + e.getMessage());
      return;
   }
   catch (NoSuchPaddingException e) {
      System.out.println("No Such Padding:" + e.getMessage());
      return;
   }
   catch (InvalidKeyException e) {
      System.out.println("Invalid Key:" + e.getMessage());
      return;
   } catch (InvalidAlgorithmParameterException ex) {
   Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
   }
break;
case ChatMessage.LOGOUT:
```

```
                    display(username + " desconectado con un mensaje de SALIR.");
                    keepGoing = false;
            break;
            case ChatMessage.WHOISIN:
               try {
                  SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
                  IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
                  ecipher = Cipher.getInstance("DES/CTR/NoPadding");
                  ecipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
                  String encrypted = encrypt("\n\nLista de usuarios conectados " + sdf.format(new
Date()) + "\n\n");
                  writeMsg(encrypted);
               } catch (NoSuchAlgorithmException ex) {
                  Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
               } catch (NoSuchPaddingException ex) {
                  Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
               } catch (InvalidKeyException ex) {
                  Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
               } catch (InvalidAlgorithmParameterException ex) {
                  Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
               }
                    // Revisa todos los usuarios conectados
               int cont=1;
                    for(int i = 0; i < al.size(); ++i) {
                  ClientThread ct = al.get(i);
                  if(!ct.username.equals("SYSTEM")){
                     try {
                        SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
                        IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
                        ecipher = Cipher.getInstance("DES/CTR/NoPadding");
```

```
                ecipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
                String encrypted = encrypt(cont + ") " + ct.username + " desde " + ct.date);
                cont++;
                writeMsg(encrypted);
            } catch (NoSuchAlgorithmException ex) {
                Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
            } catch (NoSuchPaddingException ex) {
                Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
            } catch (InvalidKeyException ex) {
                Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
            } catch (InvalidAlgorithmParameterException ex) {
                Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
                    }
break;
case ChatMessage.REGISTRY:
    try {
        String[] mensj = message.split(",");
        SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
        IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
        dcipher = Cipher.getInstance("DES/CTR/NoPadding");
        dcipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
        String decrypted = decrypt(mensj[0]);
        String decrypted2 = decrypt(mensj[1]);
        String decrypted3 = decrypt(mensj[2]);
        String [] contactos2 = decrypted3.split(",");
        if(usuarios.size()>0){
            Collection c = usuarios.values();
```

```
        Iterator itr = c.iterator();
        while (itr.hasNext()) {
           if(itr.next().equals(decrypted)){
              existe1=true;
              break;
           }
        }
        if(!existe1){
           usuarios.put(usuarios.size(), decrypted);
           passwords.put(decrypted, decrypted2);
           for(int l=0;l<contactos2.length-1;l++){
              lista.add(contactos2[l]);
           }
           contactos.put(decrypted,lista );
           writeMsg("VALIDO");
        }else{
           writeMsg("INVALIDO");
           existe1 = false;
           //JOptionPane.showMessageDialog(ventana, "Usuario invalido");
           //System.out.println("hola");
        }
     }else{
         usuarios.put(usuarios.size(), decrypted);
         passwords.put(decrypted, decrypted2);
         writeMsg("VALIDO");
     }
  }
  catch (NoSuchAlgorithmException e) {
     System.out.println("No Such Algorithm:" + e.getMessage());
```

```
                return;
            }
            catch (NoSuchPaddingException e) {
                System.out.println("No Such Padding:" + e.getMessage());
                return;
            }
            catch (InvalidKeyException e) {
                System.out.println("Invalid Key:" + e.getMessage());
                return;
            } catch (InvalidAlgorithmParameterException ex) {
            Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
            }
        break;
        case ChatMessage.LOGIN:
            try {
                String msg3 = message;
                SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
                IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
                dcipher = Cipher.getInstance("DES/CTR/NoPadding");
                dcipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
                String decrypted = decrypt(msg3);
                //System.out.println(decrypted);
                if(usuarios.size()>0){
                    for(Map.Entry<String, String> entry : passwords.entrySet()){
                        //System.out.printf("Key : %s and Value: %s %n", entry.getKey(),
entry.getValue());
                        if(entry.getKey().equals(decrypted)){
                            pass = entry.getValue();
                            existe1 = true;
                            mensajeAleatorio="";
```

```
    }
}
if(existe1){
    writeMsg("VALIDO2");
    existe1 = false;
    for(int i=0;i<5000;i++){
        mensajeAleatorio+=rndChar();
    }
    try {
        SecretKeySpec key2 = new SecretKeySpec(keyBytes, "DES");
        IvParameterSpec ivSpec2 = new IvParameterSpec(iviBytes);
        ecipher = Cipher.getInstance("DES/CTR/NoPadding");
        ecipher.init(Cipher.ENCRYPT_MODE, key2, ivSpec2);
        String encrypted = encrypt(mensajeAleatorio);
        writeMsg(encrypted);
    } catch (NoSuchAlgorithmException ex) {
        Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
    } catch (NoSuchPaddingException ex) {
        Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
    } catch (InvalidKeyException ex) {
        Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
    } catch (InvalidAlgorithmParameterException ex) {
        Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
    }
}else{
    writeMsg("INVALIDO2");
    //JOptionPane.showMessageDialog(ventana, "Usuario invalido");
    //System.out.println("hola");
}
```

```
        }else{
            writeMsg("INVALIDO2");
        }
    }
    catch (NoSuchAlgorithmException e) {
        System.out.println("No Such Algorithm:" + e.getMessage());
        return;
    }
    catch (NoSuchPaddingException e) {
        System.out.println("No Such Padding:" + e.getMessage());
        return;
    }
    catch (InvalidKeyException e) {
        System.out.println("Invalid Key:" + e.getMessage());
        return;
    } catch (InvalidAlgorithmParameterException ex) {
    Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
    }
break;
case ChatMessage.MD5:
    try {
        String msg4 = message;
        SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
        IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
        dcipher = Cipher.getInstance("DES/CTR/NoPadding");
        dcipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
        String decrypted = decrypt(msg4);
        String preHuella ="";
        preHuella+=mensajeAleatorio;
```

```
                    preHuella+=pass;
                    String huella = getMD5(preHuella);
                    sg.appendRoom("MD5 recibido: " + decrypted + "\n MD5 creado: " + huella +
"\n\n");
                    if(huella.equals(decrypted)){
                        writeMsg("IGUALES");
                    }else{
                        writeMsg("NOIGUALES");
                    }
                }
                catch (NoSuchAlgorithmException e) {
                    System.out.println("No Such Algorithm:" + e.getMessage());
                    return;
                }
                catch (NoSuchPaddingException e) {
                    System.out.println("No Such Padding:" + e.getMessage());
                    return;
                }
                catch (InvalidKeyException e) {
                    System.out.println("Invalid Key:" + e.getMessage());
                    return;
                } catch (InvalidAlgorithmParameterException ex) {
                Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
                }
            break;
                }
        }
        // remover servidor de la lista de clientes conectados
        remove(id);
        close();
```

```
  }


  // tratar de cerrar todo
  private void close() {
// tratar de cerrar la conexión
try {
            if(sOutput != null) sOutput.close();
}
catch(Exception e) {}
try {
   if(sInput != null) sInput.close();
}
catch(Exception e) {};
try {
   if(socket != null) socket.close();
}
catch (Exception e) {}
  }



  /*
  * Escribir un mesaje al cliente
  */
  private boolean writeMsg(String msg) {
// enviar el mensaje al cliente si aun está conectado
if(!socket.isConnected()) {
   close();
            return false;
}
```

```
        // escribir el mensaje
        try {
           sOutput.writeObject(msg);
        }
        // si existe un error, se notifica al usuario
        catch(IOException e) {
                    display("Error al enviar mensaje a " + username);
                    display(e.toString());
        }
        return true;
         }
}

public static String decrypt(String str) {
   try {
      // decodificar con Base64
      byte[] dec = BASE64DecoderStream.decode(str.getBytes());
      byte[] utf8 = dcipher.doFinal(dec);
      return new String(utf8, "UTF-8");
   }
   catch (Exception e) {
      e.printStackTrace();
   }
   return null;
}

public char rndChar () {
   int rnd = (int) (Math.random() * 52); // usar un numero aleatorio
   char base = (rnd < 26) ? 'A' : 'a';
```

```
    return (char) (base + rnd % 26);
}

public static String getMD5(String input) {
    try {
        MessageDigest md = MessageDigest.getInstance("MD5");
        byte[] messageDigest = md.digest(input.getBytes());
        BigInteger number = new BigInteger(1, messageDigest);
        String hashtext = number.toString(16);
        // lo rellenamos para tener los 32 caracteres
        while (hashtext.length() < 32) {
            hashtext = "0" + hashtext;
        }
        return hashtext;
    }
    catch (NoSuchAlgorithmException e) {
        throw new RuntimeException(e);
    }
}

public static String encrypt(String str) {
    try {
        byte[] utf8 = str.getBytes("UTF-8");
        byte[] enc = ecipher.doFinal(utf8);
        enc = BASE64EncoderStream.encode(enc);
        return new String(enc);
    }
    catch (Exception e) {
        e.printStackTrace();
```

```
        }
        return null;
    }
}
```

Código del cliente.

```
package chat_md5;

import com.sun.xml.messaging.saaj.packaging.mime.util.BASE64DecoderStream;
import java.net.*;
import java.io.*;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.crypto.Cipher;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

/*
 * El cliente puede correr por consola o con GUI
 */
public class Client  {
    private static Cipher dcipher;
    private byte[] keyBytes;
    private byte[] iviBytes;
    // para I/O
```

```
private ObjectInputStream sInput;              // leer desde el socket
private ObjectOutputStream sOutput;                    // esribir en el socket
private Socket socket;
// si se utiliza GUI o no
private ClientGUI cg;
private Chat_MD5 ch;
// servidor puerto y nombre de usuario
private String server, username;
private int port;
/*
 *  Constructor llamado por console
 *  server: la direccion del servidor
 *  port: el numero de puerto
 *  username: nombre de usuario
 */

Client(String server, int port, String username) {
        // constructor en caso de que no se use GUI
        this(server, port, username, null);
   keyBytes = "12345678".getBytes();
   iviBytes = "input123".getBytes();
}
/*
* Constructor llamado cuando se usa la GUI
* en modo consola, el parametro GUI es null
*/
Client(String server, int port, String username, ClientGUI cg) {
   keyBytes = "12345678".getBytes();
   iviBytes = "input123".getBytes();
```

```
        this.server = server;
        this.port = port;
        this.username = username;
        // saber si estamos en modo GUI o no
        this.cg = cg;
}

/*
* Iniciar el dialogo
*/
public boolean start() {
        // intentar la conexion con el servidor
        try {
      socket = new Socket(server, port);
        }
        catch(Exception ec) {
      display("Error al conectarse al servidor:" + ec);
      return false;
        }

        String msg = "Conexion aceptada " + socket.getInetAddress() + ":" + socket.getPort();
        display(msg);
        /* Crear los Streams */
        try
        {
      sInput  = new ObjectInputStream(socket.getInputStream());
      sOutput = new ObjectOutputStream(socket.getOutputStream());
        }
        catch (IOException eIO) {
```

```
        display("Error creando nuevos Input/output Streams: " + eIO);
        return false;
          }
    // crear el hilo para escuchar desde el servior
          new ListenFromServer().start();
          try
          {
        sOutput.writeObject(username);
          }
          catch (IOException eIO) {
        display("Error al iniciar : " + eIO);
        disconnect();
        return false;
          }
          // la conexión funciona
          return true;
}

/*
* Enviar mensaje a la consola o a la GUI
*/
private void display(String msg) {
if(cg == null)
    System.out.println(msg);
else
    cg.append(msg + "\n");
}

/*
```

```
* Enviar mensaje al servidor
*/
void sendMessage(ChatMessage msg) {
   try {
      sOutput.writeObject(msg);
        }
        catch(IOException e) {
      display("Error al escribir al servidor: " + e);
        }
}

/*
* Cuando algo sale mal
* Cerrar los streams y desconectar
*/
private void disconnect() {
   try {
      if(sInput != null) sInput.close();
        }
        catch(Exception e) {}
        try {
      if(sOutput != null) sOutput.close();
        }
        catch(Exception e) {}
   try{
      if(socket != null) socket.close();
        }
        catch(Exception e) {}
        // informar a la GUI
```

```
        if(cg != null)
      cg.connectionFailed();


  }


  public static void main(String[] args) {
        // valores por defecto
        int portNumber = 1500;
        String serverAddress = "localhost";
        String userName = "Anonimo";
        // contamos el numero de parametros
        switch(args.length) {
      // > javac Client username portNumber serverAddr
      case 3:
        serverAddress = args[2];
                // > javac Client username portNumber
      case 2:
        try {
          portNumber = Integer.parseInt(args[1]);
        }
        catch(Exception e) {
          System.out.println("Numero de puerto invalido.");
          System.out.println("Debe ser is: > java Client [nombre usuario] [puerto] [direccion
servidor]");
          return;
        }
        // > javac Client username
      case 1:
        userName = args[0];
        // > java Client
```

```
        case 0:
        break;
        // numero invalido de argumentos
        default:
          System.out.println("Usage is: > java Client [nombre usuario] [puerto] {direccion
servidor]");
          return;
         }
      // crea el objeto cliente
         Client client = new Client(serverAddress, portNumber, userName);
         // probar si sirve la conexion
         if(!client.start())
         return;
         // espera mensajes del usuario
         Scanner scan = new Scanner(System.in);
         // esperar mensajes
         while(true) {
       System.out.print("> ");
       // leer mensaje
       String msg = scan.nextLine();
       // salir
       if(msg.equalsIgnoreCase("SALIR")) {
                    client.sendMessage(new ChatMessage(ChatMessage.LOGOUT, ""));
                    // desconectar
                    break;
       }
       // mensaje WhoIsIn
       else if(msg.equalsIgnoreCase("VER")) {
          client.sendMessage(new ChatMessage(ChatMessage.WHOISIN, ""));
```

```
        }
        else {                                  // un mensaje cualquiera
           client.sendMessage(new ChatMessage(ChatMessage.MESSAGE, msg));
        }
         }
         // desconectar
         client.disconnect();
}

/*
* una clase que espera mensajes del servidor
*/
class ListenFromServer extends Thread {
         public void run() {
        while(true) {
           try {
              String msg = (String) sInput.readObject();
              if(cg == null) {
                                 System.out.println(msg);
                                 System.out.print("> ");
              }
              else {
                 SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
                 IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
                 dcipher = Cipher.getInstance("DES/CTR/NoPadding");
                 dcipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
                 String decrypted = decrypt(msg);
                 cg.append(decrypted);
              }
```

```
                }
                catch(IOException e) {
            display("El servidor ha cerrado la conexion: " + e);
            if(cg != null)
               cg.connectionFailed();
            break;
                }
                catch(ClassNotFoundException e2) {
                } catch (NoSuchAlgorithmException ex) {
            Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null, ex);
          } catch (NoSuchPaddingException ex) {
            Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null, ex);
          } catch (InvalidKeyException ex) {
            Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null, ex);
          } catch (InvalidAlgorithmParameterException ex) {
            Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null, ex);
          }
       }
        }
}

  public static String decrypt(String str) {
    try {
       byte[] dec = BASE64DecoderStream.decode(str.getBytes());
       byte[] utf8 = dcipher.doFinal(dec);
       return new String(utf8, "UTF-8");
    }
    catch (Exception e) {
       e.printStackTrace();
```

```
        }
        return null;
    }
}
```

Chat MD5

```
package chat_md5;

//import com.sun.xml.messaging.saaj.packaging.mime.util.BASE64DecoderStream;
//import com.sun.xml.messaging.saaj.packaging.mime.util.BASE64EncoderStream;
import java.awt.Color;
import java.awt.List;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import java.math.BigInteger;
import java.net.Socket;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
```

```
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import javax.swing.*;
import static javax.swing.WindowConstants.DISPOSE_ON_CLOSE;

public class Chat_MD5 extends JFrame implements Serializable{
   String[] argumentos = new String[]{""};
   private final static long serialVersionUID = 1;
   private JFrame ventana;
   private JButton registro;
   private JButton iniciarSesion;
   private JButton conexion;
   private JLabel bienvenido;
   private String contactos;
   private String registroUsuario;
   private String registroCon;
   private String loginUsuario;
   private String loginCon;
   private JFrame frame;
   private JPanel panel;
   private JLabel userLabel;
   private JTextField userText;
   private JLabel passwordLabel;
```

```
private JPasswordField passwordText;
private JLabel contactLabel;
private JTextField contactText;
private JButton aceptar;
private JFrame frame2;
private JPanel panel2;
private JLabel userLabel2;
private JTextField userText2;
private JLabel passwordLabel2;
private JPasswordField passwordText2;
private JButton aceptar2;
private JFrame frame3;
private JPanel panel3;
private JLabel userLabel3;
private JTextField userText3;
private JButton aceptar3;
private boolean existe1 = false;
String[] arguments = new String[] {""};
private static Cipher ecipher, dcipher;

private String encrypted, encrypted2, encrypted3;

private byte[] keyBytes;
private byte[] iviBytes;

// for I/O
private ObjectInputStream sInput;              // to read from the socket
private ObjectOutputStream sOutput;                    // to write on the socket
private Socket socket;
```

```
private String server;
private int port;



public Chat_MD5(){
   inicializar();
}

public void inicializar(){
   keyBytes = "12345678".getBytes();
   iviBytes = "input123".getBytes();
   port = 1500;
   server = "localhost";
   ventana = new JFrame("Ventana");
   ventana.setSize(500, 600);
   ventana.setLocationRelativeTo(null);
   ventana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   ventana.setVisible(true);
   ventana.getContentPane().setBackground(new Color(208,229,255));
   ventana.setLayout(null);

   bienvenido = new JLabel();
   ventana.add(bienvenido);
   bienvenido.setText("<html><span style='font-size:18px'>Mensajero Exprés</span></html>");
   bienvenido.setBounds(148, 80, 220, 80);

   conexion = new JButton();
   ventana.add(conexion);
   conexion.setBackground(new Color(102,255,102));
```

```
conexion.setForeground(Color.BLACK);
conexion.setText("Conectarse");
conexion.setBounds(195, 160, 115, 50);

registro = new JButton();
ventana.add(registro);
registro.setBackground(new Color(102,255,102));
registro.setForeground(Color.BLACK);
registro.setText("Registrarse");
registro.setBounds(195, 230, 115, 50);

iniciarSesion = new JButton();
ventana.add(iniciarSesion);
iniciarSesion.setBackground(new Color(102,255,102));
iniciarSesion.setForeground(Color.BLACK);
iniciarSesion.setText("Iniciar Sesion");
iniciarSesion.setBounds(195, 300, 115, 50);



registro.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mouseClicked(java.awt.event.MouseEvent evt) {
      registroMouseClicked(evt);
   }
});

iniciarSesion.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mouseClicked(java.awt.event.MouseEvent evt) {
      iniciarSesionMouseClicked(evt);
```

```
        }
    });

    conexion.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            conexionMouseClicked(evt);
        }
    });

    pack();
}

private void conexionMouseClicked(java.awt.event.MouseEvent evt) {
    ventana.setEnabled(false);
    frame3 = new JFrame("Conectarse");
    frame3.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        frame3.setSize(300, 150);
    frame3.setLocationRelativeTo(null);
        panel3 = new JPanel();
        frame3.add(panel3);
    panel3.setLayout(null);

        userLabel3 = new JLabel("IP");
        userLabel3.setBounds(10, 10, 80, 25);
        panel3.add(userLabel3);

        userText3 = new JTextField(20);
        userText3.setBounds(100, 10, 160, 25);
        panel3.add(userText3);
```

```
    aceptar3 = new JButton("Aceptar");
        aceptar3.setBounds(100, 80, 80, 25);
        panel3.add(aceptar3);

    aceptar3.addMouseListener(new java.awt.event.MouseAdapter() {
      public void mouseClicked(java.awt.event.MouseEvent evt) {
        aceptar3MouseClicked(evt);
      }
    });

    frame3.addWindowListener(new java.awt.event.WindowAdapter() {
    @Override
    public void windowClosing(java.awt.event.WindowEvent windowEvent) {
      ventana.setEnabled(true);
    }
    });
        frame3.setVisible(true);
}

private void aceptar3MouseClicked(java.awt.event.MouseEvent evt) {
    if(userText3.getText().trim().length()==0){
      JOptionPane.showMessageDialog(frame, "Datos incompletos");
    }else{
      System.out.println(server);
      String servidor = userText3.getText().trim();
      System.out.println(servidor);
      conectar(servidor);
      ventana.setEnabled(true);
```

```
        frame3.dispose();
    }
}

private void registroMouseClicked(java.awt.event.MouseEvent evt) {
    ventana.setEnabled(false);
    frame = new JFrame("Registro");
    frame.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        frame.setSize(300, 200);
    frame.setLocationRelativeTo(null);
        panel = new JPanel();
        frame.add(panel);
    panel.setLayout(null);

        userLabel = new JLabel("Usuario");
        userLabel.setBounds(10, 10, 80, 25);
        panel.add(userLabel);

        userText = new JTextField(20);
        userText.setBounds(100, 10, 160, 25);
        panel.add(userText);

        passwordLabel = new JLabel("Contraseña");
        passwordLabel.setBounds(10, 40, 80, 25);
    panel.add(passwordLabel);

        passwordText = new JPasswordField(20);
        passwordText.setBounds(100, 40, 160, 25);
        panel.add(passwordText);
```

```
    contactLabel = new JLabel("Contactos");
        contactLabel.setBounds(10, 70, 80, 25);
    panel.add(contactLabel);

        contactText = new JTextField(20);
        contactText.setBounds(100, 70, 160, 25);
        panel.add(contactText);

        aceptar = new JButton("Aceptar");
        aceptar.setBounds(100, 110, 80, 25);
        panel.add(aceptar);


    aceptar.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
       aceptarMouseClicked(evt);
    }
    });

    frame.addWindowListener(new java.awt.event.WindowAdapter() {
    @Override
    public void windowClosing(java.awt.event.WindowEvent windowEvent) {
       ventana.setEnabled(true);
    }
    });

        frame.setVisible(true);
}
```

```
   private void aceptarMouseClicked(java.awt.event.MouseEvent evt) {

      if(userText.getText().trim().length()==0 || passwordText.getText().trim().length()==0 ||
contactText.getText().trim().length() == 0){

         JOptionPane.showMessageDialog(frame, "Datos incompletos");

      }else{

         registroUsuario = userText.getText();

         //System.out.println(registroUsuario);

         registroCon = passwordText.getText();

         //System.out.println(registroCon);

         contactos = contactText.getText();

         ventana.setEnabled(true);

         frame.dispose();


         try {

            SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");

            IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);

            // generate secret key using DES algorithm

            ecipher = Cipher.getInstance("DES/CTR/NoPadding");

            // initialize the ciphers with the given key

            ecipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);

            encrypted = encrypt(registroUsuario);

            encrypted2 = encrypt(registroCon);

            encrypted3 = encrypt(contactos);

            sendMessage(new ChatMessage(ChatMessage.REGISTRY,
encrypted+","+encrypted2+","+encrypted3));

         }

         catch (NoSuchAlgorithmException e) {

            System.out.println("No Such Algorithm:" + e.getMessage());

            return;
```

```
        }
        catch (NoSuchPaddingException e) {
            System.out.println("No Such Padding:" + e.getMessage());
            return;
        }
        catch (InvalidKeyException e) {
            System.out.println("Invalid Key:" + e.getMessage());
            return;
        } catch (InvalidAlgorithmParameterException ex) {
            Logger.getLogger(Chat_MD5.class.getName()).log(Level.SEVERE, null, ex);
        }

    }
}

public static String encrypt(String str) {
    try {
        // encode the string into a sequence of bytes using the named charset
        // storing the result into a new byte array.
        byte[] utf8 = str.getBytes("UTF-8");
        byte[] enc = ecipher.doFinal(utf8);
        // encode to base64
        enc = BASE64EncoderStream.encode(enc);
        return new String(enc);
    }
    catch (Exception e) {
        e.printStackTrace();
    }
    return null;
```

```
}

private void iniciarSesionMouseClicked(java.awt.event.MouseEvent evt) {
   ventana.setEnabled(false);
   frame2 = new JFrame("Login");
   frame2.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        frame2.setSize(300, 150);
   frame2.setLocationRelativeTo(null);
        panel2 = new JPanel();
        frame2.add(panel2);
   panel2.setLayout(null);

        userLabel2 = new JLabel("Usuario");
        userLabel2.setBounds(10, 10, 80, 25);
        panel2.add(userLabel2);

        userText2 = new JTextField(20);
        userText2.setBounds(100, 10, 160, 25);
        panel2.add(userText2);

        passwordLabel2 = new JLabel("Contraseña");
        passwordLabel2.setBounds(10, 40, 80, 25);
   panel2.add(passwordLabel2);

        passwordText2 = new JPasswordField(20);
        passwordText2.setBounds(100, 40, 160, 25);
        panel2.add(passwordText2);

        aceptar2 = new JButton("Aceptar");
```

```
        aceptar2.setBounds(100, 80, 80, 25);
        panel2.add(aceptar2);

    aceptar2.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            aceptar2MouseClicked(evt);
        }
    });

    frame2.addWindowListener(new java.awt.event.WindowAdapter() {
    @Override
    public void windowClosing(java.awt.event.WindowEvent windowEvent) {
        ventana.setEnabled(true);
    }
    });
        frame2.setVisible(true);
}

private void aceptar2MouseClicked(java.awt.event.MouseEvent evt) {
    if(userText2.getText().trim().length()==0 || passwordText2.getText().trim().length()==0){
        JOptionPane.showMessageDialog(frame, "Datos incompletos");
    }else{
        loginUsuario = userText2.getText();
        //System.out.println(loginUsuario);
        loginCon = passwordText2.getText();
        //System.out.println(loginCon);
        ventana.setEnabled(true);
        frame2.dispose();
        //ventana.dispose();
```

```
    try {
        SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
        IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
        // generate secret key using DES algorithm
        ecipher = Cipher.getInstance("DES/CTR/NoPadding");
        // initialize the ciphers with the given key
        ecipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
        encrypted = encrypt(loginUsuario);
        encrypted2 = encrypt(loginCon);
        sendMessage(new ChatMessage(ChatMessage.LOGIN, encrypted));
    }
    catch (NoSuchAlgorithmException e) {
        System.out.println("No Such Algorithm:" + e.getMessage());
        return;
    }
    catch (NoSuchPaddingException e) {
        System.out.println("No Such Padding:" + e.getMessage());
        return;
    }
    catch (InvalidKeyException e) {
        System.out.println("Invalid Key:" + e.getMessage());
        return;
    } catch (InvalidAlgorithmParameterException ex) {
        Logger.getLogger(Chat_MD5.class.getName()).log(Level.SEVERE, null, ex);
    }
    //ClientGUI.main(arguments);
}
```

```
}
/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
   // TODO code application logic here
   Chat_MD5 chat = new Chat_MD5();
}

public void conectar(String servidor){
   // try to connect to the server
        try {
      socket = new Socket(servidor, port);
        }
        // if it failed not much I can so
        catch(Exception ec) {
      ec.printStackTrace();
        }
   try
        {
      sInput  = new ObjectInputStream(socket.getInputStream());
      sOutput = new ObjectOutputStream(socket.getOutputStream());
      new Chat_MD5.ListenFromServer().start();
        }
        catch (IOException eIO) {
      eIO.printStackTrace();
        }
   try
        {
```

```
        sOutput.writeObject("SYSTEM");
          }
          catch (IOException eIO) {
        eIO.printStackTrace();
          }
}

void sendMessage(ChatMessage msg) {
          try {
        sOutput.writeObject(msg);
          }
          catch(IOException e) {
        e.printStackTrace();
          }
}



class ListenFromServer extends Thread {
          public void run() {
        while(true) {
           try {
              String msg = (String) sInput.readObject();
              if(msg.equals("INVALIDO")){
                  JOptionPane.showMessageDialog(ventana, "Usuario no valido");
              }
              if(msg.equals("VALIDO")){
                 JOptionPane.showMessageDialog(ventana, "Usuario registrado");
              }
              if(msg.equals("INVALIDO2")){
```

```
   JOptionPane.showMessageDialog(ventana, "Usuario y/o contraseña incorrectos");
}
if(msg.equals("VALIDO2")){

}
if(msg.length() > 1000){
   String preHuella = "";
   SecretKeySpec key = new SecretKeySpec(keyBytes, "DES");
   IvParameterSpec ivSpec = new IvParameterSpec(iviBytes);
   //key = KeyGenerator.getInstance("DES").generateKey();
   dcipher = Cipher.getInstance("DES/CTR/NoPadding");
   // initialize the ciphers with the given key
   dcipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
   String decrypted = decrypt(msg);
   preHuella+=decrypted;
   preHuella+=loginCon;
   String huella = getMD5(preHuella);
   try {
      SecretKeySpec key2 = new SecretKeySpec(keyBytes, "DES");
      IvParameterSpec ivSpec2 = new IvParameterSpec(iviBytes);
      // generate secret key using DES algorithm
      ecipher = Cipher.getInstance("DES/CTR/NoPadding");
      // initialize the ciphers with the given key
      ecipher.init(Cipher.ENCRYPT_MODE, key2, ivSpec2);
      encrypted = encrypt(huella);
      sendMessage(new ChatMessage(ChatMessage.MD5, encrypted));
   }
   catch (NoSuchAlgorithmException e) {
      System.out.println("No Such Algorithm:" + e.getMessage());
```

```
            return;
        }
        catch (NoSuchPaddingException e) {
            System.out.println("No Such Padding:" + e.getMessage());
            return;
        }
        catch (InvalidKeyException e) {
            System.out.println("Invalid Key:" + e.getMessage());
            return;
        } catch (InvalidAlgorithmParameterException ex) {
            Logger.getLogger(Chat_MD5.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    if(msg.equals("IGUALES")){
        JOptionPane.showMessageDialog(ventana, "Sesion iniciada");
        ClientGUI.main(argumentos);
        ventana.dispose();
    }
    if(msg.equals("NO IGUALES")){
        JOptionPane.showMessageDialog(ventana, "Usuario y/o contraseña incorrectos");
    }
            }
            catch(IOException e) {
    e.printStackTrace();
            }
            // can't happen with a String object but need the catch anyhow
            catch(ClassNotFoundException e2) {
            } catch (InvalidKeyException ex) {
    Logger.getLogger(Chat_MD5.class.getName()).log(Level.SEVERE, null, ex);
```

```
            } catch (InvalidAlgorithmParameterException ex) {
                Logger.getLogger(Chat_MD5.class.getName()).log(Level.SEVERE, null, ex);
            } catch (NoSuchAlgorithmException ex) {
                Logger.getLogger(Chat_MD5.class.getName()).log(Level.SEVERE, null, ex);
            } catch (NoSuchPaddingException ex) {
                Logger.getLogger(Chat_MD5.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
         }
}

public static String getMD5(String input) {
    try {
        MessageDigest md = MessageDigest.getInstance("MD5");
        byte[] messageDigest = md.digest(input.getBytes());
        BigInteger number = new BigInteger(1, messageDigest);
        String hashtext = number.toString(16);
        // Now we need to zero pad it if you actually want the full 32 chars.
        while (hashtext.length() < 32) {
            hashtext = "0" + hashtext;
        }
        return hashtext;
    }
    catch (NoSuchAlgorithmException e) {
        throw new RuntimeException(e);
    }
}
```

```
  public static String decrypt(String str) {
    try {
      // decode with base64 to get bytes
      byte[] dec = BASE64DecoderStream.decode(str.getBytes());
      byte[] utf8 = dcipher.doFinal(dec);
      // create new string based on the specified charset
      return new String(utf8, "UTF-8");
    }
    catch (Exception e) {
      e.printStackTrace();
    }
    return null;
  }
}
```

Conclusiones

En conclusión se tiene un programa que emula el cliente y servidor no del todo completo ni el mejor pero nos ayuda a ver cómo funciona el servicio de encriptación el cual como ya se dijo es el encargado de proteger la información que no queremos que sea visto por otros usuarios.

o

GRAFICACIÓN PRIMAVERA 2015

SEGUNDO PARCIAL

Graficación

201217899

KARLA ESTEFANÍA GARRIDO GONZÁLEZ

INTRODUCCION

En esta ocasión con todo lo aprendido hasta el momento, hemos podido realizar una pequeña animación de un personaje conocido por la internet como “NyanCat” el cual va pasando de izquierda a derecha por la pantalla. Todo esto será posible gracias a los conocimientos adquiridos en nuestros cursos de programación y éste curso que nos enseña a dibujar objetos en la pantalla.

CONCEPTOS DESARROLLADOS

Los conceptos que desarrollaremos son funciones primitivas de OpenGL, las cuales ya hemos trabajado anteriormente pero que enlistaremos a continuación:

- glTranslatef(x,y,z)
- glScalef(xs,ys,zs);
- glLoadIdentity()
- Pila de matrices (glPushMatrix y glPopMatrix)
- glBegin(GL_QUADS)
- glBegin(GL_POLYGON)
- Texturas.

glTranslatef(x,y,z)

Esta función se usa para poder trasladar los objetos a un determinado punto en el eje de coordenadas. Sus parámetros son:

x=la distancia en la que moveremos en el eje x.

y=la distancia en la que moveremos en el eje y.

z=la distancia en la que moveremos en el eje z.

Por ejemplo, si queremos mover 30 en el eje y, lo haremos así:

glTranslatef(0,30,0).

Esta función la usamos para que el gato se pueda mover lateralmente por la pantalla.

glScalef(xs,ys,zs)

Esta función nos ayuda a ‘escalar’ lo que estamos dibujando en la pantalla, es decir, aumentarle o disminuirle el tamaño de acuerdo a lo que queramos mostrar.

Sus parámetros son:

xs= el valor para poder redimensionar la figura con respecto al eje x.

ys= el valor para poder redimensionar la figura con respecto al eje y.

zs= el valor para poder redimensionar la figura con respecto al eje z.

Una cosa extra que podemos hacer con esta función es hacer un reflejo en la figura, esto lo podremos conseguir con solo meter uno o más parámetros con signo negativo.

glLoadIdentity()

Ésta función nos ayudará a que si queremos aplicar más de una transformación a un objeto, al ser las transformaciones acumulativas, la matriz identidad es reiniciar el origen del objeto.

Si nosotros queremos rotar un objeto y trasladar otro, antes de trasladarlo llamaremos a la función de matriz identidad para que reinicie el origen de este objeto.

Pila de matrices (glPushMatrix y glPopMatrix)

La pila de matrices nos sirve para guardar el estado de las transformaciones, la función glPushMatrix() inserta en la pila mientras que glPopMatrix() quita el ultimo estado guardado en la pila.

glBegin(GL_QUADS)

Es una primitiva de OpenGL la cual nos permite dibujar un cuadrado. La diferencia de dibujar el cuadrado con GL_QUADS y GL_LINES es que al dibujar el cuadrado con GL_QUADS, éste saldrá relleno. Otra diferencia notable son los puntos que recibe cada uno, pues en la línea es definir cada línea (en total 8 puntos, 2 por cada línea) mientras que en GL_QUADS solo definimos los cuatro vértices de la figura.

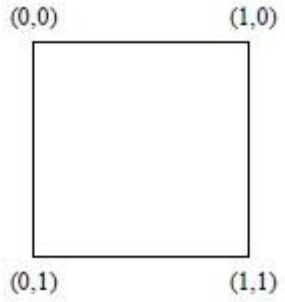
glBegin(GL_POLYGON)

Muestra un polígono relleno, cuyos vértices se proporcionan mediante funciones glVertex y se terminan mediante una función glEnd.

Texturas.

Es un estilo de relleno que nos permite personalizar aún más ciertas figuras, la cual nos permite poner una imagen con ciertas características. Por ejemplo: si queremos dibujar una pared de ladrillo, podemos poner en el área determinada una imagen de ladrillos.

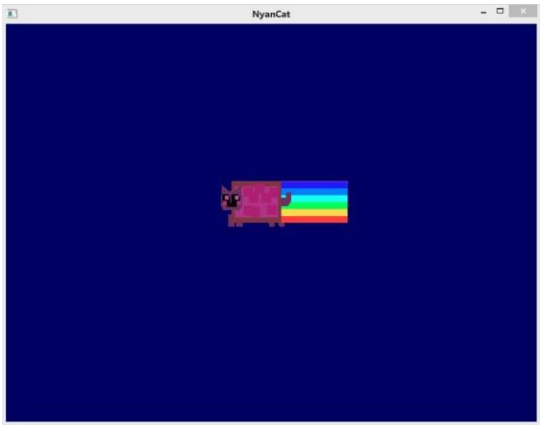
Las coordenadas de una textura son:

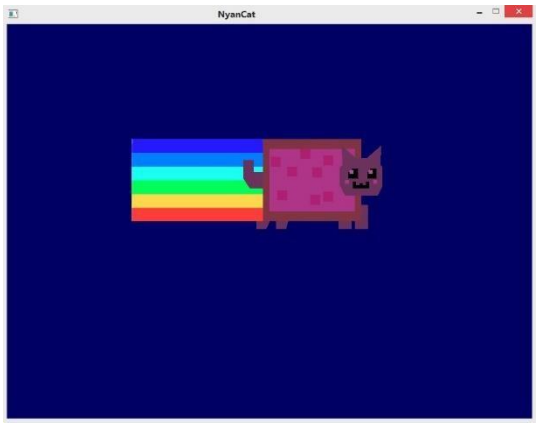


Para poder usar texturas en nuestros proyectos siguientes debemos usar librerías de imágenes como FreeImage, DevIL, etc.

CONCLUSION

Con ayuda de la pila de matrices y la traslación pudimos lograr que el gato se mueva de un lado a otro de la pantalla. La textura es la estela que en este caso es una imagen de un arcoíris, la textura me facilitó hacer el arcoíris. Para que el gato se haga más pequeño y que a imagen se refleje usamos el glScale.





BIBLIOGRAFIA

http://www.rapidtables.com/web/color/RGB_Color.htm

https://www.opengl.org/sdk/docs/man2/xhtml/glTranslate.xml

Donald Hearn, M.Pauline Baker. (2006). Gráficos por Computadora con OpenGL. Madrid: Pearson Education.

[Escriba el nombre de la compañía]

en 1

[Escriba el subtítulo del documento]

user

Introducción

Como segundo parcial se dejó la elaboración de un escenario usando las primitivas de opengl al igual que una librería grafica para cargar alguna textura y también incluyendo la traslación y rotación.

Conceptos desarrollados

Primero se había intentado hacer el proceso con la librería de freeimage pero esa librería tenía un filtro que cambiada el color y todo se veía raro ya después de varios días tratando de buscar la solución me decidí cambiar a la librería de soil donde el método para cargar texturas era mucho más pequeño y no cambiada a ningún filtro por lo tanto se veía la imagen del color original.

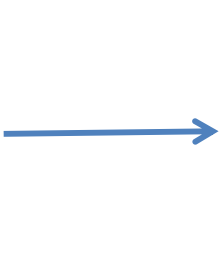
Escenario

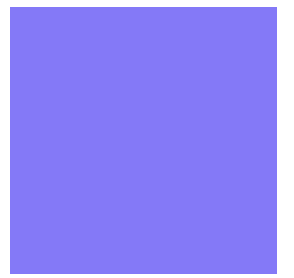
Después de haber encontrado varias imágenes para hacer el escenario y buscando de que sean el mismo color una de las primeras problemáticas fue mandar a llamar varias imágenes por lo que no sabía del arreglo era de tipo entero o char ya que en una página web decía que el arreglo era entero pero había parte que requerían un char posteriormente decidí hacer el arreglo char para poder probar y si funciono pero después al hacer el linkeo con su tempTextureId pedía un entero por lo cual solo fue hacer esa variable entera en un arreglo de variables enteras del tamaño del arreglo de las imágenes.

```
int LoadTextures(){
    for (int i=0;i<18;i++)
    {
    tempTextureID[i] = SOIL_load_OGL_texture
    (
        texturas[i],
        SOIL_LOAD_AUTO,
        SOIL_CREATE_NEW_ID,
        SOIL_FLAG_MIPMAPS | SOIL_FLAG_INVERT_Y | SOIL_FLAG_NTSC_SAFE_RGB | SOIL_FLAG_COMPRESS_TO_DXT
    );

          glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
          glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
          glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
          glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
    if( 0 == tempTextureID )
    {
       printf( "SOIL loading error: '%s'\n", SOIL_last_result() );
    }
    }


    }
```

Posteriormente fue el diseño del escenario con un dos polígonos incluyendo sus texturas:

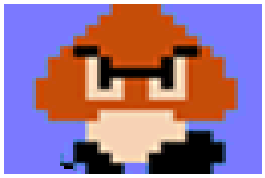
```
glBindTexture(GL_TEXTURE_2D, tempTextureID[6]);
glBegin(GL_POLYGON);
glColor3f(0.0f,0.0f,1.0f);
    glTexCoord2f(0.0f,0.0f);
    glVertex3f( -400.0f, -200.0f, 0.0f);
    glTexCoord2f(0.0f,1.0f);
    glVertex3f(-400.0f,400.0f, 0.0f);
    glTexCoord2f(1.0f,1.0f);
    glVertex3f(400.0f,400.0f, 0.0f);
    glTexCoord2f(1.0f,0.0f);
    glVertex3f(400.0f,-200.0f, 0.0f);
glEnd();
glBindTexture(GL_TEXTURE_2D, tempTextureID[7]);
glBegin(GL_POLYGON);
glColor3f(1.0f,1.0f,1.0f);
    glTexCoord2f(0.0f,0.0f);
    glVertex3f( -400.0f, -300.0f, 0.0f);
    glTexCoord2f(0.0f,1.0f);
    glVertex3f(-400.0f,-200.0f, 0.0f);
    glTexCoord2f(10.0f,1.0f);
```



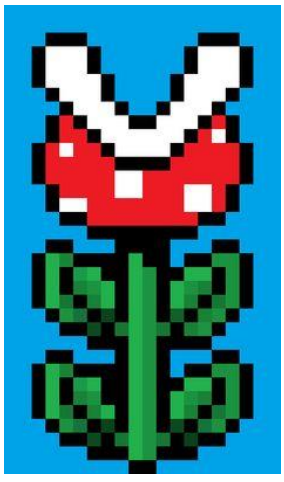


Los obstaculos

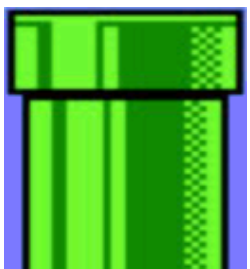
```
oid malo(){
 glPushMatrix();
 glBindTexture(GL_TEXTURE_2D, tempTextureID[12]);
 glBegin(GL_POLYGON);
 glColor3f(1.0f,1.0f,1.0f);
    glTexCoord2f(0.0f,0.25f);
                glVertex3f( 0.0f, 0.0f, 0.0f);
    glTexCoord2f(0.0f,1.0f);
                glVertex3f(0.0f,25.0f, 0.0f);
    glTexCoord2f(1.0f,1.0f);
                glVertex3f(25.0f,25.0f, 0.0f);
    glTexCoord2f(1.0f,0.25f);
                glVertex3f(25.0f,0.0f, 0.0f);
 glEnd();
```



```
void flor(){
  glPushMatrix();
  glBindTexture(GL_TEXTURE_2D, tempTextureID[11]);
  glBegin(GL_POLYGON);
  glColor3f(1.0f,1.0f,1.0f);
     glTexCoord2f(0.0f,0.25f);
                 glVertex3f( 0.0f, 0.0f, 0.0f);
     glTexCoord2f(0.0f,1.0f);
                 glVertex3f(0.0f,25.0f, 0.0f);
     glTexCoord2f(1.0f,1.0f);
                 glVertex3f(25.0f,25.0f, 0.0f);
     glTexCoord2f(1.0f,0.25f);
                 glVertex3f(25.0f,0.0f, 0.0f);
  glEnd();
```

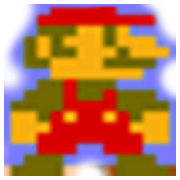


```
PushMatrix();
 glTranslatef(-300.0f,-200.0f,0.0f);
 glBindTexture(GL_TEXTURE_2D, tempTextureID[8]);
 glBegin(GL_POLYGON);
 glColor3f(1.0f,1.0f,1.0f);
    glTexCoord2f(0.0f,0.0f);
                glVertex3f( 0.0f, 0.0f, 0.0f);
    glTexCoord2f(0.0f,1.0f);
                glVertex3f(0.0f,50.0f, 0.0f);
```



Mario

```
void mario(){
    glPushMatrix();
    glBindTexture(GL_TEXTURE_2D, tempTextureID[13]);
    glBegin(GL_POLYGON);
    glColor3f(1.0f,1.0f,1.0f);
        glTexCoord2f(0.0f,0.25f);
        glVertex3f( 0.0f, 0.0f, 0.0f);
        glTexCoord2f(0.0f,1.0f);
        glVertex3f(0.0f,25.0f, 0.0f);
        glTexCoord2f(1.0f,1.0f);
        glVertex3f(25.0f,25.0f, 0.0f);
        glTexCoord2f(1.0f,0.25f);
        glVertex3f(25.0f,0.0f, 0.0f);
    glEnd();
    glPopMatrix();
    }
```



ción y traslación y algo en lo cual si se me complico fue en la traslación de arriba para abajo en donde solo pensaba que con una comparación se hacía pero no.

```
glPushMatrix();
glTranslatef(-148.0f,-150.0f,0.0f);
glPushMatrix();
glTranslatef(0.0f,yLocation,0.0f);
flor();
glPopMatrix();
```

La traslación que hacía que las flores suban y bajen,

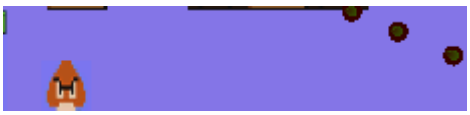
Y para eso era cuestión de moverse sobre el eje y

```
if (movingUp)
yLocation -= 0.1f;
else
yLocation += 0.1f;

if (yLocation < 0.0f)
movingUp = false;
else if (yLocation > 32.0f)
movingUp = true;
```

Tal y como se movio la flor tambien se movia el enemigo pero en el eje x

```
if (x)
xLocation -= 0.1f;
else
xLocation += 0.1f;

if (xLocation < 0.0f)
x = false;
else if (xLocation > 250.0f)
x = true;
```

```
glPushMatrix();
glTranslatef(-270.0f,-200.0f,0.0f);
glPushMatrix();
glTranslatef(xLocation,0.0f,0.0f);
malo();
glPopMatrix();
```



Movimiento de Mario

Algo que se agrego fue que mediante las flechas del teclado se moviera mario y para eso fue necesario la función de keyboardown y 2 variables de tipo flotante para mover a mario en eje x y en y.

```
float move_unit = 10.0f;

void keyboardown(int key, int x, int y) {
    switch (key) {
    case GLUT_KEY_RIGHT:
        mariox += move_unit;
        break;
    case GLUT_KEY_LEFT:
        mariox -= move_unit;
        break;
    case GLUT_KEY_UP:
        marioy += move_unit;
        break;
    case GLUT_KEY_DOWN:
        marioy -= move_unit;
        break;
    default:
        break;
    }
glutPostRedisplay();
}
```

Primero una unidad para que se mueva que en este caso lo declare de 10 unidades y después al hacer clic en la flecha derecha o izquierda se moviera en el eje x y de igual forma para arriba y abajo en el eje y y posteriormente volver a dibujar la función display.

Conclusión

Una de las principales dificultades de este programa fue cargar varias imágenes y hacer que la traslación se de arriba para abajo y de izquierda a derecha ya que yo solo pensaba que con un simple if lo haría pero no fue así, otra cosa fue el filtro en el cual se perdió mucho tiempo tratando de cambiar el color pero no funciono y al final tuve que migrar a soil, y por ultimo quise agregar el sonido para darle un toque me gusto el sonido y le da un toque más realista.

Escenario

en 1

[Escriba el subtítulo del documento]

user

Introducción

Este proyecto está basado en un escenario que se crea en base a librerías creadas por el programador con los operadores básicos para diversas figuras usando OpenGL aprenderemos a pintar estas figuras como son:

- Triángulos
- Cuadriláteros
- Círculos

Básicamente todo debe estar dibujado con puntos glBegin(GL_POINTS).

Y desde el Programa principal (Main.cpp) deben de ser llamadas cada una de estas librerías para ir formando y diseñando el escenario.

Desarrollo del Proyecto

Antes de empezar a diseñar nuestro escenario se tuvo que crear las diferentes librerías que llevaran cada una de las figuras para usar en nuestro escenario.

La función que se muestra a continuación se usa para graficar “rectas” en el plano ya que estas serán usadas para la creación del escenario

```
//*******Rectas********
void Linea2 (int a1, int b1,int a2, int b2){
//glClear(GL_COLOR_BUFFER_BIT);
glColor3f(0.0,0.0,0.0); //color negro
glMatrixMode(GL_PROJECTION);

            int  x[a2-a1], y[a2-a1],a , i;
            float m;

            m  = (float)(b2 -b1)/(float)(a2 -a1);
            a = a1 -1;

            for(i=1; i<=a2-a1; i++){
            x[i] = a + i;
            y[i] = (int) ((m*x[i])-(m*a1) + b1);
           }


            for(i= 1; i<=a2 -a1; i++){ //Se procede a graficar los puntos
               glBegin(GL_POINTS);
                       glVertex2i(x[i], y[i]);
               glEnd();
                          }
}
```

****

****

```
//Puntos Verticales
void DibujarPuntos2(int x, int y, int b){

     int i;

    glColor3f(0.0,0.0,0.0);
      for(i=y;i< y+b; i++){     //A partir
        glBegin(GL_POINTS);
            glVertex2i(x,i);
        glEnd();

         }

       }
```

****

****

```
void DibujarPuntos(int x, int y, int a){

     int i;

    glColor3f(0.0,0.0,0.0);
      for(i=x;i< x+a; i++){          //A par
        glBegin(GL_POINTS);          //Grafi
            glVertex2i(i,y);
        glEnd();
   }
         }
```

****

****

```
float p1x=1.5, p1y=8.5, radio1 =1 ,corx1,cory1;
float p2x=1.5, p2y=8.5, radio2 =3 ,corx2,cory2; //Pies del Bebe
float p3x=1.5, p3y=8.5, radio3 =7 ,corx3,cory3; //Cabeza del Bebe
float p4x=1.5, p4y=8.5, radio4 =2 ,corx4,cory4; //OjoS del Bebe
float p5x=1.5, p5y=8.5, radio5 =0.2 ,corx5,cory5; //cornia del Bebe
float p11x=1.5, p11y=8.5, radio11 =0.1 ,corx11,cory11; // Pezon del Bebe
```

La imagen anterior muestra cómo se debe declarar las variables de la creación de un circulo en este caso se pusieron punto 1 y punto 2 las coordenadas donde estas estarían y el radio para crear un circulo de diferentes tamaños

```
//-----------------CABEZA---------
for(double i=0.0;i<10;i+=0.001)
 {


corx3= radio3 *cos(i);
cory3= radio3 *sin(i);
glVertex2f(corx3+30,cory3+22);

}
```

En esta imagen se puede observar que se manda a llamar la variable “corx3” y “cory3” que son las variables de coordenadas donde se guarda el radio y esta se multiplica por la función cos o sen para la creación del círculo, a lo que se prosigue graficar mediante el glVertex2f es aquí donde el programador decide donde colocar los círculos en el área de trabajo con las coordenadas en X y Y del plano.

Se puede observar que estas librerías son únicas ya que cada una desempeña una función diferente y una figura o línea (puntos) en diferente postura.

Ya que se ha creado cada una de estas librerías.h ahora es momento de mandar a llamarlas desde nuestro programa (cabecera) principal del proyecto mismo.

```
#include <GL/glut.h>
#include <stdlib.h>
#include  <math.h>
#include <Phorizontal.h>
#include <Recta.h>
#include <Pvertical.h>
```

Se puede observar que en estas librerías están también las que se crearon para el diseño del escenario ya que se mencionaba que desde este programa principal se mandan a llamar cada una de estas para que simplemente se coloquen estas figuras en las coordenadas deseadas en el proyecto.

El proyecto o escenario que se diseñó esta vez es UN BEBE JUNTO A SU CUNA el diseño es muy poco creativo ya que con la poca herramienta para poder usar nos limitó a que el diseño fuera mejor

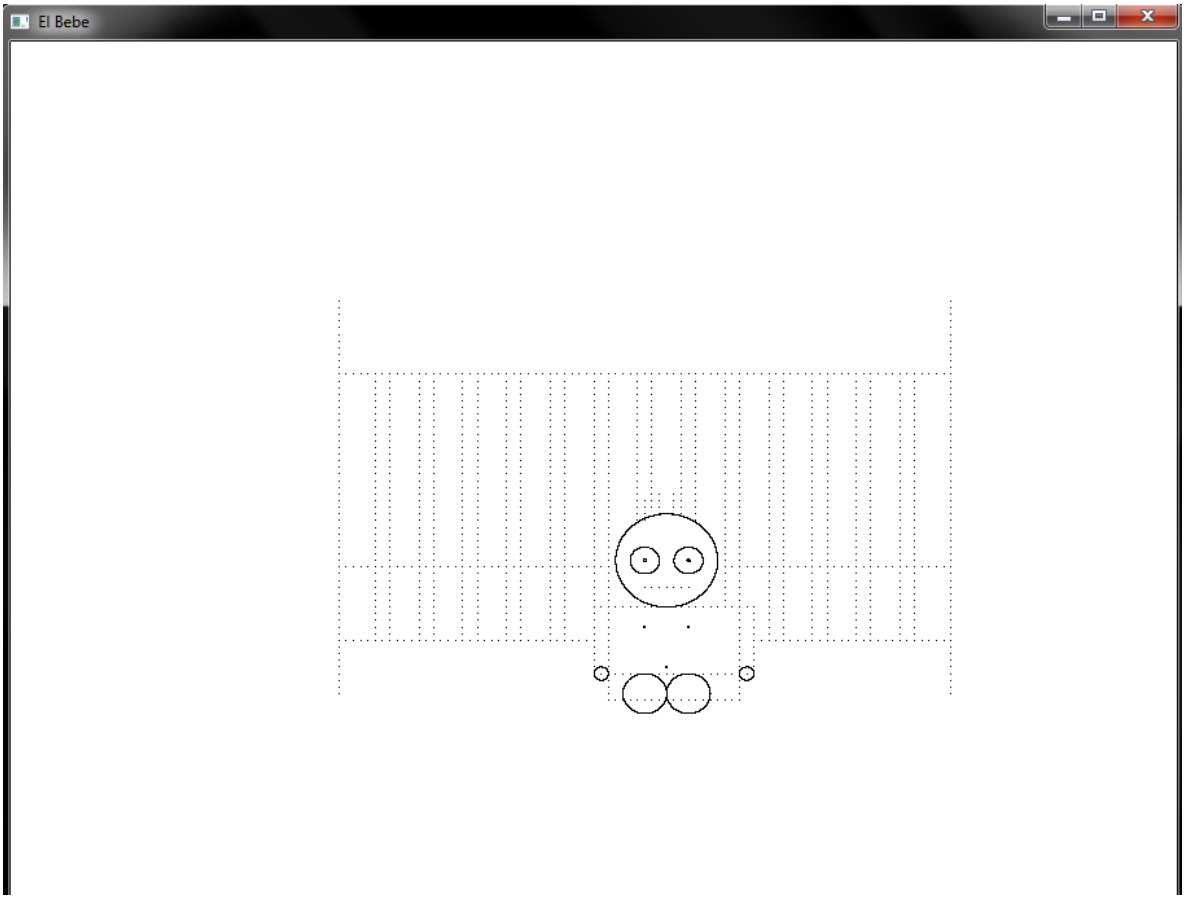
```
//--------------ESCENARIO CON PUNTOS----------------------
 DibujarPuntos(27,18,7);//Boca del bebe
 DibujarPuntos(20,15,23);//Torax del Bebe Alta
 DibujarPuntos(22,1,18);// Torax del Bebe Baja
 DibujarPuntos2(22,1,15);//Torax Izquierdo del Bebe
 DibujarPuntos2(40,1,15);//Torax Derecho del Bebe
 DibujarPuntos2(27,28,4);//pelo del bebe
 DibujarPuntos2(29,29,4);//pelo del bebe
 DibujarPuntos2(31,29,4);//pelo del bebe
 DibujarPuntos(20,5,23);//Pañal del bebe
 DibujarPuntos2(20,5,11);//Brazo izquierdo bebe
 DibujarPuntos2(42,5,11);//Brazo derecho del bebe
```

Lo que se hizo para diseñar al bebe fueron líneas con puntos y círculos como se puede observar en la imagen anterior se usa la instrucción DibujarPuntos2 para crear una línea de puntos en vertical.

```
//----------------  CUNA  ---------------------
        DibujarPuntos2(-15,2,60);//Parte Izquierda de la cuna
         DibujarPuntos2(69,2,60);//Parte derecha de la cuna



      DibujarPuntos(-15,50,85);//Linea cuna alta
      DibujarPuntos(-15,10,36);//Linea cuna baja/izquierda
      DibujarPuntos(42,10,28);//Linea cuna baja/derecha
```

DibujarPuntos es una instrucción de línea de puntos en forma horizontal y también fue usada para la construcción de la cuna del escenario

En base a todo lo creado dentro de este proyecto y librerías se muestra a continuación la forma en la que al mezclar toda la programación de OpenGL y las figuras con líneas el escenario que se pudo realizar.



Conclusión

Los resultados de este proyecto fueron los esperados aunque no se realizó al 100% de su desarrollo ya que hubo diferentes problemas para la creación de librerías para las otras figuras que se tenían que mostrar, se pensaba que el escenario fuera más completo ya que al usar las demás figurar faltantes se podrían crear mejores resultados, esperando que con el paso del tiempo se pueda mejorar la técnica de la graficacion se logre un escenario más creativo y detallista posible, aunque para poder lograrlo se necesita mucho tiempo y tener mayor conocimiento para graficar un escenario

25 de Marzo del 2015

Benemérita Universidad Autónoma de Puebla

Reporte del 2° Proyecto

Graficación

Miguel Ángel Hilario Xelano

Introducción:

La meta de este proyecto es que como estudiante de Graficación ponga en práctica lo estudiado en estos meses de clase, en este caso el trabajo consiste en realizar un escenario con figuras geométricas, animaciones y texturizacíon en 2D.

Desarrollo:

Para lograr nuestro cometido se hizo uso de las librerías de Opengl:

- **FreeImage:** Nos permite utilizar texturas en el proyecto.
- **Gl/glut:** Contiene las instrucciones para inicializar el ambiente gráfico.

También para lograr nuestro cometido se hizo uso de las primitivas de figuras:

- **GL_QUADS:** Función que nos permite definir un cuadrado.
- **GL_TRIANGLES:** Función que nos permite definir un triángulo.
- **glutSolidSphere:** Función que me permite dibujar un círculo.

Entre las funciones de animación usamos:

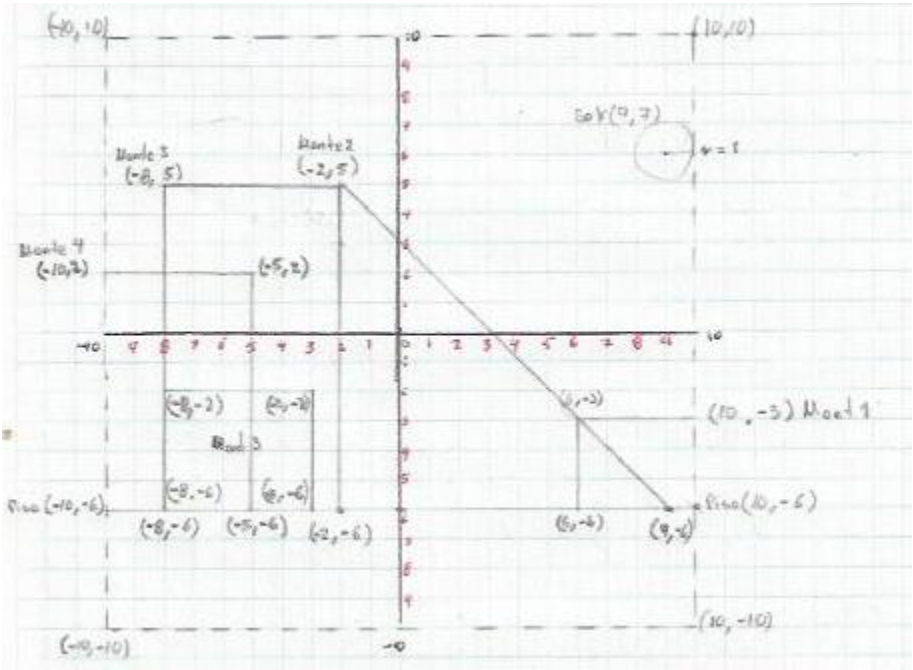
- **glTranslated:** Permite trasladar una figura a otro punto.
- **glRotatef:** Nos permite rotar la figura en un ángulo.

Ahora el escenario que usamos como referencia fue:

Claro está que no todas las figuras estarán en el dibujo ya que solo es como referencia, ya que se pretende hacer algunas modificaciones del mismo.

Antes de comenzar todo se necesita hacer un trazado especificando donde iría cada figura en el plano “x, y” y las partes que conformarían todo el escenario, al igual que el tamaño de este.

Trazado de estructura:



La estructura anterior nos sirvió como base para dibujar el escenario base.

Para poder dibujar el escenario lo primero es preparar el ambiente para poder dibujar, lo cual es mediante:

Main():

```
int main(int argc, char **argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
   glutInitWindowPosition(200, 150);
   glutInitWindowSize(700, 500);
   glutCreateWindow("Triangulo a color");
   //if(!LoadTextures()) return 0;
   init();
   glutDisplayFunc(display);

         glutIdleFunc(idle);     //no hacer nada
   glutMainLoop();
   return 0;
}
```

En esta parte se declara la posición de la ventana, el tamaño, el nombre, que se repita, entre otras cosas.

Lo segundo es inicializar el ambiente:

Función Init():

```
void init()
{
int r;
glClearColor(1.0, 1.0, 1.0, 1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(-10.0,10.0, -10.0,12.0);
cargarTexturas();
}
```

Esta función permite iniciar todo el ambiente en el cual se estará desarrollando el escenario, con estos dos pasos se ha iniciado el ambiento y ahora podemos dibujar todo lo que querremos.

Ahora para poder dibujar todo lo que contiene el escenario necesitamos el uso de algunas funciones las cuales son:

Display():

Esta función se encarga de llamar todas las demás que serán dibujadas, al igual que la ocupe para definir el tamaño del fondo, el cual no es más que un cuadrado de 10*12 en todos los ejes (x, y, -x, -y).

```
void display()
{
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glClear(GL_COLOR_BUFFER_BIT);
    //Codigo del Fondo:
    glPushMatrix();
      glEnable(GL_TEXTURE_2D);
      glBindTexture(GL_TEXTURE_2D,textures[0]);
      glColor3d(1,1,1);
      glBegin(GL_QUADS);
        glTexCoord2f(1.0f,1.0f);
        glVertex2d(10,12);
        glTexCoord2f(1.0f,0.0f);
        glVertex2d(10,-3);
        glTexCoord2f(0.0f,0.0f);
        glVertex2d(-10,-3);
        glTexCoord2f(0.0f,1.0f);
        glVertex2d(-10,12);
      glEnd();
      glDisable(GL_TEXTURE_2D);
    glPopMatrix();
  sol();
  Escenario();
  nubes();
  dragones();
  mario();

  glFlush();
}
```

El fondo de esta parte (Cuadrado) fue texturizado con la siguiente imagen:

Sol():

Esta función me permite dibujar un círculo que aparenta ser el sol, y rotarla para lograr un efecto de tiempo, lo cual se logra con la primitiva **glRotatef()** la cual tiene como parámetros **ángulo, x, y, z** y para lograr el efecto de movimiento se le asigna el ángulo como una variable de incremento.

```
void sol()
{
   //Codigo del Sol:
     glPushMatrix();
     glColor3d(1,1,0);
       glRotatef(angsol,0.0f,0.0f,1.0f);
       glTranslated(solx,8,0);
       glutSolidSphere(1,slices,stacks);
       angsol+=0.001f;
    glPopMatrix();
}
```

Escenario():

Esta función tiene como tarea dibujar todo el escenario, para esto se usó como modelo la traza antes mencionada.

Las figuras que conforman esta función son:

- 5 Cuadrados
- 1 triangulo

Las cuales son definidas de diferente tamaño y como son estáticas no se necesita hacer alguna operación en esta parte.

Trozo del código:

```
void Escenario()
{
  //Codigo del Montaña 1:
     glEnable(GL_TEXTURE_2D);
     glBindTexture(GL_TEXTURE_2D,textures[2]);
     glColor3d(1,1,1);
     glBegin(GL_TRIANGLES);
        glTexCoord2f(1.0f,1.0f);
        glVertex2d(-2,5);
        glTexCoord2f(1.0f,0.0f);
        glVertex2d(-2,-6);
        glTexCoord2f(0.0f,1.0f);
        glVertex2d(9,-6);
     glEnd();
     glDisable(GL_TEXTURE_2D);
```

.

.

Ahora las figuras fueron texturizadas con la siguiente imagen:

Nubes():

La función Nubes como su nombre lo dice crea las nubes, para poder crearlas se necesitó de cuadrados con diferentes posiciones, una vez que se determinó las posiciones se les aplico un **glTranslated()** que tiene como parámetros **"x, y, z"**, los cuales son los puntos de avance, para dar el efecto de recorrido, para esto se usó una variable incrementada en x.

Trozo del Código:

```
void nubes()
{
   //Codigo de la Nube 4:
     glPushMatrix();
       glEnable(GL_TEXTURE_2D);
       glBindTexture(GL_TEXTURE_2D,textures[5]);
       glTranslated(n4,-1,0);
       glColor3d(1,1,1);
       glBegin(GL_QUADS);
          glTexCoord2f(1.0f,1.0f);
          glVertex2f(5,8);
          glTexCoord2f(1.0f,0.0f);
          glVertex2f(5,7);
          glTexCoord2f(0.0f,0.0f);
          glVertex2f(4,7);
          glTexCoord2f(0.0f,1.0f);
          glVertex2f(4,8);
       glEnd();
       glBegin(GL_QUADS);
          glTexCoord2f(1.0f,1.0f);
          glVertex2f(4,8);
          glTexCoord2f(1.0f,0.0f);
          glVertex2f(4,7);
          glTexCoord2f(0.0f,0.0f);
          glVertex2f(3,7);
          glTexCoord2f(0.0f,1.0f);
          glVertex2f(3,8);
       glEnd();
       glDisable(GL_TEXTURE_2D);
       n4-=0.001f;
   //Condición del limite
       if(n4<-16.0)
       {
            n4=17.0f;
       }
     glPopMatrix();
```

Ahora bien para que estas nubes no se fueran al infinito se les puso una condicional en la cual se evaluaba que x no avanzara más allá del límite establecido y si era así que la retornara a su valor inicial.

Ahora las figuras fueron texturizadas con la siguiente imagen:

Dragones():

Como su nombre lo dice esta función me permite crear dragones, los cuales son cuadrados con una textura y movimiento de izquierda a derecha, para poder hacer visible ese efecto de movimiento se usó la misa técnica que el de las nubes solo que esta vez con 2 condicionales "**xmin<Movimiento<xmax**" y así poder tener control sobre el movimiento.

Trozo de código:

```
// Codigo Dragon 3
  glPushMatrix();
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D,textures[c3]);
    glColor3d(1,1,1);
    glTranslated(d3,0,0);
    glBegin(GL_QUADS);
      glTexCoord2f(1.0f,1.0f);
      glVertex2f(1,9);
      glTexCoord2f(1.0f,0.0f);
      glVertex2f(1,5);
      glTexCoord2f(0.0f,0.0f);
      glVertex2f(2.5,5);
      glTexCoord2f(0.0f,1.0f);
      glVertex2f(2.5,9);
    glEnd();
    glDisable(GL_TEXTURE_2D);


//Código de Evaluación Limites
 if(d3>=-4.5f)
    {
      aux3=-0.001f;
      c3=6;
    }
    if(d3<=-9.0f)
    {
      aux3=0.001f;
      c3=7; //Variable de cambio de textura
    }
    d3+=aux3;
  glPopMatrix();
```

En este caso como se cambia las imágenes cuando llegan a su límite se usó una variable global que cambiara su valor según el lado al que avanzara.

Texturas que se usaron:

Mario():

Esta función es la encargada de dibujar el personaje de Mario Bros el cual tiene la misma lógica que la función de los dragones, solo que este función contien mas condiciones de evaluación de limites ya que el personaje se mueve en el eje "x" y luego de un tiempo en el eje "x,y" y de nuevo en el eje "x".

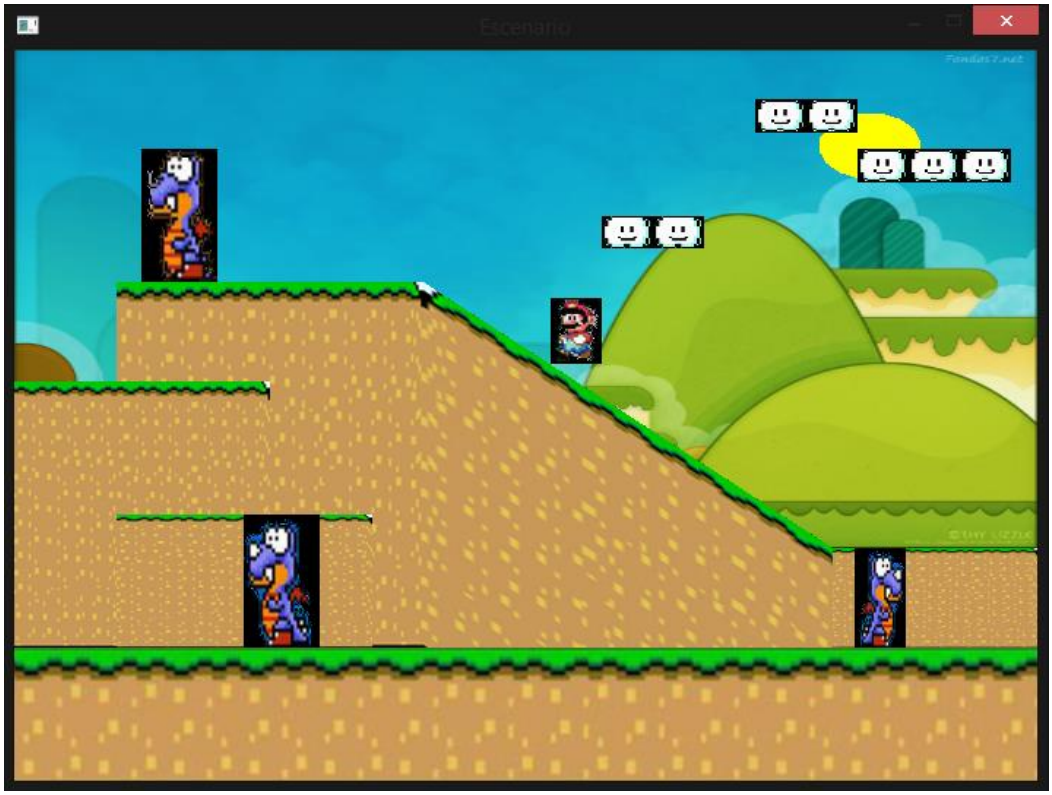
Código:

```
void mario()                                    {
```

```
  glPushMatrix();
    glEnable(GL_TEXTURE_2D);
    glAlphaFunc(GL_GREATER, 0.0f);
    glBindTexture(GL_TEXTURE_2D,textures[m1]);
    glColor3d(1,1,1);
    glTranslated(m,my,0);
    glBegin(GL_QUADS);
      glTexCoord2f(1.0f,1.0f);
      glVertex2f(1,-3);
      glTexCoord2f(1.0f,0.0f);
      glVertex2f(1,-5);
      glTexCoord2f(0.0f,0.0f);
      glVertex2f(0,-5);
      glTexCoord2f(0.0f,1.0f);
      glVertex2f(0,-3);
    glEnd();
    glDisable(GL_TEXTURE_2D);




  //Condiciones de limites y cambio de imagenes
    if(m<=9.1f)
    {
      maux=-0.001f;
    }
    if(m<=6.0f)
    {
      mauxy=0.001f;
    }
    if(my>10.0f)
    {
      mauxy=0.0f;
    }
    if(m<-3.5)
    {
      maux=0.0f;
      mauxy=-0.01f;
      m1=8;
    }
    if(my<-7.0f)
    {
      m=9.0f;
      my=2.0f;
      maux=0.0f;
      mauxy=0.0f;
      m1=4;
    }
    m+=maux;
    my+=mauxy;
}
```

Texturas que se usaron:

Proyecto Final:



Conclusión:

Este Proyecto- Examen me permitió como estudiante reafirmar mis conocimientos obtenidos en clase y ponerlos en práctica para poder resolver este proyecto, este trabajo me causo varias dificultades a la hora de texturizar ya que no sabía cómo hacerlo al igual que cada imagen tenía que tener parámetros específicos, pero al final pude resolver, uno

de los problemas que no pude resolver fue el de la transparencia ya que no me respetaba esas marcar y me las ponía con un fondo negro, por lo que el escenario toma una mala perspectiva por ese detalle.

Por lo demás estoy satisfecho ya que al final aprendí como texturizar y hacer animaciones es un ambiente 2D.

[Seleccionar fecha]

[ESCRIBIR EL NOMBRE DE LA COMPAÑÍA]

EN 1

[Escribir el subtítulo del documento] | user

Introducción

OpenGL (Open Graphics Library) es una especificación estándar que define una **API multilenguaje** y **multiplataforma** para escribir aplicaciones que produzcan gráficos 2D y 3D. La interfaz consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples, tales como puntos, líneas y triángulos.

Fundamentalmente OpenGL es una especificación, es decir, un documento que describe un conjunto de funciones y el comportamiento exacto que deben tener. Partiendo de ella, los fabricantes de hardware crean implementaciones, que son bibliotecas de funciones que se ajustan a los requisitos de la especificación, utilizando aceleración hardware cuando es posible. Dichas implementaciones deben superar unos test de conformidad para que sus fabricantes puedan calificar su implementación como conforme a OpenGL y para poder usar el logotipo oficial de OpenGL.

Básicamente OpenGL consiste en una serie de librerías y rutinas de clases por lo cual, OpenGL no es un paquete de software de renderizado y modelado como Blender o 3D Max, básicamente es una API de bajo nivel que proporciona una interfaz de hardware de gráficos. No es por lo tanto ningún lenguaje de programación, sino tan sólo un conjunto de librerías que son utilizadas a través de lenguajes de programación como VisualC++ para conseguir un interfaz software entre las aplicaciones y el hardware gráfico.

OpenGL tiene dos propósitos esenciales:

1. Ocultar la complejidad de la interfaz con las diferentes tarjetas gráficas, presentando al programador una API única y uniforme.

2. Ocultar las diferentes capacidades de las diversas plataformas hardware, requiriendo que todas las implementaciones soporten la funcionalidad completa de OpenGL (utilizando emulación software si fuese necesario).

Desarrollo y Objetivo del Proyecto

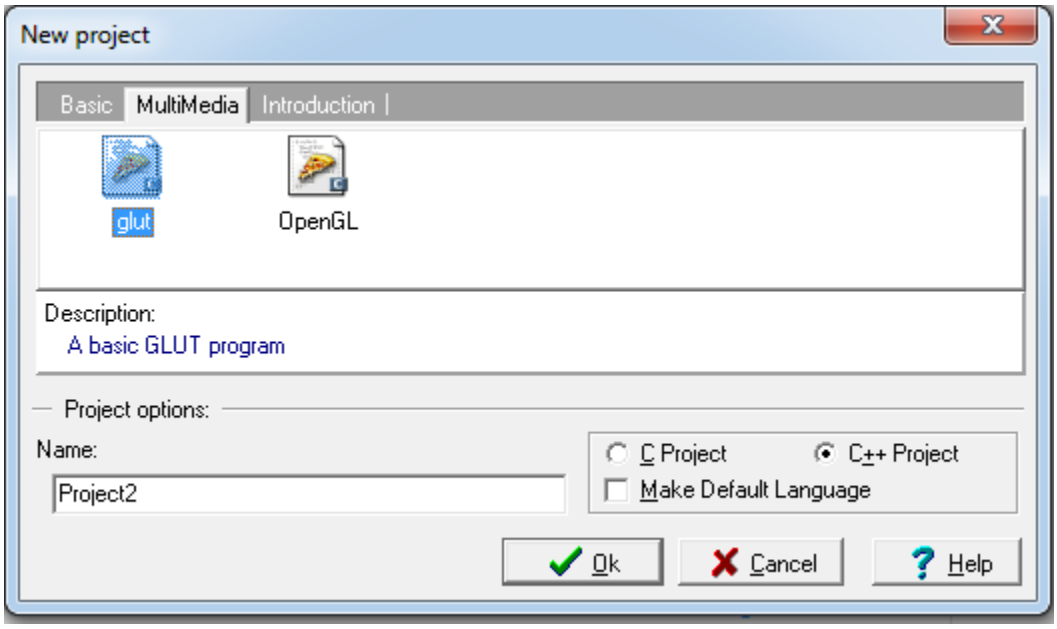
Se trata de un proyecto realizado para la asignatura de graficacion de la Benemérita Universidad Autónoma de Puebla. El lenguaje de programación usado es C y el api usado es OpenGL.

Es un programa en el cual se mostrara todo lo visto en el curso ya que ahora se puede usar todo tipo de librerías, primitivas, etc.

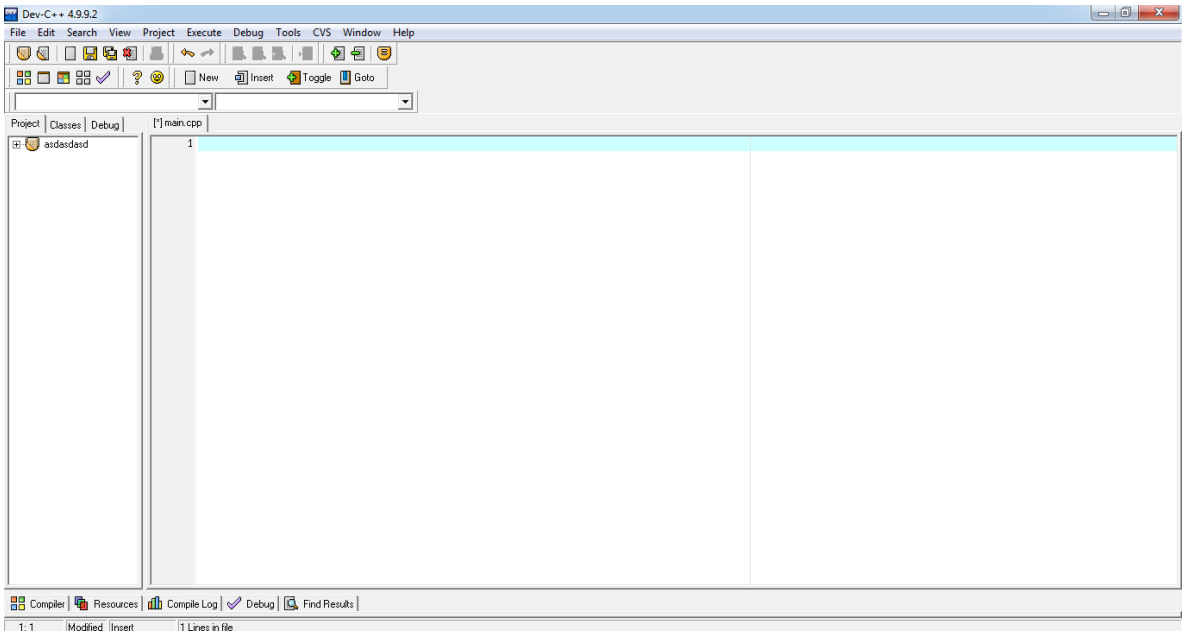
Además mostraremos como agregar texturas mediante imágenes (.BMP) las cuales darán un escenario más realista.

El desarrollo de este proyecto se explica a continuación:

Se creó un proyecto OpenGl (Glut), teniendo en cuenta que se debe tener instalado el "glut" para que se pueda visualizar todo los entornos de dibujos y animaciones



Se guarda con el nombre deseado antes de pasar a "programar" nuestro escenario 2D o 3D en este caso será un escenario lo más cercano a 3D.



Tendremos nuestro lugar de trabajo para empezar a trabajar, lo primo que se hizo fue poner nuestras bibliotecas necesarias

#include <stdio.h>
#include <GL/glut.h>
#include <malloc.h>

Para el funcionamiento correcto de este proyecto usaremos las bibliotecas usadas anteriormente por nuestros reportes pasados, pero ahora también usaremos la biblioteca “malloc” que esta es den gran importancia para asignar el numero especificado de bytes.

En nuestro código la usaremos para mandar a llamar a nuestras imágenes usadas en el escenario.

image = (GLubyte *) malloc (imageSize);

Empezamos con unas variables de tipo flotante para poder darle el movimiento a nuestro cuerpo 3D que diseñamos con vértices junto con el raton de nuestra computadora

float alpha, beta;
int x0, y0;

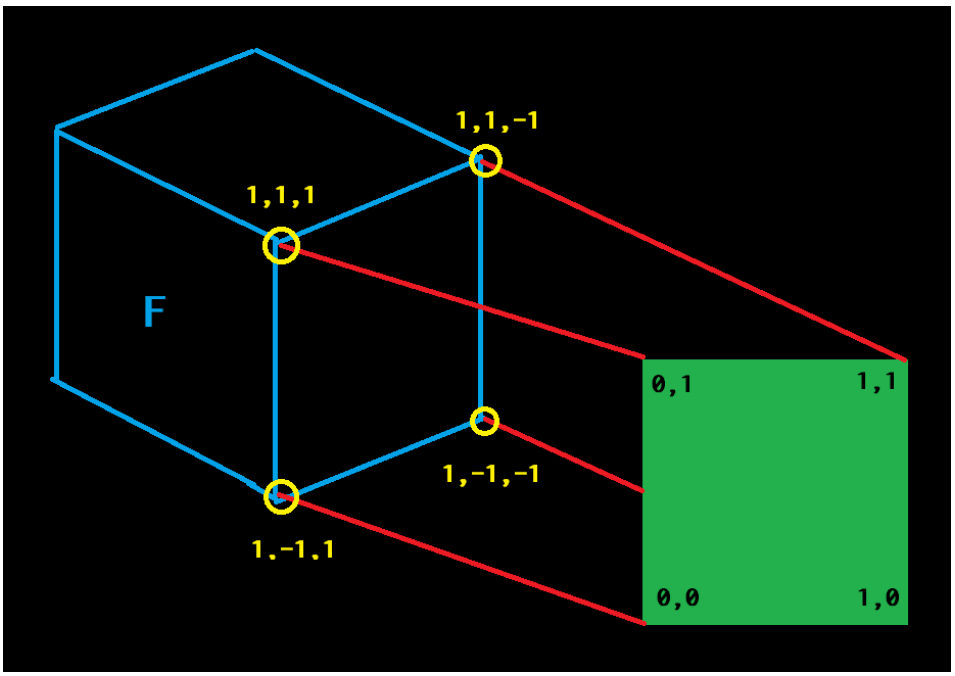
GLuint texture_id [];

Que es donde declaramos nuestra textura del escenario dando un valor arriba de 10 ya que si es un número menor a 10 tendremos errores para la carga de textura

GLfloat texcoords [] = {0,0, 1,0, 1,1, 0,1};

Primero debemos de colocar las coordenadas deseadas que pueden ser definidas como una matriz.
Para cada vértice de una cara del polígono en este caso de cubos debemos de declarar las coordenadas que corresponden de esta forma ir formando poco a poco el polígono deseado.

Para el cubo se debe de proporcionar las coordenadas de los 6 lados de este



En donde en cada matriz iremos guardando las coordenadas de nuestra figura hasta ir logrando la forma deseada (Castillo)

NOTA: esto es una de las formas en las que se pueden ir dibujando cada parte de un objeto, otra manera más sencilla es simplemente usar dos puntos del cuadrado y de

igual forma se ira formando nuestro cubo ya que le agregaremos una tercera vista que es 'z' que en el mundo de 3D se le conoce como la profundidad del objeto dando esa vista en 3D.

```
GLfloat vertices1[]={-.40f,-.5f, .4f,
                                        -.16f,-.5f, .4f,
                                        -.16f,-.1f, .4f,
                                        -.40f,-.1f, .4f};
        GLfloat vertices2[]={.16f,-.5f, .4f,
                                        .40f,-.5f, .4f,
                                        .40f,-.1f, .4f,
                                        .16f,-.1f, .4f};
        GLfloat vertices3[]={-.40f,-.10f, .20f,
                                        -.16f,-.10f, .20f,
                                        -.16f, .46f, .20f,
                                        -.40f, .46f, .20f};
```

Como podemos observar usamos un arreglo de matrices para ir formando nuestro objeto cada coordenada declarada pinta cada parte del castillo.

Toda esta técnica la usamos para hacer todo nuestro objeto en 3D y después mandar a llamar a esta misma para finalizar poniéndole su respectiva textura.

```
int  LoadBMP(char *filename, int tex_name)
 {
   #define SAIR       {fclose(fp_archivo); return -1;}
   #define CTOI(C)     (*(int*)&C)

   GLubyte     *image;
   GLubyte     Header[0x54];
   GLuint      DataPos, imageSize;
   GLsizei     Width,Height;
```

Es aquí donde declaramos nuestra variable LoadBMP que nos servirá para que cargue nuestras texturas para nuestro escenario

```
FILE * fp_archivo = fopen(filename,"rb");
   if (!fp_archivo)
```

```
    return -1;
  if (fread(Header,1,0x36,fp_archivo)!=0x36)
    SAIR;
  if (Header[0]!='B' || Header[1]!='M')
    SAIR;
  if (CTOI(Header[0x1E])!=0)
    SAIR;
  if (CTOI(Header[0x1C])!=24)
    SAIR;
```

Con esta parte de código abrimos un archivo y se hace la lectura del encabezado del archivo BMP

```
Width   = CTOI(Header[0x12]);
  Height  = CTOI(Header[0x16]);
  ( CTOI(Header[0x0A]) == 0 ) ? ( DataPos=0x36 ) : ( DataPos =
CTOI(Header[0x0A]) );

  imageSize = Width*Height*3;
```

De cierta manera debemos recuperar los atributos de la altura y ancho de la imagen para al momento de compilar nuestro código esta se reajuste a la pantalla.

```
image = (GLubyte *) malloc ( imageSize );
  int retorno;
  retorno = fread(image,1,imageSize,fp_archivo);

  if (retorno !=imageSize)
   {
    free (image);
    SAIR;
```

Este pedazo de código es de gran importancia ya que con él se pudo mandar a llamar a la imagen (texturas), como podemos apreciar usaremos aquí nuestra nueva biblioteca “malloc” que es la que se encarga de asignar el número de bytes asignados para las imágenes.

```
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
```

```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, Width, Height, 0, GL_RGB,
GL_UNSIGNED_BYTE, image);

   fclose (fp_archivo);
   free (image);
   return 1;
}
```

Esta es la parte en donde opengl nos ayuda a empezar a dar texturas a nuestro escenario manipulando la memoria de las texturas.

Se crearon dos funciones sencillas en opengl en donde se pueda usar el mouse para jugar con el objeto 3D y tenga un movimiento hacia todos los ejes del ambiente grafico que se muestra a continuación como se usó.

```
void onMouse(int button, int state, int x, int y)
{
        if ( (button == GLUT_LEFT_BUTTON) & (state == GLUT_DOWN) ) {
        x0 = x; y0 = y;}
}

void onMotion(int x, int y)
{
        alpha = (alpha + (y - y0));
        beta = (beta + (x - x0));
        x0 = x; y0 = y;
        glutPostRedisplay();
```

Nuestra función onMotion es la que nos da el efecto de movimiento de nuestro objeto en donde a nuestras variables le sumamos los ejes X o Y, y volvemos a mandar a imprimir el escenario con los valores nuevos.

En nuestra función myinit hicimos manejo de todas las cargas de textura, además del manejo de luces para que se visualizara mejor el escenario. Además de que el profesor del curso nos pidió que esta función o manejo de luces fuera agregada.

```
        GLfloat ambiente1[] = { 0.5f,0.5f,0.5f};
        GLfloat difusa1[] = { 0.5f,0.5f,0.5f};
        GLfloat especular1[] = { 1.0f,0.0f,0.0f};
```

```
GLfloat posicion1[] = { 0.0f,0.0f,5.0f,0.0};

glLightfv(GL_LIGHT0, GL_AMBIENT, ambiente1);
glLightfv(GL_LIGHT0, GL_DIFFUSE, difusa1);
glLightfv(GL_LIGHT0, GL_SPECULAR, especular1);
```

En donde estas líneas de código especifican una luz de direccionamiento para que todo el escenario este iluminado.

```
GLfloat ambiente2[] = { 1.0f,1.0f,0.0f};
GLfloat difuso2[] = { 1.0f,0.0f,0.0f};
GLfloat posicion2[] = { 1.0f,0.0f,5.0f,1.0};
GLfloat direccion2[] = {0.0f,0.0f,-5.0f};

glLightfv(GL_LIGHT1, GL_AMBIENT, ambiente2);
glLightfv(GL_LIGHT1, GL_DIFFUSE, difuso2);
glLightfv(GL_LIGHT1, GL_POSITION, posicion2);
glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, direccion2);
glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, 15.0f);
```

Tenemos nuestras lineas de codigo donde mandamos a llamar a cada una de las imagines (.bmp) que se usaron en el escenario poniéndolas en una misma carpeta para que no existan errores de compilación

```
glBindTexture(GL_TEXTURE_2D,texture_id[0]);
LoadBMP("pared.bmp",0);
glBindTexture(GL_TEXTURE_2D,texture_id[1]);
LoadBMP("paredgrande.bmp",1);
  glBindTexture(GL_TEXTURE_2D,texture_id[2]);
LoadBMP("puerta.bmp",2);
glBindTexture(GL_TEXTURE_2D,texture_id[3]);
LoadBMP("paredatras.bmp",3);
glBindTexture(GL_TEXTURE_2D,texture_id[4]);
LoadBMP("techo2.bmp",3);
glBindTexture(GL_TEXTURE_2D,texture_id[5]);
LoadBMP("borde.bmp",3);
glBindTexture(GL_TEXTURE_2D,texture_id[6]);
LoadBMP("paisaje4.bmp",3);
```

OpenGl nos brinda la oportunidad de cargar las imágenes que queramos usar mediante una variable

glBindTexture (GL_TEXTURE_2D,texture_id[6]);
LoadBMP("borde.bmp",3);

En la función de castillo la usamos para ir declarando cada uno de los cuadros con los que se fue armando poco a poco (frente, laterales, techo, etc.) es aquí donde los vértices los fusionamos con los cuadros usando una de las funciones

glDrawArrays(GL_QUADS,0,4);

Con glDrawArrays se puede especificar varias primitivas geométricas para renderizar. En lugar de llamar a funciones OpenGL separadas para pasar cada vértice individual, normal, o el color , puede especificar matrices independientes de vértices, normales y colores para definir una secuencia de primitivas (todas del mismo tipo) con una sola llamada a glDrawArrays .

En la función fondo simplemente mandamos a llamar a nuestra imagen que usamos para el fondo con el siguiente pedazo de código

```
glBindTexture(GL_TEXTURE_2D,texture_id[6]);
        glVertexPointer(3,GL_FLOAT,0,vertices141);
        glDrawArrays(GL_QUADS,0,4);
```

Por ultimo tenemos nuestras funciones que sirven para poder mover al objeto, mandar a llamarlo y que este se redibuje con nuevos parámetros (rotación, translación, escalamiento) dentro de un push y pop matrix para encapsular dichas operaciones. Nuestra función main principal que es aquí donde mandamos a llamar a todas nuestras funciones usadas en el código.

CODIGO FUENTE

```
#include <GL/gl.h>
#include <GL/glut.h>
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

float alpha, beta;
int x0, y0;
GLuint texture_id[15];

        GLfloat texcoords[] ={0,0, 1,0, 1,1, 0,1};
////////////frente/////////////
```

```
GLfloat vertices1[]={-.40f,-.5f, .4f,

 -.16f,-.5f, .4f,

 -.16f,-.1f, .4f,

 -.40f,-.1f, .4f};
GLfloat vertices2[]={.16f,-.5f, .4f,

 .40f,-.5f, .4f,

 .40f,-.1f, .4f,

 .16f,-.1f, .4f};
GLfloat vertices3[]={-.40f,-.10f, .20f,

 -.16f,-.10f, .20f,

 -.16f, .46f, .20f,

 -.40f, .46f, .20f};
GLfloat vertices4[]={.16f,-.10f, .20f,

 .40f,-.10f, .20f,

 .40f, .46f, .20f,

 .16f, .46f, .20f};
GLfloat vertices5[]={-.40f,-.50f, -.20f,

 -.35f,-.50f, -.20f,

 -.35f, .22f, -.20f,

 -.40f, .22f, -.20f};
GLfloat vertices6[]={.35f,-.50f, -.20f,

 .40f,-.50f, -.20f,

 .40f, .22f, -.20f,

 .35f, .22f, -.20f};
GLfloat vertices7[]={-.16f,-.50f, .2f,

  .16f,-.50f, .2f,

  .16f, .22f, .2f,

 -.16f, .22f, .2f};
GLfloat vertices8[]={-.40f,.22f, -.2f,

 -.16f,.22f, -.2f,

 -.16f,.46f, -.2f,

 -.40f,.46f, -.2f};
GLfloat vertices9[]={.16f,.22f, -.2f,

 .40f,.22f, -.2f,

 .40f,.46f, -.2f,

  .16f,.46f, -.2f};
//////////lado izquierdo/////////
  GLfloat vertices10[]={.4f,-.5f, .4f,

  .4f,-.5f, .2f,

  .4f,-.1f, .2f,

  .4f,-.1f, .4f};
  GLfloat vertices11[]={-.16f,-.5f, .4f,

  -.16f,-.5f, .2f,

  -.16f,-.1f, .2f,

  -.16f,-.1f, .4f};
  GLfloat vertices12[]={.4f,-.50f, .2f,

  .4f,-.50f, .0f,

  .4f, .46f, .0f,

  .4f, .46f, .2f};
  GLfloat vertices13[]={.4f,-.50f, -.2f,

  .4f,-.50f, -.4f,

  .4f, .46f, -.4f,

  .4f, .46f, -.2f};
  GLfloat vertices14[]={.35f,-.50f,  .0f,

  .35f,-.50f, -.2f,

  .35f, .22f, -.2f,

  .35f, .22f,  .0f};
 GLfloat vertices15[]={-.16f,.22f, .2f,

  -.16f,.22f, .0f,

  -.16f,.46f, .0f,

  -.16f,.46f, .2f};
  GLfloat vertices16[]={-.16f,.22f, -.2f,

  -.16f,.22f, -.4f,

  -.16f,.46f, -.4f,

  -.16f,.46f, -.2f};
////////////////Atras//////////
  GLfloat vertices17[]={ .4f,-.50f, -.4f,

  -.4f,-.50f, -.4f,

  -.4f, .22f, -.4f,

   .4f, .22f, -.4f};
  GLfloat vertices18[]={.40f,-.50f, .0f,
```

```
                .35f,-.50f, .0f,

                .35f, .22f, .0f,

                .40f, .22f, .0f};
            GLfloat vertices19[]={-.35f,-.50f, .0f,

                -.40f,-.50f, .0f,

                -.40f, .22f, .0f,

                -.35f, .22f, .0f};
            GLfloat vertices20[]={.40f,.22f, -.4f,

                .16f,.22f, -.4f,

                .16f,.46f, -.4f,

                .40f,.46f, -.4f};
            GLfloat vertices21[]={-.16f,.22f, -.4f,

                -.40f,.22f, -.4f,

                -.40f,.46f, -.4f,

                -.16f,.46f, -.4f};
            GLfloat vertices22[]={.40f,.22f, -.0f,

                .16f,.22f, -.0f,

                .16f,.46f, -.0f,

                .40f,.46f, -.0f};
            GLfloat vertices23[]={-.16f,.22f, -.0f,

                -.40f,.22f, -.0f,

                -.40f,.46f, -.0f,

                -.16f,.46f, -.0f};
//////////lado izquierdo/////////
            GLfloat vertices24[]={.16f,-.5f, .2f,

                .16f,-.5f, .4f,

                .16f,-.1f, .4f,

                .16f,-.1f, .2f};
            GLfloat vertices25[]={-.4f,-.5f, .2f,

                -.4f,-.5f, .4f,

                -.4f,-.1f, .4f,

                -.4f,-.1f, .2f};
            GLfloat vertices26[]={-.4f,-.50f, .0f,

                    -.4f,-.50f, .2f,

                    -.4f, .46f, .2f,

                    -.4f, .46f, .0f};
            GLfloat vertices27[]={-.35f,-.50f, -.2f,

                -.35f,-.50f,  .0f,

                -.35f, .22f,  .0f,

                -.35f, .22f, -.2f};
            GLfloat vertices28[]={-.4f,-.50f, -.4f,

                -.4f,-.50f, -.2f,

                -.4f, .46f, -.2f,

                -.4f, .46f, -.4f};
            GLfloat vertices29[]={.16f,.22f, -.4f,

                .16f,.22f, -.2f,

                .16f,.46f, -.2f,

                .16f,.46f, -.4f};
            GLfloat vertices30[]={.16f,.22f, .0f,

                .16f,.22f, .2f,

                .16f,.46f, .2f,

                .16f,.46f, .0f};
//////////////////Techo////////////
            ///torres chicas/////
            GLfloat vertices31[]={-.40f,-.1f,.4f,

                -.16f,-.1f,.4f,

                -.16f,-.1f,.2f,

                -.40f,-.1f,.2f};
            GLfloat vertices32[]={.16f,-.1f,.4f,

                .40f,-.1f,.4f,

                .40f,-.1f,.2f,

                .16f,-.1f,.2f};
            ///torres grandes/////
            GLfloat vertices33[]={-.40f,.46f,.2f,

                -.16f,.46f,.2f,

                -.16f,.46f,.0f,

                -.40f,.46f,.0f};
            GLfloat vertices34[]={.16f,.46f,.2f,

                .40f,.46f,.2f,

                .40f,.46f,.0f,

                .16f,.46f,.0f};
            GLfloat vertices35[]={-.40f,.46f,-.2f,
```

```
-.16f,.46f,-.2f,

-.16f,.46f,-.4f,

-.40f,.46f,-.4f};
GLfloat vertices36[]={.16f,.46f,-.2f,

.40f,.46f,-.2f,

.40f,.46f,-.4f,

.16f,.46f,-.4f};
/////cuerpo/////////
GLfloat vertices37[]={-.16f,.225f, .2f,

.16f,.225f, .2f,

.16f,.225f,-.4f,

-.16f,.225f,-.4f};
GLfloat vertices38[]={-.35f,.22f, .0f,

.35f,.22f, .0f,

.35f,.22f,-.2f,

-.35f,.22f,-.2f};
//////////////////bordes/////////////////
////////torres chicas////////
GLfloat vertices39[]={-.40f,-.10f, .4f,

-.36f,-.10f, .4f,

-.36f,-.06f, .4f,

-.40f,-.06f, .4f};
GLfloat vertices40[]={-.35f,-.10f, .4f,

-.31f,-.10f, .4f,

-.31f,-.06f, .4f,

-.35f,-.06f, .4f};
GLfloat vertices41[]={-.30f,-.10f, .4f,

-.26f,-.10f, .4f,

-.26f,-.06f, .4f,

-.30f,-.06f, .4f};
GLfloat vertices42[]={-.25f,-.10f, .4f,

-.21f,-.10f, .4f,

-.21f,-.06f, .4f,

-.25f,-.06f, .4f};
GLfloat vertices43[]={-.20f,-.10f, .4f,

-.16f,-.10f, .4f,

-.16f,-.06f, .4f,

-.20f,-.06f, .4f};
GLfloat vertices44[]={.16f,-.10f, .4f,

.20f,-.10f, .4f,

.20f,-.06f, .4f,

.16f,-.06f, .4f};
GLfloat vertices45[]={.21f,-.10f, .4f,

.25f,-.10f, .4f,

.25f,-.06f, .4f,

.21f,-.06f, .4f};
GLfloat vertices46[]={.26f,-.10f, .4f,

.30f,-.10f, .4f,

.30f,-.06f, .4f,

.26f,-.06f, .4f};
GLfloat vertices47[]={.31f,-.10f, .4f,

.35f,-.10f, .4f,

.35f,-.06f, .4f,

.31f,-.06f, .4f};
GLfloat vertices48[]={.36f,-.10f, .4f,

.40f,-.10f, .4f,

.40f,-.06f, .4f,

.36f,-.06f, .4f};
////////torres grandes////////
GLfloat vertices49[]={-.40f,.46f, .2f,

-.36f,.46f, .2f,

-.36f,.50f, .2f,

-.40f,.50f, .2f};
GLfloat vertices50[]={-.35f,.46f, .2f,

-.31f,.46f, .2f,

-.31f,.50f, .2f,

-.35f,.50f, .2f};
GLfloat vertices51[]={-.30f,.46f, .2f,

-.26f,.46f, .2f,

-.26f,.50f, .2f,

-.30f,.50f, .2f};
GLfloat vertices52[]={-.25f,.46f, .2f,
```

```
 -.21f,.46f, .2f,

 -.21f,.50f, .2f,

 -.25f,.50f, .2f};
GLfloat vertices53[]={-.20f,.46f, .2f,

 -.16f,.46f, .2f,

 -.16f,.50f, .2f,

 -.20f,.50f, .2f};
GLfloat vertices54[]={.16f,.46f, .2f,

 .20f,.46f, .2f,

 .20f,.50f, .2f,

 .16f,.50f, .2f};
GLfloat vertices55[]={.21f,.46f, .2f,

 .25f,.46f, .2f,

 .25f,.50f, .2f,

 .21f,.50f, .2f};
GLfloat vertices56[]={.26f,.46f, .2f,

 .30f,.46f, .2f,

 .30f,.50f, .2f,

 .26f,.50f, .2f};
GLfloat vertices57[]={.31f,.46f, .2f,

 .35f,.46f, .2f,

 .35f,.50f, .2f,

 .31f,.50f, .2f};
GLfloat vertices58[]={.36f,.46f, .2f,

 .40f,.46f, .2f,

 .40f,.50f, .2f,

 .36f,.50f, .2f};
GLfloat vertices59[]={-.40f,.46f, .0f,

 -.36f,.46f, .0f,

 -.36f,.50f, .0f,

 -.40f,.50f, .0f};
GLfloat vertices60[]={-.35f,.46f, .0f,

 -.31f,.46f, .0f,

 -.31f,.50f, .0f,

 -.35f,.50f, .0f};

GLfloat vertices61[]={-.30f,.46f, .0f,

 -.26f,.46f, .0f,

 -.26f,.50f, .0f,

 -.30f,.50f, .0f};
GLfloat vertices62[]={-.25f,.46f, .0f,

 -.21f,.46f, .0f,

 -.21f,.50f, .0f,

 -.25f,.50f, .0f};
GLfloat vertices63[]={-.20f,.46f, .0f,

 -.16f,.46f, .0f,

 -.16f,.50f, .0f,

 -.20f,.50f, .0f};
GLfloat vertices64[]={.16f,.46f, .0f,

 .20f,.46f, .0f,

 .20f,.50f, .0f,

 .16f,.50f, .0f};
GLfloat vertices65[]={.21f,.46f, .0f,

 .25f,.46f, .0f,

 .25f,.50f, .0f,

 .21f,.50f, .0f};
GLfloat vertices66[]={.26f,.46f, .0f,

 .30f,.46f, .0f,

 .30f,.50f, .0f,

 .26f,.50f, .0f};
GLfloat vertices67[]={.31f,.46f, .0f,

 .35f,.46f, .0f,

 .35f,.50f, .0f,

 .31f,.50f, .0f};
GLfloat vertices68[]={.36f,.46f, .0f,

 .40f,.46f, .0f,

 .40f,.50f, .0f,

 .36f,.50f, .0f};
GLfloat vertices69[]={-.40f,.46f, -.2f,

 -.36f,.46f, -.2f,

 -.36f,.50f, -.2f,
```

```
 -.40f,.50f, -.2f};
GLfloat vertices70[]={-.35f,.46f, -.2f,

 -.31f,.46f, -.2f,

 -.31f,.50f, -.2f,

 -.35f,.50f, -.2f};
GLfloat vertices71[]={-.30f,.46f, -.2f,

 -.26f,.46f, -.2f,

 -.26f,.50f, -.2f,

 -.30f,.50f, -.2f};
GLfloat vertices72[]={-.25f,.46f, -.2f,

 -.21f,.46f, -.2f,

 -.21f,.50f, -.2f,

 -.25f,.50f, -.2f};
GLfloat vertices73[]={-.20f,.46f, -.2f,

 -.16f,.46f, -.2f,

 -.16f,.50f, -.2f,

 -.20f,.50f, -.2f};
GLfloat vertices74[]={.16f,.46f, -.2f,

 .20f,.46f, -.2f,

 .20f,.50f, -.2f,

 .16f,.50f, -.2f};
GLfloat vertices75[]={.21f,.46f, -.2f,

 .25f,.46f, -.2f,

 .25f,.50f, -.2f,

 .21f,.50f, -.2f};
GLfloat vertices76[]={.26f,.46f, -.2f,

 .30f,.46f, -.2f,

 .30f,.50f, -.2f,

 .26f,.50f, -.2f};
GLfloat vertices77[]={.31f,.46f, -.2f,

 .35f,.46f, -.2f,

 .35f,.50f, -.2f,

 .31f,.50f, -.2f};
GLfloat vertices78[]={.36f,.46f, -.2f,

 .40f,.46f, -.2f,

 .40f,.50f, -.2f,

 .36f,.50f, -.2f};
GLfloat vertices79[]={-.40f,.46f, -.4f,

 -.36f,.46f, -.4f,

 -.36f,.50f, -.4f,

 -.40f,.50f, -.4f};
GLfloat vertices80[]={-.35f,.46f, -.4f,

 -.31f,.46f, -.4f,

 -.31f,.50f, -.4f,

 -.35f,.50f, -.4f};
GLfloat vertices81[]={-.30f,.46f, -.4f,

 -.26f,.46f, -.4f,

 -.26f,.50f, -.4f,

 -.30f,.50f, -.4f};
GLfloat vertices82[]={-.25f,.46f, -.4f,

 -.21f,.46f, -.4f,

 -.21f,.50f, -.4f,

 -.25f,.50f, -.4f};
GLfloat vertices83[]={-.20f,.46f, -.4f,

 -.16f,.46f, -.4f,

 -.16f,.50f, -.4f,

 -.20f,.50f, -.4f};
GLfloat vertices84[]={.16f,.46f, -.4f,

 .20f,.46f, -.4f,

 .20f,.50f, -.4f,

 .16f,.50f, -.4f};
GLfloat vertices85[]={.21f,.46f, -.4f,

 .25f,.46f, -.4f,

 .25f,.50f, -.4f,

 .21f,.50f, -.4f};
GLfloat vertices86[]={.26f,.46f, -.4f,

 .30f,.46f, -.4f,

 .30f,.50f, -.4f,

 .26f,.50f, -.4f};
GLfloat vertices87[]={.31f,.46f, -.4f,
```

```
 .35f,.46f, -.4f,

 .35f,.50f, -.4f,

 .31f,.50f, -.4f};
GLfloat vertices88[]={.36f,.46f, -.4f,

 .40f,.46f, -.4f,

 .40f,.50f, -.4f,

 .36f,.50f, -.4f};
GLfloat vertices89[]={-.16f,.22f,.2f,

 -.11f,.22f,.2f,

 -.11f,.26f,.2f,

 -.16f,.26f,.2f};
GLfloat vertices90[]={-.09f,.22f,.2f,

 -.04f,.22f,.2f,

 -.04f,.26f,.2f,

 -.09f,.26f,.2f};
GLfloat vertices91[]={-.02f,.22f,.2f,

  .03f,.22f,.2f,

  .03f,.26f,.2f,

 -.02f,.26f,.2f};
GLfloat vertices92[]={.05f,.22f,.2f,

 .10f,.22f,.2f,

 .10f,.26f,.2f,

 .05f,.26f,.2f};
GLfloat vertices93[]={.12f,.22f,.2f,

 .17f,.22f,.2f,

 .17f,.26f,.2f,

 .12f,.26f,.2f};
GLfloat vertices94[]={-.16f,.22f,-.4f,

 -.11f,.22f,-.4f,

 -.11f,.26f,-.4f,

 -.16f,.26f,-.4f};
GLfloat vertices95[]={-.09f,.22f,-.4f,

 -.04f,.22f,-.4f,

 -.04f,.26f,-.4f,

 -.09f,.26f,-.4f};
GLfloat vertices96[]={-.02f,.22f,-.4f,

  .03f,.22f,-.4f,

  .03f,.26f,-.4f,

 -.02f,.26f,-.4f};
GLfloat vertices97[]={.05f,.22f,-.4f,

 .10f,.22f,-.4f,

 .10f,.26f,-.4f,

 .05f,.26f,-.4f};
GLfloat vertices98[]={.12f,.22f,-.4f,

 .17f,.22f,-.4f,

 .17f,.26f,-.4f,

 .12f,.26f,-.4f};
//////////lado//////////
GLfloat vertices99[]={.4f,-.10f, .40f,

 .4f,-.10f, .35f,

 .4f,-.06f, .35f,

 .4f,-.06f, .40f};
GLfloat vertices100[]={.4f,-.10f,.33f,

 .4f,-.10f,.28f,

 .4f,-.06f,.28f,

 .4f,-.06f,.33f};
GLfloat vertices101[]={.4f,-.10f,.26f,

 .4f,-.10f,.21f,

 .4f,-.06f,.21f,

 .4f,-.06f,.26f};
GLfloat vertices102[]={.16f,-.10f,.40f,

 .16f,-.10f, .35f,

 .16f,-.06f, .35f,

 .16f,-.06f, .40f};
GLfloat vertices103[]={.16f,-.10f,.33f,

 .16f,-.10f,.28f,

 .16f,-.06f,.28f,

 .16f,-.06f,.33f};
GLfloat vertices104[]={.16f,-.10f,.26f,

 .16f,-.10f,.21f,

 .16f,-.06f,.21f,
```

```
  .16f,-.06f,.26f};
GLfloat vertices105[]={-.4f,-.10f, .40f,

  -.4f,-.10f, .35f,

  -.4f,-.06f, .35f,

  -.4f,-.06f, .40f};
GLfloat vertices106[]={-.4f,-.10f,.33f,

  -.4f,-.10f,.28f,

  -.4f,-.06f,.28f,

  -.4f,-.06f,.33f};
GLfloat vertices107[]={-.4f,-.10f,.26f,

  -.4f,-.10f,.21f,

  -.4f,-.06f,.21f,

  -.4f,-.06f,.26f};
GLfloat vertices108[]={-.16f,-.10f,.40f,

  -.16f,-.10f, .35f,

  -.16f,-.06f, .35f,

  -.16f,-.06f, .40f};
GLfloat vertices109[]={-.16f,-.10f,.33f,

  -.16f,-.10f,.28f,

  -.16f,-.06f,.28f,

  -.16f,-.06f,.33f};
GLfloat vertices110[]={-.16f,-.10f,.26f,

  -.16f,-.10f,.21f,

  -.16f,-.06f,.21f,

  -.16f,-.06f,.26f};
GLfloat vertices111[]={.4f,.46f,.20f,

  .4f,.46f,.15f,

  .4f,.50f,.15f,

  .4f,.50f,.20f};
GLfloat vertices112[]={.4f,.46f,.13f,

  .4f,.46f,.08f,

  .4f,.50f,.08f,

  .4f,.50f,.13f};
GLfloat vertices113[]={.4f,.46f,.06f,

  .4f,.46f,.00f,

  .4f,.50f,.00f,

  .4f,.50f,.06f};
GLfloat vertices114[]={.4f,.46f,-.20f,

  .4f,.46f,-.25f,

  .4f,.50f,-.25f,

  .4f,.50f,-.20f};
GLfloat vertices115[]={.4f,.46f,-.27f,

  .4f,.46f,-.32f,

  .4f,.50f,-.32f,

  .4f,.50f,-.27f};
GLfloat vertices116[]={.4f,.46f,-.34f,

  .4f,.46f,-.40f,

  .4f,.50f,-.40f,

  .4f,.50f,-.34f};
GLfloat vertices117[]={.16f,.46f,.20f,

  .16f,.46f,.15f,

  .16f,.50f,.15f,

  .16f,.50f,.20f};
GLfloat vertices118[]={.16f,.46f,.13f,

  .16f,.46f,.08f,

  .16f,.50f,.08f,

  .16f,.50f,.13f};
GLfloat vertices119[]={.16f,.46f,.06f,

  .16f,.46f,.00f,

  .16f,.50f,.00f,

  .16f,.50f,.06f};
GLfloat vertices120[]={.16f,.46f,-.20f,

  .16f,.46f,-.25f,

  .16f,.50f,-.25f,

  .16f,.50f,-.20f};
GLfloat vertices121[]={.16f,.46f,-.27f,

  .16f,.46f,-.32f,

  .16f,.50f,-.32f,

  .16f,.50f,-.27f};
GLfloat vertices122[]={.16f,.46f,-.34f,
```

```
  .16f,.46f,-.40f,

  .16f,.50f,-.40f,

  .16f,.50f,-.34f};
GLfloat vertices123[]={-.4f,.46f,.20f,

  -.4f,.46f,.15f,

  -.4f,.50f,.15f,

  -.4f,.50f,.20f};
GLfloat vertices124[]={-.4f,.46f,.13f,

  -.4f,.46f,.08f,

  -.4f,.50f,.08f,

  -.4f,.50f,.13f};
GLfloat vertices125[]={-.4f,.46f,.06f,

  -.4f,.46f,.00f,

  -.4f,.50f,.00f,

  -.4f,.50f,.06f};
GLfloat vertices126[]={-.4f,.46f,-.20f,

  -.4f,.46f,-.25f,

  -.4f,.50f,-.25f,

  -.4f,.50f,-.20f};
GLfloat vertices127[]={-.4f,.46f,-.27f,

  -.4f,.46f,-.32f,

  -.4f,.50f,-.32f,

  -.4f,.50f,-.27f};
GLfloat vertices128[]={-.4f,.46f,-.34f,

  -.4f,.46f,-.40f,

  -.4f,.50f,-.40f,

  -.4f,.50f,-.34f};
GLfloat vertices129[]={-.16f,.46f,.20f,

  -.16f,.46f,.15f,

  -.16f,.50f,.15f,

  -.16f,.50f,.20f};
GLfloat vertices130[]={-.16f,.46f,.13f,

  -.16f,.46f,.08f,

  -.16f,.50f,.08f,

  -.16f,.50f,.13f};
GLfloat vertices131[]={-.16f,.46f,.06f,

  -.16f,.46f,.00f,

  -.16f,.50f,.00f,

  -.16f,.50f,.06f};
GLfloat vertices132[]={-.16f,.46f,-.20f,

  -.16f,.46f,-.25f,

  -.16f,.50f,-.25f,

  -.16f,.50f,-.20f};
GLfloat vertices133[]={-.16f,.46f,-.27f,

  -.16f,.46f,-.32f,

  -.16f,.50f,-.32f,

  -.16f,.50f,-.27f};
GLfloat vertices134[]={-.16f,.46f,-.34f,

  -.16f,.46f,-.40f,

  -.16f,.50f,-.40f,

  -.16f,.50f,-.34f};
GLfloat vertices135[]={.35f,.22f, .00f,

  .35f,.22f,-.05f,

  .35f,.26f,-.05f,

  .35f,.26f, .00f};
GLfloat vertices136[]={.35f,.22f,-.07f,

  .35f,.22f,-.12f,

  .35f,.26f,-.12f,

  .35f,.26f,-.07f};
GLfloat vertices137[]={.35f,.22f,-.14f,

  .35f,.22f,-.19f,

  .35f,.26f,-.19f,

  .35f,.26f,-.14f};
GLfloat vertices138[]={-.35f,.22f, .00f,

  -.35f,.22f,-.05f,

  -.35f,.26f,-.05f,

  -.35f,.26f, .00f};
GLfloat vertices139[]={-.35f,.22f,-.07f,

  -.35f,.22f,-.12f,

  -.35f,.26f,-.12f,
```

```
                -.35f,.26f,-.07f};
              GLfloat vertices140[]={-.35f,.22f,-.14f,

                -.35f,.22f,-.19f,

                -.35f,.26f,-.19f,

                -.35f,.26f,-.14f};
////////////////fondo/////////////////
              GLfloat vertices141[]={-1.5f,-1.0f, -.5f,

                1.5f,-1.0f, -.5f,

                1.5f,1.0f, -.5f,

                -1.5f,1.0f, -.5f};

int  LoadBMP(char *filename, int tex_name)
 {
   #define SAIR       {fclose(fp_archivo); return -1;}
   #define CTOI(C)    (*(int*)&C)

   GLubyte    *image;
   GLubyte    Header[0x54];
   GLuint     DataPos, imageSize;
   GLsizei    Width,Height;

   int nb = 0;


   // Abre un archivo y efectua la lectura del encabezado del
archivo BMP
   FILE * fp_archivo = fopen(filename,"rb");
   if (!fp_archivo)
      return -1;
   if (fread(Header,1,0x36,fp_archivo)!=0x36)
      SAIR;
   if (Header[0]!='B' || Header[1]!='M')
      SAIR;
   if (CTOI(Header[0x1E])!=0)
      SAIR;
   if (CTOI(Header[0x1C])!=24)
      SAIR;

   // Recupera los atributos de la altura y ancho de la imagen

      Width   = CTOI(Header[0x12]);
      Height  = CTOI(Header[0x16]);
      ( CTOI(Header[0x0A]) == 0 ) ? ( DataPos=0x36 ) : (
DataPos = CTOI(Header[0x0A]) );

      imageSize=Width*Height*3;

   // Llama a la imagen
      image = (GLubyte *) malloc ( imageSize );
   int retorno;
   retorno = fread(image,1,imageSize,fp_archivo);

   if (retorno !=imageSize)
    {
      free (image);
      SAIR;
    }

   // Invierte los valores de R y B
   int t, i;

   for ( i = 0; i < imageSize; i += 3 )
    {
      t = image[i];
      image[i] = image[i+2];
      image[i+2] = t;
    }

   // Tratamiento de textura para OpenGL


glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FI
LTER,GL_LINEAR);

glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FIL
TER,GL_LINEAR);

glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_
S,GL_MODULATE);

glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_
T,GL_MODULATE);

   glTexEnvf ( GL_TEXTURE_ENV,
GL_TEXTURE_ENV_MODE, GL_REPLACE );

   // Manipulacion en memoria de la textura
   glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, Width,
Height, 0, GL_RGB, GL_UNSIGNED_BYTE, image);

   fclose (fp_archivo);
   free (image);
   return 1;
 }


void onMouse(int button, int state, int x, int y)
{
            if ( (button == GLUT_LEFT_BUTTON) & (state ==
GLUT_DOWN) ) {
            x0 = x; y0 = y;}
}

void onMotion(int x, int y)
{
            alpha = (alpha + (y - y0));
            beta = (beta + (x - x0));
            x0 = x; y0 = y;
            glutPostRedisplay();
}


void myinit(void)
{

   // Propiedades del material
            GLfloat mat_especular[] = { 1.0f, 0.0f, 0.0f, 1.0f};
```

```
            GLfloat mat_shininess[] = { 50.0f};
            GLfloat mat_ambiente[] = { 0.25f, 0.25f, 0.25f,
1.0f};
            GLfloat mat_difuso[] = { 0.5f, 0.5f, 0.5f, 1.0f};

            glClearColor (0.0, 0.0, 0.0, 0.0);

            glMaterialfv(GL_FRONT, GL_SPECULAR,
mat_especular);
            glMaterialfv(GL_FRONT, GL_SHININESS,
mat_shininess);
            glMaterialfv(GL_FRONT, GL_DIFFUSE,
mat_difuso);
            glMaterialfv(GL_FRONT, GL_AMBIENT,
mat_ambiente);

            // Parametros de las luces

            // Habilitando la luz
            glEnable(GL_LIGHTING);

            // Especifica una luz de direccion simple
            GLfloat ambiente1[] = { 0.5f,0.5f,0.5f};
            GLfloat difusa1[] = { 0.5f,0.5f,0.5f};
            GLfloat especular1[] = { 1.0f,0.0f,0.0f};
            GLfloat posicion1[] = { 0.0f,0.0f,5.0f,0.0};

            glLightfv(GL_LIGHT0, GL_AMBIENT, ambiente1);
            glLightfv(GL_LIGHT0, GL_DIFFUSE, difusa1);
            glLightfv(GL_LIGHT0, GL_SPECULAR,
especular1);
            glLightfv(GL_LIGHT0, GL_POSITION, posicion1);


            glEnable(GL_LIGHT0);

            // Especifica un solo proyector posicional
            GLfloat ambiente2[] = { 1.0f,1.0f,0.0f};
            GLfloat difuso2[] = { 1.0f,0.0f,0.0f};
            GLfloat posicion2[] = { 1.0f,0.0f,5.0f,1.0};
            GLfloat direccion2[] = {0.0f,0.0f,-5.0f};

            glLightfv(GL_LIGHT1, GL_AMBIENT, ambiente2);
            glLightfv(GL_LIGHT1, GL_DIFFUSE, difuso2);
            glLightfv(GL_LIGHT1, GL_POSITION, posicion2);

            glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION,
direccion2);
            glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, 15.0f);

            glEnable(GL_LIGHT1);
            glDepthFunc(GL_LESS);
   glEnable(GL_DEPTH_TEST);
   glTexCoordPointer(2, GL_FLOAT, 0, texcoords);
            glEnableClientState(GL_VERTEX_ARRAY) ;
            glEnableClientState(GL_TEXTURE_COORD_ARR
AY);
            glEnable(GL_AUTO_NORMAL);
   glEnable(GL_NORMALIZE);

            glPolygonMode(GL_FRONT_AND_BACK,GL_FILL
);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   glTranslatef (0.0, 0.0, -5.0);
   glEnable(GL_TEXTURE_2D);
   glPixelStorei(GL_UNPACK_ALIGNMENT,1);
            glGenTextures(10,texture_id);

            glBindTexture(GL_TEXTURE_2D,texture_id[0]);
            LoadBMP("pared.bmp",0);
            glBindTexture(GL_TEXTURE_2D,texture_id[1]);
            LoadBMP("paredgrande.bmp",1);
   glBindTexture(GL_TEXTURE_2D,texture_id[2]);
            LoadBMP("puerta.bmp",2);
            glBindTexture(GL_TEXTURE_2D,texture_id[3]);
            LoadBMP("paredatras.bmp",3);
            glBindTexture(GL_TEXTURE_2D,texture_id[4]);
            LoadBMP("techo2.bmp",3);
            glBindTexture(GL_TEXTURE_2D,texture_id[5]);
            LoadBMP("borde.bmp",3);
            glBindTexture(GL_TEXTURE_2D,texture_id[6]);
            LoadBMP("paisaje4.bmp",3);
}

void castillo(){
            glColor3ub(192,192,192);//Gris claro(chica)
            glEnableClientState(GL_VERTEX_ARRAY);
//////////frente//////////////////
            glBindTexture(GL_TEXTURE_2D,texture_id[0]);
            glVertexPointer(3,GL_FLOAT,0,vertices1);
            glDrawArrays(GL_QUADS,0,4);
            glVertexPointer(3,GL_FLOAT,0,vertices2);
            glDrawArrays(GL_QUADS,0,4);
            glBindTexture(GL_TEXTURE_2D,texture_id[1]);
            glVertexPointer(3,GL_FLOAT,0,vertices3);
            glDrawArrays(GL_QUADS,0,4);
            glVertexPointer(3,GL_FLOAT,0,vertices4);
            glDrawArrays(GL_QUADS,0,4);
            glBindTexture(GL_TEXTURE_2D,texture_id[0]);
            glVertexPointer(3,GL_FLOAT,0,vertices5);
            glDrawArrays(GL_QUADS,0,4);
            glVertexPointer(3,GL_FLOAT,0,vertices6);
            glDrawArrays(GL_QUADS,0,4);
            glBindTexture(GL_TEXTURE_2D,texture_id[2]);
            glVertexPointer(3,GL_FLOAT,0,vertices7);
            glDrawArrays(GL_QUADS,0,4);
            glBindTexture(GL_TEXTURE_2D,texture_id[0]);
            glVertexPointer(3,GL_FLOAT,0,vertices8);
            glDrawArrays(GL_QUADS,0,4);
            glVertexPointer(3,GL_FLOAT,0,vertices9);
            glDrawArrays(GL_QUADS,0,4);
            ///////izquierdo/////////
            glBindTexture(GL_TEXTURE_2D,texture_id[0]);
            glVertexPointer(3,GL_FLOAT,0,vertices10);
            glDrawArrays(GL_QUADS,0,4);
            glVertexPointer(3,GL_FLOAT,0,vertices11);
            glDrawArrays(GL_QUADS,0,4);
            glBindTexture(GL_TEXTURE_2D,texture_id[1]);
            glVertexPointer(3,GL_FLOAT,0,vertices12);
            glDrawArrays(GL_QUADS,0,4);
            glVertexPointer(3,GL_FLOAT,0,vertices13);
            glDrawArrays(GL_QUADS,0,4);
            glVertexPointer(3,GL_FLOAT,0,vertices14);
```

```
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices15);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices16);
glDrawArrays(GL_QUADS,0,4);
////////////////Atras////////////
glBindTexture(GL_TEXTURE_2D,texture_id[3]);
glVertexPointer(3,GL_FLOAT,0,vertices17);
glDrawArrays(GL_QUADS,0,4);
glBindTexture(GL_TEXTURE_2D,texture_id[0]);
glVertexPointer(3,GL_FLOAT,0,vertices18);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices19);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices20);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices21);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices22);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices23);
glDrawArrays(GL_QUADS,0,4);
/////////derecho/////////
glBindTexture(GL_TEXTURE_2D,texture_id[0]);
glVertexPointer(3,GL_FLOAT,0,vertices24);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices25);
glDrawArrays(GL_QUADS,0,4);
glBindTexture(GL_TEXTURE_2D,texture_id[1]);
glVertexPointer(3,GL_FLOAT,0,vertices26);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices27);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices28);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices29);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices30);
glDrawArrays(GL_QUADS,0,4);
////////////////Techo////////////
glBindTexture(GL_TEXTURE_2D,texture_id[4]);
glVertexPointer(3,GL_FLOAT,0,vertices31);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices32);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices33);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices34);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices35);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices36);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices37);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices38);
glDrawArrays(GL_QUADS,0,4);
//////////////////////////bordes//////////////
//////////frente/////////////
glBindTexture(GL_TEXTURE_2D,texture_id[5]);
glVertexPointer(3,GL_FLOAT,0,vertices39);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices40);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices41);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices42);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices43);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices44);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices45);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices46);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices47);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices48);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices49);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices50);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices51);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices52);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices53);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices54);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices55);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices56);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices57);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices58);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices59);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices60);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices61);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices62);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices63);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices64);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices65);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices66);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices67);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices68);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices69);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices70);
glDrawArrays(GL_QUADS,0,4);
```

```
glVertexPointer(3,GL_FLOAT,0,vertices71);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices72);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices73);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices74);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices75);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices76);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices77);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices78);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices79);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices80);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices81);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices82);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices83);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices84);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices85);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices86);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices87);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices88);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices89);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices90);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices91);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices92);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices93);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices94);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices95);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices96);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices97);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices98);
glDrawArrays(GL_QUADS,0,4);
//////////lado//////////
glVertexPointer(3,GL_FLOAT,0,vertices99);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices100);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices101);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices102);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices103);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices104);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices105);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices106);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices107);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices108);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices109);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices110);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices111);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices112);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices113);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices114);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices115);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices116);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices117);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices118);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices119);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices120);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices121);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices122);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices123);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices124);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices125);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices126);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices127);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices128);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices129);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices130);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices131);
glDrawArrays(GL_QUADS,0,4);
glVertexPointer(3,GL_FLOAT,0,vertices132);
```

```
		glDrawArrays(GL_QUADS,0,4);
		glVertexPointer(3,GL_FLOAT,0,vertices133);
		glDrawArrays(GL_QUADS,0,4);
		glVertexPointer(3,GL_FLOAT,0,vertices134);
		glDrawArrays(GL_QUADS,0,4);
		glVertexPointer(3,GL_FLOAT,0,vertices135);
		glDrawArrays(GL_QUADS,0,4);
		glVertexPointer(3,GL_FLOAT,0,vertices136);
		glDrawArrays(GL_QUADS,0,4);
		glVertexPointer(3,GL_FLOAT,0,vertices137);
		glDrawArrays(GL_QUADS,0,4);
		glVertexPointer(3,GL_FLOAT,0,vertices138);
		glDrawArrays(GL_QUADS,0,4);
		glVertexPointer(3,GL_FLOAT,0,vertices139);
		glDrawArrays(GL_QUADS,0,4);
		glVertexPointer(3,GL_FLOAT,0,vertices140);
		glDrawArrays(GL_QUADS,0,4);
}

void fondo(){
		////////////////fondo/////////////////
		glBindTexture(GL_TEXTURE_2D,texture_id[6]);
		glVertexPointer(3,GL_FLOAT,0,vertices141);
		glDrawArrays(GL_QUADS,0,4);
}

void display(void)
{
  glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);
  glPushMatrix();
		fondo();
		glRotatef(alpha, 1.0f, 0.0f, 0.0f);
		glRotatef(beta, 0.0f, 1.0f, 0.0f);
		glRotatef(30, 0.0f, 1.0f, 0.0f);
		glRotatef(4, 1.0f, 0.0f, 0.0f);
		glTranslatef(-0.5f,0.15f,-0.7f);
		glScalef(0.75f,0.75f,0.75f);
		castillo();
  glPopMatrix();
  glutSwapBuffers();
}

void reshape(int w, int h)
{
  glViewport(0, 0, w, h);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
		gluPerspective(20.0f,(GLdouble)w/(GLdouble)h,
1.0f, 10.0f);
  glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE |
GLUT_RGB);
		glutInitWindowSize(1270,1024);
  glutCreateWindow(" Castillo 3D ");
  myinit();
  glutReshapeFunc(reshape);
  glutDisplayFunc(display);
		glutMouseFunc(onMouse);
		glutMotionFunc(onMotion);
  glutMainLoop();
  return 0;
}
```

IMPRESIONES DE PANTALLA

CONCLUSIONES

Gracias a OpenGl aprendimos y ahora sabemos dominar todo tipo de escenario grafico pudiendo así diseñar e implementar un escenario 3D junto con texturas movimiento, y por último también se pudo hacer manejo de luces para que este tenga un aspecto más realista y más iluminado.

Sabemos que OpenGL es un gran **API multilenguaje** y **multiplataforma** para escribir aplicaciones que produzcan gráficos 2D y 3D. Consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples, tales como puntos, líneas y triángulos, hasta importar objetos en 3D.

20 de Febrero del 2015

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

REPORTE TÉCNICO:

SEGUNDO EXAMEN PARCIAL

PRESENTA:

LUIS ALBERTO SILVA ESCAMILLA

MATERIA:

GRAFICACIÓN

PRIMAVERA 2015

Introducción

Durante la etapa correspondiente al segundo parcial, se paso del empleo de unicamente la primitiva GL_POINTS para a partir de ellas dibujar las demas figuras que necesitaramos, a utilizar las demas primitivas que OpenGL provee. Recordemos que una primitiva es simplemente la interpretación de un conjunto de vértices dibujados de una manera específica en pantalla. Hay diez primitivas distintas en OpenGL[1], pero para este caso se utilizaron solamente las más comunes: puntos (GL_POINTS), líneas (GL_LINES), triángulos (GL_TRIANGLES) y cuadrados (GL_QUADS). Se utilizarón también las primitivas GL_LINES_STRIP, GL_TRIANGLE_STRIP y GL_QUAD_STRIP, utilizadas para definir “tiras” de líneas, triángulos y de cuadrados respectivamente.

La utiilización de las primitivas de OpenGL no tiene mayor complejidad mas que recordar los nombres y el funcionamiento, aunque dado que el nombre de las primitivas es bastante intuitivo, estos dan una idea de su utilización. Por ejemplo, la mayoría de las primitivas de OpenGL utilizan esta estructura.

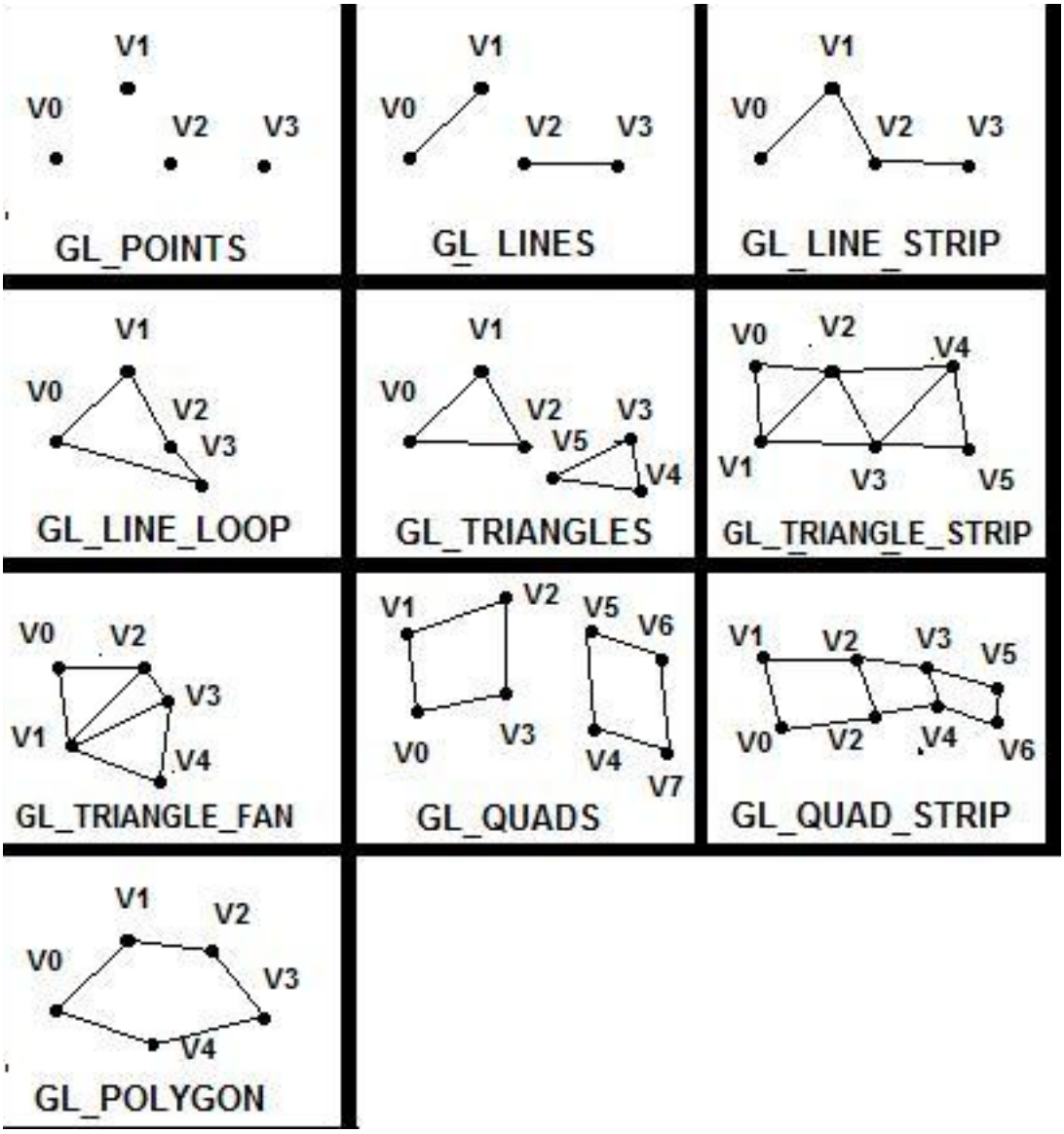
```
glBegin(<tipo de primitiva>);
   glVertex(...);
   glVertex(...);
   ...
   glVertex(...);
glEnd();
```

De manera específica las 10 primitivas de OpenGL y de manera particular, las antes mencionadas trabajan con esta estructura. Como mencionaba arriba, los nombres de las primitivas dan una idea de la utilización de las mismas.

Nombre	Descripción
GL_POINTS	Los vértices que dibujemos solo dibujaran un punto.
GL_LINES	Cada dos vértices dibujara una línea, si la cantidad de vértices indicados es impar se elimina el ultimo.
GL_LINE_STRIP	se usan los vértices para

	dibujar líneas. Después del segundo vértice, cada vértice subsiguiente especifica el punto al que se extiende la línea.
GL_LINE_LOOP	igual que GL_LINE_STRIP pero el ultimo vértice de todos se unirá con el primero de todos.
GL_TRIANGLES	Dibujara triángulos cada 3 vértices, si el numero de vértices indicados no se puede dividir en 3 ignora los últimos.
GL_TRIANGLE_STRIP	Dibujara un triangulo con los 3 primeros vértices después con los 2 siguientes vértices y el ultimo de los 3 anteriores vuelve a dibujar un triangulo.
GL_TRIANGLE_FAN	Dibuja un abanico de triángulos. Cada vértices después del tercero se usa para dibujar otro triangulo.
GL_QUADS	Dibujara cuadrados cada 4 vértices. Si el numero de vértices no se puede dividir

	en 4 ignora los últimos vértices.
GL_QUAD_STRIP	Dibuja cuadrados consecutivos es decir, coge los dos últimos vértices indicados con los dos nuevos y va dibujando cuadrados.
GL_POLYGON	Con los vértices que indiquemos se dibuja un polígono convexo. El ultimo vértice indicado se cierra con el primero para asegurar que ha quedado cerrado.



Por otro lado, aunque en el primer parcial se vieron los operadores de transformaciones tales como lo son: traslaciones, rotaciones y escalamientos, e incluso se intento programarlos; los resultados y la manipulación era en general bastante complicados.

Sin embargo, con las funciones que OpenGL provee, el manejo de estos conceptos es bastante mas sencillo ya que existen las funciones **glTranslatef(), glRotatef()** y **glScalef()**. Recordemos también que OpenGL trabaja con matrices, por lo cual las transformaciones son en realidad matrices que permiten la transformación sobre el objeto. Anque su manejo es bastante mas sencillo, es necesario no olvidar que en OpenGL se maneja el concepto de objetos unitarios, por lo cual todas las figuras se manejan como un todo. Es importante no perder esta noción ya que de lo contrario, a la hora de manipular las figuras con las funciones anteriores se pueden obtener resultados inesperados.

Se vio de manera escueta la función **glutSolidSphere()** y **glutWireSphere()** que se utilizarón en la práctica del sistema solar para dibujar circulos y curvas.

Finalmente se vierón texturas. Las texturas son básicamente bibliotecas que permiten cargar una imagen externa para utilizarla dentro de un programa en OpenGL. Dicho de otro modo, aplicar una imagen externa a cualquier objeto en OpenGL. Para esto, OpenGL cuenta con una librería llamada gl/aux.h desarrollada junto a OpenGL, sin embargo, cuando esta dejo de tener soporte gl/aux también. Por otro lado, existen varia librerias tales como FreeImage, que permiten importar imágenes para texturas, con el agregado con que aún cuentan con soporte aunque algunas no estan disponibles para todas las distribuciones, los compiladores y algunas son mas difíciles de emplear que otras.

Practica de laboratorio:

El programa correspondiente al segundo parcial consiste en realizar un escenario utilizando para ello las primitivas de OpenGL. El programa debe de utilizar todo (o al menos) la mayor parte de los visto durante esta etapa del curso es decir, el escenario debe de contener escalamientos, traslaciones, rotaciones y texturas.

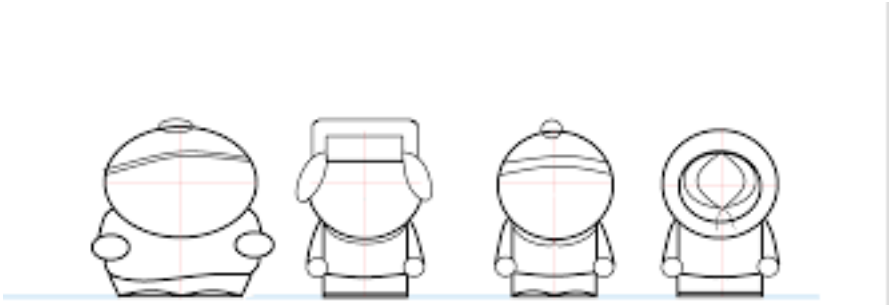
El diseño del escenario es libre, sin embargo, debe de contar con las características antes mencionadas.

El escenario propuesto por el alumno es el siguiente:

Con al agregado que para mostrar el empleo de traslaciones y rotaciones se anexara al escenario un auto en movimiento por la calle y un pedazo de paja circular girando y atravezando la calle.

El procedimiento para la creación del escenario consistio en los siguientes pasos.

1. Elegir el escenario.
2. Dibujar los bocetos de las figuras a emplear para cada uno de los personajes,



dado que no existe una manera sencilla de dibujar figuras en OpenGL, se tiene que planear que figuras predefinidas en OpenGL se pueden utilizar dibujar alguna parte en específico, y cuales tendrían que dibujar utilizando otras formas.

3. El orden de dibujado fue el siguiente de derecha a izquierda empezando por el niño gordo (Cartman/Nelson), Kenny/Rafa, Keil/Milhouse, Stan/Bart, la acera, la casa, los cerros y los pinos.
4. Se dibujo el automóvil y se movió utilizando la función **glTranslatef(_, _, _)** desplazando el dibujo sobre el eje de las X variando el valor de el parametro x y multiplicándolo por -1 para que se moviera de derecha a izquierda:
   `glTranslatef(-incremento, -22.0, 1.0);`
   Todos los objetos que forman parte del automóvil se colocarón dentro de un **glPushMatrix()**, **glPopMatrix()** para que tambien se desplazaran como una unidad.

Para la estrucuración del programa se utilizó un *templete* de programas ya utilizados con anterioridad tales como el sistema solar y el girasol; por lo cual el programa consta en general de las siguientes partes:

```
/* Segundo Parcial; South Park

   Author: Luis Alberto Silva Escamilla

*/

//Cabeceras necesarias para las funciones

#include <GLUT/glut.h>

#include <stdlib.h>

#include <math.h>


#define xmin -200

#define xmax 200

#define ymin -200

#define ymax 200


//Se utiliza para el redibujado

void reshape()

{
```

```
}

//Aquí va el código correspondiente al dibujo, es decir todas //las
figuras que forman al escenario.

void dibuja()

{

}


//Se utiliza para cuando la máquina esta osciosa.

void idle()

{

}


//Donde se inicizalizan los valores

void init()

{

}


//Es como un main de cualquier programa en C o C++ y llama a las funciones
anteriores, además de llamar a la función glutMainLoop() para el repintado
del escenario.

int main()

{

glutInit(&argc, argv);


    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
```

```
    glutInitWindowSize(600, 600);
    glutCreateWindow("South Park LUIS SILVA");
    init();
    glutDisplayFunc(dibuja);
     glutReshapeFunc(reshape);
     glutIdleFunc(idle);



    glutMainLoop();
return 0;
}
```

En la función dibujo() cada figura esta estructurada llevando el siguiente orden:

```
void dibujo()
{
glClear(GL_COLOR_BUFFER_BIT);      //borra la ventana de visualizacion
 //###### P A I S A J E #########//############################
Código(Montañas, casa, acera, calle, etc)

//P E R S O N A J E S ###### P E R S O N A J E S ###########
//&&&&&&&&&&&&&&&&&& CARTMAN &&&&&&&&&&&&&&&&&&&&&&&&&&&&
Código (Cabeza, cuerpo, detalles de cara y ropa)

//&&&&&&&&&&&&&&&&&& KENNY &&&&&&&&&&&&&&&&&&&&&&&&&&&&
Código (Cabeza, cuerpo, detalles de cara y ropa)

//&&&&&&&&&&&&&&&&&& KEIL &&&&&&&&&&&&&&&&&&&&&&&&&&&&
```

```
Código (Cabeza, cuerpo, detalles de cara y ropa)


//&&&&&&&&&&&&&&&&&& STAN &&&&&&&&&&&&&&&&&&&&&&&&&&&&

Código (Cabeza, cuerpo, detalles de cara y ropa)

//############## F I N  ###  P E R S O N A J E S ######




//&&&&&&&&&&&&&&&&&&&&&&&&&&&&AUTOMOVIL&&&&&&&&&&&&&&&&&&&&&&&&&&&&

Código(Silueta, ventanas, faros, ruedas)

glFlush();


     //glutSvapBuffers();

}
```

Conceptos desarrollados

- Máquina de estados: Es un concepto relativo a automátas y que tiene una aplicación en OpenGL ya que prácticamente todas las operaciones en OpenGL funcionan mediante estados, ejemplos de las funciones que funcionan así son glColor3f(). Para manejar los estados se utilizan las funciones **glPushMatrix()** y **glPopMatrix().**
- Pila: Se utilizo para manipular la máquina de estados. Utilizan las funciones anterirores.
- Primitiva: Es la interpretación de un conjunto de vértices dibujados de una manera específica en pantalla. Existen un total de 10.
- Transformaciones: Funciones utilizadas para la manipulación de los objetos unitarios en OpenGL. Tales funciones son **glTranslatef(), glRotatef()** y **glScalef().** Recordemos que las transformaciones se aplican en orden inverso.

Experimentos:

Las texturas fue lo mas complicado de esta tarea, ya que gl/aux.h el nativo de OpenGL no se pudo utilizar con el compilador DevC++ en Windows. Por otra parte, se intento instalar en Mac OS X, sin embargo en este la instalación es aún mas complicada y su complilación lo es también. De hecho en Mac OS X para compilar en programa en C o C++ y que utilice OpenGL, es necesario llamar a todos los frameworks de manera explicita al compilar desde consola.

```
MacBook-Pro-de-Luis:SegundoParcial siel_alb$ c++ /System/Library/Frameworks/GLUT
.framework/GLUT /System/Library/Frameworks/OpenGL.framework/OpenGL segundoLuis.c
pp -o segundoLuis
```

además de cambiar la cabera de OpenGL, así en lugar de utilizar GL/glut.h en Windows, en Mac OS X sería GLUT/glut.h

Se trato de instalar FreeImage en Windows, fue difícil bajarlo desde los repositorios de Dev C++ aunque finalmente se pudo instalar. Desafortunadamente la instalación se dio a destiempo, por lo cual la tarea de utilizar texturas para el escenario no se pudo llevar a cabo.

Un detalle aparte, es que no se utilizaron funciones para las partes del escenario, lo cual hubiera justificado bastante el tomar ciertas partes que se repetian en un dibujo y podrían, posteriormente, utilizarse en otras. Se trato de realizar esto, sin embargo, no funciono ya que no se planeo de manera rigurosa cada una de las piezas del escenario.

Conclusiones

La figura obtenida fue la siguiente:

Como conclusión podemos mencionar que se llevo a cabo un primer escenario completo utilizando la mayor parte de las librerías de OpenGL. Se utilizarón transformaciones tales como rotación, traslación y escalamiento para la mayoria de las figuras. Por ejemplo, para los pinos se les dibujo utilizando triangulos, se escalo toda la figura y posteriormente se le copio para trasladar los diferentes pinos resultantes a lo largo del escenario. Un procedimiento similar se llevo a cabo con el resto de las figuras, se les dibujo en un sitio específico, se les escalo a algunas y se les traslado a diferentes puntos del escenario. No obstante, no se llevo a cabo lo de texturas por la problemática mencionada anteriormente, sin embargo se considera que en general el escenario fue completado en un 90% aunque no fielmente. Esto se puede justificar debido a la falta de experiencia para la manipulación de colores así como al díseño del escenario y la planeación del tiempo.

Finalmente, se considera que con mas práctica y una planeación mas anticipada y fielmente seguida, se puede obtener un mejor resultado.

Bibliografía

[1] http://sabia.tic.udc.es/gc/Tutorial%20OpenGL/tutorial/cap3.htm

Colores:

https://www.opengl.org/discussion_boards/showthread.php/132502-Color-table

Capítulo 3: Dibujando en 3D:

http://sabia.tic.udc.es/gc/Tutorial%20OpenGL/tutorial/cap3.htm

Primitivas:

http://opengl.infojosep.es/primitivas/primitivas.php

Transformaciones:

http://openglenfichas.uji.es/ejemplos/transf.pdf

Texturas:

http://informatica.uv.es/iiguia/AIG/docs/texturas.htm

20 de Febrero del 2015

Benemérita Universidad Autónoma de Puebla

Reporte del Proyecto

Graficación

Miguel Ángel Hilario Xelano

Introducción:

La meta de este proyecto es que como estudiante de Graficación ponga en práctica lo estudiado en estos meses de clase, por lo que se hará uso de la primitiva punto para poder dibujar: líneas o figuras y con esto poder realizar operaciones de rotación, traslación, escalamiento, deformación y reflexión para así poder dibujar escenarios un poco complejos.

Desarrollo:

Para lograr nuestro cometido se hizo uso de las siguientes Clases:

-Punto

-Línea

-Triangulo

-Cuadrilátero

-Circulo

-Matrices

Punto:

En esta función se pretende dibujar dos puntos, posicionando en un plano cartesiano de segunda dimensión, para ello se declara de la siguiente manera:

Variables de la clase:

Int x;

Int y;

-Que son almacenados en un arreglo denominado: “Coord”

Métodos:

punto() : Declaración del constructor por defecto.

~punto() : Destructor por defecto.

setValues(int x, int y) : Método en el que recibe los datos en este caso 2 enteros.

getX() : Devuelve el valor de “X”, para poder hacer uso de el.

getY() : Devuelve el valor de “Y”, para poder hacer uso de el.

*getValues() : Devuelve el arreglo con los valores de “X,Y”, para poder hacer uso de ellos .

draw() : Metodo que dibuja el punto con los valores recibidos.

Con todo esto estamos listos para poder dibujar una infinidad de puntos.

Ejemplo:

x=0 : y=0

Dibujado de un punto: ▪

Línea:

En la clasa se hace uso de la clase punto para poder definir el punto inicial (x,y) y el punto final (x1,y1), con estos datos podremos calcular la trayectoria de la línea y para ello se hace uso del siguiente código el cual va incrementando el valor de “y” o “x” según sea el caso y de la fórmula:

-Formula: m = (y2-y1)/(x2-x1);

-Código:

```
if(x2-x1==0)
 {
    for(float i=y;i<=supY;i++)
    {

       glBegin(GL_POINTS);
          glVertex2i(x2,i);
       glEnd();
    }
 }
 if(x2>x1)
    for(float i=x1;i<=x2;i++)
       {
          glBegin(GL_POINTS);
             glVertex2i(i,y1);
          glEnd();
          y1=y1+m;
       }
 else
    if(x1>x2)
       for(float i=x2;i<=x1;i++)
       {
          glBegin(GL_POINTS);
             glVertex2i(i,y2);
          glEnd();
          y2=y2+m;
       }
```

En este código se comparan los valores de x2 y x1 en la primera condición se verifica que x2-x1 sea igual a 0 por lo que se toman como punto inicial y1 y como punto final y2, en la 2 condición se verifica si x2 es mayor que x1 si es asi al punto y1 se le suma m hasta que x1 sea igual a x2 y en la 3 condición se compara que x1 sea mayor que x2 si esto es cierto entonces el incremento se hace en y2.

Valores clase:

-2 puntos: Valores que son del tipo punto(x,y).

Métodos:

-linea(): Declaración del constructor por defecto.

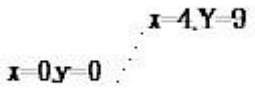
-~linea(): Destructor por defecto.

-setValues(int x,int y,int x1,int y1): Metodo que recibe los valores en este caso son de tipo entero x,y,x1 y y1.

-draw(); Metodo que dibuja la clase haciendo uso de la formula.

Con todo esto podremos dibujar líneas:

Ejemplo:

Trazado de una línea: 

Triangulo:

En la clase Triangulo se hace uso de la clase línea para poder definir las líneas, ya que esta hacen uso de la clase punto, por lo tanto para graficar un triángulo se necesitan 3 puntos con los cuales podemos definir el tipo de triangulo a dibujar: escaleno, isósceles o equilátero.

Valores de la clase:

-3 variables del tipo Punto(x,y).

Métodos:

En la clase triangulo se definen los siguientes métodos:

-triangulo(): Constructor por defecto del triangulo

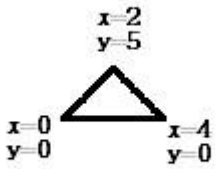
-~triangulo(): Destructor

-setValues(punto,punto,punto): método con el cual podemos asignar valores.

-draw(): Metodo para dibujar un triangulo

Con estos métodos se puede definir los valores de un triángulo y así poder graficarlo, en este caso se hace un llamado a la función línea para ir uniendo punto por punto y con esto obtener un resultado como este:

Trazado de un Triángulo:



Cuadrilátero:

Esta clase tiene como función dibujar un cuadrado o rectángulo, para poder realizar esta acción hace usó de los métodos línea y punto, por lo que se puede decir que para dibujar un cuadrilátero se necesitan 2 puntos (4 coordenadas) y 4 lineas.

Valores de la clase:

- 2 Variables del tipo Punto(x,y).

Metodos:

Dentro de los métodos de la clase tenemos:

- cuadrilatero(): Este es el constructor por defecto.

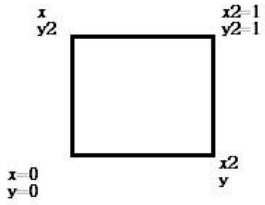
-~cuadrilatero(): Destructor de la clase.

-setValues(punto,punto): Este es el método que ingresa los valores a las variables de tipo punto.

-draw(): Este es el método que dibuja el cuadrado, esta función utiliza como parámetros 2 puntos con los cuales se definen los dos restantes y se agregan respectivamente a una variable del tipo line y se trazan con su función línea draw().

Como resultado obtendríamos lo siguiente:

Trazado de un cuadrilátero:



Circulo:

En la clase Círculo se pretende dibujar un círculo, para ello se hace uso de la clase punto y del siguiente código:

```
Incremento =2*M_PI/n;
    for (int i=0;i<=n;i++)
{
        teta  =i*incremento;
          x1   =r*(cos(teta));
          y1=r*(sin(teta));
          p1.setValues((x1+xDesplaz),(y1+yDesplaz));
          p1.draw();
 }
```

Este código va incrementando y haciendo cálculos para encontrar el punto(x,y) en el cual debe ir el primer punto y así respectivamente hasta calcular 360 los cuales son los grados que abarca una circunferencia en radianes, también se debe mencionar que se necesita implementar la función setValues() del punto al igual que la de draw() para poder ir dibujando cada punto(x,y) en cada incremento.

Variables de la clase:

-Centro del tipo punto(x,y).

-Radio.

Métodos:

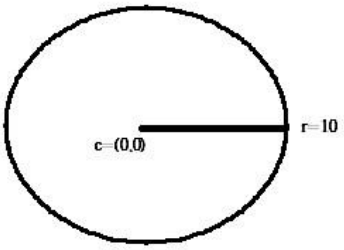
-circulo(): Este es el constructor por defecto de la clase.

-~circulo(): Destructor de la clase.

-setValues(punto,int): Agrega los valores a punto y radio.

-draw(): Dibuja el circulo haciendo uso del codigo para agregar datos al punto y usa su función punto draw() para ir dibujando cada punto en cada incremento.

Una vez teniendo esto podemos dibujar el círculo, ejemplo:



Trazado de un Círculo:

Matrices:

Esta clase tiene como objetivo englobar todas las operaciones que vimos en todo el curso las cuales son:

- Traslación
- Rotación
- Escalamiento
- Reflexión (x,y,xy)
- Deformación (x,y)

Bueno para poder hacer todo esto y optimizar se necesita la ayuda de una matriz la cual lleva por nombre Matriz de Proyección que no es más que una matriz identidad, la cual es definida como variable de la clase y es afectado por cualquiera de sus métodos, por lo que para realizar cambios en una figura se necesitaría afectar a cada uno de los puntos que forman esa figura.

Las matrices correspondientes a esas operaciones son:

Traslación:

$$\begin{pmatrix} 1 & 0 & movX \\ 0 & 1 & movY \\ 0 & 0 & 1 \end{pmatrix}$$

movX = int

movY = int

Escalamiento:

$$\begin{pmatrix} e & 0 & 0 \\ 0 & e & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

e = int

Rotación:

ang = int

Reflexión X:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Reflexión Y:

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Reflexión XY:

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Deformación X:

sh= int

$$\begin{pmatrix} 1 & sh & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Deformación en Y:

sh= int

$$\begin{pmatrix} 1 & 0 & 0 \\ sh & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Una vez definidas estas matrices con las cuales al operarlas nos devuelven las nuevas coordenadas del punto.

Para poder operar entre ellas también necesitamos una matriz donde guardar los resultados:

Matriz Proyección o de Resultados:

Matriz P: $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

La cual nos servirá para almacenar la matriz resultante al multiplicar matrices para poder optimizar el modificado de valores en los puntos.

Una vez definido todo podemos dar paso al contenido dentro de la clase Matrices:

Variables de la clase:

-MatrizP[3][3]

Métodos:

-matrices(): Define los valores por defecto en la matriz.

-~matrices(): Destructor de la clase.

-MTraslacion(int,int): Engloba la matriz para trasladar una figura a otro punto del plano.

-MEscalamiento(int): Engloba la matriz de Escalamiento la cual permite agrandar una imagen a n tamaño.

-MRotacion(int): Engloba la matriz de Rotación con la cual podemos rotar una figura.

-MReflexion(int): Engloba la matriz de Reflexión la cual se puede definir en 3 parámetros: Reflexión en "X", Reflexión en "Y" y Reflexión en "XY"

- MDeformacion(int,int): Engloba la matriz de Deformación la cual se define en 2 parámetros: "X","Y".

-MultiproyPunto(punto): En esta se hace la operación de multiplicación de la Matriz de Proyección una ves que se a definido todo lo que se quiere hacer por el punto, después retorna ese punto modficado.

-MPOriginal(): Devuelve a la matriz Proyección a su estado original.

PD: Es importante que después de afectar a todos los puntos con las operaciones deseadas se devuelva a los valores originales la Matriz de Proyección ya que si no se hace esto seguirá multiplicando valores tras valores y no devolverá lo deseado.

Conclusión:

Este Proyecto- Examen me permitió como estudiante reafirmar mis conocimientos obtenidos en clase y ponerlos en práctica para poder resolver este problema, también me permitió aprender a definir y crear clases en c++ ya que no sabía mucho al respecto, por otra parte pienso que el problema principal que me encontré aparte de mi falta de conocimiento en la programación orientada a objetos en c++ fue la programación de la Matriz Proyección, ya que tenía muchas ideas pero todas fallaban a la hora de devolver valores, hasta que por la ayuda del profesor pude resolver ese problema y así poder implementarlo en el proyecto dándome así una gran amplitud de funciones con estas matrices de operaciones.

Una vez que termine mi proyecto me sentí satisfecho por el arduo trabajo empleado en él ya que pude aprender cosas nuevas y emplear lo aprendido.

[Seleccionar fecha]

[ESCRIBIR EL NOMBRE DE LA COMPAÑÍA]

EN 1

Introducción

OpenGL (Open Graphics Library) es una especificación estándar que define una **API multilenguaje** y **multiplataforma** para escribir aplicaciones que produzcan gráficos 2D y 3D. La interfaz consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples, tales como puntos, líneas y triángulos.

Fundamentalmente OpenGL es una especificación, es decir, un documento que describe un conjunto de funciones y el comportamiento exacto que deben tener. Partiendo de ella, los fabricantes de hardware crean implementaciones, que son bibliotecas de funciones que se ajustan a los requisitos de la especificación, utilizando aceleración hardware cuando es posible. Dichas implementaciones deben superar unos test de conformidad para que sus fabricantes puedan calificar su implementación como conforme a OpenGL y para poder usar el logotipo oficial de OpenGL.

Básicamente OpenGL consiste en una serie de librerías y rutinas de clases por lo cual, OpenGL no es un paquete de software de renderizado y modelado como Blender o 3D Max, básicamente es una API de bajo nivel que proporciona una interfaz de hardware de gráficos. No es por lo tanto ningún lenguaje de programación, sino tan sólo un conjunto de librerías que son utilizadas a través de lenguajes de programación como VisualC++ para conseguir un interfaz software entre las aplicaciones y el hardware gráfico.

OpenGL tiene dos propósitos esenciales:

3. Ocultar la complejidad de la interfaz con las diferentes tarjetas gráficas, presentando al programador una API única y uniforme.

4. Ocultar las diferentes capacidades de las diversas plataformas hardware, requiriendo que todas las implementaciones soporten la funcionalidad completa de OpenGL (utilizando emulación software si fuese necesario).

Desarrollo y Objetivo del Proyecto

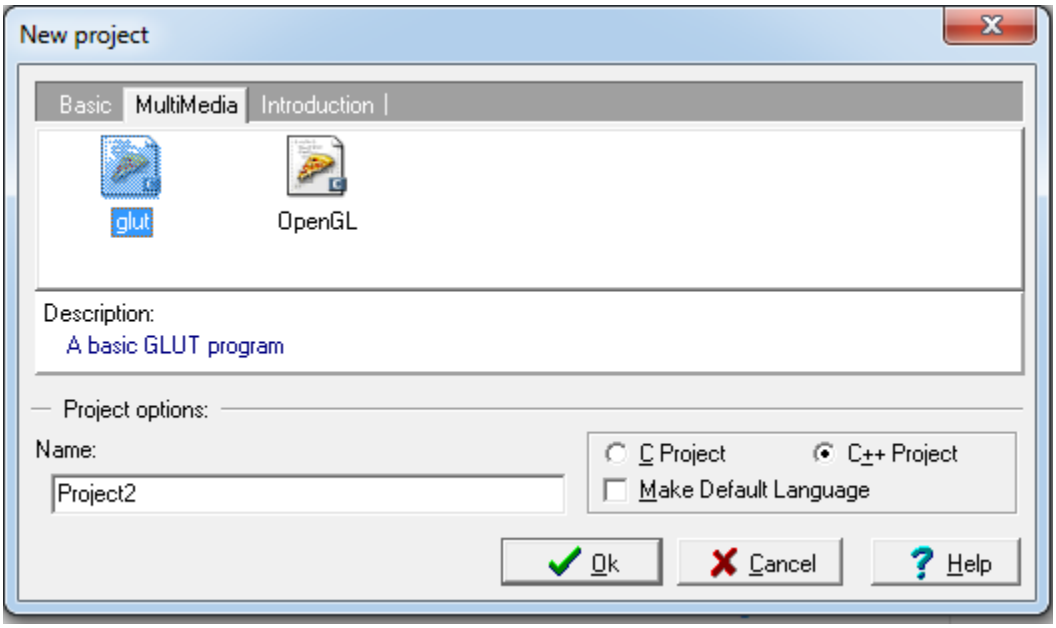
Se trata de un proyecto realizado para la asignatura de graficacion de la Benemérita Universidad Autónoma de Puebla. El lenguaje de programación usado es C y la api OpenGL.

Es un programa realmente básico, se trata de un entorno en 2D en el que aparece una casa junto a una carretera, el sol y unas nubes las cuales estas se mueven en dirección a las manecillas del reloj cuando termina este movimiento, nuevamente este se vuelve a re dibujar.

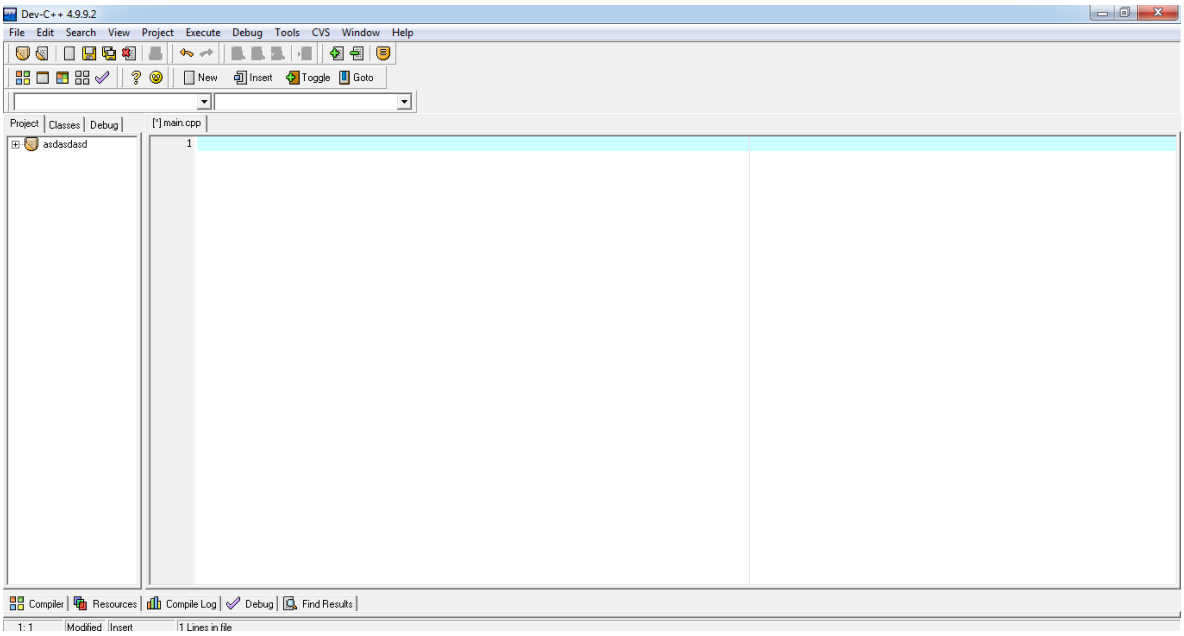
Teniendo en cuenta que para este proyecto se tiene permitido usar todo tipo de librerías, es decir, se puede usar las primitivas que nos ofrece OpenGL.

El desarrollo de este proyecto se explica a continuación:

Se creó un proyecto OpenGl (Glut), teniendo en cuenta que se debe tener instalado el "glut" para que se pueda visualizar todo los entornos de dibujos y animaciones



Se guarda con el nombre deseado antes de pasar a "programar" nuestro escenario 2D.



Tendremos nuestro lugar de trabajo para empezar a trabajar, lo primo que se hizo fue poner nuestras bibliotecas necesarias

```
#include <GL/glut.h>
#include <GL/gl.h>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
```

Usaremos nuestras variables para asignar las diferentes coordenadas y mediante ellas podamos hacer el movimiento de nuestros objetos

```
int rad=100;
int x=1, y=1,z=1,w=1,e=1,f=1,g=1,h=1;  //Variables para las nubes
int x2=1, y2=1,z2=1,w2=1,e2=1,f2=1,g2=1,h2=1;  //Variables para la nube 2
int x3=1, y3=1,z3=1,w3=1,e3=1,f3=1,g3=1,h3=1;  //Variables para la nube 3
int x4=1, y4=1,z4=1,w4=1,e4=1,f4=1,g4=1,h4=1;  //Variables para la nube 4
int x5=1, y5=1,z5=1,w5=1,e5=1,f5=1,g5=1,h5=1;  //Variables para la nube 5
int x6=1, y6=1,z6=1,w6=1,e6=1,f6=1,g6=1,h6=1; //Variables para las LLANTAS
int x7=1, y7=1,z7=1,w7=1,e7=1,f7=1,g7=1,h7=1; //Variables para las LLANTAS


void inicializa(void)
{
glClearColor(0.0,0.3,1.0,0.0);   //color de fondo
```

```
glMatrixMode(GL_PROJECTION);          //Modo de proyección
glLoadIdentity();                     //Establece los parámetros de proyección
gluOrtho2D(0.0, 2000.0, 0.0, 2000.0);     //vista ortogonal
}
```

Tenemos nuestra función "inicializa " que es donde se establece los parámetros del escenario como son el color de fondo, el modo de visualización (en este caso trabajremos con una vista ortogonal 2D), la matriz de proyección que en este caso se usara "GL_PROJECTION".

Seguido de nuestra primera función podemos empezar a crear todas aquellas funciones que queramos insertar en nuestra escena aqui se muestran las que se usaron:

- **void sol(int x,int y,int radio)**
- **void circulo(int x, int y, int radio)**

NOTA: Se tenía en mente hacer una función llamada "carrito" que esta era la que tendríamos que trasladar (movimiento) pero al modificar o agregar más variables funciones y primitivas nuestro escenario echo se modificaba drásticamente.

Estas dos funciones se usaron para fines diferentes aunque solo se creía que se usaría una de ellas ya que bastaba una función para hacer o dibujar círculos en nuestro escenario pero ya que no se pudo hacer un movimiento con el carro implementado se creó la otra función.

- **void dibuja(void)**

Nuestra función dibuja es donde tenemos que implementar todo aquello que queremos que se dibuje en nuestro escenario, y es aqui donde se tuvo que dibujar nuestro escenario echo.
Simplemente lo que se hizo fue mediante polígonos, cuadrados, triángulos, se fue formando cada parte del escenario junto con las texturas para ir diferenciando y hacer un poco más real el dibujo que se presenta.

también dentro de esta misma función ajustamos las coordenadas de movimiento para nuestra figura de sol, las nubes y el carrito que se mueve sobre la carretera que se dibujo.

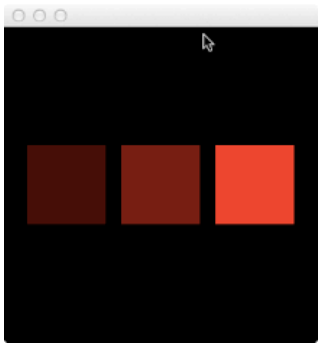
Se muestra lo que se hizo para estas coordenadas

```
if(x1>=1 && x1<1800){
sol(0+x1,1770,200);
x1++;
if(x1==1800){
x1=1;
            }

//nube 1
glColor3f(1.0,1.0,1.0);
if(x>=1 && x<2000){circulo(100+1*x,1800,60);x++;}
if(y>=1 && y<2000){circulo(140+1*y,1790,60);y++;}
if(z>=1 && z<2000){circulo(120+1*z,1790,60);z++;}


if(x==2000){ x=1;}
if(y==2000){ y=1;}
if(z==2000){ z=1;}
```

Las funciones que se usaron se muestras a continuación para tener más idea de cómo se hizo

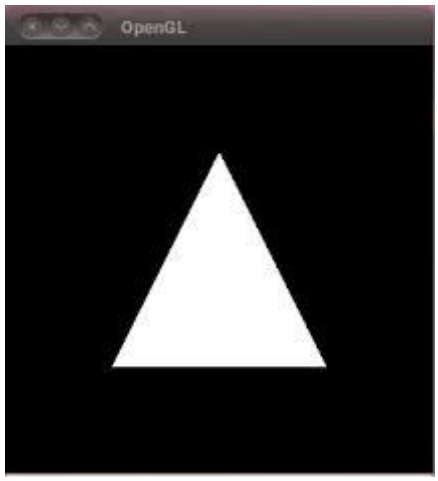
```
glColor3f(1.0,0.0,0.0);
glBegin(GL_QUADS);
glVertex2f(0,680);
glVertex2f(0,640);
glVertex2f(2000,640);
```

```
glVertex2f(2000,680);
glEnd();
```



se le pone color, se declara la primitiva de cuadrado se dibujan las coordenadas en los 4 vértices que este utiliza.
En el proyecto esto fue lo más laborioso ya que se tuvo que diseñar y acomodar cada figura para que se fuera formando el escenario

```
glColor3f(0.0,0.6,0.2);
glBegin(GL_TRIANGLES);
glVertex2f(1725,1100);
glVertex2f(1600,600);
glVertex2f(1850,600);
glEnd();
```



Estas primitivas se usaron para pintar las hojas de un árbol ya que el segundo árbol se hizo con cuadramos y se muestra que se puede hacer de las dos formas solo que en el de cuadrados es mas código que solo poner un triangulo que simula las hojas. de igual forma solo se pone color (verde) y ahora se calculan las 3 coordenadas para que se cree el triangulo

Lo que se puso por ultimo en la función que se describió "dibuja" fue las variables de movimiento y detalles de cada circulo que se uso para hacer el prototipo del auto y sus respectivas llantas

Por último mandamos a llamar a nuestra función dibuja desde el método main el cual se muestra a continuación

```
int main (int argc, char** argv)        //metodo main
{
glutInit(&argc, argv);                          //incializa GLUT
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);     //establece el modo de
visualización
  glutInitWindowSize(800,600);              //tamaño de la ventana
  glutInitWindowPosition(0,0);            //posicion inicial de la ventana
  glutCreateWindow("Examen2");              //nombre de la ventana
  inicializa();
  glutDisplayFunc(dibuja);
```

```
  glutIdleFunc(dibuja);              //Envia los graficos a la ventana de
visualización
  glutMainLoop();                    //muestra todo y espera
  return 0;                         //retorna un valor de cero
}
```

CODIGO FUENTE

```
#include <GL/glut.h>
#include <GL/gl.h>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>


int rad=100;
int x=1, y=1,z=1,w=1,e=1,f=1,g=1,h=1;  //Variables para las nubes
int x2=1, y2=1,z2=1,w2=1,e2=1,f2=1,g2=1,h2=1;  //Variables para la nube 2
int x3=1, y3=1,z3=1,w3=1,e3=1,f3=1,g3=1,h3=1;  //Variables para la nube 3
int x4=1, y4=1,z4=1,w4=1,e4=1,f4=1,g4=1,h4=1;  //Variables para la nube 4
```

```
int x5=1, y5=1,z5=1,w5=1,e5=1,f5=1,g5=1,h5=1;  //Variables para la nube 5

int x6=1, y6=1,z6=1,w6=1,e6=1,f6=1,g6=1,h6=1; //Variables para las LLANTAS
int x7=1, y7=1,z7=1,w7=1,e7=1,f7=1,g7=1,h7=1; //Variables para las LLANTAS


int x8=1, y8=1,z8=1,w8=1,e8=1,f8=1,g8=1,h8=1;  //Variables carro
int x9=1, y9=1,z9=1,w9=1,e9=1,f9=1,g9=1,h9=1;  //Variables carro
int x10=1, y10=1,z10=1,w10=1,e10=1,f10=1,g10=1,h10=1;  //Variables para las nubes
int x11=1, y11=1,z11=1,w11=1,e11=1,f11=1,g11=1,h11=1;  //Variables para las nubes
int x12=1, y12=1,z12=1,w12=1,e12=1,f12=1,g12=1,h12=1;  //Variables para las nubes




double ang=0,b=0,c=0;



int x1=1;
int a2=2;


void inicializa(void)
{
glClearColor(0.0,0.3,1.0,0.0);   //color de fondo
glMatrixMode(GL_PROJECTION);           //Modo de proyeccion
glLoadIdentity();                    //Establece los parametros de proyeccion
gluOrtho2D(0.0, 2000.0, 0.0, 2000.0);     //vista ortogonal
}



void sol(int x,int y,int radio){

        int angulo = 1;
        glColor3f(1.0,1.0,0.0);
        glBegin(GL_TRIANGLE_FAN); //glColor3f(0.0,0.0,0.0);
        glVertex2f(x,y);



         for(angulo=0;angulo<=360;angulo+=5)
{glVertex2f(x+sin(angulo)*radio,y+cos(angulo)*radio);}
         glEnd();
```

```
	}

 void circulo(int x, int y, int radio)
	{
		int angulo=0;
		glBegin(GL_TRIANGLE_FAN);
	 // glColor3f (1.0, 0.0, 1.0);
			glVertex2f(x,y);

			for (angulo=0;angulo<=360; angulo+=6){ glVertex2f(x + sin(angulo) * radio, y +
cos(angulo) * radio);}
		glEnd();
		}



void dibuja(void)			//funcion dibuja
{
 glClear(GL_COLOR_BUFFER_BIT);



//-------------------------------------Sol---------------------------------------
//Ajustamos coordenadas de movimiento para nuestra figura de Sol

if(x1>=1 && x1<1800){
sol(0+x1,1770,200);
x1++;
if(x1==1800){

x1=1;

}

//nube 1
glColor3f(1.0,1.0,1.0);
if(x>=1 && x<2000){circulo(100+1*x,1800,60);x++;}
if(y>=1 && y<2000){circulo(140+1*y,1790,60);y++;}
if(z>=1 && z<2000){circulo(120+1*z,1790,60);z++;}
if(w>=1 && w<2000){circulo(140+1*w,1810,60);w++;}
if(e>=1 && e<2000){circulo(140+1*e,1770,60);e++;}
if(f>=1 && f<2000){circulo(170+1*f,1810,60);f++;}
if(g>=1 && g<2000){circulo(190+1*g,1800,60);g++;}
if(h>=1 && h<2000){circulo(180+1*h,1780,60);h++;}
```

```
if(x==2000){ x=1;}
if(y==2000){ y=1;}
if(z==2000){ z=1;}
if(w==2000){ w=1;}
if(e==2000){ e=1;}
if(f==2000){ f=1;}
if(g==2000){ g=1;}
if(h==2000){ h=1;}

//nube 2
glColor3f(1.0,1.0,1.0);
if(x2>=1 && x2<2000){circulo(400+1*x2,1700,60);x2++;}
if(y2>=1 && y2<2000){circulo(440+1*y2,1690,60);y2++;}
if(z2>=1 && z2<2000){circulo(420+1*z2,1690,60);z2++;}
if(w2>=1 && w2<2000){circulo(440+1*w2,1710,60);w2++;}
if(e2>=1 && e2<2000){circulo(440+1*e2,1670,60);e2++;}
if(f2>=1 && f2<2000){circulo(470+1*f2,1710,60);f2++;}
if(g2>=1 && g2<2000){circulo(490+1*g2,1700,60);g2++;}
if(h2>=1 && h2<2000){circulo(480+1*h2,1680,60);h2++;}


if(x2==2000){ x2=1;}
if(y2==2000){ y2=1;}
if(z2==2000){ z2=1;}
if(w2==2000){ w2=1;}
if(e2==2000){ e2=1;}
if(f2==2000){ f2=1;}
if(g2==2000){ g2=1;}
if(h2==2000){ h2=1;}

//nube 3
glColor3f(1.0,1.0,1.0);
if(x3>=1 && x3<2000){circulo(600+1*x3,1800,60);x3++;}
if(y3>=1 && y3<2000){circulo(640+1*y3,1790,60);y3++;}
if(z3>=1 && z3<2000){circulo(620+1*z3,1790,60);z3++;}
if(w3>=1 && w3<2000){circulo(640+1*w3,1810,60);w3++;}
if(e3>=1 && e3<2000){circulo(640+1*e3,1770,60);e3++;}
if(f3>=1 && f3<2000){circulo(670+1*f3,1810,60);f3++;}
if(g3>=1 && g3<2000){circulo(690+1*g3,1800,60);g3++;}
if(h3>=1 && h3<2000){circulo(680+1*h3,1780,60);h3++;}


if(x3==2000){ x3=1;}
if(y3==2000){ y3=1;}
if(z3==2000){ z3=1;}
if(w3==2000){ w3=1;}
```

```
if(e3==2000){ e3=1;}
if(f3==2000){ f3=1;}
if(g3==2000){ g3=1;}
if(h3==2000){ h3=1;}

//nube 4
glColor3f(1.0,1.0,1.0);
if(x4>=1 && x4<2000){circulo(820+1*x4,1900,60);x4++;}
if(y4>=1 && y4<2000){circulo(850+1*y4,1890,60);y4++;}
if(z4>=1 && z4<2000){circulo(840+1*z4,1890,60);z4++;}
if(w4>=1 && w4<2000){circulo(860+1*w4,1910,60);w4++;}
if(e4>=1 && e4<2000){circulo(810+1*e4,1870,60);e4++;}
if(f4>=1 && f4<2000){circulo(880+1*f4,1910,60);f4++;}
if(g4>=1 && g4<2000){circulo(890+1*g4,1900,60);g4++;}
if(h4>=1 && h4<2000){circulo(887+1*h4,1880,60);h4++;}


if(x4==2000){ x4=1;}
if(y4==2000){ y4=1;}
if(z4==2000){ z4=1;}
if(w4==2000){ w4=1;}
if(e4==2000){ e4=1;}
if(f4==2000){ f4=1;}
if(g4==2000){ g4=1;}
if(h4==2000){ h4=1;}

//Nube 5
glColor3f(1.0,1.0,1.0);
if(x5>=1 && x5<2000){circulo(700+1*x5,1600,60);x5++;}
if(y5>=1 && y5<2000){circulo(740+1*y5,1590,60);y5++;}
if(z5>=1 && z5<2000){circulo(720+1*z5,1590,60);z5++;}
if(w5>=1 && w5<2000){circulo(740+1*w5,1610,60);w5++;}
if(e5>=1 && e5<2000){circulo(740+1*e5,1570,60);e5++;}
if(f5>=1 && f5<2000){circulo(770+1*f5,1610,60);f5++;}
if(g5>=1 && g5<2000){circulo(790+1*g5,1600,60);g5++;}
if(h5>=1 && h5<2000){circulo(780+1*h5,1580,60);h5++;}


if(x5==2000){ x5=1;}
if(y5==2000){ y5=1;}
if(z5==2000){ z5=1;}
if(w5==2000){ w5=1;}
if(e5==2000){ e5=1;}
if(f5==2000){ f5=1;}
if(g5==2000){ g5=1;}
if(h5==2000){ h5=1;}
```

```
//----------------------------Barandal Horizontal----------------------------
glColor3f(1.0,0.0,0.0);
glBegin(GL_QUADS);
glVertex2f(0,680);
glVertex2f(0,640);
glVertex2f(2000,640);
glVertex2f(2000,680);
glEnd();


//-------------------------Techo-----------------------------------------------
glColor3f(0.8 , 0.6, 0.0);
glBegin(GL_POLYGON);
glVertex2f(920,1200);
glVertex2f(840,1280);
glVertex2f(400,1280);
glVertex2f(320,1200);
glEnd();
//--------------------Cuerpo de casa-----------------------------------------

glColor3f(1.0 , 1.0, 1.0);
glBegin(GL_QUADS);
glVertex2f(360,1200);
glVertex2f(880,1200);
glColor3f(0.0 , 0.0, 1.0);
glVertex2f(880,600);
glVertex2f(360,600);
glEnd();

glColor3f(1.0 , 1.0, 1.0);
glBegin(GL_QUADS);
glVertex2f(360,1120);
glVertex2f(360,1000);
glColor3f(0.0 , 0.0, 1.0);
glVertex2f(480,1000);
glVertex2f(480,1120);
glEnd();

glColor3f(1.0 , 1.0, 1.0);
glBegin(GL_QUADS);
glVertex2f(760,1120);
glVertex2f(760,1000);
glColor3f(0.0 , 0.0, 1.0);
glVertex2f(880,1000);
```

```
glVertex2f(880,1120);
glEnd();
//-------------------Linea Cuerpo Izquierda y Derecha--------------------------

glColor3f(0.0 , 0.0, 1.0);
glBegin(GL_QUADS);
glVertex2f(480,1160);
glVertex2f(480,600);
glVertex2f(520,600);
glVertex2f(520,1160);
glEnd();

glColor3f(0.0 , 0.0, 1.0);
glBegin(GL_QUADS);
glVertex2f(720,1160);
glVertex2f(720,600);
glVertex2f(760,600);
glVertex2f(760,1160);
glEnd();

//-------------------------Triangulo Techo-------------------------------------
glColor3f(0.8 , 0.6, 0.0);
glBegin(GL_TRIANGLES);
glVertex2f(480,1160);
glVertex2f(600,1240);
glVertex2f(760,1160);
glEnd();
//-----------------Ventanas-------------------------------------------------
glColor3f(0.0, 1.0,1.0);
glBegin(GL_QUADS);
glVertex2f(520,1160);
glVertex2f(520,1000);
glColor3f(1.0 , 1.0,1.0);
glVertex2f(720,1000);
glColor3f(0.0 ,1.0,1.0);
glVertex2f(720,1160);
glEnd();

glColor3f(1.0, 1.0,1.0);
glBegin(GL_QUADS);
glVertex2f(600,1160);
glVertex2f(600,1000);
glVertex2f(640,1000);
glVertex2f(640,1160);
glEnd();
```

```
//-----------------------Cuadrado debajo de Ventanas-----------------------------------------
glColor3f(0.8 , 0.6, 0.0);
glBegin(GL_QUADS);
glVertex2f(520,1000);
glVertex2f(520,920);
glVertex2f(720,920);
glVertex2f(720,1000);
glEnd();

//----------------------Linea horizontal cuerpo casa---------------------------

glColor3f(0.8 , 0.6, 0.0);
glBegin(GL_QUADS);
glVertex2f(360,920);
glVertex2f(360,880);
glVertex2f(880,880);
glVertex2f(880,920);
glEnd();

//-------------------------Ventana izquierda abajo-----------------------------
glColor3f(0.0,1.0,1.0);
glBegin(GL_QUADS);
glVertex2f(360,800);
glVertex2f(360,680);
glVertex2f(480,680);
glVertex2f(480,800);
glEnd();
//---------------------Ventana derecha abajo-----------------------------------
glColor3f(0.0,1.0,1.0);
glBegin(GL_QUADS);
glVertex2f(760,800);
glVertex2f(760,680);
glVertex2f(880,680);
glVertex2f(880,800);
glEnd();
//--------------------------------Puerta---------------------------------------
glColor3f(0.6,0.4,0.2);
glBegin(GL_QUADS);
glVertex2f(520,840);
glVertex2f(520,600);
glVertex2f(720,600);
glVertex2f(720,840);
glEnd();
//-------------------------Dentro de puerta-----------------------------------
glColor3f(0.0,1.0,1.0);
glBegin(GL_QUADS);
```

```
glVertex2f(560,800);
glVertex2f(560,640);
glVertex2f(640,640);
glVertex2f(640,800);
glEnd();
//------------------------------Manija----------------------------------------
glColor3f(0.5,0.5,0.5);
glBegin(GL_QUADS);
glVertex2f(640,720);
glVertex2f(640,680);
glVertex2f(680,680);
glVertex2f(680,720);
glEnd();

//------------------------------Cochera----------------------------------------

glColor3f(0.8 , 0.6, 0.0);
glBegin(GL_POLYGON);
glVertex2f(360,880);
glVertex2f(80,880);
glVertex2f(40,800);
glVertex2f(360,800);
glEnd();
//----------------------Entrada Casa----------------------------------------
glColor3f(0.5 , 0.5, 0.5);
glBegin(GL_QUADS);
glVertex2f(520,320);
glVertex2f(520,600);
glVertex2f(720,600);
glVertex2f(720,320);
glEnd();
//------------------------------Banqueta----------------------------------------
glColor3f(0.5 , 0.5, 0.5);
glBegin(GL_QUADS);
glVertex2f(0,320);
glVertex2f(0,280);
glVertex2f(2000,280);
glVertex2f(2000,320);
glEnd();


//--------------------------Cochera----------------------------------

glColor3f(0.0,0.0,1.0);
glBegin(GL_POLYGON);
glVertex2f(80,800);
```

```
glVertex2f(80,600);
glColor3f(1.0 , 1.0, 1.0);
glVertex2f(360,600);
glVertex2f(360,800);
glEnd();

//--------------------------------Carretera----------------------------------
glColor3f(0.0 , 0.0,0.0);
glBegin(GL_QUADS);
glVertex2f(0,240);
glVertex2f(0,0);
glVertex2f(2000,0);
glVertex2f(2000,240);
glEnd();
//-------------------------------Orilla Carretera--------------------------------
glColor3f(1.0 , 1.0,0.0);
glBegin(GL_QUADS);
glVertex2f(0,280);
glVertex2f(0,240);
glVertex2f(2000,240);
glVertex2f(2000,280);
glEnd();


//-------------------------Barras de carretera--------------------------------
glColor3f(1.0 , 1.0,1.0);
glBegin(GL_QUADS);
glVertex2f(40,160);
glVertex2f(40,120);
glVertex2f(440,120);
glVertex2f(440,160);

glEnd();
glColor3f(1.0 , 1.0,1.0);
glBegin(GL_QUADS);
glVertex2f(520,160);
glVertex2f(520,120);
glVertex2f(920,120);
glVertex2f(920,160);
glEnd();

glColor3f(1.0 , 1.0,1.0);
glBegin(GL_QUADS);
glVertex2f(1080,160);
glVertex2f(1080,120);
glVertex2f(1560,120);
```

```
glVertex2f(1560,160);
glEnd();

glColor3f(1.0 , 1.0,1.0);
glBegin(GL_QUADS);
glVertex2f(2000,160);
glVertex2f(2000,120);
glVertex2f(1650,120);
glVertex2f(1650,160);
glEnd();

//------------------------------Pasto----------------------------------------
glColor3f(0.0 , 1.0,0.0);
glBegin(GL_QUADS);
glVertex2f(0,600);
glVertex2f(0,320);
glVertex2f(520,320);
glVertex2f(520,600);
glEnd();
glColor3f(0.0 , 1.0,0.0);
glBegin(GL_QUADS);
glVertex2f(720,600);
glVertex2f(720,320);
glVertex2f(2000,320);
glVertex2f(2000,600);
glEnd();

//-----------------Divisiones de la casa---------------------------------------
glColor3f(0.8 , 0.6, 0.0);
glBegin(GL_QUADS);
glVertex2f(360,1080);
glVertex2f(360,1040);
glVertex2f(480,1040);
glVertex2f(480,1080);
glEnd();
glColor3f(0.8 , 0.6, 0.0);
glBegin(GL_QUADS);
glVertex2f(360,760);
glVertex2f(360,720);
glVertex2f(480,720);
glVertex2f(480,760);
glEnd();
glColor3f(0.8 , 0.6, 0.0);
glBegin(GL_QUADS);
glVertex2f(760,1080);
glVertex2f(760,1040);
```

```
glVertex2f(880,1040);
glVertex2f(880,1080);
glEnd();
glColor3f(0.8 , 0.6, 0.0);
glBegin(GL_QUADS);
glVertex2f(760,760);
glVertex2f(760,720);
glVertex2f(880,720);
glVertex2f(880,760);
glEnd();



//----------------------------Barandales Verticales---------------------------------------
glColor3f(1.0,0.0,0.0);
glBegin(GL_QUADS);
glVertex2f(0,720);
glVertex2f(0,600);
glVertex2f(40,600);
glVertex2f(40,720);
glEnd();
glColor3f(1.0,0.0,0.0);
glBegin(GL_QUADS);
glVertex2f(920,720);
glVertex2f(920,600);
glVertex2f(960,600);
glVertex2f(960,720);
glEnd();
glColor3f(1.0,0.0,0.0);
glBegin(GL_QUADS);
glVertex2f(1000,720);
glVertex2f(1000,600);
glVertex2f(1040,600);
glVertex2f(1040,720);
glEnd();
glColor3f(0.6,0.4,0.2);
glBegin(GL_QUADS);
glVertex2f(1080,720);
glVertex2f(1080,600);
glVertex2f(1240,600);
glVertex2f(1240,720);
glEnd();
//----------------------Barandal despues del Tronco----------------------------
glColor3f(1.0,0.0,0.0);
glBegin(GL_QUADS);
glVertex2f(1280,720);
```

```
glVertex2f(1280,600);
glVertex2f(1320,600);
glVertex2f(1320,720);
glEnd();
glColor3f(1.0,0.0,0.0);
glBegin(GL_QUADS);
glVertex2f(1360,720);
glVertex2f(1360,600);
glVertex2f(1400,600);
glVertex2f(1400,720);
glEnd();
glColor3f(1.0,0.0,0.0);
glBegin(GL_QUADS);
glVertex2f(1440,720);
glVertex2f(1440,600);
glVertex2f(1480,600);
glVertex2f(1480,720);
glEnd();
glColor3f(1.0,0.0,0.0);
glBegin(GL_QUADS);
glVertex2f(1520,720);
glVertex2f(1520,600);
glVertex2f(1560,600);
glVertex2f(1560,720);
glEnd();
glColor3f(1.0,0.0,0.0);
glBegin(GL_QUADS);
glVertex2f(1600,720);
glVertex2f(1600,600);
glVertex2f(1640,600);
glVertex2f(1640,720);
glEnd();

glColor3f(1.0,0.0,0.0);
glBegin(GL_QUADS);
glVertex2f(1680,720);
glVertex2f(1680,600);
glVertex2f(1720,600);
glVertex2f(1720,720);
glEnd();

glColor3f(1.0,0.0,0.0);
glBegin(GL_QUADS);
glVertex2f(1760,720);
glVertex2f(1760,600);
glVertex2f(1800,600);
```

```
glVertex2f(1800,720);
glEnd();

glColor3f(1.0,0.0,0.0);
glBegin(GL_QUADS);
glVertex2f(1840,720);
glVertex2f(1840,600);
glVertex2f(1880,600);
glVertex2f(1880,720);
glEnd();

glColor3f(1.0,0.0,0.0);
glBegin(GL_QUADS);
glVertex2f(1920,720);
glVertex2f(1920,600);
glVertex2f(1960,600);
glVertex2f(1960,720);
glEnd();

//--------------------------Troncos Arbol--------------------------------------
glColor3f(0.6,0.4,0.2);
glBegin(GL_QUADS);
glVertex2f(1120,720);
glVertex2f(1120,600);
glVertex2f(1160,600);
glVertex2f(1160,720);
glEnd();


glColor3f(0.6,0.4,0.2);
glBegin(GL_QUADS);
glVertex2f(1700,620);
glVertex2f(1700,400);
glVertex2f(1750,400);
glVertex2f(1750,620);
glEnd();
//-----------------------------Hojas Arbol-----------------------------------------
//glBegin(GL_LINES);

glColor3f(0.0,0.6,0.2);
glBegin(GL_QUADS);
glVertex2f(1000,760);
glVertex2f(1000,720);
glVertex2f(1400,720);
glVertex2f(1400,760);
glEnd();
```

```
glColor3f(0.0,0.6,0.2);
glBegin(GL_QUADS);
glVertex2f(1040,840);
glVertex2f(1040,800);
glVertex2f(1320,800);
glVertex2f(1320,840);
glEnd();
glColor3f(0.0,0.6,0.2);
glBegin(GL_QUADS);
glVertex2f(1080,920);
glVertex2f(1080,880);
glVertex2f(1280,880);
glVertex2f(1280,920);
glEnd();
glColor3f(0.0,0.6,0.2);
glBegin(GL_QUADS);
glVertex2f(1120,1000);
glVertex2f(1120,960);
glVertex2f(1240,960);
glVertex2f(1240,1000);
glEnd();
glColor3f(0.0,0.6,0.2);
glBegin(GL_QUADS);
glVertex2f(1160,1080);
glVertex2f(1150,1040);
glVertex2f(1200,1040);
glVertex2f(1200,1080);
glEnd();

//-------------------Arbol con Triangulos--------------------------------------
glColor3f(0.0,0.6,0.2);
glBegin(GL_TRIANGLES);
glVertex2f(1725,1100);
glVertex2f(1600,600);
glVertex2f(1850,600);
glEnd();

 //-------------------------Carro-----------------------------------

glColor3f(1.0,0.0,0.0);//rojo

glBegin(GL_POLYGON);
glVertex2f(400,440);
glVertex2f(240,440);
glVertex2f(160,360);
glVertex2f(480,360);
```

```
glColor3f(1.0 ,0.0,0.0);
glBegin(GL_QUADS);
glVertex2f(160,360);
glVertex2f(160,280);
glVertex2f(600,280);
glVertex2f(600,360);
glEnd();



//------------Llantas Auto ----------------------------------------------------

glColor3f(0.5,0.5,0.5);
circulo(240,240,50);
circulo(240,240,25);
glEnd();
glColor3f(0.5,0.5,0.5);
circulo(520,240,50);
circulo(520,240,25);
glEnd();


//----------LLANTAS-----------------------------------------
glColor3f(0.5,0.5,0.5);
if(x8>=1 && x8<1600){circulo(100+1*x8,50,40);x8++;}
if(x8==1600){ x8=1;}

}

glColor3f(0.5,0.5,0.5);
if(x9>=1 && x9<1600){circulo(280+1*x9,50,40);x9++;}
if(x9==1600){ x9=1;}

//------------CUERPO CARRO-------------------------

glColor3f(0.0,0.0,1.0);
if(x12>=1 && x12<1600){circulo(100+1*x12,140,60);x12++;}
if(x12==1600){ x12=1;}

glColor3f(0.0,0.0,1.0);
if(x10>=1 && x10<1600){circulo(180+1*x10,140,60);x10++;}
if(x10==1600){ x10=1;}

glColor3f(0.0,0.0,1.0);
if(x11>=1 && x11<1600){circulo(260+1*x11,140,60);x11++;}
if(x11==1600){ x11=1;}
```

```
 glFlush();
 glutSwapBuffers();                    //forza dibujo
      }

int main (int argc, char** argv)       //metodo main
{

 glutInit(&argc, argv);                  //incializa GLUT
 glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);    //establece el modo de visualización
 glutInitWindowSize(800,600);            //tamaño de la ventana
 glutInitWindowPosition(0,0);           //posicion inicial de la ventana
 glutCreateWindow("Examen2");            //nombre de la ventana
 inicializa();
 glutDisplayFunc(dibuja);
 glutIdleFunc(dibuja);                 //Envia los graficos a la ventana de visualización
 glutMainLoop();                        //muestra todo y espera
 return 0;                             //retorna un valor de cero
}
```

IMPRESIONES DE PANTALLA

CONCLUSIONES

Gracias a OpenGl aprendimos y dominamos un poco mas con cada practica que se hace en el curso de graficacion a manejar cada herramienta que este nos ofrece.

Sabemos que mediante el uso de primitivas podemos crear cualquier tipo de escenario ya sea 2D o 3D sin embargo se tiene en cuenta que aun no se ha explorado cada función, herramienta que en la que OpenGL nos proporciona.

El resultado de este proyecto fue el esperado ya que se cumplió con el objetivo descrito por el profesor y se uso la mayoría de primitivas para formar este escenario.

en 1

[Escriba el subtítulo del documento]

[Seleccione la fecha]
[Escriba el nombre de la compañía]
user

Introducción

Anteriormente tanto en clase como en trabajos ya se han manejado estos conceptos, (navegación por escenario, iluminación y texturizado). Por lo tanto hoy me he encomendado a la tarea de unir todo en un solo proyecto.

Objetivo

Generar un escenario 3D con texturas, movimiento tanto de cámara como del usuario así como iluminación.

CONTENIDO

Movimiento e interacción del usuario.

Para mayor facilidad de navegación del usuario se buscó una manera sencilla de interactuar con el escenario de manera que sea intuitivo. Se anexaron funciones como DollyCamera cuya función es seguir al usuario durante su desplazamiento por el escenario.

```
void dollyCamera( GLfloat dollyBy, GLfloat dollyAngle )
{
 GLfloat actualAngleInRadians;
 actualAngleInRadians = ( ( camYaw + dollyAngle ) * M_PI / 180.0 );
 camX -= sin( actualAngleInRadians ) * dollyBy * DEF_dollyStepSize;
 camZ -= cos( actualAngleInRadians ) * dollyBy * DEF_dollyStepSize;
}
```

Así como una función para que la cámara siga el desplazamiento del mouse.

```
void mouseFunc( int button, int state, int x, int y )
{
 if( button == GLUT_LEFT_BUTTON && state == GLUT_DOWN )
    {
    if( !isInMouseDrag )
        enterMouseDrag( x, y );
    else
        exitMouseDrag( x, y );
    }
}

void allMotionFunc( int x, int y )
{
 int deltaX, deltaY;

 if( !isInMouseDrag )
    return;
 deltaX = x - viewPortCenterX;
 deltaY = y - viewPortCenterY;
 if( deltaX == 0 && deltaY == 0 )
    return;
 glutWarpPointer( viewPortCenterX, viewPortCenterY );
 camYaw -= DEF_angleSensitivity * deltaX;
 camPitch -= DEF_angleSensitivity * deltaY * ( mouseIsInverted ? -1.0 : 1.0 );
 clampCamera();
 glutPostRedisplay();
```

Como extra para lograr un mayor ambiente se incluyó una función para agregar sonido.

```
mciSendString("open Datos/Sonido.mp3 type  sequencer", NULL,0, NULL);
mciSendString("play Datos/Sonido.mp3", NULL,0, NULL);
```

Texturización.

Para texturizar se utilizó la librería nativa de OpenGL "<GL/glaux.h>":

Preparación de las texturas:

```
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_MODULATE);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_MODULATE);

glTexEnvf ( GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE );
```

Cargando las texturas:

```
glPolygonMode(GL_FRONT_AND_BACK,GL_FILL);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef (0.0, 0.0, -5.0);
glPixelStorei(GL_UNPACK_ALIGNMENT,1);
glGenTextures(20,texture_id);
glBindTexture(GL_TEXTURE_2D,texture_id[0]);
LoadBMP("bien.bmp",0);
glBindTexture(GL_TEXTURE_2D,texture_id[1]);
LoadBMP("piso2.bmp",0);
glBindTexture(GL_TEXTURE_2D,texture_id[2]);
LoadBMP("piso2.bmp",0);
glBindTexture(GL_TEXTURE_2D,texture_id[3]);
LoadBMP("paredexterna.bmp",0);
glBindTexture(GL_TEXTURE_2D,texture_id[4]);
LoadBMP("S.bmp",0);
```

Iluminación.

Se definen los parámetros de la iluminación a utilizar.

```
float LightPos0[] = { 0.75f, 1.0f, 1.0f, 0.0f};
float LightAmb0[] = { 0.75f, 0.75f, 0.75f, 1.0f};
float LightDif0[] = { 1.0f, 1.0f, 1.0f, 1.0f};
float LightSpc0[] = { 1.0f, 1.0f, 1.0f, 1.0f};
```

Se habilitan las luces mandándolas a llamar.

```
 glLightfv(GL_LIGHT0, GL_POSITION, LightPos0);
glLightfv(GL_LIGHT0, GL_AMBIENT, LightAmb0);
glLightfv(GL_LIGHT0, GL_DIFFUSE, LightDif0);
glLightfv(GL_LIGHT0, GL_SPECULAR, LightSpc0);
glEnable(GL_LIGHT0);
glEnable(GL_LIGHTING);
```

Ejemplo del manejo del escenario.

```
glBindTexture(GL_TEXTURE_2D, texture_id[10]);
glBegin(GL_QUADS);
        glTexCoord2f(0, 1);
        glVertex3f(115.0f, 100.0f, 31.0f+400.0f);
        glTexCoord2f(0, 0);
        glVertex3f(115.0f, 40.0f, 31.0f+400.0f);
        glTexCoord2f(1, 0);
        glVertex3f(155.0f, 40.0f, 31.0f+400.0f);
        glTexCoord2f(1, 1);
        glVertex3f(155.0f, 100.0f, 31.0f+400.0f);
glEnd();
```

En la primera línea podemos ver que se indica que se utilizarán texturas y qué textura se utilizará. En la tercera vemos que se llaman las coordenadas de la textura.

Graficación

Proyecto Final

Andrea Estephany Sánchez Hernández 201225072

Iván Olmos Pineda

Primavera 2015

Contenido

Introducción

Para realizar este proyecto utilizaremos: esferas, conos, carga de texturas, rotación y traslación, movimiento de observador e iluminación.

Conceptos Desarrollados

Texturas

La carga de textura se refiere a cargar una imagen en un polígono con el fin de personalizar el polígono lo cual nos permitirá obtener imágenes mucho más realistas.

Para aplicar una textura a un objeto 2D deberemos seguir los siguientes pasos:

1.- Creación de la textura

2.- Definimos las condiciones en las que se va a aplicar la textura

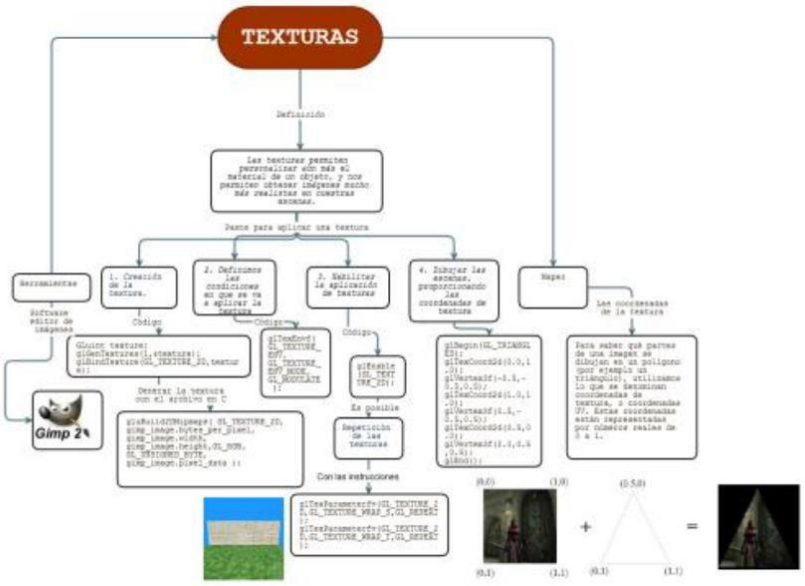
3.- Habilitar la aplicación de texturas

4.- Dibujar las escenas, proporcionando las coordenadas de textura.

Las coordenadas de textura

Para saber qué partes de una imagen se dibujan en un polígono (por ejemplo un triángulo), utilizamos lo que se denominan coordenadas de textura, o coordenadas UV. Estas coordenadas están representadas por números reales de 0 a 1.

A continuación se muestra un diagrama el cual representa los pasos a seguir para cargar texturas:


TEXTURAS
Definición
Las texturas permiten personalizar aún más el material de un objeto, y nos permiten obtener imágenes mucho más realistas en nuestras escenas.
Pasos para aplicar una textura
Herramientas
1. Creación de la textura.
2. Definimos las condiciones en que se va a aplicar la textura
3. Habilitar la aplicación de texturas
4. Dibujar las escenas, proporcionando las coordenadas de textura
Mapas
Las coordenadas de la textura
Software editor de imágenes
Código
Código
Código
Gimp 2
Con las instrucciones
Es posible
Repetición de las texturas
(0,0)
(1,0)
(0.5,0)
(0,1)
(1,1)
(0,1)
(1,1)

DE LA TEXTURA DEL DIAGRAMA TENEMOS LA TEXTURA:

*Crear textura:
Int texture;

*Condiciones de textura

Se define la cantidad de texturas que se manejaran

#define NTextures 2 GLuint texture [NTextures];
Variables para manejo de texturas

Char *texturefiles [] = {

"pastito.bmp","canasta.bmp"

};

*Activar una textura:

GlBindTexture (GL_TEXTURE_2D, texture);
GlEnable (GL_TEXTURE_2D);

GlBegin (GLTRIANGLES);

glTextCoord2d (0.0, 1.0);
glVertex3f (-0.5,-0.5, 0.5);
glTexCoord2d (1.0, 1.0);
glVertex3f (0.5,-0.5, 0.5);
glTexCoord2d (0.5, 1.0);
glVertex3f (0.0, 0.5, 0.5);

Proyección Perspectiva

La proyección perspectiva delimita un volumen de visualización dado por un ángulo de cámara, y una relación alto/ancho. La distancia al observador determinará el tamaño con el que un objeto se visualiza.

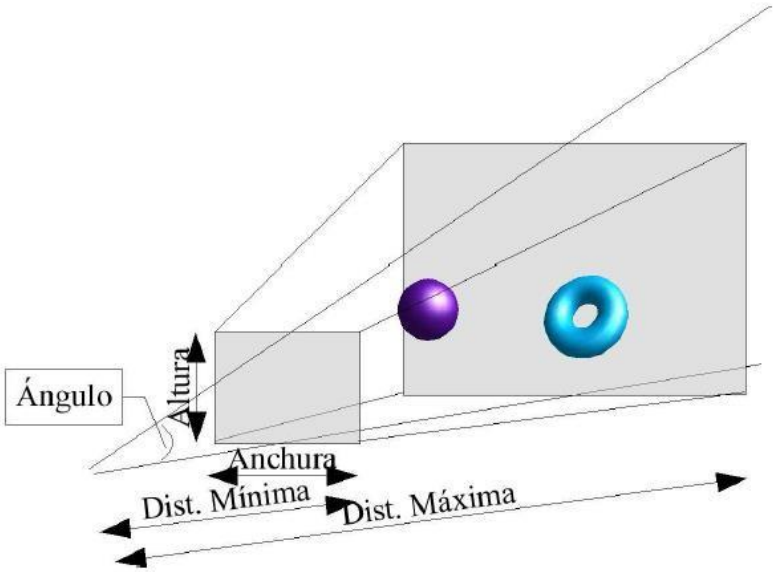


Figura 3.7. Volúmen de visualización en perspectiva

Utilizando esta proyección incluiremos también el uso del teclado para la interacción con el observador. La función que nos permite registrar este

evento es *glutSpecialFunc*, que se encarga de manejar la pulsación de teclas "especiales", como los cursores.

La declaración de esta función es: glutSpecialFunc (void (*func) (int key, int x, int y)); Es decir, recibe un puntero a función, que no devuelve nada, y recibe tres enteros: la tecla pulsada, y la posición X e Y del ratón en la pantalla. La rotación alrededor de los ejes X e Y la vamos a controlar mediante dos variables globales (*rot_angle_x*, *rot_angle_y*).

Para ello utilizaremos la función *glutKeyboardFunc ()*, que recibirá como parámetro un puntero a la siguiente función:

```
Static void keys (unsigned char key, int x, int y)
{
Switch (key)
{
case 27: exit(0);
break;
}
glutPostRedisplay();
}
```

Uso de glPushMatrix()

La función glPushMatrix() realiza una copia de la matriz superior y la pone encima de la pila, de tal forma que las dos matrices superiores son iguales. En la figura 1 se observa la pila en la situación inicial con una sola matriz, al llamar a la función glPushMatrix() se duplica la matriz superior. Las siguientes transformaciones que se realizan se aplican sólo a la matriz superior de la pila, quedando la anterior con los valores que tenía en el momento de llamar a la función glPushMatrix().

Uso de glPopMatrix()

La función glPopMatrix() elimina la matriz superior, quedando en la parte superior de la pila la matriz que estaba en el momento de llamar a la función glPushMatrix().

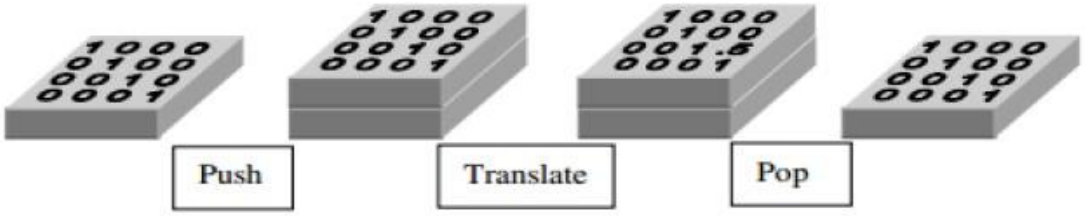


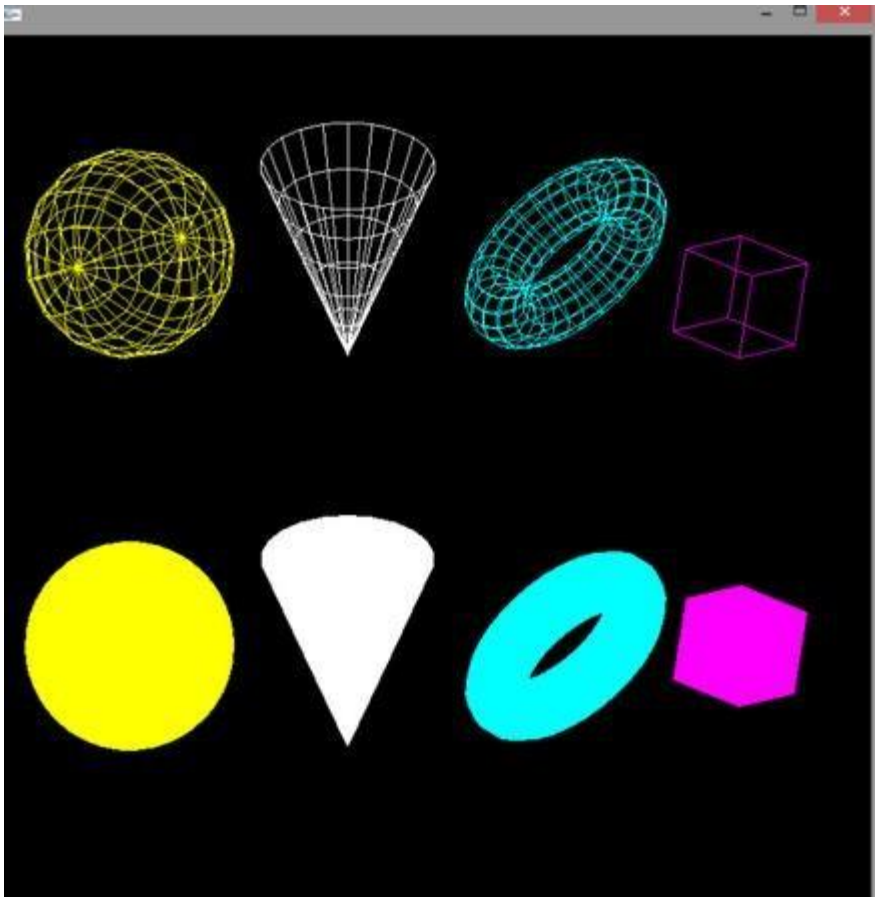
Figura 1 PushMatrix y PopMatrix

En la función display() al llamar a la función glPushMatrix() se realiza una copia de la matriz actual. La traslación en el eje Z se realiza en la matriz superior de la pila, es decir, en la copia de la matriz, de tal forma que al llamar a la función glPopMatrix(), como se muestra en la figura 1, se elimina la matriz superior, que es la que tenía el efecto de esta transformación, quedando la matriz que estaba en el momento de llamar

a glPushMatrix(). Al descomentar las llamadas a las funciones glPushMatrix() y glPopMatrix() las transformaciones realizadas entre ambas no afectan al resto de la aplicación.

Esferas y conos

OpenGL ofrece distintas funciones para crear figuras básicas entre ellas se encuentran las esferas y los conos. Las figuras pueden ser tipo 'wire' o 'rellenas'. En la imagen que se encuentra a continuación, las imágenes de la parte superior corresponden al tipo 'wire' y las inferiores son tipo 'rellenas'



Esfera

Para crear una esfera, se utiliza las siguientes líneas.

Tipo wire: glutWireSphere(radio,rebanadas,stacks)

Tipo rellena: glutSolidSphere(radio,rebanadas,stacks)

Donde:

Radio: Corresponde al radio de la esfera.

Rebanadas: El número de subdivisiones alrededor del eje z.

Stacks: El número de subdivisiones en el eje z.

Cono

Para crear un cono, utilizamos las siguientes líneas: Tipo

wire: glutWireCone(base,altura,rebanadas,stacks) Tipo

rellena: glutSolidCone(base,altura,rebanadas,stacks)

Donde:

Base: es el radio de la base del cono.

Altura: es la altura del cono.

Rebanadas: El número de subdivisiones alrededor del eje z.

Stacks: El número de subdivisiones en el eje z.

En estas figuras podemos hacer uso de distintas funciones, como las que se mencionaron en trabajos anteriores (Trasladar, rotas, escalar…etc.).

Iluminación

La iluminación de OpenGL se basa en luces y materiales. Una luz es una fuente de iluminación para la escena. Emite un haz de luz de un color determinado, dividido en las tres componentes de color RGB. Un material determina la cantidad de cada color que refleja un objeto determinado.

Luces

Como se ha dicho, una luz aporta iluminación a una escena, según unas determinadas componentes de color. A partir de ahora distinguiremos entre fuente de luz, como entidad que proporciona luz a una escena, y luz, como aportación de esa fuente a la iluminación de la escena. Una fuente puede emitir tipos diferentes de luz, que son complementarias, y pueden darse a la vez para un cierto tipo de luz.

Los tipos de luz son:

*"Emitted" (emitida): es la luz emitida por un objeto. No se ve afectada por ningún otro tipo de luz. Por ejemplo, un fuego emite una determinada luz, y si lo miramos, lo veremos de ese color, independientemente del color de las luces que estén apuntando al fuego.

* "Diffuse" (difusa): es la luz que índice sobre un objeto, y proviene de un determinado punto. La intensidad con la que se refleja en la superficie del objeto puede depender del ángulo de incidencia, dirección, etc. Una vez incide sobre un objeto se refleja en todas direcciones.

* "Specular" (especular): es la luz que, al incidir sobre un objeto, se ve reflejada con un ángulo similar al de incidencia. Podemos pensar que es la luz que produce los brillos.

* "Ambient" (ambiental): podemos considerarlo como los restos de luz que aparecen definido a la reflexión residual de una luz que ya se ha reflejado sobre muchos objetos, y es imposible determinar su procedencia. Es algo así como la iluminación global de una escena.

Materiales

Un material define, para un determinado objeto, qué componentes refleja de cada tipo de luz.

Normales

La iluminación en OpenGL se calcula a nivel de vértice, es decir, por cada vértice, se calcula su color a partir del material activo, las luces activas, y cómo estas luces inciden sobre el vértice. Para saber cómo incide una luz sobre un vértice, empleamos la normal del vértice, un vector, que, generalmente, será perpendicular a la cara que estemos dibujando, aunque podremos variarlo para conseguir distintos efectos. En la figura, podemos ver cómo incide la misma luz en

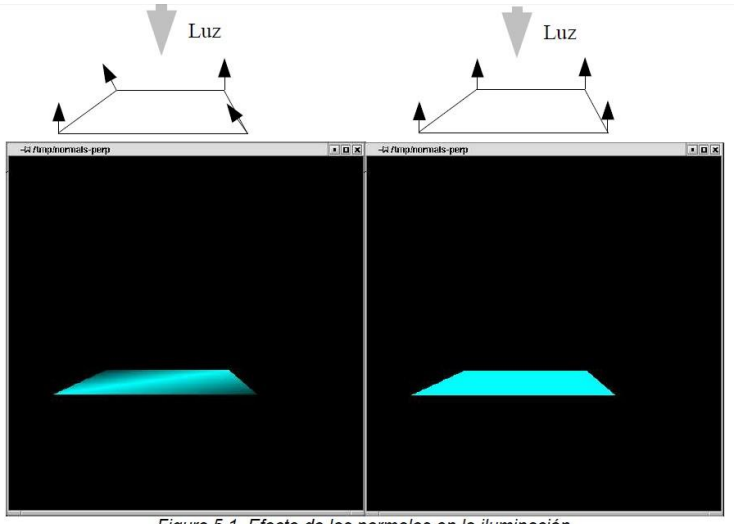


Figura 5.1. Efecto de las normales en la iluminación

el mismo objeto, pero con las normales cambiadas.

El primer paso para utilizar la iluminación en una aplicación OpenGL es activarla, mediante la llamada glEnable(GL_LINGHTING); La especificación inicial de OpenGL comtempla que, al menos, cada implementación debe de poder definir 8 luces, identificadas por las constantes GL_LIGHTn, dónde 'n' es el número de la luz, comenzando por

0. Para establecer las propiedades de una luz utilizaremos llamadas a las

funciones del tipo glLight*(). Las propiedades de toda luz son las siguientes:

Posición/Dirección:

Indica la posición/dirección de la luz, y especifica si ésta es una luz posicional o direccional. Una luz posicional tiene una posición concreta en el espacio, mientras que una luz direccional consiste en un conjunto de haces de luz paralelos.

Figura 5.2. Tipos de luces

Para establecer esta propiedad utilizaremos la llamada: glLightfv(GL_LIGHTn,GL_POSITION,val_ptr); "val_prt" es un puntero a un vector de cuatro componentes de tipo float, de la forma (x,y,z,w). En el caso de que w sea 1, estaremos ante una luz posicional, y su posición está determinada por (x,y,z). Si w es 0, la luz es direccional, y su dirección es el vector (x,y,z).

Dirección del foco:

En el caso de una luz focal, debemos establecer su dirección. Esto lo haremos con la llamada: glLightfv(GL_LIGHTn,GL_SPOT_DIRECTION,val_prt);

val_ptr" es un puntero a un vector con la dirección, en formato (x,y,z).

Apertura del foco:

El ángulo de apertura del foco se define mediante: glLightf(GL_LIGHTn,GL_SPOT_CUTOFF,val); "val" expresa en grados la mitad del ángulo de apertura del foco.

Atenuación del foco:

La atenuación del foco (degradación de la intensidad a medida que nos acercamos al borde) se define mediante: glLightf(GL_LIGHTn,GL_SPOT_EXPONENT,val);

Intensidad de la luz:

Define el color ambiental, difuso y especular de la luz. Se define mediante la llamada: glLightfv(GL_LIGHTn,GL_[AMBIENT|DIFFUSE|SPECULAR],val_ptr);

"val_ptr" es un puntero a un vector de cuatro componentes de color RGBA.

Una vez establecidas las propiedades de una determinada luz, las activaremos con la llamada: glEnable(GL_LIGHTn);

Código

//Colocamos las librerías

```
#include <GL/gl.h>

#include<stdlib.h>

#include <iostream>

#include <math.h>

#include <GL/glut.h>

#include <stdlib.h>

#include "FreeImage.h"

#include <mmsystem.h>

int hazPerspectiva=0;
```

//se define la cantidad de texturas que se manejaran

```
#define NTextures 2

GLuint       texture[NTextures];
```

//variables para manejo de texturas

```
char *texturefiles[] = {
        "pasto.bmp","camino.bmp"

};
```

//Se definen variables que nos ayudaran para el movimiento

```
float x=0;
float y=0;
float z=0;
float x2=0;
float y2=0;
float z2=0;
float xc=0;
float yc=0;
float zc=0;
```

```
float mov_pajaro=0;
float rotacion_pajaro=0;
float dist=0;

GLint ancho=100;
GLint alto=100;
```

//Definimos los vectores de color, posición y dirección para la iluminación

```
float light_color [] = {1.0,1.0,1.0,1.0};
float light_pos [] = {1.0,1.0,1.0,0.0};
float spot_dir [] = {0.0,0.0,-1.0};

float mata [] = {1.0,1.0,1.0,1.0};

float matd [] = {0.4,0.4,0.4};

float mats [] = {0.774597};
```

//Esta función sirve para configurar nuestra textura la cual fue guardada en un arreglo texturefiles[] readImage es para la textura 1 y readImage para la textura 0

```
bool readImage()

    //image format

    FREE_IMAGE_FORMAT fif = FIF_UNKNOWN;

    //pointer to the image, once loaded

    FIBITMAP *dib(0);
```

```
//pointer to the image data

BYTE* bits(0);

//image width and height
unsigned int width(0), height(0);

//check the file signature and deduce its format
fif = FreeImage_GetFileType(texturefiles[1], 0);

//if still unknown, try to guess the file format from the file extension

if(fif == FIF_UNKNOWN)
```

```
        fif = FreeImage_GetFIFFromFilename(texturefiles[1]);

//if still unkown, return failure
if(fif == FIF_UNKNOWN)
        return false;



//check that the plugin has reading capabilities and load the file
if(FreeImage_FIFSupportsReading(fif))
        dib = FreeImage_Load(fif, texturefiles[1]);

//if the image failed to load, return failure
if(!dib)
        return false;

// ** AGREGADO ** (convierto la imagen a una de 24bits) //

dib = FreeImage_ConvertTo24Bits(dib);

//retrieve the image data

bits = FreeImage_GetBits(dib);

//get the image width and height
width = FreeImage_GetWidth(dib);
height = FreeImage_GetHeight(dib);

//if this somehow one of these failed (they shouldn't), return failure
if((bits == 0) || (width == 0) || (height == 0))
```

```
        return false;


//generate an OpenGL texture ID for this texture
glGenTextures(1, &texture[1]);

//glGenTextures(1, &texture[1]);

//bind to the new texture ID

glBindTexture(GL_TEXTURE_2D, texture[1]);

//glBindTexture(GL_TEXTURE_2D, texture[1]);

//store the texture data for OpenGL use
```

```
	// ** AGREGADO ** (filtros para la textura y condicional de mipmap) //

	glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S, GL_REPEAT);

	glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T, GL_REPEAT);

	glTexParameteri(GL_TEXTURE_2D,	GL_TEXTURE_MAG_FILTER, GL_LINEAR);

	glTexParameteri(GL_TEXTURE_2D,	GL_TEXTURE_MIN_FILTER, GL_LINEAR);

	glTexImage2D(GL_TEXTURE_2D, 0, 3, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, bits);



	//Free FreeImage's copy of the data

	FreeImage_Unload(dib);



	//return success

	return true;



}

bool readImage2()
```

```
{

    //image format

    FREE_IMAGE_FORMAT fif = FIF_UNKNOWN;

    //pointer to the image, once loaded

    FIBITMAP *dib(0);

    //pointer to the image data

    BYTE* bits(0);

    //image width and height
    unsigned int width(0), height(0);

    //check the file signature and deduce its format
```

```
fif = FreeImage_GetFileType(texturefiles[0], 0);

//if still unknown, try to guess the file format from the file extension
if(fif == FIF_UNKNOWN)
        fif = FreeImage_GetFIFFromFilename(texturefiles[1]);

//if still unkown, return failure
if(fif == FIF_UNKNOWN)
        return false;



//check that the plugin has reading capabilities and load the file
if(FreeImage_FIFSupportsReading(fif))
        dib = FreeImage_Load(fif, texturefiles[0]);

//if the image failed to load, return failure
if(!dib)
        return false;

// ** AGREGADO ** (convierto la imagen a una de 24bits) //

dib = FreeImage_ConvertTo24Bits(dib);

//retrieve the image data

bits = FreeImage_GetBits(dib);
```

```
//get the image width and height
width = FreeImage_GetWidth(dib);
height = FreeImage_GetHeight(dib);

//if this somehow one of these failed (they shouldn't), return failure
if((bits == 0) || (width == 0) || (height == 0))
        return false;



//generate an OpenGL texture ID for this texture
glGenTextures(1, &texture[0]);

//glGenTextures(1, &texture[1]);

//bind to the new texture ID
```

```
	glBindTexture(GL_TEXTURE_2D, texture[0]);

	//glBindTexture(GL_TEXTURE_2D, texture[1]);

	//store the texture data for OpenGL use

	// ** AGREGADO ** (filtros para la textura y condicional de mipmap) //

	glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S, GL_REPEAT);

	glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T, GL_REPEAT);

	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);

	glTexImage2D(GL_TEXTURE_2D, 0, 3, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, bits);

		//Free FreeImage's copy of the data

	FreeImage_Unload(dib);

//return success
	return true;

}
```

//En esta función iniciamos la configuración de la pantalla, el color de fondo, las texturas y la iluminación

```
void init(void)

{

    glClearColor(0.22,0.69,0.87,0);

    glColor3f(1., 0., 1.);



    glEnable(GL_DEPTH_TEST);

    readImage(); readImage2();
    glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);
    glLightfv(GL_LIGHT3,GL_POSITION,light_color);
```

```
    glLightfv(GL_LIGHT3,GL_AMBIENT,light_color);

    glEnable(GL_LIGHT3);
    glEnable(GL_LIGHTING);
    glEnable(GL_COLOR_MATERIAL);
}

void reshape(int width, int height)

{

    glViewport(0, 0, (GLsizei) width, (GLsizei) height);

    glMatrixMode(GL_PROJECTION);

    glLoadIdentity();

    gluPerspective(50., (GLfloat) width / height, 0.01, 1000.);

    glMatrixMode(GL_MODELVIEW);

    glLoadIdentity();

    gluLookAt(0, 0, 50, 0, 0, -10, 0, 1, 0);

}
```

//Con esta función manejamos el movimiento de las aves

```
static void timer(int x)

 {
```

```
    rotacion_pajaro+=5;

    glutTimerFunc(20, timer,0);

}
```

//Creamos nuestro auto a base de polígonos y donas

```
void automovil(float x, float y, float z){
```

//AUTO

//lado izquierdo

```
    /*glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, mata);
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, matd);
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, mats);
    glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 0.6 *
128.0);*/
```

```
glBegin(GL_POLYGON);

glColor3f(0., 0., 1.);

glVertex3f((-10.+x)*-1, (2.+y)*-1, -5.5+z);
glVertex3f((0.+x)*-1, (2.+y)*-1, -5.5+z);
glVertex3f((10.+x)*-1, (6.+y)*-1, -5.5+z);
glVertex3f((15.+x)*-1, (6.+y)*-1, -5.5+z);
glVertex3f((15.+x)*-1, (10.+y)*-1, -5.5+z);
glVertex3f((10.+x)*-1, (10.+y)*-1, -5.5+z);
glVertex3f((-10.+x)*-1, (10.+y)*-1, -5.5+z);
glVertex3f((-19.+x)*-1, (10.+y)*-1, -5.5+z);
glVertex3f((-19.+x)*-1, (6.+y)*-1, -5.5+z);
glVertex3f((-16.+x)*-1, (6.+y)*-1, -5.5+z);
glEnd();
```

//techo

```
glBegin(GL_POLYGON);
glColor3f(0, 0., 1.);
glVertex3f((0+x)*-1, (2.+y)*-1, 5.5+z);

glVertex3f((7+x)*-1, (4.8+y)*-1, 5.5+z);
glVertex3f((15.+x)*-1, (6.+y)*-1, 5.5+z);
glVertex3f((15+x)*-1, (6+y)*-1, -5.5+z);
glVertex3f((7+x)*-1, (4.8+y)*-1, -5.5+z);
glVertex3f((0+x)*-1, (2+y)*-1, -5.5+z);
glEnd();
glBegin(GL_QUADS);
```

```
glColor3f(0,0,1);

glVertex3f((0+x)*-1,(2+y)*-1,5.5+z);
glVertex3f((-10+x)*-1,(2+y)*-1,5.5+z);
glVertex3f((-10+x)*-1,(2+y)*-1,-5.5+z);
```

```
glVertex3f((0+x)*-1,(2+y)*-1,-5.5+z);


glBegin(GL_POLYGON);

glColor3f(0,0,1);

glVertex3f((-10+x)*-1, (2.+y)*-1, -5.5+z);
glVertex3f((-19.+x)*-1, (6.+y)*-1, -5.5+z);
glVertex3f((-19.+x)*-1, (6.+y)*-1, 5.5+z);
glVertex3f((-10+x)*-1, (2.+y)*-1, 5.5+z);
glEnd();
```

//lado derecho

```
glBegin(GL_POLYGON);
glColor3f(0, 0., 1.);

glVertex3f((-10.+x)*-1, (2.+y)*-1, 5.5+z);
glVertex3f((0.+x)*-1, (2.+y)*-1, 5.5+z);
glVertex3f((10.+x)*-1, (6.+y)*-1, 5.5+z);
glVertex3f((15.+x)*-1, (6.+y)*-1, 5.5+z);
glVertex3f((15.+x)*-1, (10.+y)*-1, 5.5+z);
glVertex3f((10.+x)*-1, (10.+y)*-1, 5.5+z);
glVertex3f((-10.+x)*-1, (10.+y)*-1, 5.5+z);
glVertex3f((-19.+x)*-1, (10.+y)*-1, 5.5+z);
glVertex3f((-19.+x)*-1, (6.+y)*-1, 5.5+z);
glVertex3f((-16.+x)*-1, (6.+y)*-1, 5.5+z);
glEnd();
```

```
glVertex3f((0+x)*-1,(2+y)*-1,-5.5+z);

//frente
glBegin(GL_POLYGON);
glColor3f(0, 0, 1);
glVertex3f((15+x)*-1, (10+y)*-1, -5.5+z);
```

```
glVertex3f((15+x)*-1, (10+y)*-1, 5.5+z);
glVertex3f((15+x)*-1, (6+y)*-1, 5.5+z);
glVertex3f((15+x)*-1, (6+y)*-1, -5.5+z);
glEnd();
```

//atras

```
glBegin(GL_POLYGON);
glColor3f(0, 0, 1);
glVertex3f((-19+x)*-1, (10+y)*-1, -5.5+z);
glVertex3f((-19+x)*-1, (10+y)*-1, 5.5+z);
glVertex3f((-19+x)*-1, (6+y)*-1, 5.5+z);
glVertex3f((-19+x)*-1, (6+y)*-1, -5.5+z);
glEnd();
```

//ventana frontal

```
glBegin(GL_QUADS);
glColor3f(0,0,0);
glVertex3f((6+x)*-1, (2+2+y)*-1, -5.+z);
glVertex3f((6+x)*-1, (2+1.6+y)*-1, 5.+z);
glVertex3f((1+x)*-1, (-1+3.2+y)*-1, 5.+z);
glVertex3f((1+x)*-1, (-1+3.2+y)*-1, -5.+z);
glEnd();
```

//ventana lateral derecha 1

```
glBegin(GL_QUADS);
glColor3f(0,0,0);
```

```
glVertex3f(5+x2,-2.5+y2,5.6+z2);
glVertex3f(1+x2,-2.5+y2,5.6+z2);
glVertex3f(-5+x2,-4.5+y2,5.6+z2);
```

```
glVertex3f(5+x2,-4.5+y2,5.6+z2);

//ventana lateral derecha 2
glBegin(GL_QUADS);
glColor3f(0,0,0);

glVertex3f(10+x2,-2.5+y2,5.6+z2);
glVertex3f(6+x2,-2.5+y2,5.6+z2);
glVertex3f(6+x2,-4.5+y2,5.6+z2);
glVertex3f(13+x2,-4.5+y2,5.6+z2);
glEnd();


//ventana lateral izquierda 1
glBegin(GL_QUADS);
glColor3f(0,0,0);

glVertex3f(5+x2,-2.5+y2,-5.6+z2);
glVertex3f(1+x2,-2.5+y2,-5.6+z2);
glVertex3f(-5+x2,-4.5+y2,-5.6+z2);
glVertex3f(5+x2,-4.5+y2,-5.6+z2);
glEnd();

//ventana lateral izquierda 2
glBegin(GL_QUADS);
glColor3f(0,0,0);

glVertex3f(10+x2,-2.5+y2,-5.6+z2);

glVertex3f(6+x2,-2.5+y2,-5.6+z2);
glVertex3f(6+x2,-4.5+y2,-5.6+z2);
glVertex3f(13+x2,-4.5+y2,-5.6+z2);
glEnd();
```

glVertex3f(5+x2,-4.5+y2,5.6+z2);

//ventana trasera

```
glBegin(GL_QUADS);

glColor3f(0,0,0);

glVertex3f(11+x2,-2.3+y2,-5.+z2);
glVertex3f(11+x2,-2.3+y2,5.+z2);
glVertex3f(15+x2,-4.1+y2,5.+z2);
glVertex3f(15+x2,-4.1+y2,-5.+z2);
glEnd();
```

//luz derecha

```
glPushMatrix();
glColor3f(1, 1, 0);

glLightfv(GL_LIGHT0,GL_POSITION,light_pos);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spot_dir);
glTranslatef((14.5+x)*-1, (8+y)*-1, 3.3+z);
glutSolidSphere(1.2, 20, 10);

glPopMatrix();
```

//luz izquierda

```
glPushMatrix();
glColor3f(1, 1, 0);

glTranslatef((14.5+x)*-1, (8+y)*-1, -3.3+z);

glutSolidSphere(1.2, 20, 10);

glPopMatrix();
```

//ruedas

```
glPushMatrix();
glColor3f(0, 0, 0);
```

```
glBegin(GL_QUADS);
glTranslatef((9+x)*-1, (10+y)*-1, 5.5+z);
glutSolidTorus(1.5, 1.5, 30, 30);
glPopMatrix();
glPushMatrix();

glColor3f(0, 0, 0);
```

```
        glTranslatef((-12+x)*-1, (10+y)*-1, 5.5+z);

        glutSolidTorus(1.5, 1.5, 30, 30);

        glPopMatrix();
        glPushMatrix();
        glColor3f(0, 0, 0);
        glTranslatef((9+x)*-1, (10+y)*-1, -5.5+z);
        glutSolidTorus(1.5, 1.5, 30, 30);
        glPopMatrix();
        glPushMatrix();

        glColor3f(0, 0, 0);

        glTranslatef((-12+x)*-1, (10+y)*-1, -5.5+z);
        glutSolidTorus(1.5, 1.5, 30, 30);
        glPopMatrix();



}
```

//Definimos una esfera la cual manipularemos para crear los arboles

```
void esfera(int x,int y, int z){
    glPushMatrix();
    glColor3f(0,1,0);
    glTranslatef(12+x,-2+y,20+z);
    glutSolidSphere(3,100,100);
    glPopMatrix();
```

```
        glTranslatef((-12+x)*-1, (10+y)*-1, 5.5+z);
}
```

//Definimos una función para crear manzanas

```
void manzana(int x, int y, int z)

{

   glPushMatrix();
    glColor3f(1,0,0);
    glTranslatef(2+x,10+y,22+z);
```

```
    glutSolidSphere(0.5,100,100);

    glPopMatrix();

}
```

//Le ingresamos los parámetros para trasladar las manzanas

```
void manzanas()

{

    glPushMatrix();

    //1
    manzana(0,0,0);
    manzana(0,-6,-2);
    manzana(-4,-6,1);
    manzana(-4,-1,6);
    manzana(-9,-1,-5);
    manzana(-8,-5,-1);
    manzana(0,0,-7);

    //2
    manzana(10,0,12);
    manzana(10,-6,7);
    manzana(-14,-6,11);
    manzana(-14,-1,12);
    manzana(-19,-1,11);
    manzana(-18,-5,11);
    manzana(8,4,6);

    //3
```

```
manzana(19,0,-95);
manzana(19,-6,-97);
manzana(19,-6,-96);
manzana(19,-1,-96);
manzana(19,-1,-95);
```

```
    manzana(18,-5,-96);

    manzana(19,0,-97);

    //4

    manzana(-15,0,-82);
    manzana(-15,-6,-87);
    manzana(-19,-6,-81);
    manzana(-19,-1,-82);
    manzana(-25,-1,-81);
    manzana(-23,-5,-81);
    manzana(-23,4,-86);
    glPopMatrix();


}
```

//Función con la cual definimos el sol

```
void sol()

{

glPushMatrix();
    glColor3f(1,1,0);
    glTranslatef(-1060,55,-20);

    glutSolidSphere(200,100,100);

    glPopMatrix();

}
```

//Función para definir los arboles

```
void arbol(int x, int y, int z){
    glPushMatrix();
    glColor3ub(150,75,0);
    glRotatef(90,-2000,-100,10);
    glTranslatef(0+x,-20+y,-10+z);
    glutSolidCone(1.5,30,100,100);
```

```
    esfera(-10,2,8);
    esfera(-12,2,8);
    esfera(-12,2,10);
    esfera(-12,2,12);
    esfera(-12,2,8);
    esfera(-12,-2,8);
    esfera(-14,2,8);
    esfera(-16,2,8);
    esfera(-12,4,8);
    esfera(-12,6,8);
    esfera(-12,2,6);
    esfera(-12,2,4);
    glPopMatrix();
}
```

Función para crear una esfera blanca la cual conformará a las nubes

```
void esferanube(int a, int b, int c)

{

 glPushMatrix(); glColor3f(1,1,1);
   glTranslatef(84+a,55+b,60+c);
   glutWireSphere(8,100,100);
   glPopMatrix();

   }
```

//Función para definir las nubes

```
void nube()

{

    //nube1
   esferanube(0,0,0);
```

```
esferanube(-12,-2,0);
esferanube(-11,3,0);
esferanube(-20,-2,0);

 //nube2
esferanube(40,0,-210);
esferanube(52,-2,-210);
esferanube(51,3,-210);
esferanube(60,-2,-210);


 //nube3

 esferanube(-240,0,-210);
esferanube(-252,-2,-210);
esferanube(-251,3,-210);
esferanube(-260,-2,-210);


 //nube4

esferanube(-220,0,0);
esferanube(-212,-2,0);
esferanube(-211,3,0);
esferanube(-220,-2,0);

 //nube 5
   esferanube(220,0,130);

esferanube(212,-2,130);

esferanube(211,3,130);
```

```
 esferanube(195,-2,130);

 //nube6

   esferanube(220,0,-330);
esferanube(212,-2,-330);
esferanube(211,3,-330);
```

```
 esferanube(195,-2,-330);

 //nube7

 esferanube(-60,0,250);
esferanube(-56,-2,250);
esferanube(-55,3,250);
esferanube(-60,-2,250);

 //nube8

 esferanube(-160,0,-310);
esferanube(-172,-2,-310);
esferanube(-171,3,-310);
esferanube(-180,-2,-310);

 //nube9

 esferanube(440,0,-110);
esferanube(452,-2,-110);
esferanube(451,3,-110);
esferanube(460,-2,-110);

 //nube10

 esferanube(-660,0,0);
esferanube(-656,-2,10);
esferanube(-655,3,10);
esferanube(-660,-2,10);

 //Nube12

 esferanube(-760,0,300);
esferanube(-756,-2,-310);
```

```
esferanube(-755,3,-310);
esferanube(-760,-2,-310);

    //nube13

 esferanube(-660,0,300);

esferanube(-656,-2,310);
```

```
   esferanube(-655,3,310);

    esferanube(-660,-2,310);



}
```

//Función para definir las letras que se utilizaran para mover el observador

```
void keyboard(unsigned char key, int i, int j)

{

    switch(key)

    {

    case 'a':

    case 'A':

        glRotatef(10.0f,0.0f, 1.0f, 0.0f);

    break;



    case 'd':

    case 'D':

        glRotatef(-10.0f,0.0f, 1.0f, 0.0f);
```

```
break;



case 'w':

case 'W':

    glTranslatef(1,0,0);

break;



case 's':

case 'S':

    glTranslatef(-1,0,0);

break;
```

```
case 'z':

case 'Z':
    x=x+3;
    x2=x2-3;
break;



case 'x':

case 'X': x=x-
    3;
    x2=x2+3;
break;



case 'p':

case 'P':
    hazPerspectiva=1;
    reshape(ancho,alto);
break;



case 'o':

case 'O':
    hazPerspectiva=0;
    reshape(ancho,alto);
```

```
break;



case 'e':

    case 'E':
  exit(0);
  break;

}
```

```
}

//Función para cargar la textura en un polígono

void textura(float xc, float yc, float zc)

{

//activa la textura

//cesped



   glEnable(GL_TEXTURE_2D);

   glBindTexture(GL_TEXTURE_2D, texture[0]);

glBegin(GL_POLYGON);



          glTexCoord2d(0, 0);          glVertex3f((-1128+xc)*-1, (8+yc)*-
1, -1145-17+zc);

          glTexCoord2d(175, 0);   glVertex3f((-1128+xc)*-1, (8+yc)*-1,
1113-17+zc);

          glTexCoord2d(175, 175); glVertex3f((1128+xc)*-1, (8+yc)*-1,
1113-17+zc);

          glTexCoord2d(0, 175); glVertex3f((1128+xc)*-1, (8+yc)*-1, -
1145-17+zc);
```

```
   glDisable(GL_TEXTURE_2D);

    glEnd();
```

//carretera

```
            glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, texture[1]);
glBegin(GL_POLYGON);
 glColor3f(1,1,0);

 glTexCoord2d(0,0);   glVertex3f(1128,-12.5,-15);
  glTexCoord2d(1,0);   glVertex3f(1128,-12.5,15);
  glTexCoord2d(1,1); glVertex3f(-1128,-12.5,15);
```

```
 glTexCoord2d(0,1); glVertex3f(-1128,-12.5,-15);

 glEnd();

 glDisable(GL_TEXTURE_2D);
```

//Función pájaro y pájaro 2 para definir las aves mediante polígonos

```
}
void pajaro()

 //Pajaro glPushMatrix();
 glRotatef(rotacion_pajaro,0,1,0);
 glPushMatrix();
 glColor3f(1,1,1);
 glTranslatef(10+mov_pajaro,10,10);
 glutSolidSphere(0.7,100,100);
 glPopMatrix();
 glPushMatrix(); glColor3f(1,1,1);
 glTranslatef(9+mov_pajaro,10,10);
 glutSolidSphere(0.5,100,100);
 glPopMatrix();


 glPushMatrix();
 glBegin(GL_POLYGON);
 glColor3ub(238,118,0);
 glVertex3f(8.5+mov_pajaro,9,10);
 glVertex3f(9+mov_pajaro,10,9.5);
```

```
glVertex3f(9+mov_pajaro,10,10.5);
glEnd();
```

```
glBegin(GL_QUADS);
glColor3f(1,1,1);
glVertex3f(12+mov_pajaro,10,9);
glVertex3f(12+mov_pajaro,10,11);
glVertex3f(10+mov_pajaro,10,10.6);
glVertex3f(10+mov_pajaro,10,9.4);
glEnd();


glBegin(GL_QUADS);
glColor3f(1,1,1);
glVertex3f(9.7+mov_pajaro,10,10);
glVertex3f(9.7+mov_pajaro,10,13);
glVertex3f(10.7+mov_pajaro,10,15);
glVertex3f(10.7+mov_pajaro,10,10);
glEnd();


glBegin(GL_QUADS);
glColor3f(1,1,1);
glVertex3f(9.7+mov_pajaro,10,10);
glVertex3f(9.7+mov_pajaro,10,7);
glVertex3f(10.7+mov_pajaro,10,5);
glVertex3f(10.7+mov_pajaro,10,10);
glEnd();


rotacion_pajaro=rotacion_pajaro-10;

glPopMatrix();
```

```
 glPopMatrix();

 //FIN Pajaro

}
```

```
void pajaro1()

{

   //Pajaro 2 glPushMatrix();
 glRotatef(rotacion_pajaro*-1,0,1,0);
 glPushMatrix();

 glColor3f(1,1,1);

 glTranslatef(10+mov_pajaro,10,-10);
 glutSolidSphere(0.7,100,100);
 glPopMatrix();

 glPushMatrix(); glColor3f(1,1,1);
 glTranslatef(9+mov_pajaro,10,-10);
 glutSolidSphere(0.5,100,100);
 glPopMatrix();



 glPushMatrix();
 glBegin(GL_POLYGON);
 glColor3ub(238,118,0);
 glVertex3f(8.5+mov_pajaro,9,-10);
 glVertex3f(9+mov_pajaro,10,-9.5);
 glVertex3f(9+mov_pajaro,10,-10.5);
 glEnd();



 glBegin(GL_QUADS); glColor3f(1,1,1);
 glVertex3f(12+mov_pajaro,10,-9);
```

```
glVertex3f(12+mov_pajaro,10,-11);
glVertex3f(10+mov_pajaro,10,-10.6);
```

```
glVertex3f(10+mov_pajaro,10,-9.4);

glEnd();



glBegin(GL_QUADS); glColor3f(1,1,1);
glVertex3f(9.7+mov_pajaro,10,-10);
glVertex3f(9.7+mov_pajaro,10,-13);
glVertex3f(10.7+mov_pajaro,10,-15);
glVertex3f(10.7+mov_pajaro,10,-10);
glEnd();



glBegin(GL_QUADS); glColor3f(1,1,1);
glVertex3f(9.7+mov_pajaro,10,-10);
glVertex3f(9.7+mov_pajaro,10,-7);
glVertex3f(10.7+mov_pajaro,10,-5);
glVertex3f(10.7+mov_pajaro,10,-10);
glEnd();



rotacion_pajaro=rotacion_pajaro-10;

glPopMatrix();

glPopMatrix();

//FIN Pajaro 2

  }
```

Función donde se llamaran todas las funciones creadas para imprimir el escenario final

```
void display()

{



    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

```
    textura(0,5,0);

      nube(); sol();
     automovil(0+x,0+y,0+z);
     pajaro();
     pajaro1();
    manzanas();
    arbol(0,0,-10);
    arbol(10,-10,-10);
    arbol(-10,-16,-10);
    arbol(20,100,-10);
    arbol(-10,90,-10);
    arbol(-20,80,-10);


    glutKeyboardFunc(keyboard);

    glFlush();

    glutSwapBuffers();

    glutPostRedisplay();


}


void idle()

{
```

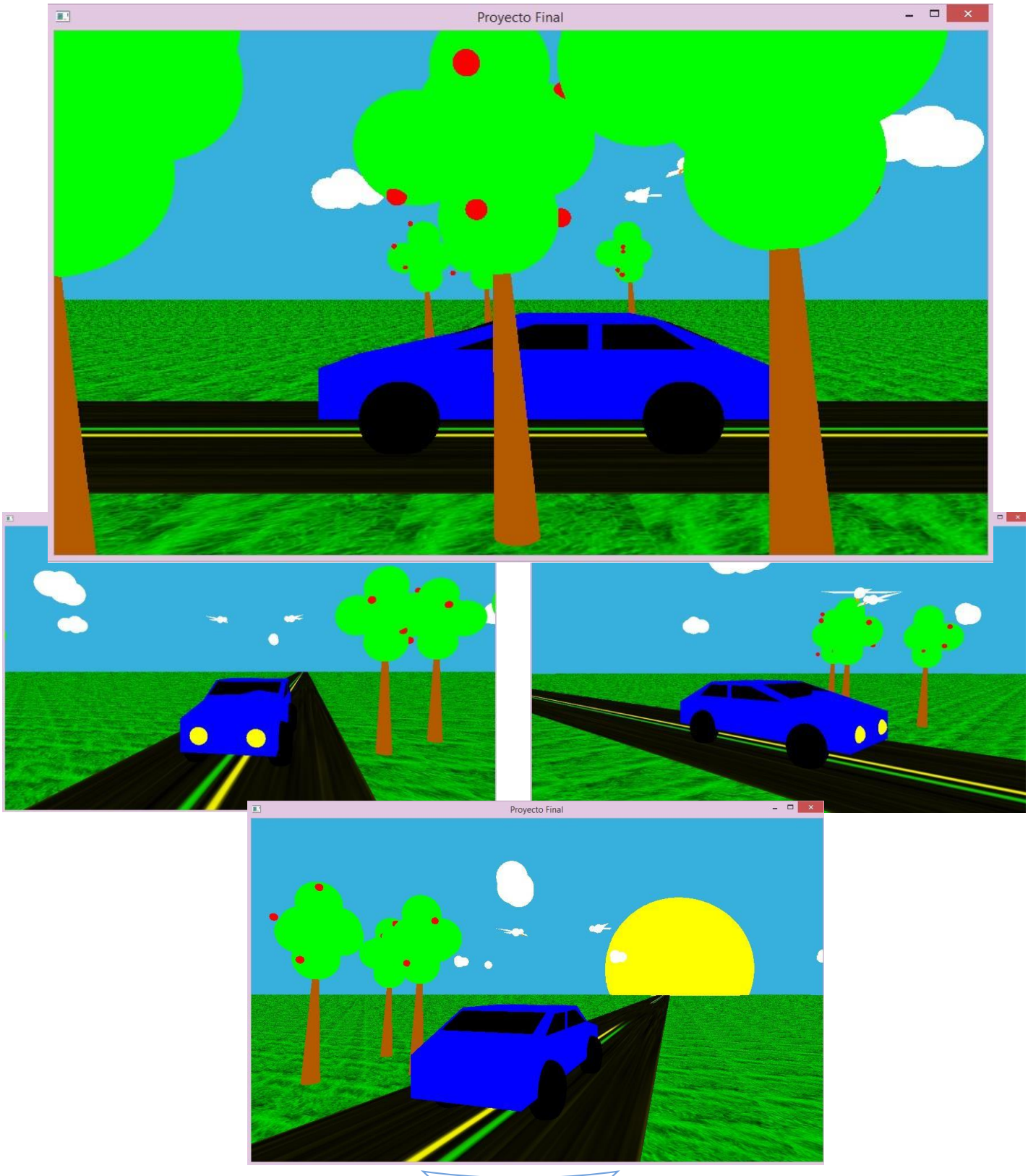
```
display();

}
```

Función main donde se definirán los parámetros finales de la ventana en donde se mostrará el escenario y se llamara a la función Display

```
int main(int argc, char **argv)

{

    PlaySound("soul.wav",NULL,SND_ASYNC | SND_LOOP);

   glutInit(&argc, argv);

   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);

   glutInitWindowPosition(100,100);
   glutInitWindowSize(1000, 600);
   glutCreateWindow("Proyecto Final");
       init();

       glutReshapeFunc(reshape);

   glutDisplayFunc(display);

glutIdleFunc(idle);



   glutMainLoop();

   return 0;

}
```

Dibujo

Vistas que mostrará el programa al ser compilado.



Conclusiones

El proyecto presentado fue el proyecto final para la clase de graficación en donde se consideró un ambiente 3D para la creación de este escenario al igual que la implementación de recursos vistos anteriormente para poder culminar este proyecto

Bibliografía

1.- Jorge García (Bardok). (2003). Texturas 2D. 20/03/2015, de Wordpress Sitio web: https://graficacion11030440.wordpress.com/texturas-2d/

2.- Jorge García (Bardok). (2003). Curso de introducción a OpenGL (v1.0). En Manual opengl (47-50). http://www.elai.upm.es/webantigua/spain/Asignaturas/InfoInd/ teoria/Manual-opengl.pdf: ESIDE Ghost.

3.- Jose Antonio Camarena Ibarrola. (2010). Notas de Graficación. En Notas de Graficación (31). Univ. Michoacana de San Nicolás de Hgo.: Facultad de Ingeniería Eléctrica.

4.- Michelle Jimenez . (2014). Figuras en OpenGL. 21/04/2015, de Wordpress Sitio web: https://michellejimenezv.wordpress.com/2014/03/07/figuras-en-opengl/

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

NOMBRE: Erik Alfredo González Gutiérrez

CICLO: primavera 2015

MATERIA: Graficación

Introducción.

El actual documento presenta la forma en como se llevo a cabo el dibujado de un escenario en 3D con elementos estaticos(es decir sin movimiento) y con movimientos de desplazamiento tambien se tendra que usar librerias para el uso de texturas. en este caso se opto por dibujar un escenario 3D que muestra un vecindario con arboles,nubes , carros , bancas , casas , el sol , la luna y estrellas cuando se oculta el sol, ademas de abarcar el tema de texturas . Para el caso de texturas solo se eligio poner el piso con una imagen que simula ser un suelo con pasto , para poder obtener este resultado se eligio la librería para la carga de texturas FreeImage.

Ademas de todo lo antes mencionado que en trabajos previos ya se han utilizado en este proyecto se ha incluido el tema de la iluminacion de objetos asi que comforme avance el documento tambien se tratara este tema.

Asimismo en el actual documento se ejemplifica mediante un diagrama de transiciones como fue el proceso para obtener el escenario final demostrando asi que efectivamente opengl esta basado en una maquina de estados.

Desarrollo.

Primero que cualquier otra cosa se crean las funciones que en todo el curso se han utilizado como lo son la función init que es la que se encarga de inicializar las variables y entornos del proyecto como la pantalla, el color, el tamaño el tipo de proyección entre otras cosas. La función reshape que sirve para redimensionar la ventana y su contenido. La función Idle que en nuestro caso lo único que hace es dibujar lo mismo que ya hay e pantalla. La función keyboard que nos sirve para la interacción de nuestro programa con el teclado. La función display que es la que se encarga de dibujar todo en pantalla. Por último la función main que es el principio de nuestro código ya

que es la que se encarga de llamar a todas las demás funciones que previamente mencione.

Dichas funciones mencionadas son parecidas a las proporcionadas por el profesor de la clase. Para este proyecto lo único que cambia es la función display que como es obvio necesitamos objetos diferentes así que dicha función es diferente.

En el código del proyecto para el dibujado de cada objeto se crea una función para poder dibujar n-objetos como se desee es decir si queremos dibujar un árbol solo mandamos a llamar a la función árbol y le mandamos las coordenadas donde queremos dibujar el árbol así mismo existe una función para cada objeto presente en el escenario, cada función contiene una estructura como se describe a continuación.

```
Void nombre([parámetros]){
glPushMatrix();
glLoadIdentity();
código de objeto a dibujar ;
glPopMatrix();
}//fin de función
```

Así que siguiendo la estructura anterior se crean las siguientes funciones:

void nube(float x , float z): aquí se dibuja una nube sin movimiento, los parámetros que recibe son las coordenadas en donde se ubicara el objeto nube en el escenario.

Void arbol(float x, float z): se encarga de dibujar una arbol de un tamaño estático determinado , los parámetros que recibe son las coordenadas x, z en el escenario , es decir es donde se dibujara el objeto(arbol).

void carro(float x, float z, float c):esta función dibuja un carro en el escenario, los parámetros que recibe son las coordenadas en el eje x y z, el tercer parámetro esta en un rango entre 1 y 3 y sirve para indicar el color del carro siendo 1 para carro azul, 2 para carro rojo y 3 para carro verde.

void sol(float x, float z): se encarga de dibujar una esfera con un tamaño determinado que simula el sol en el escenario, los parámetros son las coordenadas en donde se dibujara el objeto sol, este objeto tiene además movimiento de traslación simulando así el día y la noche.

void luna(float x, float z): en esta función se dibuja una esfera con un tamaño predeterminado que simula la luna , los parámetros recibidos son las coordenadas donde se dibujara el objeto luna , este objeto así como el objeto sol tiene movimiento de traslación simulando la noche.

void banca(float x , float z): es la función que se ocupa de dibujar una banca dentro del escenario en las coordenadas indicadas por x y z .

void casa(float x, float z, float c): esta función es la encargada de dibujar las casas que se visualizan al fondo de la pantalla, los parámetros que recibe son las coordenadas en el eje x y z , el parámetro c que recibe es para indicar el

color de cada casa, al igual que el objeto carro solo se puede recibir un dato entero que pueden ser 1,2 ó 3.

void escenario():esta función no recibe parámetros por el hecho de que solo se dibuja un escenario y los demás objetos se dibujan sobre el mismo, esta función lo que dibuja es el piso y tres muros a los lados y en el fondo del escenario.

void estrellas(float x,float y, float z): es la encargada de dibujar las estrellas en cuanto se desaparece el objeto sol y se oscurece la pantalla, para este objeto se utilizo 4 conos colocados a modo que los picos simulen los picos de las estrellas , los parámetros que recibe al igual que las demás funciones son las coordenadas donde se dibujara el objeto estrella.

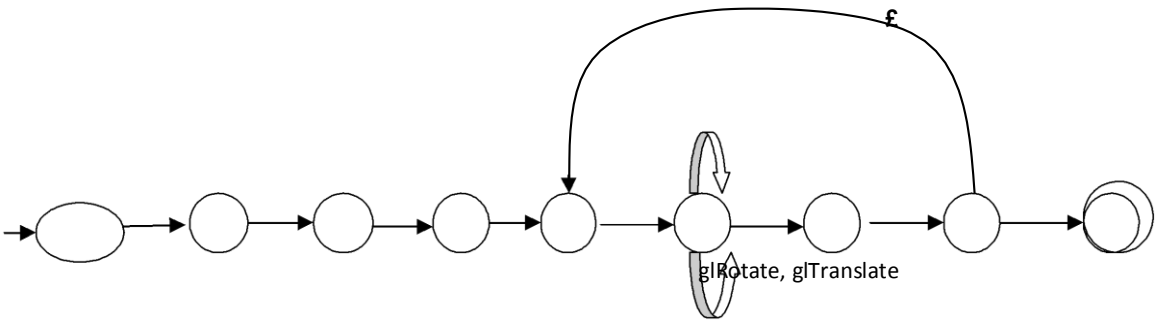
FIBITMAP *loadImage (const char *filename) es la encargada de cargar la imagen la cual se le pondrá a cualquier objeto que se desee texturizar.

GLuint LoadTexture (FIBITMAP * dib1) esta es la función que se encarga de cargar las texturas , como resultado regresa un apuntador tipo GLuint llamado tex_id donde se encuentra cargada la imagen .

Por último en la función display se invocan todas las funciones que dibujan cada objeto , se actualizan las variables globales camina, EYE_X y CENTER_X también se crean las condiciones para las variables camina, EYE_X y CENTER_X estas condiciones son que cuando cada variable llegue a -300 la variable camina se actualiza en 300 y la variable move cambie de signo permitiendo así el cambio de dirección de la cámara.

Cabe mencionar que cada objeto contiene la invocación a la función lookAt que es la encargada de mover la cámara y su dirección así como también la inclinación de la misma , también cada objeto contiene al principio la llamada de la función que activa la iluminación una vez se dibuje , es decir antes de salir de dicha función se desactiva el modo de iluminación para que de esta manera no afecte a los demás objetos en escena.

Aquí se muestra un diagrama de transiciones de cómo se llevo a cabo el proyecto :



glPushMatrix glLoadIdentity DibujaEscenario glPushMatrix dibObjeto glPopMatrix glPopMatrix

1 2 3 4 5 6 7 8 9

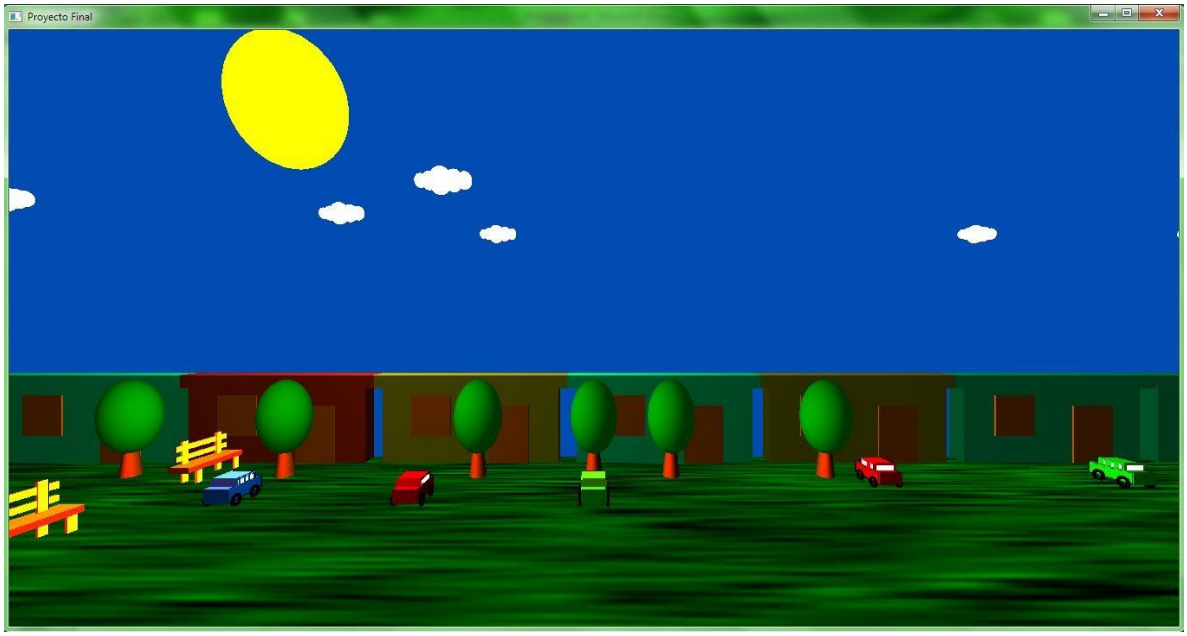
glScale

Conclusión.

Como conclusión se puede decir que con lo aprendido en el curso ya somos capaces de dibujar cualquier escenario en tercera dimensión muy completo con movimientos, texturas e iluminación.. También se espera una mejor comprensión del porque se dice que opengl es una máquina de estados y

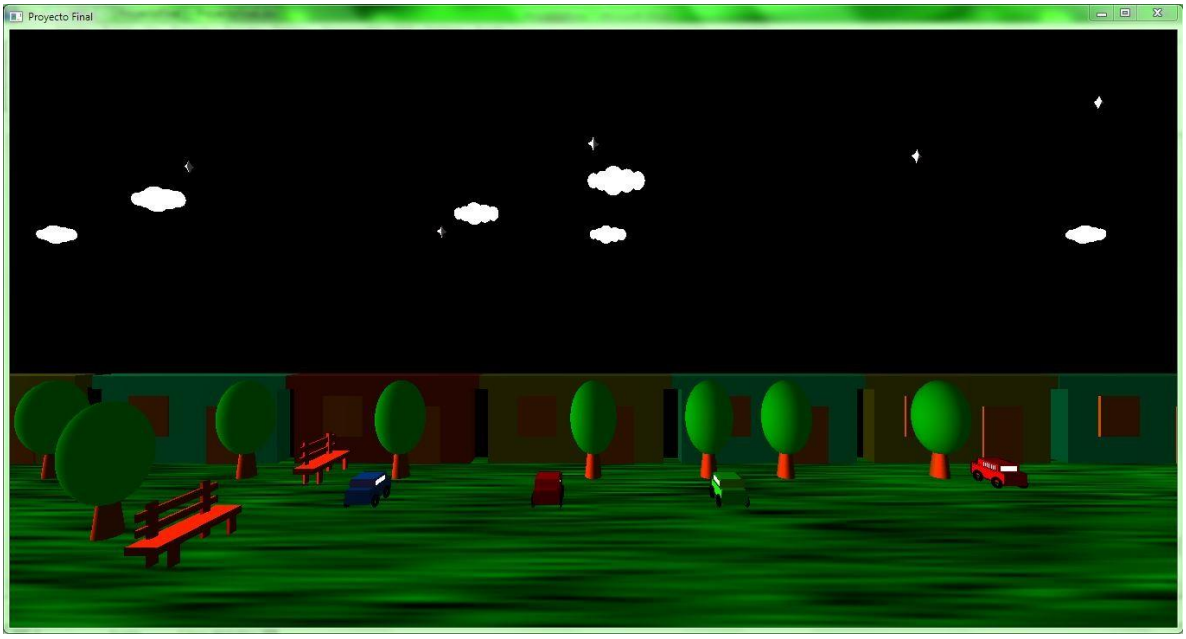
como cada objeto que dibujamos en pantalla afecta a los siguientes objetos si no se toman la debidas precauciones como lo son un buen uso de glPushMatrix, glPopMatrix glLoadIdentity entre otras funciones que existen en opengl.

Resultados obtenidos en el proyecto:



Al inicio el escenario tiene al objeto sol en el centro y por tanto simula el dia en el escenario.

Aquí el sol ya se empieza a ocultar por tanto se empieza a oscurecer el escenario y empieza a aparecer el objeto luna.



Aquí ya se simula la noche y por tanto aparecen objetos tipo estrella en el escenario.

Proyecto Final: Objetos 3D.

Garrido González Karla Estefanía

201217899

GRAFICACIÓN

Introducción

En este proyecto pondremos a prueba los conocimientos adquiridos tanto para el manejo de texturas como para las figuras que se presentarán en nuestro escenario.

Estaremos usando funciones primitivas de OpenGL, carga de texturas, iluminación y proyecciones.

Conceptos desarrollados

Usamos tanto funciones primitivas anteriormente vistas en escenarios 2D como nuevas como los anillos (Torus), cilindros, cubos, conos y esferas.

Las funciones primitivas en 3D que trae OpenGL son los anillos, cilindros, cubos, conos, esferas, entre otros. Depende del tipo de figura geométrica tienen distintos parámetros, por ejemplo el cubo solo necesita la longitud de los lados mientras que el cilindro necesita:

```
GLUquadricObj *qobj;
qobj=gluNewQuadric();
gluQuadricDrawStyle(qobj,GLU_FILL);
gluCylinder(GLUquadricObj *qobj, GLdouble radioBase, GLdouble radioTapa, GLdouble altura, GLint segmentos, GLint anillos);
```

En pocas palabras, declarar una variable con el tipo de dato GLUquadric (1), crear el objeto cuadrático (2), definir estilo GLU_LINE en maya y GLU_FILL en sólido (3), dibujar el objeto:

```
gluCylinder(qobj, 10,10,15,20,20);
```

Por la parte de iluminación, cada escena tiene varias fuentes de luz, y ésta puede estar apaga o encendida, puede iluminar toda la escena o solo una dirección dada.

Las luces es la que aporta la luminosidad a nuestra escena. Una fuente de luz puede emitir tipos diferentes de luz, las cuales son las siguientes:

1. Emitida: es la luz emitida por un objeto, no se ve afectada por ningún tipo de luz.
2. Ambiente: podemos decir que son los restos de luz que aparecen debido a la reflexión residual de una luz que ya se ha reflejado sobre muchos objetos, es algo como la iluminación global de alguna escena.

3. Difusa: es la luz que proyecta sobre un objeto y proviene de un determinado punto, la intensidad con la que se refleja en la superficie del objeto puede depender del ángulo de incidencia, dirección, etc.
4. Especular: es la luz que al incidir sobre un objeto, se ve reflejada con un ángulo similar al de incidencia.

Un material define para un objeto que componentes refleja de cada tipo de luz.

La iluminación se calcula a nivel de vértice, es decir, por cada vértice se calcula su color a partir del material activo, las luces activas y como inciden sobre el vértice.

El primer paso es activarla con glEnable(GL_LIGHTING);, una vez activada debemos establecer las propiedades de cada luz en la escena y activarlas.

La especificación contempla que al menos cada implementación debe de poder definir 8 luces, identificadas por GL_LIGHTn donde n es el número de la luz comenzando con 0.

Para establecer las propiedades utilizaremos funciones del tipo glLight*(), de las cuales las propiedades son:

- Posición/Dirección: indica la posición de la luz, además indica si es una luz posicional o direccional. glLightfv(GL_LIGHTn,GL_POSITION,val_ptr); donde val_ptr es un puntero a un vector de cuatro componentes de tipo float.
- Dirección de foco: en el caso de una luz focal, debemos establecer su dirección. glLightfv(GL_LIGHTn,GL_SPOT_DIRECTION,val_ptr); donde val_ptr es un puntero a un vector con la dirección.
- Apertura del foco: el ángulo de apertura del foco. glLightfv(GL_LIGHTn, GL_SPOT_CUTOFF, val); donde val expresa en grados la mitad del ángulo de apertura del foco.
- Atenuación del foco: es el degradado de la intensidad a medida que nos acercamos al borde, glLightfv(GL_LIGHTn, GL_SPOT_EXPONENT, val);
- Intensidad de luz: define el color ambiental, difuso y especular de la luz. glLightfv(GL_LIGHTn,GL_[AMBIENT,SPECULAR,DIFFUSE], val_ptr); donde val_ptr es un puntero a un vector de cuatro componentes de color RGB.

- Atenuación de la luz: define la pérdida de intensidad de luz a medida que nos alejamos del foco (no afecta las luces direccionales).

  glLightfv(GL_LIGHTn,GL_[CONSTANT|LINEAR|QUADRATIC]_ATTENUACION,val); y el valor de atenuación de la luz se calcula: 1/[kc+(kl*d)+(kq)^2].

Una vez que hallamos establecido estas propiedades de una determinada luz, las activamos con: glEnable(GL_LIGHTn);

Interacción con el teclado

Esto se puede hacer con funciones propias de OpenGL, en mi caso usé un switch para poder guardar alguna letra y poder usarla después, por ejemplo: con la s o S trasladar n en el eje x.

Vista del espectador

La matriz de proyección es una matriz en la que se guarda la información relativa a la “cámara” a través de la cual vamos a visualizar el mundo.

glMatrixModel(GL_PROJECTION);

La proyección ortográfica: nos permite visualizar todo aquello que se encuentre dentro de un cubo delimitado por los parámetros de la función glOrtho.

La proyección perspectiva: delimita un volumen de visualización dado por un ángulo de la cámara y una relación alto/ancho. La distancia del observador determinara el tamaño de visualización del objeto.

Conclusiones

Con todo lo aprendido hasta ahora, podemos crear o modificar escenarios más complejos, como animaciones, mini videojuegos, simulaciones entre otras cosas, claro que podemos aprender más por nosotros mismos intentando meterle sonidos, inteligencia artificial, etc.

Bibliografía

https://www.opengl.org/resources/libraries/glut/spec3/node1.html

http://educommons.anahuac.mx:8080/eduCommons/computacion-y-sistemas/programacion-de-graficas-computacionales/tema%203/copy3_of_aprendizaje_motivacion/skinless_view

http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/Opengl/tut_opengl_il umin_raton_text/iluminacion.htm

http://www.elai.upm.es/webantigua/spain/Asignaturas/InfoInd/teoria/Apuntes%20d e%20OpenGL.pdf

Manual OpenGL “Curso de Introducción a OpenGL” por Jorge García.

Benemérita Universidad Autónoma de Puebla

Facultad Ciencias de la Computación

Ingeniería en Ciencias de la Computación

Graficación

Proyecto Final: Escenario de Mario Bros

Fernando Ángel García Parra

Profesor: Dr. Iván Olmos Pineda

Introducción

Lo siguiente hace referencia al el proyecto final de la materia de Graficación en la cual se permite el uso de cada una de las funciones primitivas las cuales están integradas en opengl, el programa es desarrollado en el lenguaje de programación C++, en el entorno de desarrollo de wxDev-C++.

Así como también de la integración de texturas mediante bibliotecas externas a wxDev-C++ y a opengl ya que estos no permiten de forma nativa la incorporación de imágenes para realizar texturizaciones y permitir que nuestros programas y graficas tengan mayor realismo.

Esto nos ayudara si en un futuro decidimos realizar un videojuego, comenzaremos con la explicación de cómo desarrollo el proyecto, así como delos problemas que se encontraron.

Desarrollo

El programa que se desarrolló se basa en un escenario del conocido juego de Mario Bros.

Las partes con las que se trabajó fueron:

- Iluminación
- Texturas
- Instrucciones nativas de opengl

Para saber más sobre cada una de estas se tratara de dar una breve explicación de cada una.

Iluminación

Mediante la iluminación es como se consigue un mayor efecto de realismo.

Las instrucciones que se utilizan en una iluminación son las siguientes:

Cuando rende rizamos una primitiva disponemos de dos métodos de rellenado. Para cambiar entre uno u otro se hace con la función

void glShadeModel(GLenum mode), como parámetro le pasamos uno de los siguientes modos:

GL FLAT: Renderiza la primitiva con un solo color, elegido de uno de los vértices dependiendo de la primitiva.

GL SMOOTH: Renderiza la primitiva interpolando el color de los vértices.

Esto en cuanto a la manera de renderizar la primitiva, el color de cada uno de los vértices hasta ahora lo especificábamos con la función glColor. En el modo de iluminación activado el color de cada uno de los vértices es calculado a partir del material del vértice y las luces.

Para habilitar el modo de iluminación se realiza con la función

glEnable(GL LIGHTING), cuando renderizamos en este modo la instrucción glColor no tienen ninguna anuencia ya que el color resultante de cada vértice es calculado a partir de los materiales del vértice y las luces que haya encendidas.

Disponemos de ocho luces que podemos encender o apagar con la función glEnable(GL LIGHTi) donde i =[0. . . 7]. Cada luz la podemos configurar estableciendo los siguientes parámetros con la

funci_on glLightfifgv :

Luces

Parámetro	Valor por defecto	Significado
GL_AMBIENT	(0.0, 0.0, 0.0, 1.0)	intensidad ambiente de la luz
GL_DIFFUSE	(1.0, 1.0, 1.0, 1.0) GL_LIGTH0 (0.0, 0.0, 0.0, 1.0) GL_LIGTH1-GL_LIGHT7	intensidad difusa de la luz
GL_SPECULAR	(1.0, 1.0, 1.0, 1.0) GL_LIGTH0 (0.0, 0.0, 0.0, 1.0) GL_LIGTH1-GL_LIGHT7	intensidad especular de la luz
GL_POSITION	(0.0, 0.0, 0.0, 1.0)	(x, y, z, w) posición de la luz
GL_SPOT_DIRECTION	(0.0, 0.0, -1.0)	(x, y, z) dirección de la luz
GL_SPOT_EXPONENT	0.0	exponente del foco
GL_SPOT_CUTOFF	180.0	ángulo de abertura del foco
GL_CONSTANT_ATTENUATION	1.0	factor de atenuación constante
GL_LINEAR_ATTENUATION	0.0	factor de atenuación lineal
GL_QUADRATIC_ATTENUATION	0.0	factor de atenuación cuadrático

Componentes de luz

La luz tiene tres componentes: ambiente, difusa y especular:
Ambiente: Es la luz que se emite en todas direcciones y rebota en todas direcciones.
Difusa: Luz emitida desde el punto de luz que rebota en todas las direcciones.
Especular: Luz emitida en una dirección que rebota según la normal del polígono, provoca el brillo puntual del objeto, p.e. el brillo blanco de la bola de billar negra.

Tipos de fuentes de luz

Además de las componentes de luz es necesario saber qué tipo de fuente de luz se trata, en Opengl disponemos de dos tipos:
Direccional: Todos los rayos son paralelos e inciden en la dirección, el valor w de GL POSITION es cero, la dirección se especifica mediante (x, y, z) de GL POSITION. Un ejemplo de luz direccional es el sol.
Posicional: La luz está colocada en el punto (x, y, z) el valor w de GL POSITION es distinto de cero.

Si se trata de una luz posicional, un foco, podemos alterar una serie de parámetros:
GL SPOT DIRECTION: La dirección en que apunta.
GL SPOT EXPONENT: Concentración de la luz en el centro del foco.
GL SPOT CUTOFF: Apertura del foco en grados, el ángulo se multiplica por dos, 180 significa que la luz sale en todas direcciones

Parámetro	Valor por defecto	Significado
GL_LIGHT_MODEL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	intensidad ambiente de la luz global
GL_LIGHT_MODEL_LOCAL_VIEWER	0.0 o FALSE	como se calculan los ángulos para la reflexión de la luz especular
GL_LIGHT_MOEL_TWO_SIDE	0.0 o FALSE	permite elegir entre iluminación a una o dos caras
GL_LIGHT_MODEL_CONTROL_COLOR	GL_SINGLE_COLOR	la componente especular se calcula separado de la ambiente y difusa

En el modelo de renderizacion sin iluminación definíamos el color de cada vértice, ahora tenemos que definir los materiales. Se definen cuatro componentes de material para cada vértice.
Ambiente: Es la componente del material que refleja la luz independientemente de la dirección que venga.
Difusa: Es la componente del material que refleja según la dirección de procedencia de la luz.
Especular: Componente que refleja la luz especular, provoca el efecto brillo.
Emisiva: Componente de color que se añade independientemente de la luz recibida.

Parámetro	Valor por defecto	Significado
GL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	componente ambiente
GL_DIFFUSE	(0.8, 0.8, 0.8, 1.0)	componente difusa
GL_SPECULAR	(0.0, 0.0, 0.0, 1.0)	componente espcular
GL_SHININESS	0.0	exponente de brillo
GL_EMISSION	(0.0, 0.0, 0.0, 1.0)	componente emisiva

Se trató de dar la explicación más detallada posible ya que es un tema el cual no se había tocado durante el curso, pero me pareció interesante el incluirlo en el proyecto.

```
void ilumina_montanas(){
    glPushMatrix();
    GLfloat Posicionytipo[]={6,6,6,2};
    GLfloat Coloramiente[]={1,0,0,0};
    GLfloat Colordifuso[]={0.30,0.25,0.13};
    GLfloat direccion[]={0,0,1};
```

```
    glLightfv(GL_LIGHT0,GL_POSITION,Posicionytipo);
    glLightfv(GL_LIGHT0,GL_AMBIENT,Coloramiente);
    glLightfv(GL_LIGHT0,GL_DIFFUSE,Colordifuso);

    //atenuacion

    glLightf(GL_LIGHT0,GL_CONSTANT_ATTENUATION,.6);
    glLightf(GL_LIGHT0,GL_LINEAR_ATTENUATION,0);
    glLightf(GL_LIGHT0,GL_QUADRATIC_ATTENUATION,0);

    //DIRECCIONES
    glLightfv(GL_LIGHT1,GL_SPOT_DIRECTION,direccion);
    glLightf(GL_LIGHT1,GL_SPOT_CUTOFF,10);
    glLightf(GL_LIGHT1,GL_SPOT_EXPONENT,20);

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHT1);
    glEnable(GL_DEPTH_TEST);
    glPopMatrix();
}
```

Este es el código que si implemento para realizar la iluminación de algunos de nuestros objetos, el código es repetido para cada uno de ellos, y solo son cambiados los parámetros.

Lo siguiente que se utilizo fue la texturizarían la cual ocasiono muchos problemas ya que las opciones que se tenían pensado utilizar para realizarla eran trabajar con la biblioteca FreeImage o GLaux pero estas ocasionaban un conflicto con el sistema operativo de Windows8 así que se decidió buscar alguna otra biblioteca para realizar texturizaciones.
La cual es una de nombre loadBMP con la cual se pudo trabajar sin ningún problema.
El único inconveniente que tiene esta biblioteca es que como su nombre lo dice solo permite el uso de de imágenes BMP y con un tamaño definido de 128x128 pixeles.

A continuación se agrega el código el cual es un poco largo.

```
// Funciones para la Carga de las Texturas
class Image {
        public:
                Image(char* ps, int w, int h);
                ~Image();
                char* pixels;
                int width;
                int height;
};
```

```
Image* loadBMP(const char* filename);
Image::Image(char* ps, int w, int h) : pixels(ps), width(w), height(h) {
}
Image::~Image() {
	delete[] pixels;
}

namespace {
	int toInt(const char* bytes) {
		return (int)(((unsigned char)bytes[3] << 24) |
					((unsigned char)bytes[2] << 16) |
					((unsigned char)bytes[1] << 8) |
					(unsigned char)bytes[0]);
	}

	short toShort(const char* bytes) {
		return (short)(((unsigned char)bytes[1] << 8) |
						(unsigned char)bytes[0]);
	}
	int readInt(ifstream &input) {
		char buffer[4];
		input.read(buffer, 4);
		return toInt(buffer);
	}
	short readShort(ifstream &input) {
		char buffer[2];
		input.read(buffer, 2);
		return toShort(buffer);
	}
	template<class T>
	class auto_array {
		private:
			T* array;
			mutable bool isReleased;
		public:
			explicit auto_array(T* array_ = NULL) :
				array(array_), isReleased(false) {
			}
			auto_array(const auto_array<T> &aarray) {
				array = aarray.array;
				isReleased = aarray.isReleased;
				aarray.isReleased = true;
			}
			~auto_array() {
				if (!isReleased && array != NULL) {
					delete[] array;
				}
			}
```

```
                        T* get() const {
                                return array;
                        }
                        T &operator*() const {
                                return *array;
                        }
                        void operator=(const auto_array<T> &aarray) {
                                if (!isReleased && array != NULL) {
                                        delete[] array;
                                }
                                array = aarray.array;
                                isReleased = aarray.isReleased;
                                aarray.isReleased = true;
                        }
                        T* operator->() const {
                                return array;
                        }
                        T* release() {
                                isReleased = true;
                                return array;
                        }
                        void reset(T* array_ = NULL) {
                                if (!isReleased && array != NULL) {
                                        delete[] array;
                                }
                                array = array_;
                        }
                        T* operator+(int i) {
                                return array + i;
                        }
                        T &operator[](int i) {
                                return array[i];
                        }
        };
}

Image* loadBMP(const char* filename) {
        ifstream input;
        input.open(filename, ifstream::binary);
        assert(!input.fail() || !"Could not find file");
        char buffer[2];
        input.read(buffer, 2);
        assert(buffer[0] == 'B' && buffer[1] == 'M' || !"Not a bitmap file");
        input.ignore(8);
        int dataOffset = readInt(input);

        int headerSize = readInt(input);
        int width;
```

```
	int height;
	switch(headerSize) {
		case 40:
			width = readInt(input);
			height = readInt(input);
			input.ignore(2);
			assert(readShort(input) == 24 || !"Image is not 24 bits per
pixel");
			assert(readShort(input) == 0 || !"Image is compressed");
			break;
		case 12:
			width = readShort(input);
			height = readShort(input);
			input.ignore(2);
			assert(readShort(input) == 24 || !"Image is not 24 bits per
pixel");
			break;
		case 64:
			assert(!"Can't load OS/2 V2 bitmaps");
			break;
		case 108:
			assert(!"Can't load Windows V4 bitmaps");
			break;
		case 124:
			assert(!"Can't load Windows V5 bitmaps");
			break;
		default:
			assert(!"Unknown bitmap format");
	}

	int bytesPerRow = ((width * 3 + 3) / 4) * 4 - (width * 3 % 4);
	int size = bytesPerRow * height;
	auto_array<char> pixels(new char[size]);
	input.seekg(dataOffset, ios_base::beg);
	input.read(pixels.get(), size);

	auto_array<char> pixels2(new char[width * height * 3]);
	for(int y = 0; y < height; y++) {
		for(int x = 0; x < width; x++) {
			for(int c = 0; c < 3; c++) {
				pixels2[3 * (width * y + x) + c] =
					pixels[bytesPerRow * y + 3 * x + (2 - c)];
			}
		}
	}

	input.close();
	return new Image(pixels2.release(), width, height);
```

```
}

// FIN DE LAS FUNCIONES PARA CARGAR TEXTURAS ....

GLuint ID_de_Textura1;

GLuint loadTexture(Image* image) {
	GLuint textureId;
	glGenTextures(1, &textureId);
	glBindTexture(GL_TEXTURE_2D, textureId);
	glTexImage2D(GL_TEXTURE_2D,
				0,
				GL_RGB,
				image->width, image->height,
				0,
				GL_RGB,
				GL_UNSIGNED_BYTE,
				image->pixels);
	return textureId;
}

void initRendering() {
	glEnable(GL_DEPTH_TEST);

	Image* image1 = loadBMP("1.bmp");

	ID_de_Textura1 = loadTexture(image1);

	delete image1;
}
```

En la cual lo mas importante y lo que nos permite trabajar y cargar las texturas son las instrucciones:

GLuint ID_de_Textura1: permite asignar un número a cada una de las imágenes con las que trabajamos y de esta forma identificarlas más fácilmente.
glGenTextures(1, &textureId);
glBindTexture(GL_TEXTURE_2D, textureId);

Nos permiten trabajar con un arreglo de imágenes y así sabes con qué imagen se quieres trabajar así como el tipo de textura

Image* image1 = loadBMP("1.bmp");
ID_de_Textura1 = loadTexture(image1);
Cargad del arreglo de imágenes

Al tener demasiados problemas con texturas esta biblioteca no se terminó de entender por completo así que aún queda un poco de dudas sobre su funcionamiento.

```
glEnable(GL_TEXTURE_2D);
        glBindTexture(GL_TEXTURE_2D, ID_de_Textura1);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
        glBegin(GL_POLYGON);
        glColor3f(1.0f, 1.0f, 1.0f);
        glTexCoord2f(0.0f, 0.0f);
        glVertex3f(-BOX_SIZE / 4  , -BOX_SIZE / 4, -BOX_SIZE / 4);
        glTexCoord2f(1.0f, 0.0f);
   glVertex3f(BOX_SIZE / 4  , -BOX_SIZE / 4, -BOX_SIZE / 4);
        glTexCoord2f(1.0f, 1.0f);
   glVertex3f(BOX_SIZE / 4  , -BOX_SIZE / 4, BOX_SIZE / 4);
        glTexCoord2f(0.0f, 1.0f);
        glVertex3f(-BOX_SIZE / 4  , -BOX_SIZE / 4, BOX_SIZE / 4);
        glEnd();
        glDisable(GL_TEXTURE_2D);
```

Con esto permitidos que el objeto tenga la textura deseada.

La iluminación se manda a llamar cuando se desea dibujar algún objeto y asi poder darle más realismo

```
void montanas()
{
    glPushMatrix();
    ilumina_montanas();
    glColor3f(0.30,0.25,0.13);
    glTranslated(-3,-3,-8);
    glScalef(2,1.5,.6);
    glRotated(90,-1,0,0);
    glutSolidCone(4,7,10,3);
    desactiva_luz();
        glPopMatrix();
        //montaña 2
        glPushMatrix();
         ilumina_montanas();
    glColor3f(0.30,0.25,0.13);
    glTranslated(4,-3,-8);
    glScalef(1.5,1,.6);
    glRotated(90,-1,0,0);
    glutSolidCone(4,7,10,3);
    desactiva_luz();
        glPopMatrix();
```

```
}
```

Debido a que el dibujado de los objetos del escenario se realizaron las primitivas de opengl se decidió ya no incluir más código ya que sería lo mismo pero con diferentes coordenadas y colores.

El siguiente codigo muentra la forma en la cual se puede girar la visualización atravez de la manipulación del gluLookAt.

```
if (!lookAt){

        // eje horizontal.
        glRotatef(anguloCamaraY, 0.0f, 1.0f, 0.0f);
        // eje vertical.
        glRotatef(anguloCamaraX,1.0f,0.0f,0.0f);
    } else {
        gluLookAt(0,0,7,0,0,0,0,1,5);
    }
```

Bibliografía

Programming Principles and Practice Using C++, Bjarne Stroustrup.
http://www.codeblocks.org
http://www.wxwidgets.org
(O'Reilly) Practical C Programming (3rd Edition)
http://www.cplusplus.com
http://es.wikipedia.org/wiki/GLUT

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de computación

Documento técnico:

“Librerías y escenario”

Materia: Graficación

Nombre del alumno: Ruiz Flores Jorge Mauricio

Nombre del profesor: Olmos Pineda Iván

Proyecto Final Graficación.

Introducción

El objetivo del proyecto fue reafirmar todos los conocimientos obtenidos durante la primera unidad del curso, retomamos el concepto de que cualquier grafica (escenario) está desarrollado a través de puntos que al unirse forman figuras más complejas como son líneas, triángulos, cuadriláteros y círculos.

Conceptos:

El punto

Recordemos que la forma más básica para dibujar figuras en un plano es el punto, la cual nos permite generar diversas figuras a partir de una secuencia de puntos.

Para poder usar el punto es necesario que definamos una clase que contenga las coordenadas X e Y del objeto punto creado, además de definir una función que nos permita dibujar al objeto punto.

Variables:

- Coordenada X
- Coordenada Y

Métodos:

- Constructor (default / recibe posición dada)
- Destructor
- Draw (función para dibujar el punto)
- setValues (función para dar los valores al punto)

Variables:

- Punto p1
- Punto p2

La línea

Recordemos que para generar una línea es necesario tener dos puntos que representan los extremos de la línea que serán unidos a través de una secuencia de puntos.

La función línea usa a la librería punto para generar los puntos extremos, después a través de su función “draw”, permite dibujar la secuencia de puntos mediante la primitiva GL_POINTS de OpenGL.

Variables:

- Punto p1
- Punto p2

Métodos:

- Constructor (default / recibe puntos dados)
- Destructor
- Draw (permite dibujar la linea)
- getPointIni (que permite obtener el primer punto)
- getPointFin (que permite obtener el segundo punto)

El triángulo

El triángulo es una figura compleja formada por tres vértices (puntos) unidos a través de líneas (secuencias de puntos), por ello es necesario definir tres objetos punto, generar tres objetos línea que unan los puntos y mandar a llamar a la función Draw de la clase línea.

Variables:

- Punto p1
- Punto p2
- Punto p3

Métodos:

- Constructor (default, recibe tres puntos dados)
- Destructor
- Draw (dibuja el triángulo según p1,p2 y p3)

Variables:

- Punto p1
- Punto p2

Métodos:

- getP1 (permite obtener el primer punto)
- getP2 (permite obtener el segundo punto)
- getP3 (permite obtener el tercer punto)

Los cuadriláteros

Recordemos que un cuadrilátero es una figura compuesta por cuatro vértices que son unidos a través de líneas, pero podemos darnos cuenta que con dos puntos podemos obtener los dos puntos restantes, por ello la clase cuadriláteros sólo requiere de dos puntos con coordenadas distintas, las aristas del cuadrilátero son generadas a través del método Draw de la clase línea que usa los puntos dados y los puntos inferidos.

Variables:

- Punto p1
- Punto p2

Métodos:

- Constructor (default / recibe dos puntos)
- Destructor
- Draw (permite dibujar el cuadrilátero a través de los puntos dados)
- getPointIni (regresa el primer punto dado)
- getPointFin (regresa el segundo punto dado)

El círculo

Para generar un círculo recordemos que necesitamos un centro que será representado por un objeto punto y un radio cualquiera.

Se utiliza la relación:

$$\text{□} = \text{□} \pm \sqrt{(r^2 - (\text{□} - \text{□})^2)}$$

Para obtener todos los puntos de un cuadrante del circulo y posteriormente esos puntos se dibujan en los otros tres cuadrantes.

Variables:

- Punto p del centro
- Radio

Métodos:

- Constructor (default / con centro y radio dado)
- Destructor
- Draw (permite dibujar al círculo)
- getCentro (permite obtener el centro del circulo)
- getRadio (permite obtener el radio del circulo)

Operaciones Básicas con matrices

Métodos:
Escalamiento

Se crea una función que reciba el escalar por el cual se van a multiplicar las coordenadas del punto, luego se asigna dicho escalar en la posición [0][0] y[1][1] con el fin de lograr lo siguiente:

$$\begin{matrix} K & 0 & 0 \\ 0 & K & 0 \\ 0 & 0 & 1 \end{matrix}$$

Una vez hecho, al llamar a la función multMatrix y multPunto se logra obtener algo de la forma x'=Kx e y'=Ky.

Translación

Se crea una función que reciba el desplazamiento en "x" e "y", luego se asigna dicho valor en la posición [0][2] y[1][2] con el fin de lograr lo siguiente:

$$\begin{matrix} 1 & 0 & m\square\square\square? \\ 0 & 1 & m\square\square\square? \\ 0 & 0 & 1 \end{matrix}$$

Una vez hecho, al llamar a la función multMatrix y multPunto se logra obtener algo de la forma x'=x+movX e y'=y+movY.

Rotación

Se crea una función que reciba la cantidad de grados a rotar, luego se aplica dicho valor en los valores de seno y coseno de la matriz con el fin de lograr lo siguiente:

$$\begin{matrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{matrix}$$

Una vez hecho, al llamar a la función multMatrix y multPunto se logra obtener algo de la forma x'=x*cos α + y*sin α e y'=x*sin α + y*cosα

Reflexión

Se crea una función que reciba el tipo de reflexión, en "x", "y" o en ambos ejes, luego se asigna un valor -1 dependiendo del tipo con el fin de lograr lo siguiente:

Para el eje x:

$$\begin{matrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Para el eje y:

$$\begin{matrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Para el eje x e y:

$$\begin{matrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Una vez hecho, al llamar a la función multMatrix y multPunto se logra obtener algo de la forma:

- Caso del eje x: x’ = x ; y’ = -y
- Caso del eje y: x’ = -x ; y’ = y;
- Caso de ambo ejes: x’ = -x ; y’ = -y;

.

Deformación

Se crea una función que reciba el tipo de deformación y la longitud de la deformación en “x” o “y” según sea el tipo, luego se asigna dicho valor en la posición [0][1] o [1][0] con el fin de lograr lo siguiente:

Deformación en eje x:

$$\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Deformación en el eje y:

$$\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Una vez hecho, al llamar a la función multMatrix y multPunto se logra obtener algo de la forma:

- Caso del eje x: x’ = x + shx*(y) ; y’ = y
- Caso del eje y: x’ = x; y’ = y + shy*(x)

Pseudocódigo (para dibujar la línea)

```
lineDraw(Punto p1, Punto p2)

{

   x1,x2,y1,y2 son flotantes;

   x1 = p1.getX();    //Donde getX es una función de la clase punto que regresa la
                      //coordenada x

   y1 = p1.getY();    //Donde getY es una función de la clase punto que regresa la
                      //coordenada y

   x2 = p2.getX();

   y2 = p2.getY();

   m , y son flotantes;
```

```
Si (x2 - x1 != 0)    //comprobación para revisar el denominado de la operación
    m = (y2-y1)/(x2-x1); //Operación para obtener la pendiente
sino
    m=1;                    //Asignación de pendiente para rectas paralelas al eje “y”
y = y1;
si (x2-x1>0)                //Primer caso, donde x2 es mayor que x1
```

```
        para i = x1 hasta x2 hacer

        {

            drawPixel( i ,round(y)); // “i” es la coordenada “x”

            y += m;          //incremento para dibujar el siguiente punto

        }

    sino

        si (x2-x1<0)          //Segundo caso, donde x2 es menor que x1
            para  i = x2 hasta x1 hacer

            {

                drawPixel( i, round(y));

                y += m;

            }

        sino                  //Tercer caso, donde x1=x2
            para  i = y1 hasta y2 hacer

            {

                drawPixel(x1,i);        //La coordenada x nunca se mueve

            }

}
```

Recordemos que para dibujar la línea se toman en cuenta tres casos:

- x_2-x_1 > 0 ; el ciclo va desde la coordenada x1 a x2
- x_2-x_1 < 0 ; el ciclo va desde la coordenada x2 a x1
- x_1=x_2 ; se mueve solamente el eje y

Pseudocódigo (para dibujar el triangulo)

```
triangleDraw (punto p1, punto p2, punto p3)

{

    //Se crean los objetos Línea usando el segundo constructor de la clase
```

```
        Línea l1 (p1, p2);
        Línea l2 (p2, p3);
        Línea l3 (p3, p1);
        l1.lineDraw ();         //Se manda a dibujar la primera línea con el método
                                //lineDraw de la clase Línea
        l2.lineDraw ();
        l3.lineDraw ();
}
```

Pseudocódigo (para dibujar un cuadrilátero)

```
cuadDraw (punto p1, punto p2)
{       //siendo p1 y p2 vértices contrarios
        Punto p3 (p1.getX (), p2.getY ());
        Punto p4 (p2.getX (), p1.getY ());
        Línea l1 (p1, p3);
        Línea l2 (p3, p2);
        Línea l3 (p2, p4);
        Línea l4 (p4, p1);
        l1.lineDraw ();
        l2.lineDraw ();
        l3.lineDraw ();
        l4.lineDraw ();
}
```

Pseudocódigo (para dibujar el círculo)

circleDraw(punto p, r es flotante)

{

```
    xc, yc son flotantes;

    xc = p.getX();

    yc = p.getY();

    y, dis son flotantes;

    para i = 0 hasta n hacer                // ”i” representa la coordenada x

      {

        dis = pow(r,2)-pow((0-i),2);        //El cero representa el origen dado xc
        y = round(0+sqrt(dis));             //El cero representa el origen dado yc
        drawPixel(i+xc,y+yc);               //Primer cuadrante (+,+)
        drawPixel(-i+xc,y+yc);              //Segundo cuadrante (-,+)
        drawPixel(-i+xc,-y+yc);             //Tercer cuadrante (-,-)
        drawPixel(i+xc,-y+yc);              //Cuarto cuadrante (+,-)

      }

}
```

El método draw de círculo se encarga de dibujar el primer cuadrante y cada punto que se dibuja en el primer cuadrante también se dibuja en los otros tres cuadrantes.

Dibujado del escenario

El escenario dibujado en su mayoría está compuesto por rectángulos por lo que se utilizo la reflexión para generar rectángulos al lado contrario del que se crea y de esa manera mantener la simetría; se tiene un solo triangulo en la parte del techo del edificio y alrededor del mismo polígono una estructura de líneas.

Los pilares se dibujaron dentro de un ciclo de 3 repeticiones para generar cada pilar y se utilizo reflexión para generar los pilares del lado contrario.

El pilar central fue hecho de igual manera que los anteriores pero fuera del ciclo mencionado anteriormente dado que la distancia de las líneas que lo conforman es distinta al de los otros pilares.

Por último las capas del sol en la parte superior derecha fueron generadas mediante la clase círculo.

Conclusión

El proyecto permitió reafirmar los conocimientos obtenidos en la primera unidad, permitió recordar que el punto es el objeto principal para generar un escenario y que a través de ellos es posible generar figuras más complejas.

También permitió implementar los conceptos de escalamiento, translación, rotación, deformación y reflexión de polígonos generados a través de líneas utilizando matrices.

BENEMRITA UNIVERSIDAD AUTONOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LAS COMPUTACION

MC. IVAN OLMOS PINEDA

MATERIA: GRAFICACION ALUMNO:

SANDOVAL SALAZ CONSUELO

MATRICULA: 201023593

PERIODO: PRIMAVERA 2015

Introducción

En este trabajo se hizo uso de funciones como glPushMatrix, glPopMatrix, funciones de operadores como translación, rotación y escalamiento. Además de hacer uso de las primitivas de Opengl con la finalidad de elaborar un escenario en

3D.

Desarrollo

En este proyecto se dibujara un escenario en 3D el cual es un salón de química, con un laboratorista, unas mesas, sillas y matraces.

Para comenzar el código declaramos las librerías de Opengl y variables a utilizar, posteriormente procedemos declarar las funciones:

La Funcion void ilumina2(void), se declararon las siguientes funciones:

GLfloat Posicionytipo[], para hacer sombras y avanza la sombra puede realizar con un ciclo for.

GLfloat Colorambiente[], le da la fuerza al color como nitidez.

GLfloat Colordifuso[], cambia tonalidades y colores.

Además se utilizaron las funciones para iluminación GL_POSITION, GL_AMBIENT

y GL_DIFFUSE.

Posteriormente se procedió a declarar diferentes funciones para dibujar diferentes objetos en el escenario, se usaron las funciones:

Void pata (void), void superficie (void), void manija (void), fueron llamados para la función void mesa (void).

Void descanso (void), void tubsill (void), void tubpies (void), void cruz(void), fueron llamados para la función void silla(void).

Void cajonlab(void), void cajaagua(void), void manijalab (void), void suplab (void), void llaves (void), void mezclador (void), fueron llamados para la función void lavabo (void).

Void través (void), void cristal (void), void soportes (void), fueron llamados para la función void estante (void).

Void cuarto (void) es la función que dibujara el cuarto de nuestro escenario.

Void cuello (void), void pecho (void), void pecho (void), void pierna (void), void brazo (void), void ojos (void), void boca (void), fueron llamados en la función void laboratorista ().

Void matraz y void matrazc (double x, double y, double z) dibujaran matraces en las mesas con unas bolitas saliendo de ellas.

Opengl funciona como una máquina de estados los cuales se realizar diversas operaciones o cambios tales como cambiar colores, dibujar polígonos, esferas, trasladar, rotar y escalar dichas figuras.

En la función void conjunto (void) (es la función Display) donde se dibujara nuestro escenario.

Comenzaremos con la función glPushMatrix se llamara a las funciones previamente declaradas como ilumina2, cuarto se cierra ese estado de la máquina. El siguientes funciónes se usara glTranslated(), se llama a la función mesa y se cierra la matriz, se desactivara la iluminación.

Se activara la iluminación glDisable(GL_LITHTING) se activa de nuevo la matriz glPushMatrix(), se procede a rotar, trasladar y escalar con las funciones glRotated, glTranslated y glScalef se llama a la función predefinida lavabo para dibujar 3 diferentes lavabos en el escenario se cierra la matriz.

Procedemos a activar nuevamente la matriz, realizamos una traslación con glTranslated llamamos a la función silla y dibujamos 9 sillas en el escenario se cierra la matriz.

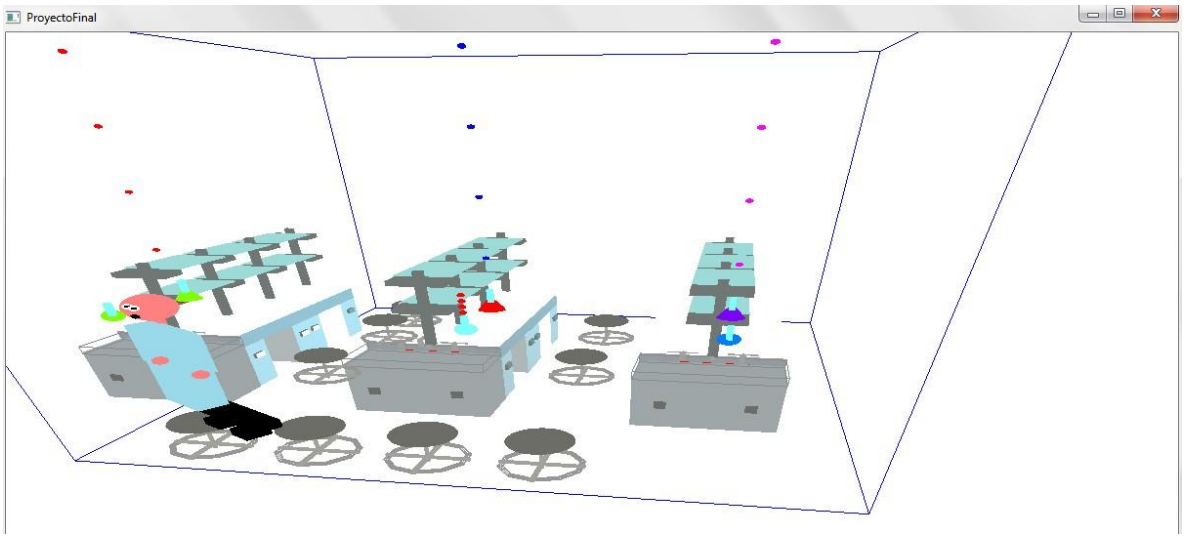
Ahora activamos la matriz y rotamos, trasladamos y escalamos con las funciones glRotated, glTranslated, glScalef y llamamos a la función estante para dibujar 3 estantes y cerramos la matriz.

Finalmente declaramos glPushMatrix, trasladamos con glTranslated, llamamos a la función laboratorista y dibujara a un personaje en el esenario cerramos la matriz con glPopMatrix().

Para lograr el movimiento del personaje, su sombra y las bolitas que salen del matraz se utiliza un for que se inicia de 1 hasta 7 es el número de veces que aparecerán dibujadas en el escenario logrando el efecto de movimiento.

Conclusión

El trabajo realizado nos ayudara a mejorar nuestras habilidades en el manejo de las funciones de Opengl, comprender de una manera más eficiente el funcionamiento de las matrices que utiliza el programa.



El producto final de dicho programa es el siguiente mostrando las respectivas rotaciones escalamientos y traslaciones.

Proyecto Final

Documento Técnico

Richard Alejandre Trigo

21/04/2015

A continuación se tiene el documento técnico del proyecto final de graficación lo que incluye los temas vistos en clase, como lo son: texturas, objetos 3D, iluminación, rotación y traslación.

Introducción

De haber concluido con el temario de la materia de graficación se dejó como proyecto un escenario con todos los conceptos que se habían visto en clase como: la carga de texturas con una librería grafica que en mi caso es soil, la traslación, la rotación la iluminación y sobre todo trabajar con objetos 3D sus respectivas texturas en cada lado.

Conceptos Desarrollados

Los principales conceptos son los que se mencionan en la introducción y los cuales han sido desarrollados en prácticas anteriores, la única diferencia es que en el examen anterior solo se manejaba una cara es decir que todos los objetos eran en

2D por lo que no se dibujaba un cubo sino un cuadro, cosa que en este proyecto sí

se quiere un cubo y por ende fue necesario aplicar la textura a cada una de sus caras para que se visualizada como cubo y no como cuadro y las demás caras en blanco. Otra cosa que también fue necesaria fue trabajar con el eje Z cosa que siempre presenta alguna dificultad que al principio si llevo algo de tiempo poder comprender.

Escenario

Init

Dentro de la función Init la que inicializa se introdujo lo de siempre, las funciones de MatrixMode, prespectiva y el lookat. De

igual forma se insertaron, declararon y habilitaron los arreglos para la iluminación.

```
void init()
{
    glClearColor(0.686275,0.933333,0.933333,0.0);
    glEnable(GL_DEPTH_TEST);
    glShadeModel (GL_SMOOTH);
    GLfloat light_ambient[] = { 1.0, 1.0, 1.0,1.0 };
    GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat light_specular[] = {1.0, 1.0, 1.0, 1.0 };
    GLfloat light_position[] = { 0.0, 100.0, 100.0, 0.0 };
    glLightfv (GL_LIGHT0, GL_AMBIENT, light_ambient);
    glLightfv (GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv (GL_LIGHT0, GL_SPECULAR, light_specular);
    glLightfv (GL_LIGHT0, GL_POSITION, light_position);
    glEnable (GL_LIGHTING);
    glEnable (GL_LIGHT0);
    glDepthFunc(GL_LESS);
    glEnable(GL_DEPTH_TEST);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0,1.0,1.0,100.0);
    glMatrixMode(GL_MODELVIEW);
    gluLookAt(0,0,1,0,0,0,0,1,0);
    glTranslatef(0.0,0.0,20.0);
}
```

Reshape

La función Reshape con sus funciones para el reajuste de las dimensiones.

```
void reshape(int w, int h)
{
glViewport (0, 0, (GLsizei) w, (GLsizei) h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(30.0, (GLfloat) w/(GLfloat) h, 0.001, 1000.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0.0,0.0,-70.0);
ancho=w;
alto=h;
}
```

Imágenes

Primero fue necesario declarar el arreglo para poder vincular la imagen con una posición del arreglo para poderla llamar y de igual forma declarar un arreglo de tipo char donde estarán todas las texturas utilizadas.

```
GLuint tex_2d;
GLuint tempTextureID[13];
char* texture[13] = {
        "9_1.png",
        "11.png",
        "pasto.png",
        "cielo.jpg",
        "metal.png",
        "brick.png",
        "reja.jpg",
        "1.jpg",
        "2.png",
        "3.png",
        "2.jpg",
        "duda.png",
        "mario.png"

      };
```

```
int LoadTextures()
{
    for (int iTex=0;iTex<13;iTex++)
    {
    tempTextureID[iTex] = SOIL_load_OGL_texture
    (
        texture[iTex],
        SOIL_LOAD_AUTO,
        SOIL_CREATE_NEW_ID,
        SOIL_FLAG_MIPMAPS | SOIL_FLAG_INVERT_Y | SOIL_FLAG_NTSC_SAFE_RGB | SOIL_FLAG_COMPRESS_TO_DXT
    );

          glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
          glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
          glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
          glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
    if( 0 == tempTextureID )
    {
       printf( "SOIL loading error: '%s'\n", SOIL_last_result() );
    }
    }
}
```

No se usaron todas las imágenes.

Piso

```
void piso_verde()

{

  glBindTexture(GL_TEXTURE_2D, tempTextureID[2]);

  //Cara frontal
  glBegin(GL_POLYGON);

        glTexCoord2f(0.0f,0.0f);
        glVertex3f(-15.0f,-7.0f,-16.0f);
        glTexCoord2f(1.0f,0.0f);
        glVertex3f(15.0f, -7.0f,-16.0f);
        glTexCoord2f(1.0f,1.0f);
        glVertex3f(15.0f, -6.0f,-16.0f);
        glTexCoord2f(0.0f,1.0f);
        glVertex3f(-15.0f, -6.0f,-16.0f);

  glEnd();

  glBegin(GL_POLYGON);
        glTexCoord2f(0.0f,0.0f);
        glVertex3f(-15.0f,-7.0f,16.0f);
        glTexCoord2f(1.0f,0.0f);
        glVertex3f(15.0f, -7.0f,16.0f);
        glTexCoord2f(1.0f,1.0f);
        glVertex3f(15.0f, -6.0f,16.0f);
        glTexCoord2f(0.0f,1.0f);
        glVertex3f(-15.0f, -6.0f,16.0f);

  glEnd();

glBegin(GL_POLYGON); glTexCoord2f(0.0f,0.0f);
      glVertex3f(15.0f,-7.0f,16.0f);
      glTexCoord2f(1.0f,0.0f); glVertex3f(15.0f, -7.0f,-
      16.0f); glTexCoord2f(1.0f,1.0f);
      glVertex3f(15.0f, -6.0f,-16.0f);
      glTexCoord2f(0.0f,1.0f);
```

```
        glVertex3f(15.0f, -6.0f,16.0f);

glEnd();




glBegin(GL_PO
        LYGON);
        glTexCoord
        2f(0.0f,0.0f);
        glVertex3f(-
        15.0f,-
        7.0f,16.0f);
        glTexCoord
        2f(1.0f,0.0f);
        glVertex3f(-
        15.0f, -7.0f,-
        16.0f);
        glTexCoord
        2f(1.0f,1.0f);
        glVertex3f(-
        15.0f, -6.0f,-
        16.0f);
        glTexCoord
        2f(0.0f,1.0f);
        glVertex3f(-
        15.0f, -
        6.0f,16.0f);

glEnd();




glBegin(GL_PO
        LYGON);
        glTexCoor
        d2f(0.0f,0.0
        f);
        glVertex3f(
        -15.0f,-
        6.0f,-16.0f);
        glTexCoord2f(1.0f,0.0f);
        glVertex3f(-15.0f, -6.0f,16.0f);
        glTexCoord2f(1.0f,1.0f);
        glVertex3f(15.0f, -6.0f,16.0f);
        glTexCoord2f(0.0f,1.0f);
        glVertex3f(15.0f, -6.0f,-16.0f);

    glEnd();




    glBegin(GL_POLYGON);
        glTexCoord2f(0.0f,0.0f);
        glVertex3f(-15.0f,-7.0f,-16.0f);
        glTexCoord2f(1.0f,0.0f);
        glVertex3f(-15.0f, -7.0f,16.0f);
        glTexCoord2f(1.0f,1.0f);
        glVertex3f(15.0f, -7.0f,16.0f);
        glTexCoord2f(0.0f,1.0f);
        glVertex3f(15.0f, -7.0f,-16.0f);

    glEnd();


}
```

Nubes

La función de las nubes fue un conjunto de varias esferas posiciones una encima de otra de diferente forma, es decir de una forma fueron llamadas nubes1, de otra nubes2 y por ultimo nubes3, para posteriormente ser llamadas las 3 en un sola función que junta las 3 con una rotación sobre el eje Y.

```
void nubes_1(){
    glColor3f(1.0,1.0,1.0);
    glPushMatrix();
    glTranslatef(8.0f,1.0f,-9.0f);
    glutSolidSphere (1, 40, 16);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(7.50f,0.0f,-9.0f);
    glutSolidSphere (1, 40, 16);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(8.5f,0.0f,-9.0f);
    glutSolidSphere (1, 40, 16);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(9.5f,1.25f,-9.0f);
    glutSolidSphere (1, 40, 16);
    glPopMatrix();
        }
```

```
void nubes_2(){
    glPushMatrix();
    glTranslatef(-8.0f,1.0f,-9.0f);
    glutSolidSphere (1, 40, 16);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-7.50f,0.0f,-9.0f);
    glutSolidSphere (1, 40, 16);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-8.5f,0.0f,-9.0f);
    glutSolidSphere (1, 40, 16);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-9.5f,1.25f,-9.0f);
    glutSolidSphere (1, 40, 16);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-8.0f,1.35f,-9.0f);
    glutSolidSphere (1, 40, 16);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-7.0f,0.9f,-9.0f);
    glutSolidSphere (1, 40, 16);
    glPopMatrix();
    }
```

```
void nubes_3(){
    glPushMatrix();
    glTranslatef(2.0f,1.0f,-9.0f);
    glutSolidSphere (1, 40, 16);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(1.50f,0.0f,-9.0f);
    glutSolidSphere (1, 40, 16);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(2.5f,0.0f,-9.0f);
    glutSolidSphere (1, 40, 16);
    glPopMatrix();


    }
```

```
void nubes(){


  glPushMatrix();

  glTranslatef(0.0f, 20.0f, 10.0f);

  glPushMatrix();

  glTranslatef(0.0f, 0.0f, -10.0f);

  glRotatef(anguloZ, 0.0f, 10.0f, 0.0f);


  glTranslatef(40.0f, 0.0f, 0.0f);
```

```
glPushMatrix();

    nubes_1(); glPopMatrix(); glPopMatrix();



     glPushMatrix(); glTranslatef(0.0f, 20.0f, 10.0f); glPushMatrix();

    glTranslatef(0.0f, 0.0f, -10.0f);

    glRotatef(anguloX, 0.0f, 10.0f, 0.0f);


    glTranslatef(40.0f, 0.0f, 0.0f);
```

```
nubes_2();
glPopMatrix();
glPopMatrix();
```

```
glTranslatef(0.0f, 20.0f, 10.0f);
glPushMatrix();
glTranslatef(0.0f, 0.0f, -10.0f);

glRotatef(anguloY, 0.0f, 10.0f, 0.0f);
glTranslatef(40.0f, 0.0f, 0.0f);
nubes_3();

glPopMatrix();


glPopMatrix();
```

```
glPushMatrix();
glTranslatef(10.0f, 20.0f, 10.0f);
glPushMatrix();
glTranslatef(0.0f, 0.0f, -10.0f);

glRotatef(anguloZ, 0.0f, 10.0f, 0.0f);
glTranslatef(40.0f, 0.0f, 0.0f);
nubes_1();

glPopMatrix();


glPopMatrix();
```

```
glPushMatrix();
glTranslatef(20.0f, 30.0f, 10.0f);
glPushMatrix();
glTranslatef(0.0f, 0.0f, -10.0f);

glRotatef(anguloX, 0.0f, 10.0f, 0.0f);


glTranslatef(40.0f, 0.0f, 0.0f);


nubes_2();
glPopMatrix();
glPopMatrix();







glPushMatrix();
glTranslatef(40.0f, 50.0f, 10.0f);
glPushMatrix();
glTranslatef(0.0f, 0.0f, -10.0f);

glRotatef(anguloY, 0.0f, 10.0f, 0.0f);


glTranslatef(40.0f, 0.0f, 0.0f);


nubes_3();
glPopMatrix();
glPopMatrix();

}
```

Cubos (casi todo)

```
void tabique(){

  glBindTexture(GL_TEXTURE_2D, tempTextureID[5]);

glBegin(GL_QUADS);

                // Frente
    glNormal3f( 0.0f, 0.0f, 1.0f);

                glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f,  1.0f);
                glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f, -1.0f,  1.0f);
                glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f,  1.0f);
                glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f,  1.0f);

                // parte de Atras
    glNormal3f( 0.0f, 0.0f,-1.0f);

                glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
                glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f,  1.0f, -1.0f);
                glTexCoord2f(0.0f, 1.0f); glVertex3f( 1.0f,  1.0f, -1.0f);
                glTexCoord2f(0.0f, 0.0f); glVertex3f( 1.0f, -1.0f, -1.0f);

                // Arriba
    glNormal3f( 0.0f, 1.0f, 0.0f);

                glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f, -1.0f);
                glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f,  1.0f,  1.0f);
                glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f,  1.0f,  1.0f);
                glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f, -1.0f);

                // Abajo
    glNormal3f( 0.0f,-1.0f, 0.0f);

                glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
                glTexCoord2f(0.0f, 1.0f); glVertex3f( 1.0f, -1.0f, -1.0f);
                glTexCoord2f(0.0f, 0.0f); glVertex3f( 1.0f, -1.0f,  1.0f);
                glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f,  1.0f);

                // lado Derecho
    glNormal3f( 1.0f, 0.0f, 0.0f);
```

```
                glTexCoord2f(1.0f,  0.0f); glVertex3f( 1.0f,  -1.0f,  -1.0f); glTexCoord2f(1.0f,
                1.0f); glVertex3f( 1.0f,  1.0f, -1.0f); glTexCoord2f(0.0f, 1.0f); glVertex3f( 1.0f,
                1.0f,  1.0f); glTexCoord2f(0.0f, 0.0f); glVertex3f( 1.0f, -1.0f,  1.0f);

                // Lado Izquierdo glNormal3f(-1.0f,
          0.0f, 0.0f);

                glTexCoord2f(0.0f,  0.0f);  glVertex3f(-1.0f,  -1.0f,  -1.0f);  glTexCoord2f(1.0f,
                0.0f); glVertex3f(-1.0f, -1.0f,  1.0f); glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f,
                1.0f,  1.0f); glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f, -1.0f);

          glEnd();
```

De esta forma fue como se creó un cubo con las coordenadas de la textura y donde la mayoría de las cosas en mi escenario son cubos por lo que las dimensiones en otros objetos varían muy poco y lo que cambia es la textura utilizada y su posición que se ve en la función display.

Castillo

El castillo fue la mandar a llamar la función ladrillo que es lo mismo que el cubo anterior solo que se cambió la textura y posteriormente solo fue cambiar la posición de cada ladrillo para tener forma de castillo con unos ladrillos en la parte de arriba con unos conos mandando a llamar la función glutSolidCone.

```
void castillo(){

glTranslatef(-10.0f,-5.0f,-8.f);

tabique();

glPushMatrix();
glTranslatef(2.01,0.0f,0.0f);
tabique();
glPopMatrix();

glPushMatrix();
glTranslatef(4.01,0.0f,0.0f);
tabique();
glPopMatrix();

glPushMatrix();
glTranslatef(0.0f,2.0f,0.0f);
tabique();
glPopMatrix();

glPushMatrix();
glTranslatef(2.01,2.0f,0.0f);
tabique();
glPopMatrix();

glPushMatrix();
glTranslatef(4.01,2.0
f,0.0f); tabique();
glPopMatrix();

glPushMatrix();
glTranslatef(0.0f,4.0
f,0.0f); tabique();
glPopMatrix();

glPushMatrix(); glTranslatef(2.01,4.0f,0.0f); tabique();
glPopMatrix();

glPushMatrix(); glTranslatef(4.01,4.0f,0.0f); tabique();
glPopMatrix();

glPushMatrix(); glTranslatef(0.0,0.0f,-2.0f); tabique();
glPopMatrix();

glPushMatrix(); glTranslatef(0.0,0.0f,-4.0f); tabique();
glPopMatrix();

glPushMatrix(); glTranslatef(0.0,0.0f,-6.0f); tabique();
glPopMatrix();

glPushMatrix(); glTranslatef(0.0,2.0f,-2.0f); tabique();
glPopMatrix();

glPushMatrix(); glTranslatef(0.0,2.0f,-4.0f); tabique();
glPopMatrix();
```

```
gl
Pu
sh
Ma
tri
x();
glT
ra
nsl
ate
f(0.
0,2
.0f,
-
6.0
f);
tab
iq
ue(
);
glPopMatrix();


 gl
Pu
sh
M
atr
ix(
);
glT
ra
nsl
ate
f(0.
0,4
.0f,
-
2.0
f);
tab
iq
u
e
(
)
;
glPopMatrix();


g
l
P
u
s
h
M
a
t
r
i
x
(
)
;
g
l
T
r
a
n
s
l
a
t
e
f
(
0
.
0
,
4
.
0
f
u
e
(
)
;
glPopMatrix();


glPushMatrix();
glTranslatef(0.0,4.0f,-6.0f);
tabique();


glPopMatrix();


glPushMatrix();
glTranslatef(2.0,0.0f,-6.0f);
tabique();


glPopMatrix();


glPushMatrix();
glTranslatef(4.0,0.0f,-6.0f);
tabique();


glPopMatrix();


glPushMatrix();
glTranslatef(6.0,0.0f,-6.0f);
tabique();


glPopMatrix();
```

```
                                                    tabique();                                          glPopMatrix();


glPushMatrix();                                                                                         glPopMatrix();
glTranslatef(2.0,2.0f,-6.0f);
tabique();
                                                    glPushMatrix();
glPopMatrix();                                      glTranslatef(8.0,0.0f,-                         glPushMatrix(); glTranslatef(8.0,2.0f,-6.0f); tabique();
                                                    6.0f); tabique();
                                                                                                        glPopMatrix();
                                                    glPopMatrix();


glPushMatrix();
glTranslatef(4.0,2.0f,-6.0f);                                                                           glPushMatrix(); glTranslatef(10.0,2.0f,-6.0f); tabique();
tabique();
                                                    glPushMatrix();                                     glPopMatrix();
glPopMatrix();                                      glTranslatef(10.0,0.0f,-
                                                    6.0f); tabique();


                                                    glPopMatrix();                                      glPushMatrix(); glTranslatef(12.0,2.0f,-6.0f); tabique();
glPushMatrix();
glTranslatef(6.0,2.0f,-6.0f);                                                                           glPopMatrix();
tabique();
                                                    glPushMatrix();
glPopMatrix();


  glPushMatrix();                                                                                       glPushMatrix(); glTranslatef(8.0,4.0f,-6.0f);
glTranslatef(2.0,4.0f,-6.0f);                                                                           tabique();
tabique();
                                                                                                        glPopMatrix();
glPopMatrix();                                                                                          glPushMatrix(); glTranslatef(10.0,4.0f,-6.0f); tabique();


glPushMatrix();                                                                                         glPopMatrix();
glTranslatef(4.0,4.0f,-6.0f);
tabique();                                                                                              glPushMatrix(); glTranslatef(12.0,4.0f,-6.0f); tabique();
glPopMatrix();                                                                                          glPopMatrix();


glPushMatrix();
glTranslatef(6.0,4.0f,-6.0f);
tabique();                                                                                              glPushMatrix(); glTranslatef(10.0,0.0f,-6.0f); tabique();
glPopMatrix();
glTranslatef(12.0,0.0f,-6.0f);                      tabique();                                          glPushMatrix();
```

```
glPopMatrix();

glPushMatrix();

glTranslatef(12.0,0.0f,-6.0f);

glTranslatef(12.0,0.0f,-6.0f);
```

```
tabique();

glPushMatrix();

glTranslatef(10.0,2.0f,-6.0f);

tabique();

glPopMatrix();

glPushMatrix();

glTranslatef(12.0,2.0f,-6.0f);

tabique();

glPopMatrix();

glPushMatrix();

glTranslatef(14.0,2.0f,-6.0f);

tabique();

glPopMatrix();

glPushMatrix();

glTranslatef(10.0,4.0f,-6.0f);

tabique();

glPopMatrix();

glPushMatrix();

glTranslatef(12.0,4.0f,-6.0f);

tabique();

glPopMatrix();

glPushMatrix();

glTranslatef(14.0,4.0f,-6.0f);

tabique();
```

```
tabique();
glPopMatrix();

glPopMatrix();

glPushMatrix();

glTranslatef(12.0,2.0f,-6.0f);

tabique();

glPopMatrix();

glPushMatrix();

glTranslatef(14.0,2.0f,-6.0f);

tabique();

glPopMatrix();

glPushMatrix();

glTranslatef(10.0,4.0f,-6.0f);

tabique();

glPopMatrix();

glPushMatrix();

glTranslatef(12.0,4.0f,-6.0f);

tabique();

glPopMatrix();

glPushMatrix();

glTranslatef(14.0,4.0f,-6.0f);

tabique();
glPopMatrix();
```

```
g
l
P
u
s
h
M
a
t
r
i
x
(
);
g
l
T
r
a
n
s
l
a
t
e
f
(
1
4
.
0
,
0
.
0
f
,-
6
.
0
f
);
```

glPopMatrix();

tabique();

glPopMatrix();

glP

ush

Ma

trix

();

glT

ran

slat

ef(

14.

0,0.

0f,-

4.0f

);

tab

iqu

e();

glPopMatrix();

glTranslatef(12.0,0.0f,-6.0f);

tabique();

glPushMatrix();

glTranslatef(14.0,0.0f,-2.0f);

tabique();

glPopMatrix();

glPushMatrix();

glTranslatef(14.0,0.0f,0.0f);

tabique();

glPopMatrix();

glPushMatrix();

glTranslatef(14.0,2.0f,-4.0f);

tabique();

glPopMatrix();

glPushMatrix();

glTranslatef(14.0,2.0f,-2.0f);

tabique();

glPopMatrix();

glPushMatrix();

glTranslatef(14.0,2.0f,0.0f);

tabique();

glPopMatrix();

glPushMatrix();

glTranslatef(14.0,4.0f,-4.0f);

tabique();

glPopMatrix();

glPopMatrix();

glPushMatrix();

glTranslatef(14.0,4.0f,-

2.0f); tabique();

glPopMatrix();

glPushMatrix();

glTranslatef(14.0,4.0f,0.0

f); tabique();

glPopMatrix();

glPushMatrix();

glTranslatef(12.0,0.0f,0.0

f); tabique();

glPushMatrix(); glTranslatef(10.0,0.0f,0.0f); tabique();

glPopMatrix();

glPushMatrix(); glTranslatef(12.0,2.0f,0.0f); tabique();

glPopMatrix();

glPushMatrix(); glTranslatef(10.0,2.0f,0.0f); tabique();

glPopMatrix();

glPushMatrix(); glTranslatef(12.0,4.0f,0.0f); tabique();

glPopMatrix();

glPushMatrix(); glTranslatef(10.0,4.0f,0.0f); tabique();

glPopMatrix();

glPushMatrix(); glTranslatef(1.75f,6.0f,0.0f); tabique();

glPopMatrix();

glPushMatrix(); glColor3f(1.0f,1.0f,1.0f);

glTranslatef(1.7,7.0f,0.0f); glRotatef(-90,1.0,0.0,0.0);

glutSolidCone(1,1,10,10); glPopMatrix();

glPushMatrix(); glTranslatef(1.5f,6.0f,-3.0f); tabique();

glutSolidCone(1,1,10,10);

glPopMatrix();

glPopMatrix();

```
glglTranslatef(14.0,0.0f,-2.0f);
Pus
hM
atri
x();
glC
olor3f(1.0f,1.0f,1.0f);
glTr
ansl
atef
(1.
5,
7.
0f,
-
3.
0f
);
glR
ot
at
ef
(-
90
,1.
0,
0,
0.
0f
);
glut
Soli
dCo
ne(
1,1
,10
,10
);
glPo
pMa
trix
();
glPu
shM
atri
x();
glTr
ansl
atef
(11.
5f,
6.0
f,-
6.0
f);
tabi
que
();
glPo
pMa
trix
();
glut
Soli
dCo
ne(
1,1
,10
,10
);
```

glP
glTranslatef(14.0,0.0f,-2.0f);

ush

Mat

rix()

;

glC

olor

3f(1.

0f,1.

0f,1.

0f);

glTr

ansl

atef

(11.

5f,7.

0f,-

6.0f)

;

glR

otat

ef(-

90,1

.0,0.

0,0.

0);

glPopMatrix();

glPopMatrix();

glutSolidCone(1,1,10,10);

```
glPopMatrix();

glPushMatrix();

glTranslatef(14.0f,6.0f,-3.0f);

tabique();

glPopMatrix();
```

```
glPushMatrix();
glColor3f(1.0f,1.0f,1.0f);
glTranslatef(14.0f,7.0f,-3.0f);
glRotatef(-90,1.0,0.0,0.0);

glutSolidCone(1,1,10,10);
```

```
glPopMatrix();

glPushMatrix();

glTranslatef(12.0f,6.0f,0.0f);

tabique();

glPopMatrix();
```

```
glPushMatrix();
glColor3f(1.0f,1.0f,1.0f);
glTranslatef(12.0f,7.0f,0.0f);
glRotatef(-90,1.0,0.0,0.0);

glutSolidCone(1,1,10,10);
```

```
glPopMatrix();

glPushMatrix();
glTranslatef(7.0f,0.0f,0.0f);
glPushMatrix();
glTranslatef(0.0f,y,0.0f);
reja();

glPopMatrix();

glPopMatrix();
```

Display

En la parte de la función display es donde se mandan a llamar todas las funciones que se crearon es decir donde se acomodan todas las coordenadas y se selecciona la textura a utilizar, donde no se habilita la textura dentro de la función sino en la función display para que habilites solo una vez para todas las texturas que vayas a utilizar.

Primero se limpian los buffers de color y de profundidad y se manda a llamar la función de nubes antes de habilitar la textura, posteriormente se habilita la textura y se mandan a llamar todas las funciones, con sus push y pop para que todo cambio que le haga a la matriz no afecte a los demás objetos que se están mandando a llamar.

```
void display()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    nubes();
    glEnable(GL_TEXTURE_2D);

    piso_verde();
```

atrix();

```
void display()

{ glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);
  nubes();
  glEnable(GL_TEXTURE_2D);



  piso_verde();



  glPushMatrix();
  glTranslatef(10.0f,10.0f,0.0f);
  cubo();
  glPopMatrix();



  glPushMatrix();
  glTranslatef(2.0f,10.0f,10.0f);
  dudas();
  glPopMatrix();



  glPushMatrix();

  glTranslatef(-2.0f,10.0f,10.0f);


  dudas();


  glPopMatrix();



  glPushMatrix();
  glTranslatef(10.0f,10.0f,-2.0f);
  dudas();
  glPopMatrix();



  glPushMatrix(); glTranslatef(0.0f,-
  5.0f,10.0f); glPushMatrix();
  glTranslatef(marioX,marioY,marioZ);
  mario();
```

```
  glPushMatrix();

  glTranslatef(0.0f,10.0f,10.0f);
  cubo();
    glPopMatrix();


    glPopMatrix();



    glPushMatrix();
    glTranslatef(10.
    0f,10.0f,2.0f);
    dudas();

    glPopMatrix();



glTranslatef(0.0f,p,0.0f);



                    glPopMatrix();

                    glEnable(GL_TEXTURE_2D);
```

```
glPopMatrix();


glPushMatrix();


glPopMatrix();


glPushMatrix(); castillo();
glPopMatrix();


glPushMatrix(); glTranslatef(10.0f,0.0f,-10.0f);
glDisable(GL_TEXTURE_2D);
glPopMatrix();


glPushMatrix();

glTranslatef(-10.0f,0.0f,10.0f);

piso2();

glPopMatrix();


glPushMatrix(); glTranslatef(10.0f,0.0f,10.0f);
glPopMatrix();
piso2();

glPopMatrix();


glPushMatrix(); glDisable(GL_TEXTURE_2D);
glTranslatef(10.0f,0.0f,10.5f); glPushMatrix();


planta(); glPopMatrix();
glEnable(GL_TEXTURE_2D); glPopMatrix();

glDisable(GL_TEXTURE_2D);
glutSwapBuffers();
```

```
glPushMatrix();
    void display()
glDisable(GL_TEXTURE_2D);
glTranslatef(-10.0f,0.0f,10.3f);
glPushMatrix(); glTranslatef(0.0f,q,0.0f);
planta();
glPushMatrix();
glDisa
ble(GL
_TEXT
URE_2
D);
glTran
slatef(1
0.0f,0.0
f,-9.5f);
glPush
Matrix
();
glTran
slatef(0
.0f,r,0.0
f);
planta(
);
glPopMatrix();
glEnable(GL_TEXTURE_2D);
glPopMatrix();
glPushMatrix();
glTranslatef(0.0f, 0.0f, 0.0f);
glPushMatrix();
glTrans
latef(-
2.5f, -
5.0f, -
4.0f);
glRotat
ef(perro
, 0.0f,
1.0f,
0.0f); perro();
glPopMatrix();
glPopMatrix();
glEnable(GL_TEXTURE_2D);
glPopMatrix();
glPushMatrix(); glTranslatef(-
6.0f,0.0f,10.0f); glPushMatrix();
glTranslatef(yLocation,1.3f,0.0f);
piso1();
glPopMatrix();
glPushMatrix();
glTranslatef(10.0f,0.0f,7.0f);
glPushMatrix();
glTranslatef(0.0f,1.3f,zLocation);
piso1();
glPopMatrix();
glPopMatrix();
glFlush();
```

Lo que se vio en la página anterior fue el llamado de las funciones con su posición usando la función de glTranslatef, en algunas la de glRotatef como se utilizó en el proyecto anterior el movimiento de ida y vuelta de rotación sobre el eje Z para el caso de perro blanco que está en medio del escenario, y a continuación están las pequeñas condiciones que hicieron el movimiento posible.

```
if (movingUp)
    yLocation -= 0.05f;
else
    yLocation += 0.05f;

if (yLocation < 0.0f)
    movingUp = false;
else if (yLocation > 12.0f)
    movingUp = true;

if (Up)
    zLocation -= 0.05f;
else
    zLocation += 0.05f;

if (zLocation < -14.0f)
    Up = false;
else if
    (zLocati
    on >
    0.0f) Up
    = true;

if (down)
    y -= 0.05f;
else
    y +=
    0.05f;

if (y < 1.0f)
    down = false; else if (y > 7.0f)
                down = true;

if (d)
        p -= 0.04f;
else
        p += 0.09f;

if (p < -0.2f)
        d = false;
    else if (p > 2.5f)
        d = true;

if (a1)
        q -= 0.09f;
else
```

```
q += 0.04f;

if (q < -0.2f)

a1 = false; else if (q > 2.5f) a1

= true;
```

```
if (a2)

r
-
=
0
.
0
5
f
;

else

r
+
=
0
.
0
5
f
;

if (r < -0.2f)

a
2
=
f
a
l
s
e
;
e
l
s
```

```
e if (r > 2.5f) a2

= true;

if (a3)

perro -= 0.5f;

else

perro += 0.5f;

if (perro < -185.f)

a3 = false;

else if (perro > 10.5f)

a3 = true;

anguloZ+=0.05f;
anguloX+=0.07f;
anguloY+=0.04f;
n1-=0.005f;
n2-=0.009f;
n3-=0.001f;
```

Teclado

Se utilizaron 2 funciones del teclado uno con las flechas con la función de glutSpecialFunc(keyboardown); y para las teclas de: a, s, w, d, o, l la función de glutKeyboardFunc(keyboard); donde las flechas fueron para mover la función lookat y las teclas (letras) para mover a mario sobre los 3 ejes.

```
GLfloat unidad =1.0f;
void keyboard(unsigned char key, int x, int y)
{
    switch(key)
    {
    case 'a':
      marioX-=unidad;
      break;
    case 'd':
      marioX+=unidad;
      break;
    case 'w':
      marioZ-=unidad;
      break;
    case 's':
      marioZ+=unidad;
      break;
    case 'o':
      marioY+=unidad;
      break;
    case 'l':
      marioY-=unidad;
      break;

    case 27:   // escape
      exit(0);
      break;
    }
}
```

```
GLfloat unidad =1.0f;
void keyboard(unsigned char key, int x, int y)
{
    switch(key)
    {
    case 'a':
      marioX-=unidad;
      break;
    case 'd':
      marioX+=unidad;
      break;
    case 'w':
      marioZ-=unidad;
      break;
    case 's':
      marioZ+=unidad;
      break;
    case 'o':
      marioY+=unidad;
      break;
    case 'l':
      marioY-=unidad;
      break;

    case 27:   // escape
      exit(0);
      break;
    }
}
```

Iluminación

Se declararon los arreglos dentro de la función de Init para posteriormente habilitar la iluminacíon:

```
GLfloat light_ambient[] = { 1.0, 1.0, 1.0,1.0 };
GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat light_specular[] = {1.0, 1.0, 1.0, 1.0 };
GLfloat light_position[] = { 0.0, 100.0, 100.0, 0.0 };
glLightfv (GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv (GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv (GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv (GL_LIGHT0, GL_POSITION, light_position);
glEnable (GL_LIGHTING);
glEnable (GL_LIGHT0);
```

Posteriormente se inserta la función de glMaterialfv en la parte donde creo que mi piso:

```
GLfloat mat_ambient_esfera2[] = {1.0, 1.0, 1.0, 1.0f};
GLfloat mat_diffuse_esfera2[] = {1.0, 1.0, 0.0, 0.0f};
GLfloat mat_specular_esfera2[] = {0.8, 0.8, 0.8, 1.0f};

glMaterialfv (GL_FRONT, GL_AMBIENT, mat_ambient_esfera2);
glMaterialfv (GL_FRONT, GL_DIFFUSE, mat_diffuse_esfera2);
glMaterialfv (GL_FRONT, GL_SPECULAR, mat_specular_esfera2);
glMaterialf (GL_FRONT, GL_SHININESS, 100.0f);
```

De igual forma a las nubes para un toque de color azul:

```
GLfloat mat_ambient_esfera2[] = {0.10, 0.10, 1.0, 1.0f};
GLfloat mat_diffuse_esfera2[] = {1.0, 1.0, 1.0, 1.0f};
GLfloat mat_specular_esfera2[] = {0.1, 0.1, 0.1, 1.0f};

glMaterialfv (GL_FRONT, GL_AMBIENT, mat_ambient_esfera2);
glMaterialfv (GL_FRONT, GL_DIFFUSE, mat_diffuse_esfera2);
glMaterialfv (GL_FRONT, GL_SPECULAR, mat_specular_esfera2);
glMaterialf (GL_FRONT, GL_SHININESS, 100.0f);
```

Y por ultimó a las esferas que suben y bajan (flores):

```
GLfloat mat_ambient_esfera2[] = {0.10, 0.1, 0.1, 1.0f};
GLfloat mat_diffuse_esfera2[] = {0.0, 1.0, 0.0, 0.0f};
GLfloat mat_specular_esfera2[] = {0.8, 0.8, 0.8, 1.0f};

glMaterialfv (GL_FRONT, GL_AMBIENT, mat_ambient_esfera2);
glMaterialfv (GL_FRONT, GL_DIFFUSE, mat_diffuse_esfera2);
glMaterialfv (GL_FRONT, GL_SPECULAR, mat_specular_esfera2);
glMaterialf (GL_FRONT, GL_SHININESS, 100.0f);
glDisable(GL_TEXTURE_2D);
glTranslatef(10.0f,0.0f,10.5f);
glPushMatrix();
glTranslatef(0.0f,p,0.0f);
planta();
```

Función Main

Lugar donde se mandan a llamar todas las funciones necesarias para la visualización.

```
int main(int argc, char **argv)
{

    glutInit(&argc, argv);
    PlaySound("mario.wav", NULL, SND_FILENAME | SND_LOOP | SND_ASYNC);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(ancho, alto);
    glutCreateWindow("Proyecto Final");
    LoadTextures();
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutIdleFunc(idle);
    glutKeyboardFunc(keyboard);
    glutSpecialFunc(keyboardown);
    glutMainLoop();
    return 0;
}
```

Sonido

Decidí volver a ponerle sonido ya le da un toque diferente amigable al momento de ver el escenario.

Conclusión

Lo más laborioso de este proyecto fue jugar con las coordenadas ya que me estaba confundiendo ya que en otra materia de matemáticas recuerdo que el eje y era el z y viceversa y como me quede con esa idea fue un poco difícil cambiar mi perspectiva pero de después de ubicarme solo fue cuestión de moverme dentro de área y ya fue todo.

Escenario Mario Bros 3D

2015

GRAFICACIÓN 2015

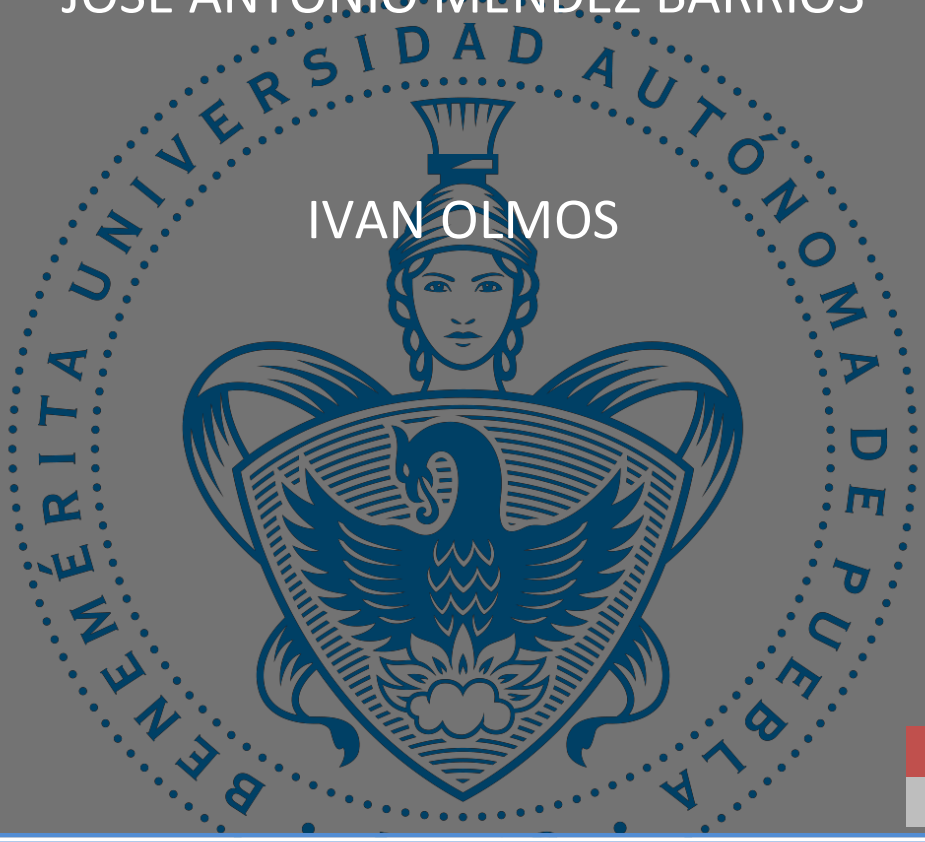
PROYECTO 2

BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA –

FACULTAD CIENCIAS DE LA COMPUTACIÓN -

JOSÉ ANTONIO MÉNDEZ BARRIOS

IVAN OLMOS



INTRODUCCIÓN:

La computación gráfica 2D es la generación de imágenes digitales por computadora - sobre todo de modelos bidimensionales (como modelos geométricos, texto e imágenes digitales 2D)

y por técnicas específicas para ellos. La palabra puede referirse a la rama de las ciencias de la computación que comprende dichas técnicas, o a los propios modelos.[1]

La computación gráfica 2D se utiliza principalmente en aplicaciones que fueron desarrolladas originalmente sobre tecnologías de impresión y dibujo tradicionales, tales

comotipografía, cartografía, dibujo técnico, publicidad, etc. En estas aplicaciones,

la imagen bidimensional no es sólo una representación de un objeto del mundo real, sino un artefacto independiente con valor semántico añadido; los modelos bidimensionales son preferidos por lo tanto, porque dan un control más directo de la imagen que los gráficos 3D por computadora (cuyo enfoque es más semejante a la fotografía que a la tipografía).

OBJETIVO DEL PROYECTO

Implementar un escenario en 2D con todas las herramientas vistas y aprendidas en clase. Nuestro escenario debe presentar las principales operaciones como rotaciones, traslaciones, etc.

ANALISIS DE LA ESTRUCTURA DE NUESTRO CODIGO

FUNCIONES

Declaramos nuestras librerías a utilizar

```
#include <unistd.h>
#include <GL/glut.h>
#include <GL/gl.h>
#include <math.h>
#include <stdlib.h>
```

Como deseamos que en nuestro escenario aparezcan nubes, entonces vamos a crear variables que nos van a permitir manipularlas.

```
int rad=100;
int x=1, y=1,z=1,w=1,e=1,f=1,g=1,h=1;          //Variables para la nube 1
int x2=1, y2=1,z2=1,w2=1,e2=1,f2=1,g2=1,h2=1;  //Variables para la nube 2
int x3=1, y3=1,z3=1,w3=1,e3=1,f3=1,g3=1,h3=1;  //Variables para la nube 3
int x4=1, y4=1,z4=1,w4=1,e4=1,f4=1,g4=1,h4=1;  //Variables para la nube 4
```

Declaramos variables que nos van a permitir disparos o balas para nuestro escenario

```
//VARIABLES PARA LAS BALAS

int x5=1, y5=1,z5=1,w5=1,e5=1,f5=1,g5=1,h5=1; //Variables para las balas
double ang=0,b=0,c=0;

//Declaramos variables que nos vana a permitir manipular el movimiento y la posición de nuestra figura
int x1=1;
```

Ahora incluimos nuestra función para inicializar nuestro escenario, donde van incluidas las medidas que proyectará este mismo.

```
void inicializa(void)
{

glClearColor(0.0,0.0,1.0,0.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0, 1000.0, 0.0, 1000.0);// el ancho y largo de nuestra pantalla
}
```

A continuación agregamos una función que nos va a permitir construir figuras de manera circular:

```
void circulo(int x, int y, int radio){
    int angulo=0;
    glBegin(GL_TRIANGLE_FAN);
    glVertex2f(x,y);

    for (angulo=0;angulo<=360; angulo+=6){ glVertex2f(x + sin(angulo) * radio, y + cos(angulo) * radio);}
    glEnd();}
```

Enseguida una función para poder construir el SOL de nuestro escenario, aclarar que el sol no va a ser construido con la función de circulo (declarada anteriormente) debido que se desea construir un sol con algunas malformaciones (rayos) y no completamente circular.

```
void sol(int x,int y,int radio){

      int angulo = 1;
      glBegin(GL_TRIANGLE_FAN); glColor3f(0.0,0.0,0.0);
      glVertex2f(x,y);
       glColor3f(1.0,1.0,0.0);


       for(angulo=0;angulo<=360;angulo+=5)  {glVertex2f(x+sin(angulo)*radio,y+cos(angulo)*radio);}
       glEnd();
       }
```

También vamos a implementar una función que nos construya el arma de nuestro personaje

```
void arma(int x,int y,int radio){

       int angulo = 100;
       glBegin(GL_TRIANGLE_FAN); glColor3f(0.0,0.0,0.0);
       glVertex2f(x,y);
        glColor3f(1.0,1.0,1.0);


        for(angulo=0;angulo<=360;angulo+=3)   {glVertex2f(x+sin(angulo)*radio,y+cos(angulo)*radio);}
        glEnd();
        }
```

CREACIÓN DE NUESTRO ESCENARIO

Ahora vamos a empezar a armar nuestro escenario, llamando nuestras funciones.

```
void escenario(void)
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

Invocando a la función círculo vamos a crear la cabeza de nuestro personaje

```
glColor3f(1.0,0.5,0.0);
circulo (300,350,80);
```

Creamos el cuerpo de nuestro personaje

```
glBegin (GL_TRIANGLES);
glColor3f(0.0,1.0,0.0);
glVertex2f(300,275);
glVertex2f(230,100);
glVertex2f(380,100);
glEnd();
```

Y enseguida volvemos a invocar nuestra función círculo, esto para generar nuestras nubes. Para la creación de estas tuvimos que pegar o unir varios círculos para poder dar forma a nuestras nubes. En total fueron 4 nubes aparecidas en nuestro escenario.

```
//nube 1
glColor3f(1.0,1.0,1.0);
if(x>=1 && x<500){circulo(100+1*x,800,20);x++;}
if(y>=1 && y<500){circulo(130+1*y,790,20);y++;}
if(z>=1 && z<500){circulo(100+1*z,790,20);z++;}
if(w>=1 && w<500){circulo(120+1*w,810,20);w++;}
if(e>=1 && e<500){circulo(120+1*e,770,20);e++;}
if(f>=1 && f<500){circulo(150+1*f,810,20);f++;}
if(g>=1 && g<500){circulo(170+1*g,800,20);g++;}
if(h>=1 && h<500){circulo(160+1*h,780,20);h++;}


if(x==500){ x=1;}
if(y==500){ y=1;}
if(z==500){ z=1;}
if(w==500){ w=1;}
if(e==500){ e=1;}
if(f==500){ f=1;}
if(g==500){ g=1;}
if(h==500){ h=1;}
```

Para los ojos de nuestro personaje, volvemos a invocar a nuestra función circulo

```
glColor3f(1.0,1.0,1.0);
circulo(270,380,25);
glColor3f(0.0,0.0,0.0);
circulo(270,380,19);
glColor3f(1.0,1.0,1.0);
circulo( 270+ sin(b) * 10,380 - cos(b) * 10,4);
glEnd();
glColor3f(1.0,1.0,1.0);
circulo(330,380,25);
glColor3f(0.0,0.0,0.0);
circulo(330,380,19);
glColor3f(1.0,1.0,1.0);
circulo( 330+ sin(b) * 10,380 - cos(b) * 10,4);
glEnd();
glBegin (GL_LINES);
glColor3f(0.0,0.0,0.0);
```

Enseguida creamos su boca

```
glVertex2i(260,320); //
glVertex2i(340,320);
glEnd();
glBegin(GL_QUADS);
glColor3f(1.0,1.0,1.0);
glVertex2f(290,320);
glVertex2f(300,320);
glVertex2f(300,310);
glVertex2f(290,310);
glEnd();
```

Y también sus brazos, estos simulando sostener un arma

```
glLineWidth(5);
glBegin (GL_LINES);

glColor3f(0.9,0.8,0.8);
glVertex2i(270,200);
glVertex2i(220,170);
glVertex2i(220,170);
glVertex2i(270,120);
glEnd();
glBegin (GL_LINES);
glColor3f(0.9,0.8,0.8);
glVertex2i(335,200);
glVertex2i(400,180);
glVertex2f(370,120);
glEnd();

glBegin (GL_LINES);
glColor3f(0.0,0.0,0.0);
glVertex2i(300,370);
glVertex2i(380,320);
glEnd();
b+=0.02;
```

Con la función círculo también vamos a crear nuestras balas de fuego:

```
//----------------BALAS-------------------------------

glColor3f(1.0,0.0,0.0);
if(x5>=1 && x5<500){circulo(390+1*x5,200,20);x5++;}
if(x5==500){ x5=1;}

glColor3f(1.0,0.0,0.0);
if(x5>=1 && x5<500){circulo(470+1*x5,200,20);x5++;}
if(x5==500){ x5=1;}

glColor3f(1.0,0.0,0.0);
if(x5>=1 && x5<500){circulo(570+1*x5,200,20);x5++;}
if(x5==500){ x5=1;}

glColor3f(1.0,0.0,0.0);
if(x5>=1 && x5<500){circulo(670+1*x5,200,20);x5++;}
if(x5==500){ x5=1;}

glColor3f(1.0,0.0,0.0);
if(x5>=1 && x5<500){circulo(770+1*x5,200,20);x5++;}
if(x5==500){ x5=1;}

glColor3f(1.0,0.0,0.0);
if(x5>=1 && x5<500){circulo(870+1*x5,200,20);x5++;}
if(x5==500){ x5=1;}
```

Ahora si mandamos a invocar nuestra función sol para crear a este mismo.

```
//----------------------SOL-----------------------------------------

//Ajustamos coordenadas de movimiento para nuestra figura

                if(x1>=1 && x1<800){
                sol(10+x1,880,80);
                 x1++;
                 if(x1==800){

                            x1=1;

                 }
                 }
```

Por último mandamos a llamar a nuestra función arma, para crear esta misma.

```
//----------------------ARMA-----------------------------------
//Ajustamos coordenadas de movimiento para nuestra figura
                glColor3f(1.0,0.0,0.0);
                if(x7>=1 && x7<200){
                arma(300+x7,170,50);
                 x7++;
                 if(x7==30){

                           x7=10;

                 }
                 }
```

Al final, en nuestra función principal, únicamente manipulamos las medidas de nuestra ventana:

```
int main (int argc, char** argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA| GLUT_DEPTH);
glutInitWindowSize(700,700);
glutInitWindowPosition(10,10);
glutCreateWindow("LOS DIAS SE PASAN VOLANDO");
inicializa();
glutDisplayFunc(escenario);
glutIdleFunc(escenario);
glutMainLoop();
return 0;
}
```

SALIDA/CAPTURA DE PANTALLA

Hoja de presentación para escenario 3D en OpenGL

Materia: Graficación

Nombre: Jesús Paleta Islas

Desarrollo del tema

En este examen se busca desarrollar un escenario en 3D mediante predirectivas de OpenGL. El escenario deberá incluir rotaciones, traslaciones, escalamiento, manejo de texturas, de materiales y de iluminación.

El bosquejo del escenario es representar el contraste entre un desierto y un bosque de Egipto. En el desierto se pueden encontrar tres pirámides a distinta profundidad y en el bosque árboles a distintas distancias.

En primera instancia, se buscó desarrollar una función para dibujar las pirámides, la cual ya se había escrito en una práctica pasada. Así que se recicló este código.

Tras crear un primer bosquejo de las partes del escenario (sin colores, ni texturas, ni iluminación) se empezó desarrollando la función init() donde colocaríamos el manejo de texturas así como la configuración predeterminada. En las funciones ilumina() y material() se colocó el manejo de iluminación y materiales respectivamente.

La carga de texturas se realizó con las funciones que ofrece glut.h, igual que en el escenario 2D. Esto debido a la facilidad que da a la hora de manejar texturas y a que se evita utilizar directivas extras.

Se utilizaron texturas para las pirámides, y para la arena del desierto. El único problema con este método es la necesidad de convertir las imágenes (ya sean de tipo .png o .jpg) a tipo “tga”. Sin embargo, gracias a Internet esto se puede hacer en un par de *clicks* y completamente en línea sin tener que descargar aplicaciones de terceros.

Otro problema que surgió al manejar texturas es que debía ser para un escenario 3D, pero con

```
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
        glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);
        glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);
        glTexGeni(GL_R, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);
```

al momento de habilitar texturas todo estaba solucionado.

Para dibujar los árboles se utilizaron varios conos dibujados fácilmente con la función glutSolidCone() y realizando traslaciones. Para el Sol se utilizó glutSolidSphere() y estuvo rotando de acuerdo a un ángulo que empezó en 0 y que iría aumentando más 0.15.

La iluminación la definimos mediante glLightfv() y los valores de la luz como su posición se almacenan en arreglos de tipo GLfloat. La activamos mediante glEnable(), en este caso fueron dos luces. De manera similar los materiales se definieron mediante glMaterialfv(). Los materiales se activan para las pirámides y para los árboles.

El principal reto a la hora de programar fue tener cuidado en como colocábamos los llamados a las funciones glPushMatrix() y glPopMatrix() de manera que la rotación y traslación de ciertos objetos no afectaran al escenario en general.

Como conclusión, se puede decir que OpenGL ofrece muchas herramientas para el dibujado en 3D, pero aun así fue necesario desarrollar varias funciones para poder darle un mejor aspecto a nuestro escenario. Finalmente, las traslaciones y rotaciones dan una apariencia mejorada a nuestro escenario y nos hacer ver la utilidad de *MainLoop*.

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

REPORTE TÉCNICO:

PROYECTO FINAL

PRESENTA:

LUIS ALBERTO SILVA ESCAMILLA

MATERIA:

GRAFICACIÓN

PRIMAVERA 2015

Introducción

El proyecto final de la materia de Graficación tiene por objetivo poner en práctica todos los recursos aprendidos a lo largo de este cuatrimestre; correspondiente al periodo Primavera 2015.

Además de poner en práctica todos los conceptos y herramientas, se tiene por objetivos que el alumno refuerce mediante la investigación los conceptos que no le hayan quedado claros, pero también adentrarse más acerca de algún concepto o herramienta que particularmente le interese. El otro objetivo es que el maestro pueda comprobar que efectivamente al alumno le quedaron claros todos los conceptos manejados a lo largo del curso, y que también el alumno se muestre capaz de hacer converger todas las herramientas, recursos y técnicas que aprendió en esta asignatura.

Practica de laboratorio:

El proyecto final consiste en realizar un escenario 3D con elementos del mismo tipo, texturas y además los últimos elementos vistos en el curso como lo son profundidad e iluminación. Adicionalmente se incluirán transformaciones en OpenGL como lo son: escalamientos, rotación y traslación.

El escenario es libre o adaptado, sin embargo si es preciso que lleve todos los elementos antes mencionados.

EL procedimiento seguido por el alumno, fue además de dividir su tiempo, también llevo a cabo una planeación del mismo. Por lo tanto, el alumno dividió el proyecto en 3 etapas:

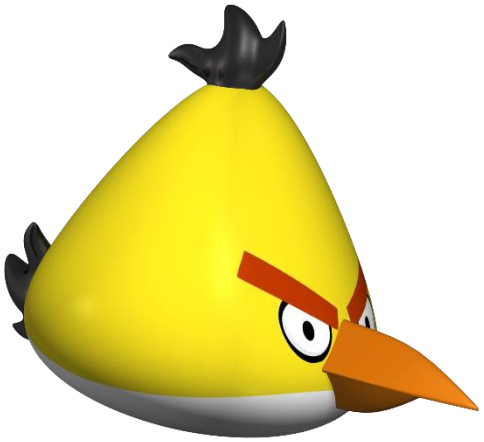
1. Elección del tipo de escenario a elegir (libre o adaptado).
2. Elección de los personajes o elementos a modelar.
3. Diseño de cada figura.
4. Implementación de cada figura.
5. Animación del escenario (traslación y rotación).
6. Investigación relativa a texturas.

7. Adición de elementos texturizados.
8. Unión de todos los componentes.

La finalidad de la planeación tomando estos 8 pasos, fue considerar el proyecto por etapas, ya que como algunos conceptos aun no estaban claros (tales como texturas) fue necesario considerar tiempos y etapas para ubicarlos y saber si es que se encontraba en una etapa crítica o no, a que se debía dedicar más tiempo y que elementos ya deberán estar para cierta fecha.

Por lo tanto, de acuerdo al listado anterior se optó por diseñar un escenario libre, sin embargo tomando como referencia los personajes del popular juego “Angry Birds”. Ahora bien, “Angry Birds” es en realidad un juego en 2D, no obstante por las formas de los personajes (no tan complejas ni necesarias de muchas figuras) los personajes son adaptables para su modelado en 3D y de hecho tomando en cuenta los tiempos y otros proyectos de otras materias, se estableció como primer requisitos, que las figuras a modelar no constaran de muchos objectos.

De tal manera, se optó por modelar únicamente 3 personajes: 2 aves y un cerdo.

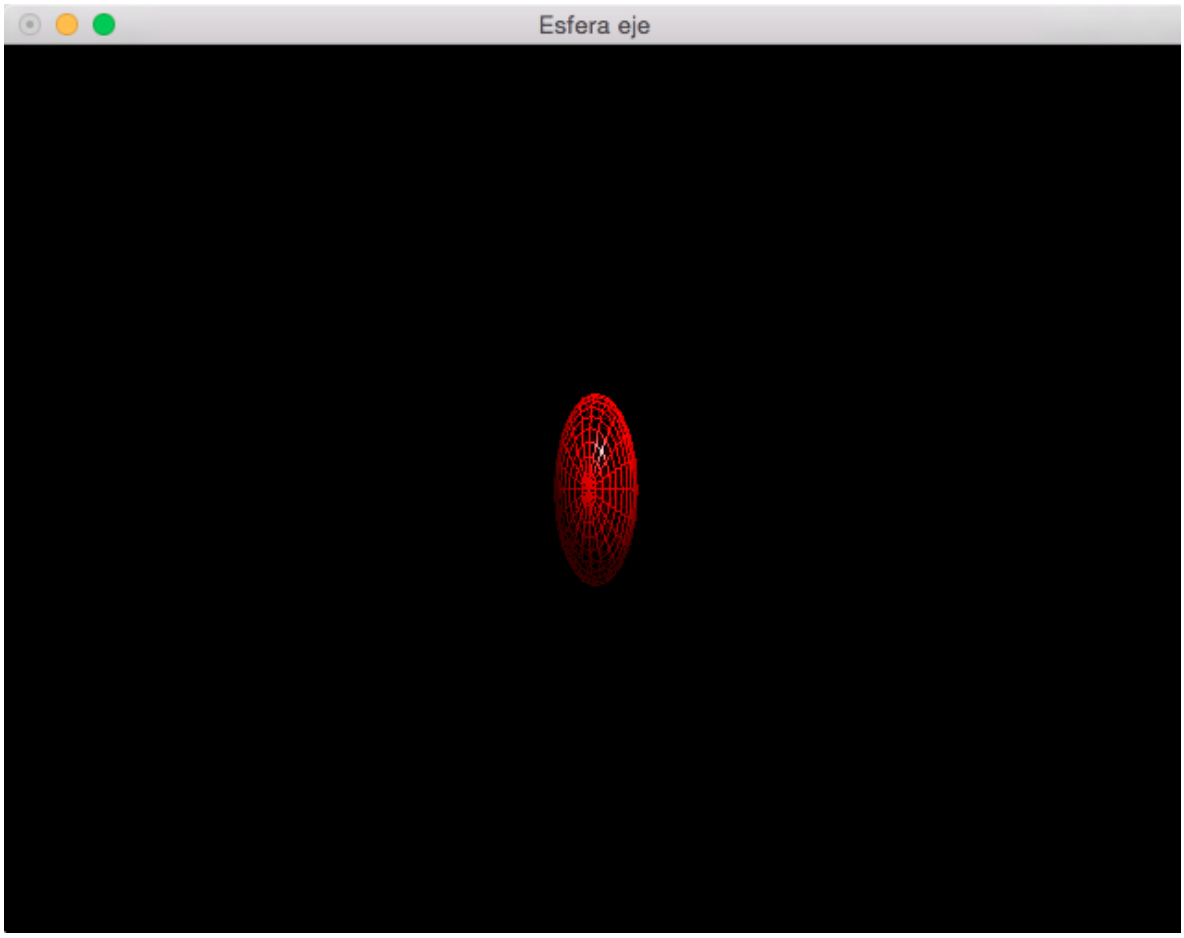


No obstante, primero se tiene que considerar el ambiente 3D; esto es, definir el espacio de trabajo. Por ejemplo, había que considerar si se utilizaría glutLookAt(), GL_PERSECTIVE, GL_PROJECTION. No obstante, para evitar tantos parámetros se eligieron tomar los parámetros establecidos por default cuando se crea una Proyecto nuevo en Dev-C++.

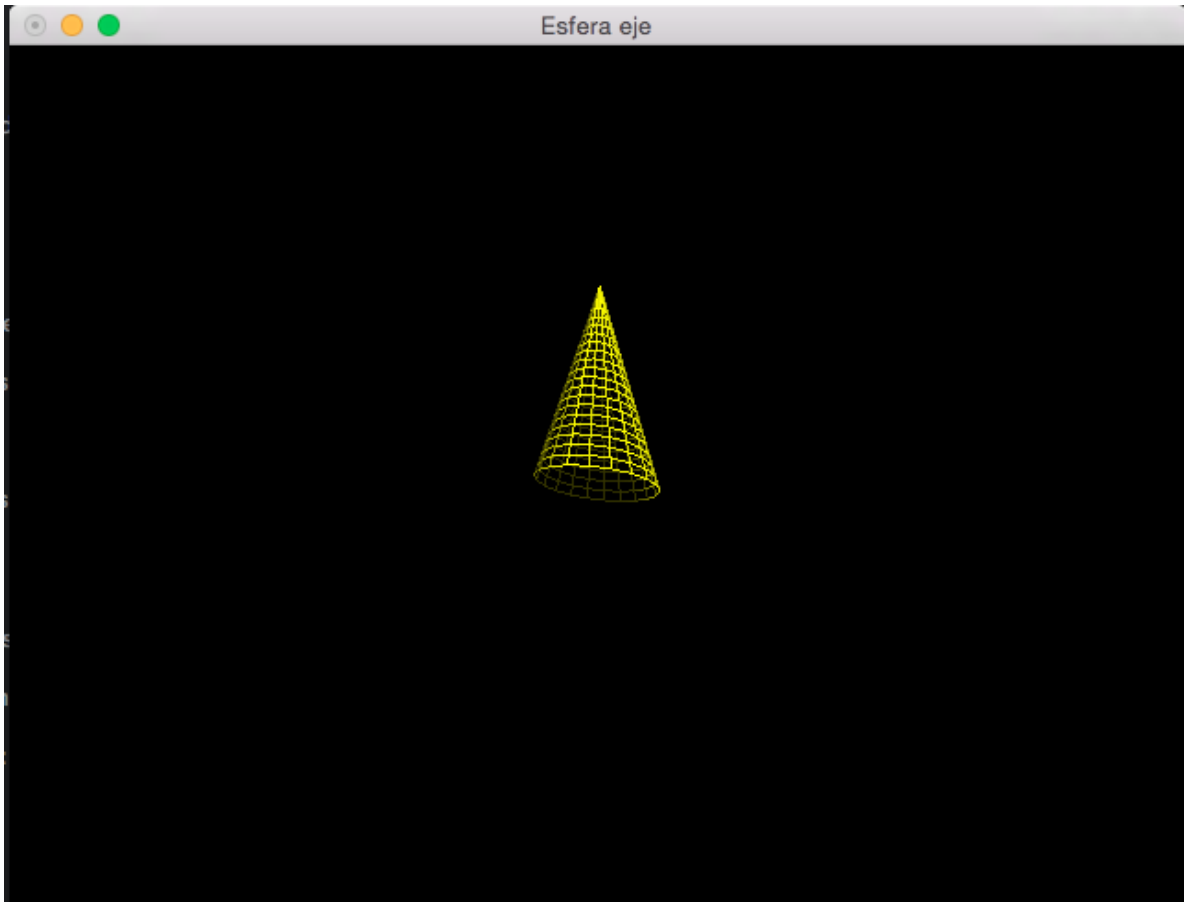
Una vez definidos estos primeros, empezamos con la primer figura a modelar.

Observado la imagen, podemos reducir el personaje esferas y conos. Esferas para el cuerpo y los ojos, además de otros detalles. Y conos para el pico.

La ventaja de reducir el personaje a estas figuras básicas, es que éstas ya vienen implementadas por default en OpenGL, por lo tanto no es necesario preocuparse por su implementación.



glutSolidSphere(1, slices, stacks);



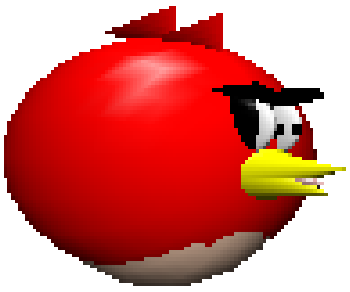
glutSolidCone(1, 1, slices, stacks);

Para los detalles como las plumas, se utilizó la función

glutSolidCube(1);

y finalmente para las pupilas se utilizó GL_POINTS.

Obteniéndose el siguiente resultado:



Para los demás personajes, se realizó el mismo procedimiento: reducirlos a objetos básico fácilmente moldeables en OpenGL y escalando, rotando y trasladando de acuerdo a donde se desea que se ubique la figura final; esto es tomando en cuenta el concepto de **figuras unitarias**.

Así del mismo modo, las figuras quedan como se presentan a continuación:

```
//PANZA
    glPushMatrix();
    glColor3d(0.96, 0.80, 0.69);
    glTranslated(0.0, -0.6, 0.0);
    glScalef(1.4, 3.9, 2.0);
    glutSolidSphere(1, slices, stacks);
    glPopMatrix();
```

El siguiente paso de acuerdo a lo planeado es proceder con la animación. Para esto, se consideró la práctica previa a este proyecto final, la cual presentaba a una esfera desplazándose dentro de un cubo. Estas son básicamente traslaciones, donde se deben se utilizan variables en lugar de parámetros estáticos, así mientras se incrementa la variable dentro de la función, se mueve el objeto.

```
glPushMatrix();
```

```
    glTranslated(angulo2, grado, -6.0);//DENTRO SE MUEVE TODO EL AVE AMARILLA


    glPushMatrix();
    glTranslated(-16.0, -25.5, 0.0);//INICIO REACOMODO POSICION


    glPushMatrix();
    glTranslated(0.0, 0.0, -6.0);//INICIO CANVAS AVE AMARILLA
    glRotated(grado2, 0, 1, 0);//ROTACION DE AVE AMARILLA
```

Ahora bien, una vez considerado el desplazamiento es necesario fijar los límites, ya que si solo se incrementa la variable de forma indefinida, llegara un momento en que la figura desaparezca de nuestra vista, ya que se estará trasladando fuera de nuestra "cámara".

Para esto, definimos los limites dentro de la función de repintado *idle(),* recordemos las imágenes en OpenGL se imprimen constantemente dentro de un ciclo, a una velocidad de 24 cuadros por segundo. Así, la función *idle()* en el tiempo en que no se repinta, esta incrementa las variables.

```
void idle()
{


    if(angulo<29.9 && ban == 0)
    {
        display();
        angulo+=0.1;
        angulo2+=0.1;
```

```
    grado+=0.2;
    if(subida<19.0)
    subida += 0.05;
    /*if(subida > -0.9)
    subida -= 4.5;
    */
    if(angulo > 29.8)
    ban = 1;
}

if(ban == 1)
{
      angulo -= 0.1;
      angulo2 -=0.1;
      grado -= 0.5;
      display();
      if(angulo < 7.0)
      {
            ban = 0;
            angulo = 0.0;
            angulo2 = 0.0;
            grado = 0.0;
      }
```

```
    }
    grado2 += 1.0;
}
```

De manera breve, el primer IF establece que la imagen se ira trasladando desde un punto en la izquierda hasta la derecha y ahí se detendrá. Esto lo hace incrementando las variables.

El segundo IF hace lo contrario, permite el desplazamiento de las figuras desde el punto anterior hasta la izquierda.

Ahora bien, como OpenGL se puede considerar un ciclo, no es necesario meter la función dentro de un ciclo.

Por otro lado, la iluminación va de la mano del desarrollo de las figuras. Recordemos que OpenGL trabaja por default con figuras 3D sin embargo, para apreciarlas es necesario utilizar la iluminación correctamente.

La manera de lograr mostrar un objeto en 3D como tal es utilizando correctamente la iluminación. La razón de que la iluminación sea tan prescindible es porque básicamente así funciona la vista humana. Por ejemplo, al entrar en una habitación con objetos como mesas o sillas, parcialmente obscura, lo que podríamos lograr apreciar

son siluetas de los objetos dentro.

Sin embargo a medida que entra la luz, (o nos acostumbramos a la obscuridad, lo cual es básicamente ir concentrado la mayor cantidad de luz del ambiente) se va apreciando poco a poco los objetos: la profundidad, textura y reflejos.

A un escenario en OpenGL y sus figuras en él, podemos verlo como una habitación obscura, y las figuras evidentemente serían las sillas y mesas. Por lo tanto, dependerá del programador ir manejado la luz de tal manera que se pueda apreciar el objeto 3D como tal. A esta tarea la podemos denominar como **iluminación**.

Para este proyecto, como el escenario es un exterior se utilizó el tipo de luz *especular* la cual tiene similarmente el mismo efecto que la luz solar tiene cuando cae sobre un objeto. También se trabajó con profundidad y brillo.

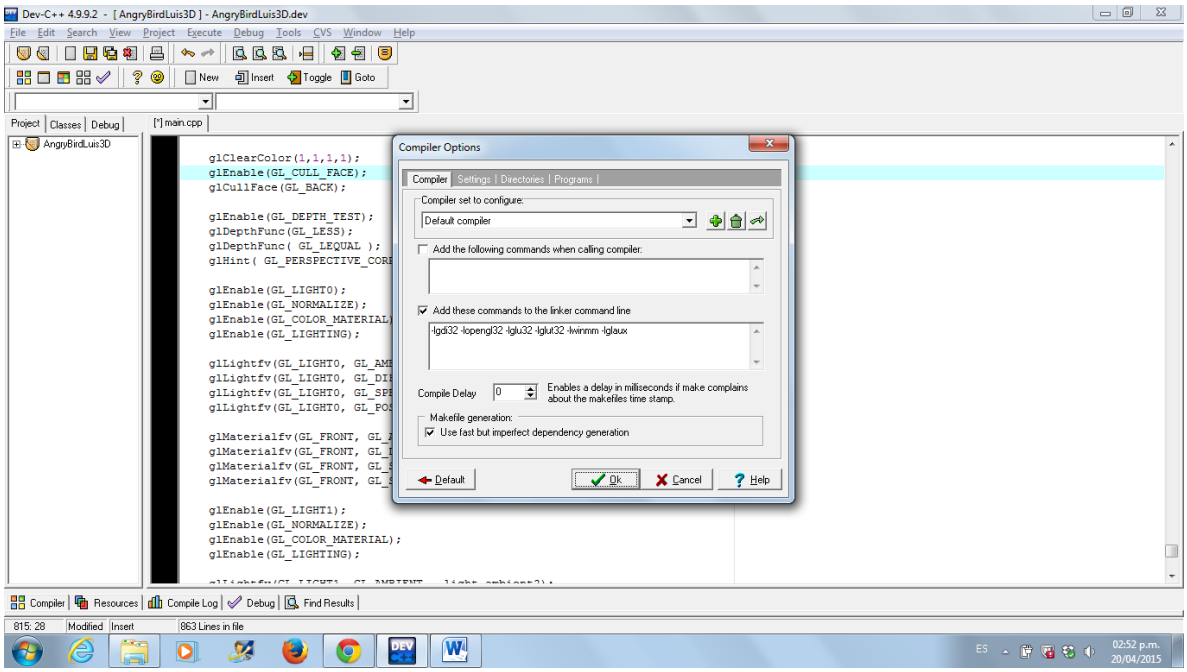
```
glEnable(GL_LIGHT0);
    glEnable(GL_NORMALIZE);
    glEnable(GL_COLOR_MATERIAL);
```

```
glEnable(GL_LIGHTING);


glLightfv(GL_LIGHT0, GL_AMBIENT,  light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE,  light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

De manera específica se utiliza 2 luces para este escenario.

Por otra parte, lo relacionado a la parte de texturización fue lo que requirió de más investigación y trabajo previo. Esto se debió al hecho de que aunque se supone que texturización se debió de haber implementado para el escenario del segundo parcial, ya no se llevó por falta de tiempo. OpenGL incluye por default la librería <*gl.glaux.h*> la cual no obstante, se considera obsoleta y ya no se recomienda su uso. Existen otras variantes como: SOIL, DevIL o FreeImage, las cuales aún continua su actualización, no obstante se encontró difícil su uso e incluso su instalación; tanto como en Mac OS X como en Windows 7.

De hecho, la dificultad en la instalación fue el mayor peso para no haber trabajado con texturas antes. Ni siquiera *glaux.h* funcionaba en Dev-C++. Sin embargo se optó por tratar de emplear este último ya que las función para obtener las imágenes fueron dadas en clase. Es necesario destacar como se solucionó el problema del uso en Dev-C++ de esta librería. La solución únicamente consistió en linkear la librería con el Proyecto. Específicamente, se crea un nuevo proyecto en OpenGL, se elige que sea de tipo **glut** y finalmente se crea. Ahora bien, procedemos a seleccionar en la barra de herramientas **Tools-> Compiler Options**



y en el segundo recuadro de la ventana que nos aparece tecleamos lo siguiente:

-lgdi32 -lopengl32 -lglu32 -lglut32 -lwinmm –lglaux

Recordando seleccionar el *checkbox.* De esta manera, cada vez que llamemos una función de la librería *glaux.h* NO habrá CONFLICTO con el linkeo.

Ahora bien, se utilizaron 4 imágenes que tiene que ir declaradas en el arreglo que utiliza *glaux.h* para jalar las texturas.

```
#define NTextures 4

GLuinttexture[NTextures];


//variables para manejo de texturas

char *texturefiles[] = {

        "treesForest.bmp", "cielo.bmp", "cesped.bmp", "ladrillos.bmp"

};
```

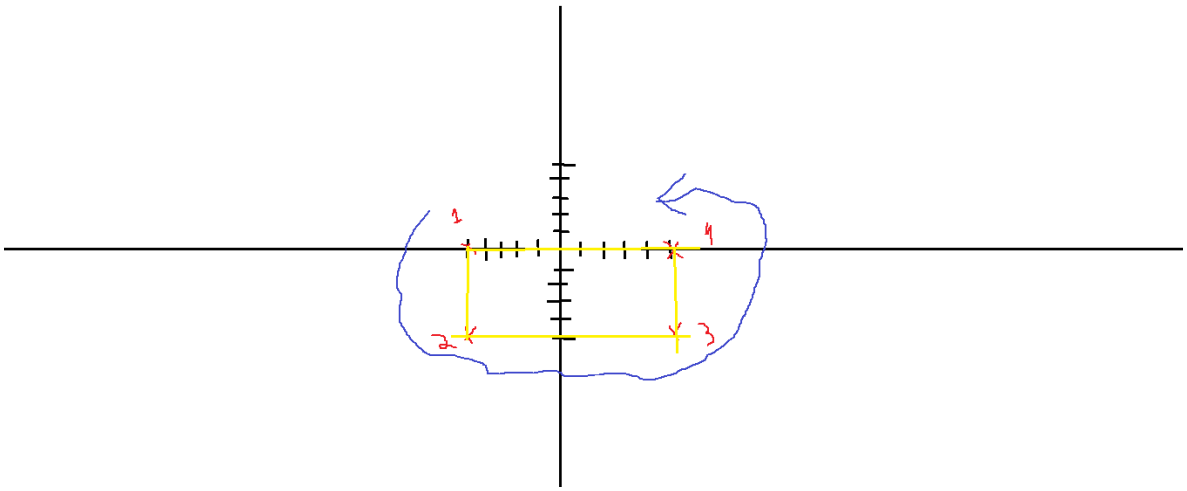
Pasto

Fondo

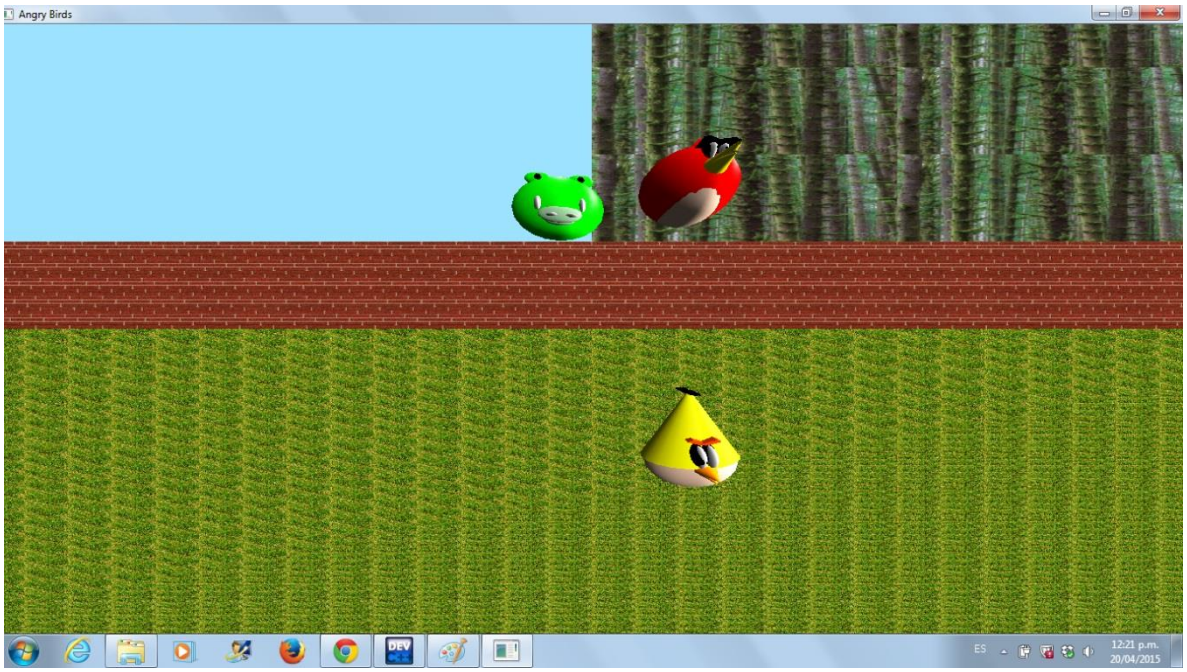
Ladrillos

Bosque

Cada una se dibujó dentro de un rectángulo, donde es importante destacar que para que se visualice correctamente, se deben de dibujar en contra sentido de las agujas del reloj.



Finalmente, unimos todos los componentes y el escenario final es el siguiente:



Conceptos desarrollados

- Máquina de estados: Es un concepto relativo a automátas y que tiene una aplicación en OpenGL ya que prácticamente todas las operaciones en OpenGL funcionan mediante estados, ejemplos de las funciones que funcionan así son glColor3f(). Para manejar los estados se utilizan las funciones **glPushMatrix()** y **glPopMatrix().**
- Pila: Se utilizo para manipular la máquina de estados. Utilizan las funciones anterirores.
- Primitiva: Es la interpretación de un conjunto de vértices dibujados de una manera específica en pantalla. Existen un total de 10.
- Figuras unitarias. Es la idea de que en OpenGL los objetos se manejan como un todo.
- Transformaciones: Funciones utilizadas para la manipulación de los objetos unitarios en OpenGL. Tales funciones son **glTranslatef(), glRotatef()** y **glScalef().** Recordemos que las transformaciones se aplican en orden inverso.

Experimentos:

Básicamente estos consistieron de modificar los parámetros de las transformaciones así como de jugar con la iluminación y los límites de la traslación.

Conclusiones.

Este proyecto me permitió recuperar conceptos aprendidos en el curso y que no quedaron claros anteriormente, o que no se llegaron a implementar en su momento: como el caso de la texturización. Finalmente y como cualquier proyecto, me permitió hacer converger todo lo aprendido a lo largo del curso, reafirmar el hecho de planear mis tiempos y aunque se pudo haber logrado más, sin duda se obtuvo el conocimiento de las herramientas que OpenGL provee, otras técnicas de programación y el hecho de que lo aprendido este curso, se suma a mi formación académica.

Bibliografía

[1] http://sabia.tic.udc.es/gc/Tutorial%20OpenGL/tutorial/cap3.htm

Colores:

https://www.opengl.org/discussion_boards/showthread.php/132502-Color-table

Capítulo 3: Dibujando en 3D:

http://sabia.tic.udc.es/gc/Tutorial%20OpenGL/tutorial/cap3.htm

Primitivas:

http://opengl.infojosep.es/primitivas/primitivas.php

Transformaciones:

http://openglenfichas.uji.es/ejemplos/transf.pdf

Texturas:

(Uso de Texturas con OpenGl)http://informatica.uv.es/iiguia/AIG/docs/texturas.htm

(OpenGL, textura de fondo) http://www.stratos-ad.com/forums/index.php?topic=9418.0

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Ingeniería en Tecnologías de la Información

Graficación

EXÁMEN 1

Aca Xique Wendy | 201245715

acaxique@gmail.com

Introducción

Se realizó un escenario con objetos sencillos cada uno en una función, hechos únicamente con puntos y líneas a los cuales se les aplicaron transformaciones geométricas básicas como la translación, escalamiento y rotación.

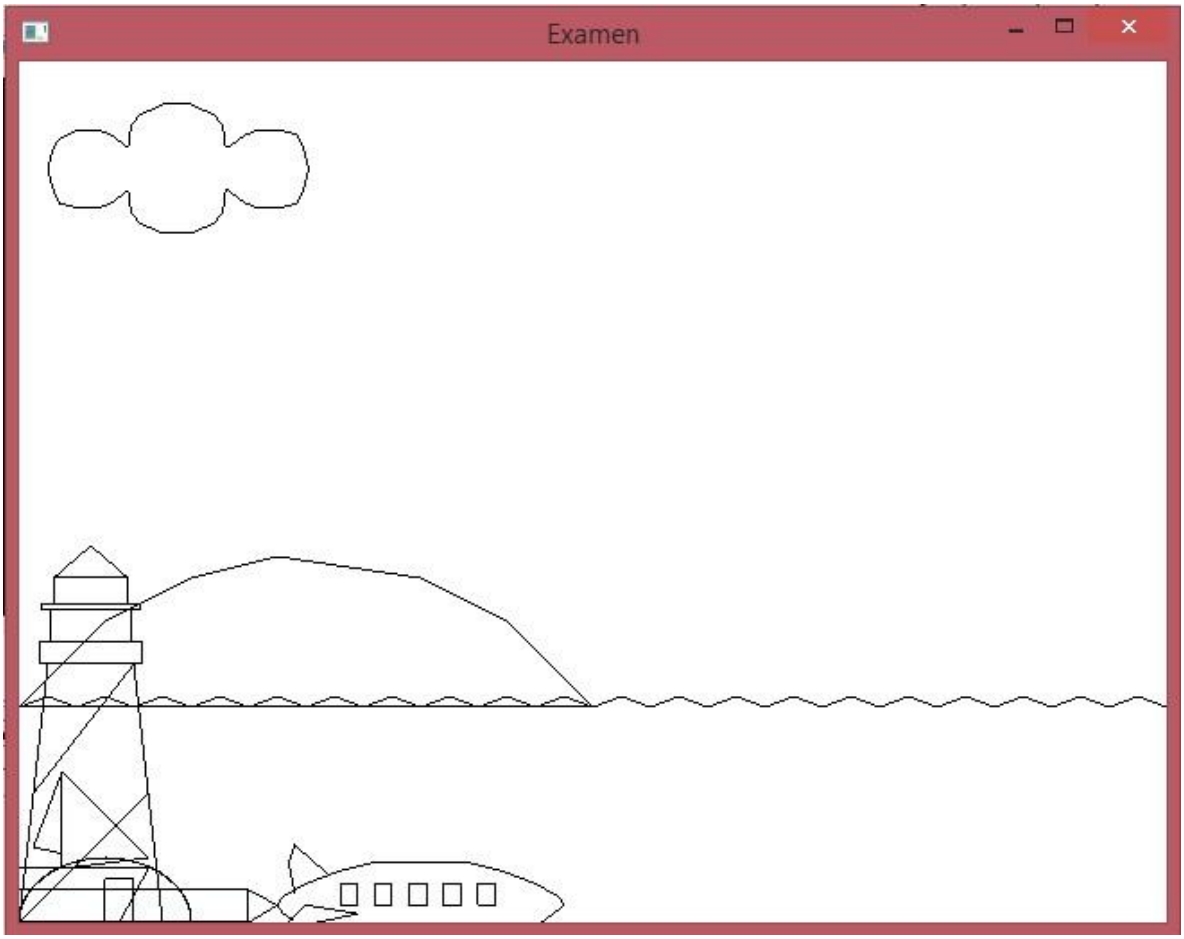
Desarrollo

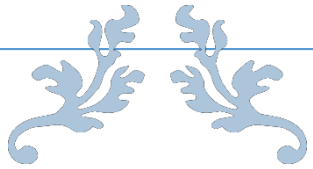
Para crear el escenario se crearon las siguientes funciones:

- Inicializa: Establece el fondo de la pantalla y las coordenadas en las que se trabajó.
- Faro: Crea el objeto faro hecho con líneas.
- Barco: Crea el objeto hecho con líneas.
- Monte: Crea el objeto hecho con líneas.
- Mar: Crea el objeto hecho con líneas.
- Avión: Crear el objeto hecho con líneas.
- Sol: Crea el objeto sol hecho con puntos, transformando coordenadas polares a cartesianas.
- Nube: Crea el objeto nube hecho con líneas.
- Principal: En esta función se mandan a llamar todas las funciones anteriores.

Conclusión

Los objetos creados en el origen se les aplicaron las transformaciones básicas y los objetos que no se crearon en el origen no se les aplico ninguna transformación.


Examen



PRIMER PARCIAL

Graficación

201217899

KARLA ESTEFANÍA GARRIDO GONZÁLEZ

INTRODUCCION

En este parcial aplicamos los conocimientos adquiridos hasta ahora construyendo nuestras funciones para dibujar un escenario.

Las funciones principalmente están hechas a base de puntos (por medio de GL_POINTS), las transformaciones están hechas con matrices.

Clasificaremos cada tipo de figura en una clase para poder tener un código mas limpio y mas ordenado.

CONCEPTOS DESARROLLADOS

Clases

Los conceptos que hemos estado aplicando, los aplicamos desde que entramos: CLASES.

Clase: abstracción que define un tipo de objeto especificando qué propiedades (atributos) y operaciones disponibles va a tener.

Una clase nos permite definir un nuevo tipo de dato,

Ejemplo:

```
class Cliente
 {
 char nombre[30];
 char calle[30];
 double importe_factura;
 char telefono[20] ;
 int edad;
 void visualizar_datos_cliente ( );
 void modificar_edad ( );
 void hacer_descuento (double);
} ;
cliente c;
```

Una diferencia entre una clase class y una estructura struct es que en las clases los miembros son privados por defecto mientras que en una estructura los miembros son públicos por defecto.

Multiplicación de Matrices.

Las operaciones que estaremos usando son multiplicaciones con matrices ya que necesitamos la matriz identidad de 3x3 y una matriz de puntos 3x1. Como estamos trabajando en escenarios en 2D aun, el valor de [3][1] en nuestra matriz de puntos será 1.

Las operaciones solo las podremos aplicar en puntos, ya que solo se nos permite alterar el valor de un punto, además que es la base de todas nuestras demás clases.

Las operaciones que haremos serán las siguientes:

Traslación

Esta operación nos permite cambiar el punto de lugar.

$$\begin{matrix} 1 & 0 & \square & \square & \square + \square \end{matrix}$$

$$\begin{pmatrix} 0 & 1 & \square \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \square \\ 1 \end{pmatrix} = \begin{pmatrix} \square + \square \\ 1 \end{pmatrix}$$ v1=valor para mover el punto x.

v2=valor para mover el punto y.

Escalamiento

Esta operación nos permite alterar el tamaño de una figura.

$$\begin{pmatrix} r_1 & 0 & 0 \\ 0 & r_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x * r_1 \\ y * r_2 \\ 1 \end{pmatrix}$$

r1=valor de cambio de tamaño en x.

r2=valor de cambio de tamaño en y.

Rotación

Esta operación, como su nombre lo indica, rotará la figura dado cierto ángulo.

$$\begin{pmatrix} \cos\propto & -\sin\propto & 0 \\ \sin\propto & \cos\propto & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x\cos\propto - y\sin\propto \\ x\sin\propto + y\cos\propto \\ 1 \end{pmatrix}$$

∝= [illegible]

Reflexión

Esta operación matricial nos sirve para reflejar en el eje x, en el eje y, o en ambos, una figura y solo es multiplicar nuestra matriz de puntos con alguna de estas matrices

$$\begin{matrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{matrix}$$ Para reflejar la figura en el eje x.

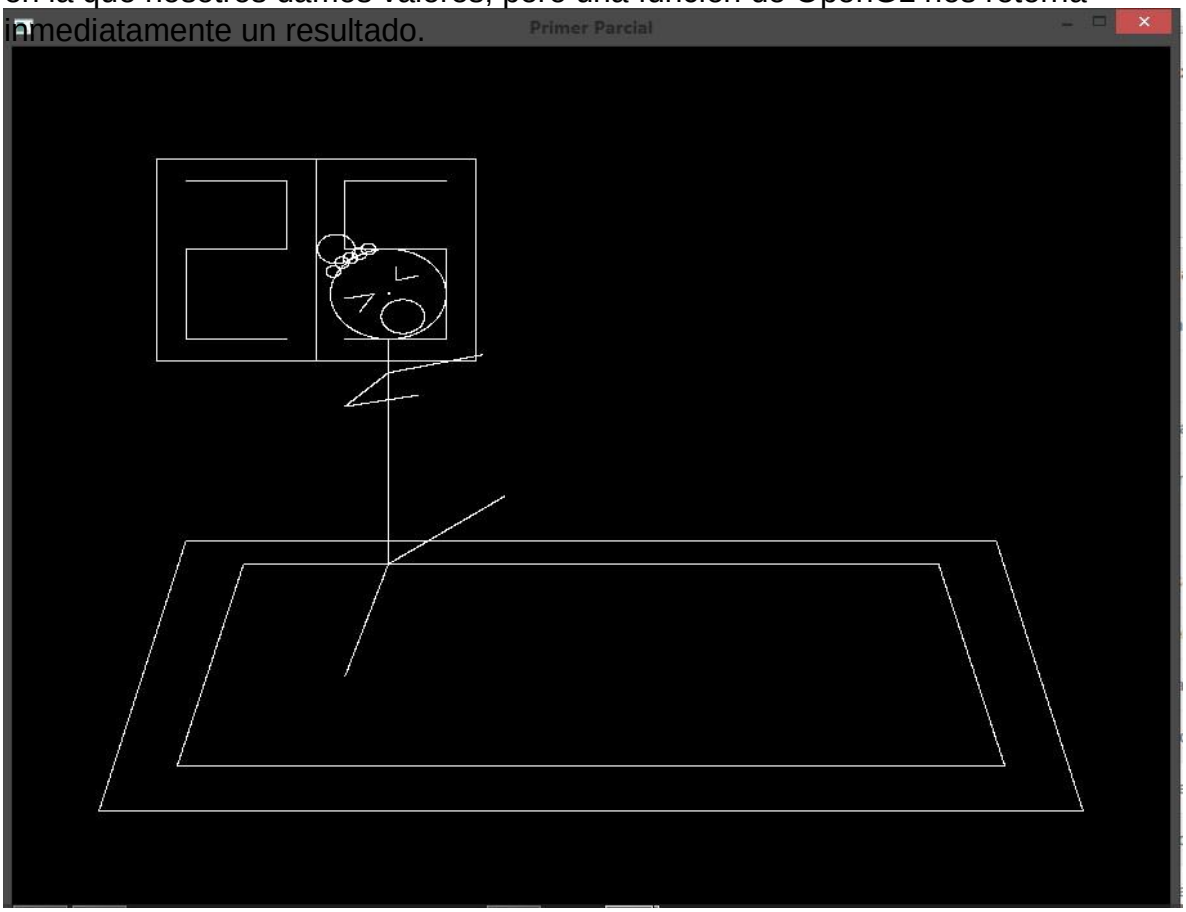
$$\begin{matrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$ Para reflejar la figura en el eje y.

$$\begin{matrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{matrix}$$ Para reflejar la figura en el eje x y en el eje y.

CONCLUSION

Gracias a los conocimientos adquiridos anteriormente en nuestros cursos tanto de programación como de matemáticas, se nos ha hecho fácil comprender como funciona cada operación y como aplicarlo en la programación (para el caso de las multiplicaciones de matrices).

Al hacer esto, nos podemos dar cuenta cómo es que trabajan otras primitivas de OpenGL, podemos darnos una idea de que es lo que ocurre en esa “caja negra” en la que nosotros damos valores, pero una función de OpenGL nos retorna inmediatamente un resultado.



BIBLIOGRAFIA

http://www.cs.buap.mx/~iolmos/

http://www.nebrija.es/~abustind/Informatica/MetodologiaII/Introduccion.pdf

http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=411:conceptos-de-objetos-y-clases-en-java-definicion-de-instancia-ejemplos-basicos-y-practicos-cu00619b&catid=68:curso-aprender-programacion-java-desde-cero&Itemid=188

PRIMAVERA 2015

BENEMERITA UNIVERSIDAD AUTÓNOMA

DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

MATERIA: GRAFICACIÓN

DOCENTE: DR. IVÁN OLMOS PINEDA

ALUMNA: JOCELYN DÁVILA HERNÁNDEZ

MATRÍCULA: 201011077

PROYECTO FINAL

Documento Técnico

PRIMAVERA 2015

ÍNDICE

INTRODUCCIÓN:

INTRODUCCIÓN:

Este documento presenta la información recopilada de todo lo que se realizó para hacer posible el proyecto que pidió el Doctor Iván Olmos al finalizar el curso de Graficación.

Este documento presenta la información recopilada de todo lo que se realizó para hacer posible el proyecto que pidió el Doctor Iván Olmos al finalizar el curso de Graficación.

El proyecto se realiza en wxDevC++, con todos los conocimientos que adquirimos para trabajar con primitivas de OpenGl.

El proyecto se realiza en wxDevC++, con todos los conocimientos que adquirimos para trabajar con primitivas de OpenGl.

El proyecto pretende ser muy visible ya que contiene todo lo que vimos en el curso, así que tendría que ser completo para que abarque todos los conocimientos de lo aprendido.

El proyecto pretende ser muy visible ya que contiene todo lo que vimos en el curso, así que tendría que ser completo para que abarque todos los conocimientos de lo aprendido.

CONCEPTOS:

CONCEPTOS:

TEXTURA Una textura, desde el punto de vista de su almacenamiento en memoria es un array de datos. Cada uno de los valores de este array lo llamamos 'textel'. Este array de datos representa una imagen, que utilizaremos para mapearla sobre un polígono.

Podemos rellenar dicho array bien cargando una imagen desde un fichero, o bien dándole nosotros los valores de color a cada texel.

La información de cada uno de los componentes puede ser:

- Las componentes RGB del color.
- Indices de color.
- Niveles de luminancia (grises).

Además, podemos tener texturas unidimensionales (un solo pixel de alto o de ancho) bidimensionales (imagen de tamaño mxn) o tridimensionales (con volumen), aunque lo habitual será utilizar las bidimensionales.

En OpenGl las dimensiones de una textura deben ser siempre potencia de 2= 64, 128,256...

Los pasos necesarios para mapear texturas sobre polígonos son los siguientes

Dibujar la escena.

POLÍGONOS: Los polígonos son las áreas comprendidas en bucles de líneas cerrados, donde los segmentos de líneas están especificados por los vértices de sus extremos. Normalmente los polígonos se dibujan con los píxeles de su interior rellenos, pero también pueden dibujarse sólo sus aristas o los puntos de los vértices.

ROTACIÓN : Para rotar, tenemos también una función de alto nivel que construye la matriz de transformación y la multiplica por la matriz activa, glRotate. Lleva como parámetros el ángulo a rotar (en grados, sentido horario), y después x, y y z del vector sobre el cual se quiere rotar el objeto. Una rotación simple, sobre el eje y, de 10º sería

glRotatef(10, 0.0f, 1.0f, 0.0f).

TRANSLACIÓN: Las traslaciones pueden entenderse como movimientos directos sin cambios de orientación, es decir, mantienen la forma y el tamaño de las figuras u objetos trasladados, a las cuales deslizan según el vector. Dado el carácter de isometría para cualquier punto P y Q se cumple la siguiente identidad entre distancias.

3D: OpenGL utiliza una representación de objetos 3D a partir de un espacio en cuatro dimensiones.

3 OpenGL utiliza una representación de objetos 3D a partir de un espacio en cuatro dimensiones. El espacio 3D se representa a través de un sistema 3D ortonormal, donde los ejes son perpendiculares y cada unidad en cada eje está representado por un vector de módulo 1. La cuarta coordenada se utiliza para representar la perspectiva.

PROYECCIÓN: Proyección ortográfica: permite visualizar todo lo que se encuentre dentro de un cubo, delimitado por los parámetros de la función glOrto.

PROYECCIÓN PERSPECTIVA: Delimita un volúmen de visualización dado por un ángulo de cámara y una relación alto/ancho. La distancia al observador delimitará el tamaño con el que un objeto se visualiza.

PROYECCIÓN PERSPECTIVA: Delimita un volúmen de visualización dado por un ángulo de cámara y una relación alto/ancho. La distancia al observador delimitará el tamaño con el que un objeto se visualiza.

DESARROLLO:

Una de las primeras cosas que se manejan en el código son las texturas que vamos a utilizar en este proyecto:

```
/ Cargar Texturas
char *texturefiles[] = {

                "asfalto.bmp", "amarilloBPM.bmp", "adoquin.bmp", "grass.bmp", "wall.bmp", "oxxo.bmp",
                "techoOxxo.bmp", "carro.bmp", "vidrioNegro.bmp", "shelbyIzq.bmp", "shelbyDer.bmp","atardecer.bmp",

                "eggs.bmp","cielo.bmp"



};


// MANEJO DE TEXTURAS
int LoadTextures()
{
  AUX_RGBImageRec *TextureImage;
  //se recorre el arreglo de imagenes para cargarlas en memoria
  for(int i=0; i < NTextures; i++)
  {
    TextureImage=auxDIBImageLoad(texturefiles[i]);
    if ( TextureImage )
                {
                  //glGenTextures(1, texture[i]);
                  glGenTextures(1, texture+i);
                  glBindTexture(GL_TEXTURE_2D, texture[i]);

                  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
```

DESARROLLO:

Una de las primeras cosas que se manejan en el código son las texturas que vamos a utilizar en este proyecto:

```
// Cargar Texturas
char *texturefiles[] = {
                "asfalto.bmp", "amarilloBPM.bmp", "adoquin.bmp", "grass.bmp", "wall.bmp", "oxxo.bmp",
                "techoOxxo.bmp", "carro.bmp", "vidrioNegro.bmp", "shelbyIzq.bmp", "shelbyDer.bmp","atardecer.bmp",
                "eggs.bmp","cielo.bmp"
};

//NAMERO DE TEXTURAS
int LoadTextures()
{
  AUX_RGBImageRec *TextureImage;
  //se recorre el arreglo de imagenes para cargarlas en memoria
  for(int i=0; i < NTextures; i++)
  {
    TextureImage=auxDIBImageLoad(texturefiles[i]);
    if ( TextureImage )
                        {
                          //glGenTextures(1, texture[i]);
                          glGenTextures(1, texture+i);
                          glBindTexture(GL_TEXTURE_2D, texture[i]);
                          glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
                          glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
                          glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);

        glTexImage2D(GL_TEXTURE_2D, 0, 3,TextureImage->sizeX, TextureImage->sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE,
TextureImage->data);
                          free(TextureImage->data);
                          free(TextureImage);
                          glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE);
                        }
                      else
    return 0;
  }
  return 1;
}
```

Por otro lado explico la función KEYBOARD, es muy importante ya que nos podemos mover dentro del programa, pero el movimiento que hace es cuando se oprimen las teclas “a”, ”s”, ”d”, ”w”.

```
// Teclado
void keyboard(unsigned char key, int x, int y)
{
  switch (key) {
// Movimiento hacia adelante
                    case 'w':
                    case 'W':
                        dollyCamera( DEF_dollyStepSize, 0.0 );
                        break;

// Movimiento hacia atras
                    case  's':
                    case 'S':
                        dollyCamera( DEF_dollyStepSize, 180.0 );
                        break;

// Strafe hacia la izquierda
                    case  'a':
                    case 'A':
                        dollyCamera( DEF_dollyStepSize, 90.0 );
                        break;
// Strafe derecha
                    case 'd':
                    case 'D':
                        dollyCamera( DEF_dollyStepSize, 270.0 );
                        break;
//Esc salir
                       }
```

Por otro lado explico la función KEYBOARD, es muy importante ya que nos podemos mover dentro del programa, pero el movimiento que hace es cuando se oprimen las teclas “a”, ”s”, ”d”, ”w”.

```
        case 27:
            exit( 0 );
            break;
}

// Teclado
void keyboard(unsigned char key, int x, int y)
{
  switch (key) {
// Movimiento hacia adelante
            case 'w':
            case 'W':
                dollyCamera( DEF_dollyStepSize, 0.0 );
                break;
```

Veamos ahora la función reshape(). Esta función, al ser pasada a glutReshapeFunc, será llamada cada vez que se reescale a ventana. La función siempre debe ser definida con el siguiente esqueleto:

void reshape(int, int) { ... }

```
// Movimiento hacia atras
            case 's':
            case 'S':
                dollyCamera( DEF_dollyStepSize, 180.0 );
                break;
```



```
// Strafe hacia la izquierda
            case 'a':
            case 'A':
                dollyCamera( DEF_dollyStepSize, 90.0 );
                break;

// Strafe derecha
            case 'd':
            case 'D':
                dollyCamera( DEF_dollyStepSize, 270.0 );
                break;

//Esc salir
            case 27:
                exit( 0 );
                break;
  }
}
```

Veamos ahora la función reshape(). Esta función, al ser pasada a glutReshapeFunc, será llamada cada vez que se reescale a ventana. La función siempre debe ser definida con el siguiente esqueleto:

void reshape(int, int) { ... }

El primer parámetro será el ancho y el segundo el alto, después del reescalado (estos valores los manda el sistema de ventanas como parámetros del callback glutReshapeFunc). Con estos dos valores trabajara la función cuando, en tiempo de ejecución, el usuario reescale la ventana.

El primer parámetro será el ancho y el segundo el alto, después del reescalado (estos valores los manda el sistema de ventanas como parámetros del callback glutReshapeFunc). Con estos dos valores trabajara la función cuando, en tiempo de ejecución, el usuario reescale la ventana.

```
void reshape(int w, int h)
{

 glViewport(0, 0,  (GLsizei) w, (GLsizei) h);
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
 gluPerspective(60.0, (GLfloat) w/(GLfloat) h, 1.0, 1000.0);
 glMatrixMode(GL_MODELVIEW);
 glLoadIdentity();
   glTranslatef (0.0, -20.0, -60.0);
}
```

```
void reshape(int w, int h)
{
 glViewport(0, 0,  (GLsizei) w, (GLsizei) h);
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
 gluPerspective(60.0, (GLfloat) w/(GLfloat) h, 1.0, 1000.0);
 glMatrixMode(GL_MODELVIEW);
 glLoadIdentity();
   glTranslatef (0.0, -20.0, -60.0);
}
```

Bueno y después de mostrar las funciones esenciales para el proyecto, la mayor parte del código trata solo de graficar polígonos, para que después a estos polígonos se les pueda agregar las texturas que se cargaron al principio del programa.

FOTO DEL PROYECTO:

Bueno y después de mostrar las funciones esenciales para el proyecto, la mayor parte del código trata solo de graficar polígonos, para que después a estos polígonos se les pueda agregar las texturas que se cargaron al principio del programa.

FOTO DEL

PROYECTO:

CONCLUSIONES:

CONCLUSIONES:

El proyecto cumple con las expectativas, tiene lo que vimos en el curso, translaciones, rotaciones, graficar polígonos, las diferentes posiciones de la cámara y una proyección diferente.

El proyecto cumple con las expectativas, tiene lo que vimos en el curso, translaciones, rotaciones, graficar polígonos, las diferentes posiciones de la cámara y una proyección diferente.

Agradeciendo los conocimientos proporcionados para realizar este proyecto, finalizo el documento técnico mencionando que los programas resultan más visibles y carismáticos para un usuario cuando se cuenta con diferentes objetos que lo hagan ver bien.

Agradeciendo los conocimientos proporcionados para realizar este proyecto, finalizo el documento técnico mencionando que los programas resultan más visibles y carismáticos para un usuario cuando se cuenta con diferentes objetos que lo hagan ver bien.

BIBLIOGRAFÍA:

http://sabia.tic.udc.es/gc/Tutorial%20OpenGL/tutorial/cap4.htm#_Toc535127351

http://sabia.tic.udc.es/gc/Tutorial%20OpenGL/tutorial/cap2.htm

http://gsii.usal.es/~igrafica/descargas/temas/Tema05.pdf

http://informatica.uv.es/iiguia/AIG/docs/texturas.htm


FCC
BUAP

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Ing. en Tecnologías de la Información

Asignatura:

Graficacion *CCOM 259 - 101*

Practica:

3 Parcial

Alumno:

Jesús García Licona 201212661

Índice

Objetivo

-Graficar un Escenario 3D

-Utilizar Traslaciones, Texturas, Figuras 3D, Figuras complejas.

-Implementar Look At y Iluminación.

Estrategia de desarrollo

1- Se realizara un bosquejo previo de nuestro escenario.

2- Procederá a la colocación de las figuras.

3- Se ocupara la Iluminación y el movimiento Look at

4- Al final se colocara las traslaciones.

Desarrollo

Un punto en 3D: el vertex

Para especificar un punto dibujado en la paleta 3D, se utiliza la función OpenGL glVertex, sin ninguna duda la función más utilizada de todas las que ofrece la API de OpenGL. Es el mínimo común denominador de todas las primitivas de OpenGL: un simple punto en el espacio. La función glVertex toma como argumentos de 2 a 4 parámetros de tipo numérico, de bytes a dobles.

El siguiente código establece un punto en un sistema de coordenadas a 20 unidades en el eje x, 20 en el eje y y 0 en el eje z:

```
glVertex3f(20.0f, 20.0f, 0.0f);
```

El siguiente paso es dibujar algo y lo más sencillo son puntos. El código que se puede ver a continuación dibujará 2 puntos en la pantalla, indicando que se va a utilizar puntos como primitiva.

```
glBegin(GL_POINTS);          glVertex3f(0.0f, 0.0f, 0.0f);
glVertex3f(50.0f, 50.0f, 50.0f);

glEnd();
```

El argumento GL_POINTS indica a OpenGL que los siguientes vértices deben ser interpretados y dibujados como puntos. En este

caso se especifican dos vértices que van a ser dibujados como dos puntos. Es importante señalar que se puede realizar múltiples

llamadas a primitivas del mismo tipo entre llamadas a glBegin y glEnd.

Dibujando líneas en 3D

La primitiva GL_POINTS que hemos venido utilizando hasta ahora es bastante directa, en tanto en cuanto para cada vértice especificado,

dibuja un punto. El siguiente paso lógico es especificar dos vértices y dibujar una línea recta entre ellos.

Ésta operación es llevada a cabo por la primitiva GL_LINES. La siguiente sección de código dibuja una línea entre los puntos (0,0,0) y (20,20,20)

```
   glBegin(GL_LINES);            glVertex3f(0.0f, 0.0f, 0.0f);
glVertex3f(20.0f, 20.0f, 20.0f);

glEnd();
```

Nótese que una línea es definida únicamente por dos vértices, pero la primitiva puede contener tantas parejas de vértices como se desee, tomándose de dos en dos de manera consecutiva. En el caso de que se especifique un mayor impar de vértices, el último será ignorado.

A continuación se muestra una sección de código más compleja que muestra una serie de líneas a modo de radios de una rueda

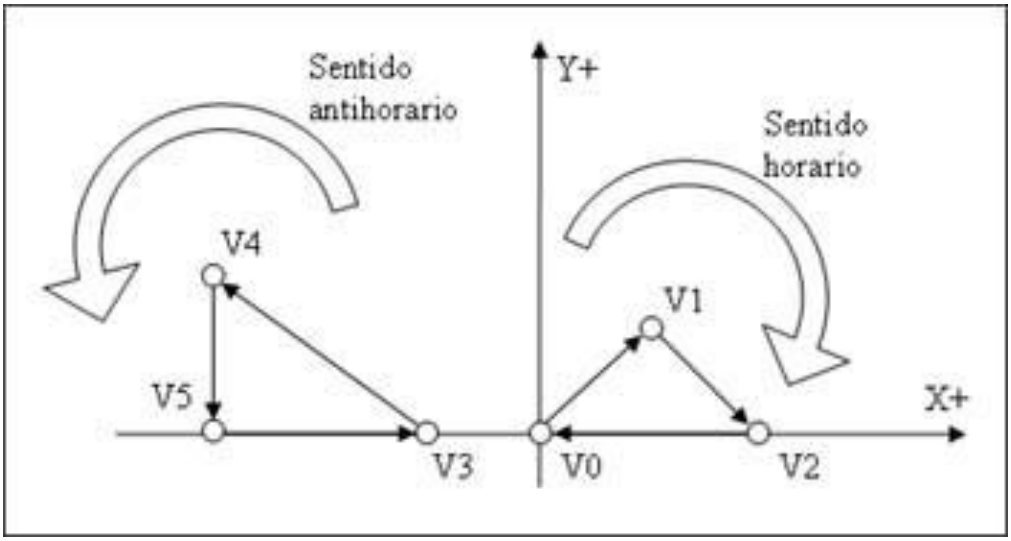
```
//Llamar una única vez para todos los puntos glBegin(GL_LINES);
//Todas las líneas se definen en el plano xy z = 0.0f;

for(angulo = 0.0f; angulo <= GL_PI*3.0f; angulo += 0.5f) {

//Puntos en la mitad superior de la circunferencia x = 40.0f*sin(angulo);
y = 40.0f*cos(angulo);
glVertex3f(x, y, z);

//Puntos en la mitad inferior de la circunferencia x =
```

```
40.0f*sin(angulo+3.1415f);
y = 40.0f*cos(angulo+3.1415f);
glVertex3f(x, y, z);

} glEnd();
```

Dibujo de polígonos

En la creación de objetos sólidos, el uso de puntos y líneas es insuficiente. Se necesitan primitivas que sean superficies cerradas, rellenas de uno o varios colores que, en conjunto, modelen el objeto deseado. En el campo de la representación 3D de los gráficos en computación, se suelen utilizar polígonos (que a menudo son triángulos) para dar forma a objetos “semisólidos” (ya que en realidad son superficies, están huecos por dentro). Ahora se verá la manera de hacer esto mediante las primitivas GL_TRIANGLES y GL_QUADS.

```
glBegin(GL_TRIANGLES);
    glVertex3f(0.0f, 0.0f, 0.0f); // V0
    glVertex3f(1.0f, 1.0f, 0.0f); // V1
    glVertex3f(2.0f, 0.0f, 0.0f); // V2
    glVertex3f(-1.0f, 0.0f, 0.0f); // V3
    glVertex3f(-3.0f, 2.0f, 0.0f); // V4
    glVertex3f(-2.0f, 0.0f, 0.0f); // V5
glEnd();
```


Sentido
antihorario
Y+
Sentido
horario
V4
V1
V5
X+
V3
V0
V2

Es muy importante el orden en que se especifican los vértices. En el primer triángulo (el de la derecha), se sigue la política de “sentido horario”, y en el segundo, “sentido antihorario”. Cuando un polígono cualquiera, tiene sentido horario (los vértices avanzan en el mismo sentido que las agujas del reloj), se dice que es positivo; en caso contrario, se dice que es negativo. OpenGL considera que, por defecto, los polígonos que tienen sentido negativo tienen un “encare frontal”. Esto significa que el triangulo de la izquierda nos muestra su cara frontal, y el de la derecha su cara trasera. Más adelante se verá que es sumamente importante mantener una consistencia en el sentido de los triángulos al construir objetos sólidos.

En OpenGL, por defecto, se muestran las caras frontales y traseras de los polígonos. Sin embargo, es habitual querer que únicamente se rendericen las caras frontales y no las traseras. Para habilitar que OpenGL sólo visualice las caras frontales se utiliza la llamada glEnable(GL_CULL_FACE) que, utilizada en el ejemplo anterior, provoca que sólo se visualice el triángulo izquierdo (Puedes ponerla en la sección de inicialización-init del programa).

Si se necesita invertir el comportamiento por defecto de ogl, basta con una llamada a la función glFrontFace(). Ésta acepta como parámetro GL_CW (considera los polígonos positivos con encare frontal) ó GL_CCW (considera los polígonos negativos con encare frontal).

Texturas

Podemos rellenar dicho array bien cargando una imagen desde un fichero, o bien dándole nosotros los valores de color a cada texel. La información de cada uno de los componente puede ser:

- Las componentes RGB del color.
- Índices de color.
- Niveles de luminancia (grises).

Además, podemos tener texturas unidimensionales (un solo pixel de alto o de ancho) bidemensionales (imagen de tamaño mxn) o tridimensionales (con volumen), aunque lo habitual será utilizar las

bidimensionales.

En OpenGl las dimensiones de una textura deben ser siempre potencia de 2= 64,128,256...

Los pasos necesarios para mapear texturas sobre polígonos son los siguientes

- Indicar los parámetros de aplicación de la textura y activar el mapeado de texturas.
- Especificar la textura

Dibujar la escena.

Pasos en la texturización

Los pasos a la hora de texturizar son los siguientes

- Crear un objeto textura y especificar la textura para este objeto
- Indicar como como se aplica la textura a cada vértice
- Habilitar el mapeo de texturas
- Dibujar la escena, indicando las coordenadas de la textura y las coordenadas geométricas (vértices)

Objetos textura

Los objetos textura son la encapsulación de una textura, al objeto textura le identifica un entero sin signo. Los objetos textura son creados y destruidos, pero no se comparten entre los distintos contextos de OpenGL (distintas ventanas). Las funciones relacionadas con los objetos texturas son las siguientes:

Para crear objetos textura:

void glGenTextures(GLsizei n, GLuint *textureNames) : Solicitamos n objetos textura, el nombre o identificador de estos objetos son

devueltos en textureNames que previamente le habremos reservado espacio.

Para eliminar objetos textura: void glDeleteTextures(GLsizei n, GLuint

*textures) : Solicitamos que los n objetos textura especificados en textures sean borrados.

Para usar un objeto textura: void glBindTexture(GLenum target,

GLuint textureName) : Selecciona la textura textureName como textura activa para el target especificado (GL_TEXTURE_1D,

GL_TEXTURE_2D, GL_TEXTURE_3D). Con esta función marcamos el objeto textura como el objeto activo, tener un objeto textura como objeto activo tiene dos funcionalidades, indicar que textura se utiliza y sobre que textura modificamos los parámetros de texturización, estos parámetros los veremos más adelante.

Para especificar la textura disponemos de tres funciones, una para cada dimensión, veremos la función para 2D glTexImage2D y es extensible lo mismo para 1D y 3D.

void glTexImage2D(GLenum target, GLint level, GLint internalFormat, GLsizi width, GLsizei height, GLint border, GLenum format, GLenum type, const GLvoid *texels)

target debe ser GL_TEXTURE2D.

level nivel de detalle, se usa cuando aplicamos técnicas de mipmapping, en nuestro caso cero.

internalFormat podemos elegir entre una variedad, generalmente GL_RGB. Hace referencia al tipo de textura que deseamos. width, height y border indican el tamaño que debe ser $(2^n+b)x(2^m+b)$.

Normalmente el borde (b) es cero, un tamaño posible es 64x128. Si no especificamos un tamaño correcto la textura no se muestra. format generalmente GL_RGB o GL_EXT_BGR. type tipo en el que esta almacenada la textura, puede ser GL_BYTE, GL_UNSIGNED_BYTE, GL_SHORT, GL_UNSIGNED_SHORT, GL_INT, GL_UNSIGNED_INT, GL_FLOAT, GL_BITMAP. texels array que contiene la textura, de izquierda a derecha y de abajo a arriba.

Especificamos cual es la textura para el objeto textura activo. Es conveniente por cuestiones de rendimiento que por cada textura que utilicemos tengamos un objeto textura y mantener los objetos textura durante toda la aplicación, en lugar de crearlo y destruirlo cada vez que se usa la textura, o tener un solo objeto textura y modificar la textura que se utiliza.

Habilitar texturas

Ya hemos visto como crear texturas, especificar cual es la textura que

contiene el objeto. Para habilitar o deshabilitar la texturización lo indicamos con la instrucción glEnable() pasando como parámetro el tipo de textura GL_TEXTURE_1D, GL_TEXTURE_2D o GL_TEXTURE_3D. En caso de que este hablitada más de uno se aplicará el tipo de mayor dimensión.

Dibujar la primitiva

Cuando renderizamos la primitiva, debemos asignar una coordenada textura a cada vértice. Podemos especificar la coordenada o dejar a OpenGL que la genere automáticamente a partir de un método seleccionado.

Para especificarla nosotros mismos lo hacemos con la función glTexCoord*(). A cada pixel se le asigna un texel interpolando las coordenadas textura de los vértices de la misma forma que se hacia para el color. Recordar que qa coordenada de la textura igual que el color, la normal u otros parámetros se especifican antes de indicar el vértice, una vez indicado no se puede modificar.

Las coordenadas de la textura (s, t) varían de 0 a 1. Por ejemplo un textura de 128x64 su rango viene dado por [0,1]x[0x1], los números mayores a 1 representan que la textura se repite, p.e:

```
glBegin(GL_QUADS);
glColor3f(1.0, 1.0, 1.0); glTexCoord2f(0.0, 0.0);// s = 0 t = 0 glVertex3f(-
1.0, -2.0, 0.0);
```

```
glTexCoord2f(0.0, 3.0);// s = 0 t = 3 glVertex3f(-1.0, 2.0, 0.0);
glTexCoord2f(2.0, 3.0);// s = 2 t = 3 glVertex3f(1.0, 2.0, 0.0);
glTexCoord2f(2.0, 0.0);// s = 2 t = 0 glVertex3f(1.0, -2.0, 0.0);

glEnd();
```

Rotación

Para rotar, tenemos también una función de alto nivel que construye la matriz de transformación y la multiplica por la matriz activa, glRotate. Lleva como parámetros el ángulo a rotar (en grados, sentido horario),

y después x, y y z del vector sobre el cual se quiere rotar el objeto. Una rotación simple, sobre el eje y, de 10º sería

glRotatef(10, 0.0f, 1.0f, 0.0f);

gluLookAt

nombre

gluLookAt - definir una transformación de visualización

C Especificación

void **gluLookAt**(GLdouble *eyeX*,
GLdouble *eyeY*,
GLdouble *eyeZ*,
GLdouble *centerX*,
GLdouble *centerY*,
GLdouble *centerZ*,
GLdouble *upX*,
GLdouble *upY*,
GLdouble *upZ*);

Parámetros

eyeX, *eyeY*, *eyeZ*

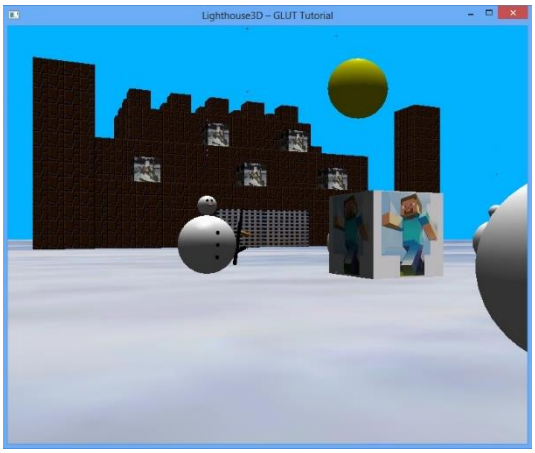
Especifica el la vista

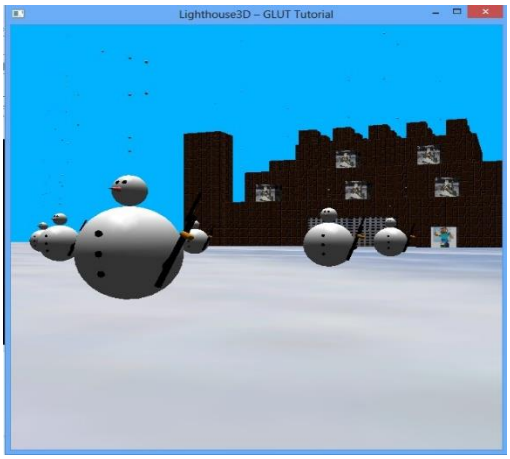
centerX, *centerY*, *centerZ*

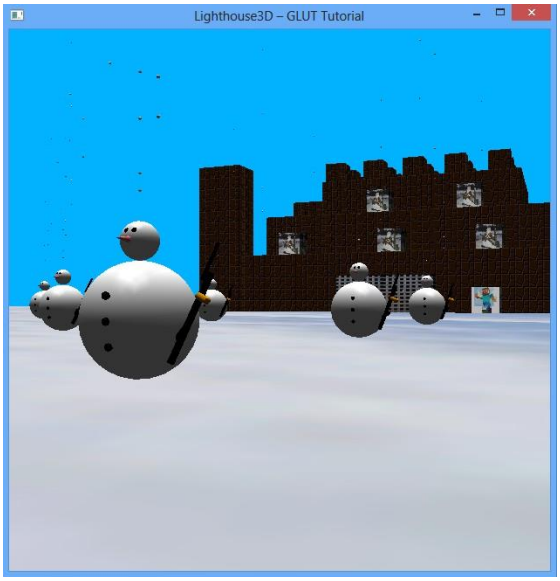
Especifica el punto de referencia

upX, *upY*, *upZ*
Sprecifica el vector

Resultados


Lighthouse3D – GLUT Tutorial


Lighthouse3D – GLUT Tutorial


Lighthouse3D – GLUT Tutorial

Conclusiones

Gracias al desarrollo de este tercer examen parcial pudimos utilizar todos los conceptos vistos en el Cuatrimestre:

-Objetos en 2D

-Rotación

-Traslación

-Texturas

-Objetos en 3D

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACION

ING. EN CIENCIAS DE LA COMPUTACION

GRAFICACIÓN

PROYECTO FINAL

ALUMNO: TITLA TLATELPA JOSE
DE JESÚS

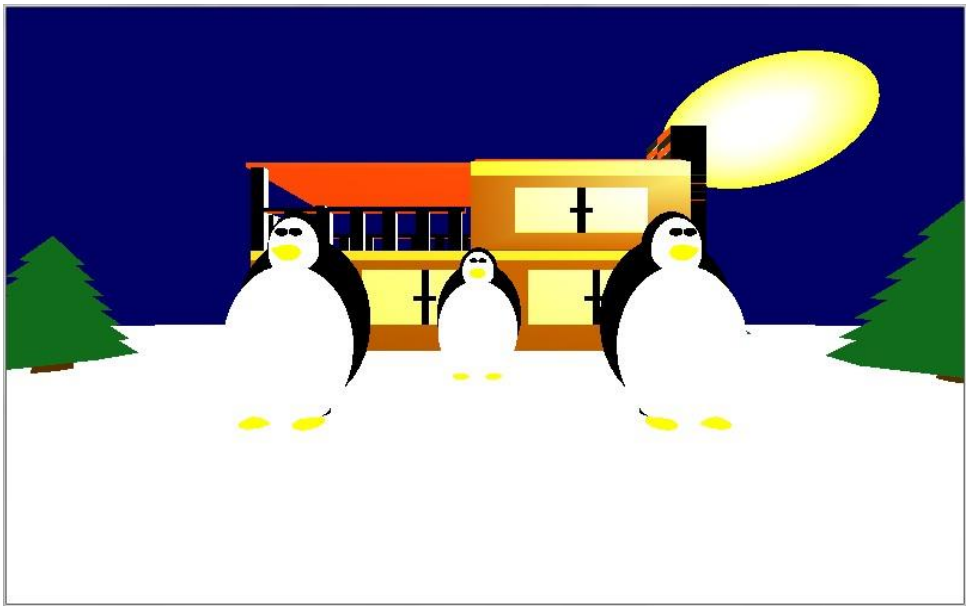
PROFESOR: OLMOS PINEDA IVAN

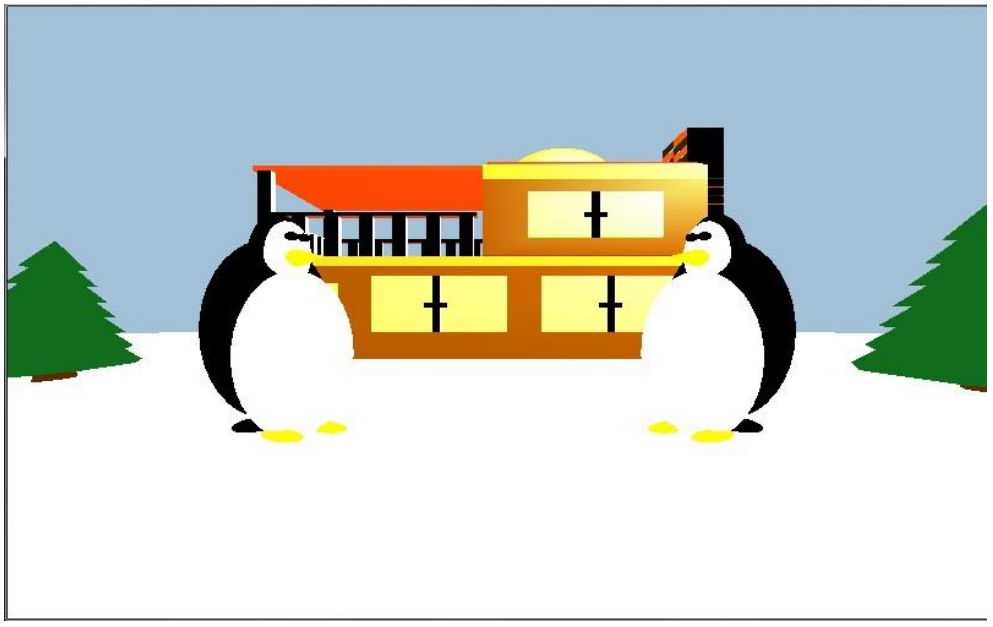
INTRODUCCIÓN

En este documento se explica cuál fue el proceso para graficar un escenario 3D usando las primitivas de OpenGL.

Se realizaron diversos experimentos para lograr los efectos deseados en este escenario y se usó un archivo de cabecera que contiene las funciones para graficar el escenario para que solo sean llamadas en el main.

El escenario que se grafica a continuación es el paisaje de una casa nevada con pingüinos en el cual se hace de día y de noche.





DESARROLLO

Como esta vez se está creando un escenario 3D el init() se inició de manera diferente, se hace uso de gluLookAt(eyex,eyey,eyez,centerx,centery,centerz,upx,upy,upz) y de gluPerspective(fovy,aspect,znear,zfar) para que se maneje la vista en el escenario de una mejor forma.

```
void init(void)
{
 glClearColor(0.0,0.0,0.0,0.0); glMatrixMode(GL_MODELVIEW);
 gluLookAt(0,0,5,0,0,0,0,1,0);/*eyex,eyey,eyez,centerx,centery,centerz,upx,upy,upz*/
 glMatrixMode(GL_PROJECTION);
 gluPerspective(105,1,1,22);/*fovy,aspect,znear,zfar*/
}
```

Algo nuevo que se incluyó en este programa fue la iluminación, con la iluminacion podemos crear mejores efectos, todo esto también depende del lugar en el que se coloque la fuente de luz. En esta ocasión la iluminación está en la zona derecha, es por eso que en el escenario algunas partes de la casa se ven más claras.

Las funciones que se encargan de este efecto son **ilumina()** y **material()**.

A continuación se explicara cómo se pensó y realizó la creación del escenario, algo que cabe destacar es que las únicas figuras que se usaron para la creación del escenario fueron: conos, cubos y esferas.

Nieve(): Simplemente se usa una esfera de color blanco.

Casa(): Para dibujar la casa se usaron los 3 objetos ya mencionados, dependiendo de la parte de la casa que se fuera a dibujar es como se fueron utilizando. La casa se fue graficando por partes, todo esto para llevar orden y para que se fueran acomodando también por posición.

Pino(): Para los pinos solo se usaron conos apilados, el tronco es el único de color café y los demás son de color verde.

Pinguino(): Al igual que la casa, en esta funcion se fue graficando por partes para que se respetara un cierto orden, se tuvo que ser cuidadoso al momento de graficar para que quedara bien unido el objeto a graficar .

Luna(): Al igual que la nieve, se usa solo una esfera, de color blanco, con el efecto de la iluminación se pinta de color amarilla.

Cielo: **cielo(float red, float green, float blue)**

Para lograr el efecto dia-noche se operó directamente con el fondo del escenario, se modificaron los colores para aclarar u oscurecer según el caso, es por eso que se necesita pasar argumentos para controlar el color del cielo.

Escena3D(): Aquí es donde se manda a llamar todo y se coloca para formar el escenario, se usan push y pops por que las traslaciones y rotaciones pueden modificar la maquina de estados y por consiguiente el escenario en un futuro

Finalmente en una sola funciones donde se manda a dibujar con el glutDisplayFunc(), esa funcion es:

```
void Graficacion3D(void)
{
 glPushMatrix();
 Escena3D();
 glPopMatrix();
 glFlush();
}
```

Y el main() queda así:

```
int main(int argc, char *argv[]){
 glutInit(&argc, argv);
 glutInitWindowSize(800,500);
 glutCreateWindow("Escena 3D");
 init();
 glutDisplayFunc(Graficacion3D);
 glutKeyboardFunc(key);
 glutIdleFunc(idle);
 glutMainLoop();
 return 0;
}
```

EXPERIMENTOS

En esta ocasión se realizaron experimentos con el teclado y con el control del día y la noche.

Uso del teclado.

Para controlar la perspectiva del escenario girando hacia cualquiera de las 4 direcciones (izquierda, derecha, arriba, abajo), se hace uso de la funcion para manejar el teclado. Las direcciones se asignaron así: a (izquierda), s (abajo), d (derecha), w (arriba), según la tecla, se modifica un valor en gluLookAt().

```
static void key(unsigned char key){
    switch (key)
    {
        case 27 :
        case 'q':
            exit(0);
            break;


        case 'w':
            if (k!=0){
          gluLookAt(0,3,3,0,0,0,0,1,0);/*eyex,eyey,eyez,centerx,centery,centerz,upx,upy,upz*/
            }
            break;
        case 'a':
            if (k!=0){
          gluLookAt(3,0,3,0,0,0,0,1,0);/*eyex,eyey,eyez,centerx,centery,centerz,upx,upy,upz*/
            }
            break;
        case 's':
```

```
            if (k!=0){
                gluLookAt(0,-3,3,0,0,0,0,1,0);
            }
            break;

        case 'd':
            if (k!=0) {
                gluLookAt(-3,0,3,0,0,0,0,1,0);
            }
            break;
    }

}
```

Control del día y la noche.

Para controlar el color del cielo y el movimiento de los objetos según el día y la noche se hizo uso de condicionales y banderas, a continuación se explica la lógica que se siguió.

En esta funcion se comprueba que el color del cielo ya sea negro o más oscuro, si ese es el caso, la bandera se prende.

```
if(red<-20 || blue<-20 || green<-20){
      flag=1;
  }
```

Si la bandera está encendida, es decir, es de noche, entonces todo se hace en el orden contrario para hacer la transición noche/día.

```
if(fla
  g=
  =1)
  {
  red
  +=
  0.0
  1;
  gre
  en
  +=
  0.0
  08;
  blu
  e+
  =0.
  00
  04;
  }
```

Si el color del cielo es del azul claro que se estableció al inicio, entonces se apaga la bandera y eso indica que es de día.

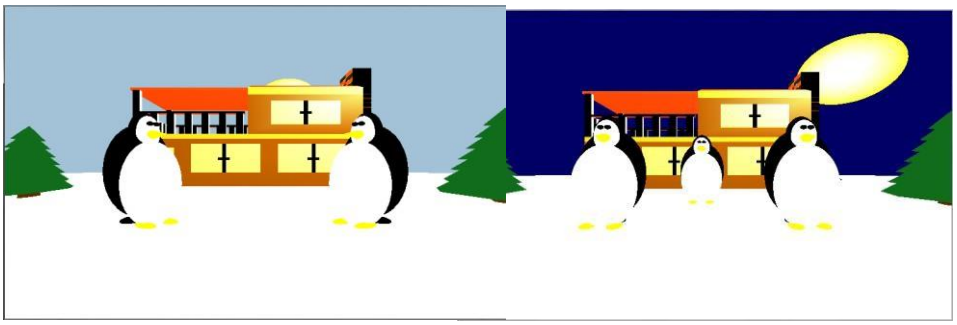
```
if(red>0.04902 || blue>0.147059 || green>0.147059){
      flag=0;
   }
```

Si la bandera está apagada, indicando que es de día, entonces las rotaciones y actualizaciones se hacen en orden para hacer que se haga de noche.

```
if(fl
  a
  g
  =
  =
  0)
  {
  re
  d-
  =
  0.
  0
  1;
  gr
  e
  e
  n-
  =
  0.
  0
  0
  8;
  bl
  u
  e-
```

=
0.
0
0
0
4;
}

Estos ciclos se van a estar realizando una y otra vez para que el ciclo día/noche no pare.



en 1

wxDev-C++ es un entorno de desarrollo integrado libre basado en el popular Dev-C++ para C y C++.

user

Introducción

A continuación se mostrara una breve explicación de la realización de un escenario con rotaciones, texturas e iluminación. El profesor con anterioridad nos explicó algunos tips de cómo se debía realizar. De esta manera si se presentaba alguna duda podíamos consultar con el profesor o buscar en internet como se implementaría. Se utilizó la librería SOIL, para la implementación de las texturas.

Desarrollo

Las texturas que se ocuparon fueron las siguientes.

Lo primero que se realizo fue inicializar las funciones de la característica ambiental de nuestro escenario, de esta manera se movió la posición de la candela que estamos ocupando en nuestro ambiente de trabajo.

A continuación se muestra cómo quedaría.

Inicializa las propiedades de la fuente de luz.

```
void init(void)
{
    GLfloat light_ambient[] = { 0.75, 0.75, 0.75, 1.0 };
    GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat light_position[] = { 0.45, 1.0, 0.45, 0.0 };

    glLightfv (GL_LIGHT0, GL_AMBIENT, light_ambient);
    glLightfv (GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv (GL_LIGHT0, GL_SPECULAR, light_specular);
    glLightfv (GL_LIGHT0, GL_POSITION, light_position);

    glEnable (GL_LIGHTING);
    glEnable (GL_LIGHT0);
    glDepthFunc(GL_LESS);
    glEnable(GL_DEPTH_TEST);
}
```

De esta manera se inicializa nuestro escenario para incluir la iluminación.

```
void reshape(int w, int h)
{

    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(30.0, (GLfloat) w/(GLfloat) h, 0.001, 1000.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0,-0.0,-50.0);
}
```

S encarga de dimensionar la pantalla al iniciarla y como se modifica.

```
char* texture[15] = {
            "1.jpg",
            "2.jpg",
            "3.png",
            "4.jpg",
            "6.jpg",
            "7.jpg",
            "8.jpg",
            "9.jpg",
            "10.jpg",
            "5.jpg",
            "11.jpg",
            "12.jpg",
            "13.jpg",
            "14.png"
      };
```

De igual manera se agregó la librería SOIL a nuestro documento ya lo habíamos hecho previamente.

De esta manera para jalar varias imágenes se debe declarar un arreglo de caracteres, como se muestra a continuación.

En esta parte se incluirán todas las imágenes que se manejaran en el escenario, hay que recordar que debemos anotar el nombre correctamente para que puedan ser llamadas de igual manera las imágenes deben de estar guardadas en la carpeta que se crea de MingW.

A continuación se mostrará las funciones que se utilizaron para cada objeto del escenario.

```
void arboles()
{
    glPushMatrix();
    glTranslatef(-5.5,-1.5,2.0);
    glRotatef(aY, 0.0f, 1.0f, 0.0f);
    drawPiramide();
    tronco();
    glPopMatrix();

    glPushMatrix();
    glTranslatef(5.5,-1.5,2.0);
    glRotatef(aY1, 0.0f, 1.0f, 0.0f);
    drawPiramide();
    tronco();
    glPopMatrix();
    aY+=0.16f;
    aY1+=0.016f;
}
```

Se dibujaran los arboles con una pirámide que ya teníamos anteriormente, se le agregara movimiento en el eje y para que le dé un efecto agradable al escenario.

Se dibujara la casa flotante utilizando un rectángulo ya establecido con anterioridad, solo que se declaró una bandera para que cambie la textura, así como también el escalamiento y el movimiento de la casa.

Debemos de tener en cuenta que para dibujar un objeto en 3D se deben crear cada lado de nuestro objeto con las coordenadas especificas en el eje x, y , z. para que de esta manera de el efecto que de que se encuentra en 3D.

```
void df()
{
    glPushMatrix();
        glPushMatrix();
        glTranslatef(6.0,-3.0,-2.0);
        glTranslatef(0.0,angulo1,0.0);
        glColor3f(1.0,1.0,1.0);

          glPushMatrix();
           glTranslatef(15.0,2,2.0);
           glScalef(0.8,0.8,0.8);
           band=1;
           drawC();
           glPopMatrix();

            glPushMatrix();
            glTranslatef(15.0,-1,2.0);
            glScalef(0.8,0.8,0.8);
            band=0;
            draw_pp();
            glPopMatrix();

             glPushMatrix();
             glTranslatef(15.0,-1.6,2.0);
             band=1;
             glScalef(0.5,0.8,0.6);
```

```
void molino_b()
{
    //glLoadIdentity();
    glPushMatrix();
    glBindTexture(GL_TEXTURE_2D, tempTextureID[2]);
    //glClearColor(1.0, 1.0,1.0, 1.0);

    glLineWidth(5);
    glBegin(GL_LINES);
    glColor3f(0.0, 0.0, 0.0);
    glVertex3f(0,-0.0,1.0);
    glVertex3f(0,-0.0,-1.0);
    glEnd();
    glRotatef(angulo,0.0f,0.0f,1.0f);
    glPushMatrix();
    glTranslatef(0.0,0.0,1.0);
    molino_b1();
    mov();
    glPopMatrix();
```

Para la realización del molino se utilizó la escalación para que se mueva respecto al eje x o y. de esta manera se reutiliza el código y se hace más óptimo.

A continuación se genera en una función en el cual se definirán nuestro sol y el material que se ocupa de igual manera se le agrego movimiento para que se vea más creíble el escenario, en el efecto de la nuve y el sol no se le agrego textura.

```
void sol()
{
    glPushMatrix();
    GLfloat mat_ambient_esfera2[] = {1.0, 0.1, 0.1, 1.0f};
    GLfloat mat_diffuse_esfera2[] = {1.0, 1.0, 0.0, 0.0f};
    GLfloat mat_specular_esfera2[] = {0.8, 0.8, 0.8, 1.0f};

    glMaterialfv (GL_FRONT, GL_AMBIENT, mat_ambient_esfera2);
    glMaterialfv (GL_FRONT, GL_DIFFUSE, mat_diffuse_esfera2);
    glMaterialfv (GL_FRONT, GL_SPECULAR, mat_specular_esfera2)
    //glMaterialf (GL_FRONT, GL_SHININESS, 900.0f);

    glPushMatrix();
    glTranslatef(0.0,0.0,-20.0);
   // glColor3f(1.0,1.0,1.0);
    //glBindTexture(GL_TEXTURE_2D, tempTextureID[15]);

    glPushMatrix();
        glRotatef(angulo6, 0.0f, 0.0f, 1.0f);
       // glColor3f(1.0, 1.0, 0.0);
        glTranslatef(30,-5.0f,0.0f);
        glutSolidSphere(2,20,20);
    glPopMatrix();
    angulo6+=.005;
```

Una de las funciones principales fue la utilización del objeto de metal gris en el cual se iban modificando las texturas para que lo vuélvamos a utilizar su forma, de esta manera se realizó la casa.

```
 void draw_obj()
{
   glPushMatrix();
   glBindTexture(GL_TEXTURE_2D, tempTextureID[7]);
   //glClearColor(1.0, 1.0, 1.0, 1.0);
   //Cara frontal
   glBegin(GL_POLYGON);
         glTexCoord2f(0.0f,0.0f);
         glVertex3f(-2.0f,-10.0f,-2.0f); // vertex 1
          glTexCoord2f(1.0f,0.0f);
         glVertex3f(2.0f, -10.0f,-2.0f); // vertex 2
          glTexCoord2f(1.0f,1.0f);
         glVertex3f(2.0f,   -7.0f,-2.0f); // vertex 3
          glTexCoord2f(0.0f,1.0f);
         glVertex3f(-2.0f, -7.0f,-2.0f);
   glEnd();

   glBegin(GL_POLYGON);
         glTexCoord2f(0.0f,0.0f);
```

```
        glVertex3f(-2.0f,-10.0f,0.0f);
// vertex 1
         glTexCoord2f(1.0f,0.0f);

        glVertex3f(2.0f,  -10.0f,0.0f);
// vertex 2
         glTexCoord2f(1.0f,1.0f);

        glVertex3f(2.0f,    -7.0f,0.0f);
// vertex 3
         glTexCoord2f(0.0f,1.0f);

        glVertex3f(-2.0f, -7.0f,0.0f);
   glEnd();

   glBegin(GL_POLYGON);
        glTexCoord2f(0.0f,0.0f);
        glVertex3f(2.0f,-10.0f,0.0f);
// vertex 1
         glTexCoord2f(1.0f,0.0f);

        glVertex3f(2.0f, -10.0f,-2.0f);
// vertex 2
         glTexCoord2f(1.0f,1.0f);

        glVertex3f(2.0f,   -7.0f,-2.0f);
// vertex 3
         glTexCoord2f(0.0f,1.0f);

        glVertex3f(2.0f, -7.0f,0.0f);
   glEnd();

   glBegin(GL_POLYGON);
        glTexCoord2f(0.0f,0.0f);
        glVertex3f(-2.0f,-10.0f,0.0f);
// vertex 1
         glTexCoord2f(1.0f,0.0f);

        glVertex3f(-2.0f,       -10.0f,-
2.0f); // vertex 2
         glTexCoord2f(1.0f,1.0f);

        glVertex3f(-2.0f,  -7.0f,-2.0f);
// vertex 3
         glTexCoord2f(0.0f,1.0f);

        glVertex3f(-2.0f, -7.0f,0.0f);
   glEnd();
   glPopMatrix();
}
```

```
if (mov1)
angulo1 -= 0.002f;
else
angulo1 += 0.002f;

if (angulo1 < 0.0f)
mov1 = false;
else if (angulo1 > 20.0f)
mov1 = true;

if (mov2)
angulo2 -= 0.005f;
else
angulo2 += 0.005f;

if (angulo2 < -15.0f)
mov2 = false;
else if (angulo2 > 0.0f)
mov2 = true;
```

En el display se agregaron los límites de cada objeto que está en movimiento de nuestro escenario.

Se ingresaron condiciones para cambiar la textura.

```
void draw_pp()
{

   glPushMatrix();
   //int inc=0;
   //glClearColor(1.0, 1.0, 1.0, 1.0);
   if(band==1)
   {glBindTexture(GL_TEXTURE_2D,
tempTextureID[6]);
   inc=0;}
   else
   {
   band=0;
```

```
   glBindTexture(GL_TEXTURE_2D,
tempTextureID[10]);

   }
```

```
void keyboard(int key, int x, int y) {
    switch (key) {
    case GLUT_KEY_RIGHT:
        gluLookAt(.1,0,.5,0,0,0,0,1,0);
        break;
    case GLUT_KEY_LEFT:
        gluLookAt(-.1,0,.5,0,0,0,0,1,0);
        break;
    case GLUT_KEY_UP:
        gluLookAt(0,.1,.5,0,0,0,0,1,0);
        break;
    case GLUT_KEY_DOWN:
        gluLookAt(0,-.1,.5,0,0,0,0,1,0);
        break;
    case GLUT_KEY_F1:
        gluLookAt(0,0,.5,0,0,0,0,1,0);
        break;
    case GLUT_KEY_F2:
        gluLookAt(0,0,0.1,0,0,0,0,1,0);
    default:
        break;
    }
```

Se utiliza el glLookAt para que nuestro observador se vaya moviendo alrededor del escenario, de esta manera se mueve haia el eje x positivo, asi como también negativo, hacia el eje y z.

Del eje z solo pudo moverse asía atrás ya que no encontré la forma de que se mueva hacia adelante.

Realizando lo dicho anteriormente nos quedaría de la siguiente manera.

Podemos observar que se cumplió lo establecido, el escenario es en 3D, tiene el efecto de iluminación el sol y la nube y tiene movimiento.

Conclusión

Debemos de aprender a usar las funciones de opengl para que de esta manera se nos facilite entenderle, esta práctica se realizó para que vayamos viendo cómo funcionan las fuentes de luz, así como también las texturas, y tener practica en la realización de objetos 3D, ya que al principio fue un poco confuso porque siempre habíamos trabajado con objetos en 2D pero poco a poco le fuimos entendiendo. Lo complicado en el escenario es la utilización de las funciones que ya tiene determinadas opengl ya que si no las conocemos bien es posible que eso haga que nos atrasemos en la realización de algún ejercicio a futuro.

BENEMERITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

MATERIA: GRAFICACIÓN

DOCENTE: DR. IVÁN OLMOS PINEDA

ALUMNA: JOCELYN DÁVILA HERNÁNDEZ

MATRÍCULA: 201011077

PRIMER EXAMEN PARCIAL

Documento Técnico

PRIMAVERA 2015

INTRODUCCIÓN:

El documento que se presenta muestra cómo se realiza un escenario (de alguna película, caricatura, o cualquier dibujo) de lo que nosotros decidiéramos hacer.

Lo que se realizará es en el lenguaje wxDev-C++ utilizando OpenGL y algunas funciones, no todas podían estar presentes.

El programa desarrollado pretende trabajar sin la función GL_LINES(), ya que solamente nos permiten la utilización de puntos y sobre esos puntos hacer líneas, ya que como bien sabemos una línea es una sucesión de puntos, por lo tanto, una vez teniendo una clase "Punto" se puede realizar una línea y un círculo, y así a su vez teniendo una clase llamada "línea" crear cualquier tipo de figura, como triangulo, rectángulo y cuadrado.

CONCEPTOS:

PUNTO: Es uno de los entes fundamentales, junto con la recta y el plano. Son considerados conceptos primarios, es decir, que sólo es posible describirlos en relación

con otros elementos similares o parecidos. Se suelen describir apoyándose en los postulados característicos, que determinan las relaciones entre los entes geométricos fundamentales.

LINEA: Es una sucesión continua de puntos trazados, como por ejemplo un trazo o un guion. Las líneas suelen utilizarse en la composición artística, se denomina en cambio «*raya*» a trazos rectos sueltos, que no forman una figura o forma en particular.

CIRCULO: En geometría euclídea, es el lugar geométrico de los puntos del plano cuya distancia a otro punto fijo, llamado centro, es menor o igual que una cantidad constante, llamada radio. En otras palabras, es la región del plano delimitada por una circunferencia y que posee un área definida.

TRIANGULO: En geometría, es la reunión de tres segmentos que determinan tres puntos del plano y no colineales. Cada punto dado pertenece a dos segmentos exactamente. Los puntos comunes a cada par de segmentos se denominan vértices del triángulo y los segmentos de recta determinados son los lados del triángulo. Dos lados contiguos forman uno de los ángulos interiores del triángulo. Un triángulo es una figura estrictamente convexa.

CUADRADO: Es un paralelogramo que tiene sus lados iguales y además sus cuatro ángulos son iguales y rectos, tiene 4 ejes de simetría, 4 vértices y 4 aristas.

COORDENADAS: Es un sistema que utiliza uno o más números (*coordenadas*) para determinar unívocamente la posición de un punto o de otro objeto geométrico.

OPERACIONES MATRICIALES: Las operaciones que se pueden hacer con matrices provienen de sus aplicaciones, sobre todo de las aplicaciones en álgebra

lineal. De ese modo las operaciones, o su forma muy particular de ser implementadas, no son únicas.

CLASES: Es una plantilla para la creación de objetos de datos según un modelo predefinido. Las clases se utilizan para representar entidades o conceptos, como los sustantivos en el lenguaje. Cada clase es un modelo que define un conjunto de variables -el estado, y métodos apropiados para operar con dichos datos -el comportamiento. Cada objeto creado a partir de la clase se denomina instancia de la clase.

FUNCIONES: Las funciones y métodos en java sirven para la optimización de código y la reutilización del mismo. Si quisiéramos realizar sumas de dos o más números en diferentes puntos de la aplicación sería un problema estar realizando el mismo código de la sumatoria a cada instante que lo necesitemos, por lo tanto lo adecuado sería realizar una sola vez la suma y llamar a tal función cada vez que lo necesitemos.

HERENCIA: Es, después de la agregación o composición, el mecanismo más utilizado para alcanzar algunos de los objetivos más preciados en el desarrollo de software como lo son la reutilización y la extensibilidad. A través de ella los diseñadores pueden crear nuevas clases partiendo de una clase o de una jerarquía de clases preexistente (ya comprobadas y verificadas) evitando con ello el rediseño, la modificación y verificación de la parte ya implementada. La herencia facilita la creación de objetos a partir de otros ya existentes e implica que una subclase obtiene todo el comportamiento (métodos) y eventualmente los atributos (variables) de su superclase.

DESARROLLO:

Primero realice la clase “Punto” de ahí se parte para cualquier otra función o en este caso otra clase como “Línea”.

La clase “Punto” es la siguiente:

```
class Punto
{
```

```
private:    // Aquí es donde se declaran las variables a utilizar.
int x;
int y;

public:
    Punto();    //Constuctor
    ~Punto();
    int regresaX();  //  y a partir de esta línea de código empiezan todas las funciones que se
    int regresaY();     van a utilizar en nuestra clase y a su vez en otras.
    void DaValores(int, int);
    void Translada(int, int);
    void Rota(int);
    void Escala(int, int);
    void DeformaX(int);
    void DeformaY(int);
    void DeformaXY(int, int);
    void DeformaLinea(int, int);

};
```

La clase “Línea” es la siguiente:

```
class Linea
{
    private : Punto pi,pf;

    public:

        Linea();
        ~Linea();
        void DaValoresLinea(int, int, int, int);
        void TransladaLinea(int, int);
        void EscalaLinea(int, int );
        void DibujaLinea();

};
```

Y así se realizan las clases restantes tomando en cuenta estas dos primeras clases que son las principales, posterior se crean las siguientes clases que ayudarán a realizar el escenario:

- Circulo.h
- Cuadrado.h
- Triangulo.h

Para la clase “Circulo” no se utiliza la clase “Línea”, pero para Cuadrado y Triangulo si es fundamental dicha clase.

Después de que se realizan todas las clases, se crean los .cpp de cada clase, por lo tanto, si ya había creado la clase “Punto” se hace su respectivo .cpp y es el siguiente:

```
Punto :: Punto()
  { x=0;
    y=0;
       }

 Punto :: ~Punto()
  { }

 int Punto :: regresaX()
   {
     return x;
   }

 int Punto :: regresaY()
   {
      return y;
   }

  void Punto :: DaValores(int p1, int p2)
 {
      x=p1;
      y=p2;
  }

 void Punto ::Translada(int p1, int p2)
 {
```

```
 int a[3][3];
 int resul[3][1];
 int aPunto[3][1];

 int i,j,z,temp;

//matriz a
 a[0][0]=1;
 a[1][0]=0;
 a[2][0]=0;

 a[0][1]=0;
 a[1][1]=1;
 a[2][1]=0;

 a[0][2]=p1;
 a[1][2]=p2;
 a[2][2]=1;

 //matriz aPunto

 aPunto[0][0]=x;
 aPunto[1][0]=y;
 aPunto[2][0]=1;

 //matriz Resultante

 for(i=0; i<3; i++)
 {
      for (j=0; j<1; j++)
      {
         temp=0;

      for (z=0; z<3; z++)
      {
         temp += a[i][j]*aPunto[j][z];
         resul[i][z]=temp;
      }
      }
 }
 }

 void  Punto :: Rota(int angulo)
 {

 int a[3][3];
 int aPunto[3][1];
 int resul[3][1];

 int i,j,z,temp;

//matriz a
 a[0][0]=cos(angulo*3.1416/180);
 a[1][0]=sin(angulo*3.1416/180);
 a[2][0]=0;

 a[0][1]= (-1)*(sin(angulo*3.1416/180));
 a[1][1]= cos(angulo*3.1416/180);
 a[2][1]= 0;
```

```
a[0][2]=0;
a[1][2]=0;
a[2][2]=1;

//matriz aPunto

aPunto[0][0]=x;
aPunto[1][0]=y;
aPunto[2][0]=1;

 //matriz Resultante

for(i=0; i<3; i++)
{
     for (j=0; j<1; j++)
     {
        temp=0;

     for (z=0; z<3; z++)
     {
        temp += a[i][j]*aPunto[j][z];
        resul[i][z]=temp;
     }
     }
}

 }

void Punto :: Escala(int r1, int r2)
{

int a[3][3];
int aPunto[3][1];
int resul[3][1];

int i,j,z,temp;

//matriz a
a[0][0]=r1;
a[1][0]=0;
a[2][0]=0;

a[0][1]=0;
a[1][1]=r2;
a[2][1]=0;

a[0][2]=0;
a[1][2]=0;
a[2][2]=1;

//matriz aPunto

aPunto[0][0]=x;
aPunto[1][0]=y;
aPunto[2][0]=1;

 //matriz Resultante

for(i=0; i<3; i++)
{
```

```
      for (j=0; j<1; j++)
      {
         temp=0;

      for (z=0; z<3; z++)
      {
         temp += a[i][j]*aPunto[j][z];
         resul[i][z]=temp;
      }
      }
 }

 }


   void Punto :: DeformaX(int shx)
   {
 int a[3][3];
 int aPunto[3][1];
 int resul[3][1];

 int i,j,z,temp;

//matriz a
 a[0][0]=1;
 a[1][0]=0;
 a[2][0]=0;

 a[0][1]=shx;
 a[1][1]=1;
 a[2][1]=0;

 a[0][2]=0;
 a[1][2]=0;
 a[2][2]=1;

 //matriz aPunto

 aPunto[0][0]=x;
 aPunto[1][0]=y;
 aPunto[2][0]=1;

  //matriz Resultante

 for(i=0; i<3; i++)
 {
      for (j=0; j<1; j++)
      {
         temp=0;

      for (z=0; z<3; z++)
      {
         temp += a[i][j]*aPunto[j][z];
         resul[i][z]=temp;
      }
      }
 }
   }

void Punto :: DeformaY(int shy)
```

```
{

  int a[3][3];
  int aPunto[3][1];
  int resul[3][1];

  int i,j,z,temp;

 //matriz a
  a[0][0]=1;
  a[1][0]=shy;
  a[2][0]=0;

  a[0][1]=0;
  a[1][1]=1;
  a[2][1]=0;

  a[0][2]=0;
  a[1][2]=0;
  a[2][2]=1;

  //matriz aPunto

  aPunto[0][0]=x;
  aPunto[1][0]=y;
  aPunto[2][0]=1;

   //matriz Resultante

  for(i=0; i<3; i++)
  {
       for (j=0; j<1; j++)
       {
          temp=0;

       for (z=0; z<3; z++)
       {
          temp += a[i][j]*aPunto[j][z];
          resul[i][z]=temp;
       }
       }
  }

}

void Punto :: DeformaXY(int shx, int shy)
{

int a[3][3];
int aPunto[3][1];
int resul[3][1];

  int i,j,z,temp;

 //matriz a
  a[0][0]=1;
  a[1][0]=shy;
  a[2][0]=0;

  a[0][1]=shx;
```

```
  a[1][1]=1;
  a[2][1]=0;

  a[0][2]=0;
  a[1][2]=0;
  a[2][2]=1;

  //matriz aPunto

  aPunto[0][0]=x;
  aPunto[1][0]=y;
  aPunto[2][0]=1;

   //matriz Resultante

  for(i=0; i<3; i++)
  {
       for (j=0; j<1; j++)
       {
          temp=0;

       for (z=0; z<3; z++)
       {
          temp += a[i][j]*aPunto[j][z];
          resul[i][z]=temp;
       }
       }
  }

}

void Punto :: DeformaLinea(int shx, int Yref)
{

int a[3][3];
int aPunto[3][1];
int resul[3][1];

  int i,j,z,temp;

 //matriz a
  a[0][0]=1;
  a[1][0]=0;
  a[2][0]=0;

  a[0][1]=shx;
  a[1][1]=1;
  a[2][1]=0;

  a[0][2]=((shx)*(-1))*(Yref);
  a[1][2]=0;
  a[2][2]=1;

  //matriz aPunto

  aPunto[0][0]=x;
  aPunto[1][0]=y;
  aPunto[2][0]=1;

   //matriz Resultante
```

```
 for(i=0; i<3; i++)
 {
      for (j=0; j<1; j++)
      {
         temp=0;

      for (z=0; z<3; z++)
      {
         temp += a[i][j]*aPunto[j][z];
         resul[i][z]=temp;
      }
      }
 }

}
```

En lo que se muestra anteriormente se trabaja con matrices, las funciones con las que se trabaja son:

- **RegresaX():** Esta función solamente retorna el valor de la variable x.

- **regresaY():** Regresa el valor de Y.

- **DaValores():** Solamente se pasa como parámetro los valores que se le asignan desde un principio (x,y).

- **Translada():** Como bien su nombre lo dice es la función en donde se le dan valores en el cual va a trasladar dicho punto.

- **Rota():** En esta función el parámetro que se le asigna es de un ángulo, ya que es lo que se va a rotar.

- **Escala():** En la función escala, se pasan parámetros para que ya sea una figura o simplemente una línea puedan ser más grandes.

Las funciones mencionadas anteriormente las únicas que trabajan con matrices, es traslación, rotación y escalamiento.

Todas las figuras tienen estás funciones, no todas, pero si las que ayudarán a realizar el escenario, la que menos tiene es cuando se hace un circulo.

Y como ya se había mencionado, el principal motivo de realizar todas estas clases y funciones con matrices es que no se ocupan funciones propias de OpenGL por ejemplo la que hace las líneas, triángulos, las funciones que realizan la traslación, rotación y escala.

Por ejemplo en la función DibujaLinea() se podría utiliza GL_LINES pero como eso no es posible Para determinar las posiciones de los píxeles a lo largo de un trayecto de línea recta se utilizan las propiedades geométricas de la línea. La ecuación punto-pendiente cartesiana para una línea recta es:

M= Yend – Y0 / Xend – X0 , siendo m la pendiente de la línea y b el punto de intersección con el eje y. Puesto que los dos extremos del segmento de la línea tiene las coordenadas (x0,y0), y (Xend, Yend).

El escenario el cual yo decidí realizar fue el personaje llamado "MIKE WAZOWSKI" de la película Monsters, Inc. Ya que es un personaje que en lo personal me gusta mucho, entonces opte por presentarlo en este primer examen parcial de graficación.

Ya que se hicieron las clases y los .cpp entonces ya se presenta la clase principal, en donde en esta clase se llamaran todas las funciones que se crearon y que se van a necesitar.

Creo una función llamada "FIGURAS", y precisamente es para realizar líneas, triángulos y círculos con sus dichos parámetros.

El ejemplo más claro que presento a continuación es donde

Hace el círculo de la cabeza, ya que la cabeza de mi personaje es totalmente un círculo.

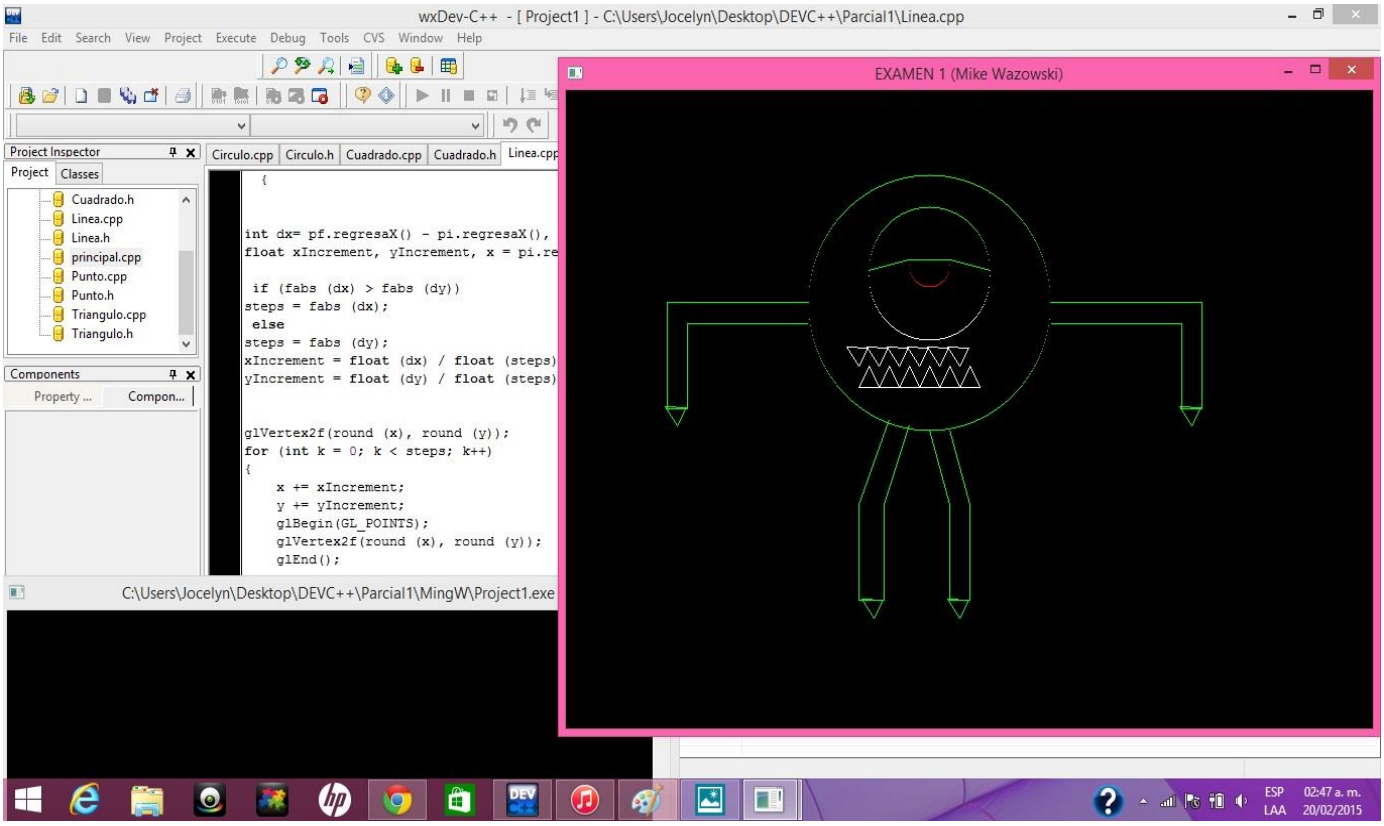
```
//Cabeza

c1.DaValoresCirculo(360,400,60);
c2.DaValoresCirculo(360,430,60);
l1.DaValoresLinea(300,430,340,440);
l2.DaValoresLinea(340,440,380,440);
l3.DaValoresLinea(380,440,420,430);
c1.EscalaCirculo(2);
c1.DibujaCirculo();
```

Aquí lo único que se hace es mandar a “llamar” a las funciones para que puedan realizar lo que se solicita, lo único diferente es que se le asignan los parámetros que se mencionaron anteriormente para cada función.

Y así en la clase principal se hacen los brazos, piernas, manos, dientes, ojo y pies de mi personaje, por supuesto usando las funciones ya antes declaradas.

La idea era realizar la siguiente imagen:

N**o salió igual, el escenario que proporciono de mi trabajo es el que muestro a continuación:**



CONCLUSIONES:

En lo personal lo que puedo mencionar acerca del trabajo que realice es que resulta un poco complicado hacer clases y

funciones para poder dibujar cualquier escenario pero sin utilizar las que ya están por default en OpenGL.

Por otro lado, es necesario trabajar con operaciones matriciales, y otro tipo de forma para hacer líneas o cualquier otra figura, para que no se olvide desde donde parten todas esas funciones.

BIBLIOGRAFÍA:

http://xcodigoinformatico.blogspot.mx/2012/09/primitivas-graficas-en-opengl-codigo-en.html

http://recursostic.educacion.es/descartes/web/materiales_didacticos/Calculo_matricial_d3/opmatr.htm

http://es.wikipedia.org/wiki/L%C3%ADnea

http://es.wikipedia.org/wiki/Monsters,_Inc.

Hoja de presentación para escenario 2D en OpenGL

Materia: Graficación

Nombre: Jesús Paleta Islas

Desarrollo del tema

En este examen se busca desarrollar un escenario en 2D mediante predirectivas de OpenGL. El escenario deberá incluir rotaciones, traslaciones y manejo de texturas.

El bosquejo del escenario es representar un parque junto a una ciudad. En dicho parque hay una montaña rusa, un rio con peces, árboles y al fondo se pueden apreciar los edificios de una ciudad. Además se pueden apreciar las estrellas y la Luna en la noche.

En primera instancia, se buscó desarrollar una función para dibujar círculos, la cual ya se había escrito en prácticas pasadas. Así que se reutilizó este código.

Tras crear un primer bosquejo de las partes del escenario (sin colores, ni texturas) se investigó cuál sería el más adecuado manejador de texturas a utilizar. Finalmente se dio prioridad a cargar texturas con las mismas funciones en glut.h. El método en sí utiliza una estructura con un apuntador a los datos de la imagen y y una función para cargar la textura a memoria.

Se utilizó este método con el fin de evitar importar librerías extras y debido a que el manejo de texturas se volvía más sencillo. Además se pueden cargar varias texturas de manera más simple.

Se utilizaron texturas para el río, para los árboles y para el camino. El único problema con este método es la necesidad de convertir las imágenes (ya sean de tipo .png o .jpg) a tipo “tga”. Sin embargo, gracias a Internet esto se puede hacer en un par de *clicks* y completamente en línea sin tener que descargar aplicaciones de terceros.

El programa cuenta con funciones establecidas para el dibujado de los edificios, de las estrellas, del árbol y de la montaña rusa.

Para dibujar los árboles y los edificios se creó uno de base y a través de un ciclo se hacían traslaciones para dibujar los siguientes en su lugar correspondiente.

El principal reto a la hora de programar fue tener cuidado en como colocábamos los llamados a las funciones glPushMatrix() y glPopMatrix() de manera que la rotación y traslación de ciertos objetos no afectaran al escenario en general.

Otro problema que se halló, fue poner texturas dentro del círculo. Esto debido a que nuestra función ya predefinida para dibujar un círculo lo hacía a través de triángulos, específicamente mediante TRIANGLE_FAN. El problema estaba en que la textura no se apreciaba en su totalidad pues se llenaba conforme a cada triángulo y no de acuerdo al círculo en general.

Como conclusión, se puede decir que OpenGL ofrece muchas herramientas para el dibujado en 2D, pero aun así fue necesario desarrollar varias funciones para poder darle un mejor aspecto a nuestro escenario. Otra cosa importante es ver cómo es que las texturas funcionan, algo que el método con glut.h nos permitió observar. Finalmente, las traslaciones y rotaciones dan una apariencia mejorada a nuestro escenario y nos hacer ver la utilidad de *MainLoop*.

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Ingeniería en Ciencias de la Computación

Graficación

Profesor: Iván Olmos Pineda

Examen 1

Alumna: Alma Amaranta Pérez Cantor

Sección 101

Febrero 2015

Para comenzar a desarrollar este trabajo lo primero que hice fue crear las figuras geométricas que considere básicas (punto, línea, cuadrilátero, triangulo y circulo). Las figuras fueron manejadas a través de clases, cada una con sus diferentes métodos y atributos.

Para construir un punto necesite de dos coordenadas, y como cada clase su respectivo constructor en el cual inicializaba los valores de las coordenadas, para el punto se inicalizaron en (0,0), también cada clase figura tiene sus respectivos métodos:

-Destructor.

-setValues() cambia los valores de las coordenadass o los puntos que definen a cada figura.

-origen() hace que la figura regrese a su valor de inicialización, es decir a la coordenada (0,0)

-draw (drawLine(), drawTriangulo(), drawCuadrilatero(), drawCircle()), mandan a dibujar cada una de la figuras geométricas. Las funciones para dibujar las figuras no reciben ningún parámetro, lo que hacen es que internamente modifican los valores de los puntos.

Volviendo a la clase Punto, es en ella donde se usa la única instrucción propia de OpenGL y la que hace que todas las figuras se puedan mostrar en pantalla. En su método draw(), el punto es dibujado mediante la instrucción glBegin(GL_POINTS) la cual recibe los valores de las coordenadas que tiene el punto que se obtienen a través de la función getX() y getY() propias de la clase Punto que se desarrollo.

Como ya podemos dibujar un Punto, lo que hice fue crear la clase Linea, la cual tiene los métodos dichos previamente. El método para dibujar una línea se llama drawLine(), este método básicamente realiza el graficado de la línea a través de una serie de puntos que va de acuerdo a los puntos p1 y p2 de una recta y con respecto a la inclinación de ella misma. En drawLine() se toman en cuenta varias condiciones que podrían afectar la graficación de los objetos, por ejemplo, si tenemos que x1 es igual a x2 la pendiente m resultante que tendríamos seria cero por lo tanto no podríamos usar la ecuación de la recta con respecto a la pendiente, sin embargo en este caso lo que hice fue que debido a que es una línea recta paralela al eje y podemos mandar a pintar puntos sobre x1 desde y1 hasta y2.

Otra condición que tuve que considerar fue que al estar manipulando las figuras con las transformaciones, estás hacen que al mandar a pintar una línea no sea de una forma tradicional, es decir, que las pintemos de un punto p1 que esta en una posición inferior a p2 en el plano cartesiano, las transformaciones también pueden hacer que una línea no se dibuje de forma ascendente o de izquierda a derecha como es lo común, sino que las mandan a pintar descendentemente o de derecha a izquierda. Todos esas circunstancias fueron tomadas en cuenta y para resolver ese problema lo que decidi hacer esa comparar las x’s y las y’s en el método

drawLine(), para verificar de que manera podíamos manejar los puntos para poder usar la ecuación de la pendiente que esta usando.

En la clase Triangulo podemos ver que yo decidi formar un triángulo a partir de 3 puntos e inicializando el triangulo en el origen del plano. Triangulo a diferencia de punto puede inicializar valores recibiendo objetos de tipo Punto y no necesariamente una serie de enteros para construirse, al igual que el cuadrado que recibe 4 Puntos. Para dibujar el triángulo y cuadrado fue más sencillo pues ya tenía las estructuras base, en los métodos que sirven para dibujar a cada uno, lo que se hace es llamar a la función setValues() y drawLIine() de Linea, se le envían los puntos propios de cuadrado y triangulo y la función une los puntos mediante líneas.

Para crear la clase Círculo, los únicos datos que necesitamos es el radio y el punto central, el constructor de Círculo crea un círculo con centro en el origen y con radio igual a 10. En el método para dibujar el círculo lo que hice fue espejear puntos. Tomando en cuenta el radio. Dividí el circulo en 8 para que se pudiera dibujar con mayor precisión.

Pasando al tema de las transformaciones también cree una clase llamada Oper en la cual se establecen métodos para cada una de las operaciones. Fije tres matrices en la clase Oper la cual uso en cada uno de los métodos para realizar la multiplicación escalar de cada una de las transformaciones, M es la matriz extendida que se usa en cada una de las operaciones, P almacena los valores iniciales del punto que se va a trabajar y P1 guarda los resultado de x y y.

Brevemente las operaciones se definen así:

Punto trasl(Punto,int, int);	Traslada un punto, recibe como parámetros el punto que se va a manipular y dos enteros que son los vectores de traslación. A las coordenadas del punto original se le suman los vectores en x y y respectivamente. Devuelve el un punto con las coordenadas modificadas.
Punto esca(Punto,int, int);	Escala un punto, recibe como parámetros el punto que se va a manipular y dos enteros que son los vectores de escalamiento. Multiplica las coordenadas del punto original por los vectores de escalamiento. Devuelve el un punto con las coordenadas modificadas.

Punto rot(Punto,double);	Rota un punto, el parámetro de tipo doublé es el ángulo (en grados) de rotación que se aplicara, dentro de esta función tuve que cambiar el valor del ángulo a radianes ya que el programa sólo funciona con radianes. Devuelve otro punto en la posición que se asigno después de la rotación. Este método considera que la figura se encuentra en el origen.
Punto rot(Punto,Punto,double);	A diferencia del anterior, este método recibe dos puntos uno es el pivote y el otro el punto que se modificara.
Punto reflX(Punto); Punto reflY(Punto); Punto reflXY(Punto);	Reflejan a al punto o a la figura de acuerdo al eje de simetría, ya sea el eje x, el y o ambos.
Punto defX(Punto,float); Punto defY(Punto,float);	Deforman a las figuras con respecto al eje que se necesita.

Para usar cada una de las transformaciones, se uso como base la estructura del Punto, primero se modifica cada uno de los puntos de las figuras y después se unen sus puntos, se mandan a dibujar.
Cada una de las figuras tiene cada una de las operaciones.

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

REPORTE TÉCNICO:

PRIMER EXAMEN PARCIAL

PRESENTA:

LUIS ALBERTO SILVA ESCAMILLA

MATERIA:

GRAFICACIÓN

PRIMAVERA 2015

Introducción

Este examen con fecha de entrega del viernes 20 de Febrero busca que converjan todos los conceptos aprendidos hasta la fecha, del curso de

Graficación. Sin embargo, básicamente estamos manejando la practica 2 (del automóvil) con ciertas modificaciones, estas últimas hechas en base a los temas de escalamiento, rotación y desplazamiento.

El examen consiste en dibujar un escenario 2D utilizando clases en C++. Sin embargo, antes se debe de implementar una librería en C++, que me permita graficar cualquier figura que desee, con la única restricción que solo se permite utilizar la primitiva de Open GL, GL_POINTS.

Es decir, implementar las clases Punto, Línea, Triangulo, Cuadrilátero y Circulo únicamente con la primitiva antes mencionada.

Se está utilizando el paradigma orientado a objetos, ya que nos permite manejar las clases como tales y a través de ellas, realizar el trazado de objetos y consecuentemente el escenario.

Finalmente, implementar la matriz de proyección para manejar las funciones de escalamiento, rotación y traslación.

Práctica de laboratorio

Se describe a continuación algunas metodologías empleadas para el desarrollo de algunos métodos:

Circulo:

Como en el caso de la práctica número 2, se buscó optimizar el trazado de esta figura mediante la técnica de programación dinámica, es decir, la utilización de los resultados derivados de un cálculo anterior; para posteriormente deducir los siguientes.

En este caso, la forma en que se implemento fue mediante el trazado del círculo por cuadrantes:

1. Solicitamos como parámetros de entrada las coordenadas “h”, “k” del centro de la circunferencia y el radio r de la misma.

2. Dentro de un ciclo vamos calculando y, para ello empleamos una derivación (despeje) de la fórmula de la circunferencia, entonces calculamos “y” con la siguiente ecuación:

$$y = \sqrt{r^2 - ((x-h)^2) + k}$$

Donde Y son los puntos que se intersectan con X, es decir los puntos que van trazando la circunferencia.

3. Una vez obtenidos estos puntos Y, se almacenan en un arreglo. Ya tenemos trazado el primer cuadrante.
4. Para trazar los posteriores cuadrantes, se utiliza el arreglo anterior y finalmente se van trazando los demás cuadrantes.

Línea:

Se necesitan dos puntos para trazarla, se utilizó la clase Punto diseñada en clase, para la implementación de esta segunda.

Las clases Punto y Linea fueron fundamentales para el desarrollo de las clases siguientes, ya que sirvieron como base para la implementación de las otras, por ejemplo, la clase Cuadrilatero, manda a llamar a cuatro objetos Linea que a su vez cuando se unen forman al cuadrilátero. De la misma forma se utiliza Linea para trazar los objetos de tipo Triángulo.

OperacionesMatriciales:

Esta fue la clase más complicada de implementar, y de hecho desafortunadamente, por un error básico de diseño no se pudo llevar a cabo satisfactoriamente y no del todo ya que, aunque se rediseño bien, con el tiempo encima ya no se pudo implementar completamente. Sin embargo, es importante destacar el diseño.

Esta clase es un ejemplo de herencia, es decir, se implementó como como clase padre, para ser heredada a todas las demás clases Linea, Triángulo, Cuadrilatero y Poligono. Esto se pensó así, porque estas clases comparten los

métodos de traslación(), escalamiento() y rotación(); por lo cual es innecesario reescribirlos dentro de cada clase por separado.

Su constructor inicializa las matrices de Escalamiento, Traslación y Rotación. Después, cuando una clase de las llamadas “Figuras”, llama a alguna de estas funciones heredadas, estas utilizan la matriz respectiva a la operación que se quiera realizar (traslación, rotación o escalamiento) y la multiplica, almacenando el resultado en la matriz de proyección. Finalmente, cuando se decida trazar la figura y se den las coordenadas finales, estas se multiplican por la matriz de proyección , se almacenan en un arreglo el resultado de las multiplicaciones para finalmente graficar esos nuevos puntos, correspondientes las operaciones deseadas.

Algo fundamental fue el uso de las matrices de Proyección y las operaciones entre matrices, ya que simplifican bastante el número de estas, que de otra manera serían repetitivas y no serviría de mucho demasiado cálculo.

Conceptos desarrollados

- ✓ Matriz de proyección.
- ✓ Herencia.
- ✓ Operaciones matriciales.
- ✓ Matriz identidad.
- ✓ Clases repetidas.
- ✓ Pendiente de la recta.
- ✓ Ecuación de la circunferencia.
- ✓ Programación dinámica.
- ✓ Optimización.

Experimentos

Derivado de las necesidades que se presentaron a lo largo de este pequeño proyecto, se tuvieron que ir replanteando algunos conceptos y, sobre la marcha, ir implementando principalmente funciones, que se acoplaran y permitieran solucionar el trazado de algunas figuras muy particulares.

La clase Círculo consta de varios métodos, algunos bastantes particulares que no estaban contemplados al inicio, tales como:

- halfCircle1(): el cual traza solo el cuadrante I y II de la circunferencia. Este método se utilizó para trazar parte de los pies y las cabezas de algunos personajes.
- halfCircelDown(): el cual traza los cuadrantes III y IV.
- halfCircleRight(): dibuja los cuadrantes I y IV.
- halfCircleLeft(): dibuja los cuadrantes II y III.
- quarterFirst(): dibuja únicamente el primer cuadrante.

Durante la fase de diseño de la clase Línea, se había pensado en cuatro casos posibles:

1. Cuando la línea fuera inclinada hacia la izquierda: /
2. Cuando la línea sea inclinada a la derecha: \
3. Cuando la línea fuese vertical: l
4. Cuando la línea es horizontal:

Sin embargo luego se consideró, y se pudo reducir a dos casos, utilizando la fórmula de la ecuación de la recta:

$$m = \frac{y_1 - y_2}{x_1 - x_2}$$

De esta manera solo se tienen dos casos:

1. Cuando la pendiente no existe, es decir cuando no se puede calcular m.
2. El caso general, que es cuando sí se puede calcular m.

Sin embargo algo curioso y que provoco demasiados problemas durante el proyecto, fue que a la hora de trazar las líneas dando los parámetros, estos tenían que ir en un orden, de otro modo, no se trazaban. Así por ejemplo para trazar

líneas así: / se tiene que empezar poniendo los valores de abajo x1, y1 y después los de arriba, x2, y2; para las líneas verticales se da el mismo caso.

Para el caso de: \ se empieza agregando de arriba hacia abajo. La línea horizontal no se ve afectada de alguna manera.

Por otro lado y como ya se adelantaba, en cuando a la implementación de la clase OperacionesMatriciales, debido primero a un manejo erróneo de los conceptos iniciales y posteriormente a la falta de tiempo para corregir correctamente el error, no se pudo llevar a cabo su total implementación. Algunos de los errores que surgieron y se intentaron solucionar sin éxito fueron:

- Liberación de memoria, que causo que la última imagen se permaneciera.
- Manejo repetitivo de estructuras innecesarias.
- Cuando se intentó eliminar esta clase, el compilador marco error de Linkeo, lo que provoco que ya no funcionara todo el proyecto, hasta que se volvió a añadir la clase al proyecto.

Finalmente, se utilizaron las directivas del prepocesador *#ifndef, #define* y *#endif,* para manejar las clases duplicadas que marcaba el compilador.

Conclusiones

Aunque no completado con éxito por parte del estudiante, este examen permito recordar el manejo de estructuras básicas como los son arreglos y matrices, las operaciones con ellas, y el recordar y aplicar los conceptos y prácticas adquiridas hasta lo que se lleva del curso. También se corrigieron errores y considero que quedaron más claros los conceptos, aunque ya no se pudo demostrar cómo se hubiese deseado.

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

NOMBRE: Erik Alfredo González Gutiérrez **CICLO:**

Primavera 2015

MATERIA: Graficación

Introduccion.

En el presente trabajo se pone en práctica la creación de algunas primitivas básicas utilizadas en lenguajes de programación como opengl, lenguajes diseñados para el graficado por computadora. En este trabajo se utilizara como única herramienta la primitiva de opengl que grafica puntos (glVertex) la cual será utilizada para graficar todas y cada una de las figuras en el graficado resultante, para ello es imprescindible crear nuestras propias librerías con sus operaciones y métodos ya definidos para cada clase, las cuales ya estarán listas solo para ser invocadas y obtener el graficado deseado.

El trabajo consta en crear librerías para dibujado de líneas, triángulos, cuadriláteros, y círculos cada librería constara de sus propios atributos y funciones de operadores que se encargan de manipular cada figura.

El entorno en el cual se llevara a cabo dicho proyecto será el entorno Dev-c++ utilizando la librería de OpenGl <GL/glut.h> .

El propósito de este proyecto es que el alumno sea capaz de trabajar en cualquier otro lenguaje de programación a partir de una única primitiva la cual será el graficado de puntos y que a partir de ella pueda trabajar sin ningún problema ya que estará preparado para ello.

Desarrollo.

Como ya se había mencionado anteriormente el trabajo consta de crear librerías que sean capaces de graficar líneas, triángulos, cuadriláteros y círculos. Entonces lo primero que hay que especificar es la estructura de cada clase que se podrá manejar en nuestro proyecto.

La clase línea tendrá como atributos cuatro variables enteras las cuales contendrán x1, y1, x2, y2. También tendrá su función setValues que se encarga de recibir los valores de los puntos. Así como también su función draw la cual se encarga de graficar cada punto obtenido mediante el algoritmo de dibujado de líneas.

La clase Triangulo tiene como atributo un vector de tamaño 6 el cual contendrá las coordenadas del triangulo.

Entre sus funciones la clase triangulo tendrá su función setValues() que recibirá 6 entero que corresponderán a las coordenadas del triangulo. Su función traslada() la cual recibe dos valores enteros que representan el vector de traslación. Su función escala() que recibe como parámetro un entero que será el factor de escalamiento. Su función rota() que recibe un entero el cual tomara su lugar como el grado en el cual queremos rotar la figura. La función reflexión() que recibe un carácter para indicar con respecto a que eje queremos la transformación de la figura, en este caso se debe ser muy cuidadoso ya que solo están definidas tres acciones la reflexión con respecto a X , la reflexión con respecto a Y y la reflexión con respecto al origen en cada caso se debe de recibir un carácter comprendido entre ‘X’, ’Y’ o bien ‘O’ este ultimo indica que su reflexión será con respecto al origen. La función deformación() recibe como parámetros un carácter que puede ser ‘X’ o ‘Y’ para indicar con respecto a que eje se quiere la deformación , así como también recibe un entero que será el factor de

deformación de la figura. por último se encuentra la función drawTri() que es la encargada de dibujar el triangulo definitivo que se verá en pantalla.

Para la clase Cuadrilátero su atributo será un vector de tamaño 8 el cual guardara las coordenadas del cuadrilátero.

Sus funciones serán análogas a las funciones mencionadas en la clase triangulo con excepción de la función setValues() que recibe como parámetro 8 valores enteros los cuales serán las coordenadas del cuadrilátero.

En la clase circulo se tendrá como atributos un vector de tamaño 2 que será el encargado de guardar las coordenadas del punto de origen del circulo y un entero r que será el radio del circulo.

Entre las funciones contenidas en la clase circulo están la función setValues() que recibe como parámetro tres entero los cuales representaran el X y Y del punto origen del circulo y el tercer parámetro será el tamaño del radio. Su función traslada es análoga a la función con el mismo nombre traslada de la clase triangulo previamente definida así que se omitirá en este apartado. La función escala() debido a que un circulo se dibuja por el largo del radio, en este caso solo recibe como parámetro el factor de escalamiento el cual es multiplicado por el radio del circulo previamente definido. Y por último la función drawCir() que se encarga del dibujado del circulo final por medio del algoritmo de bresenham.

Para el dibujado de líneas se tenía dos opciones el graficado por medio de la ecuación de la pendiente de la recta la cual es y=mx+b, donde m es la pendiente y b la intersección de la recta con el eje y. y la otra es utilizar el algoritmo de bresenham el cual es más eficiente a la hora de graficar la línea. En este caso como se está empezando a trabajar por primera vez se opto por el algoritmo de bresenham para estar seguros de que no se tuvieran problemas más adelante, por ejemplo a la hora de rotar una figura ya que al ser demasiado inclinada se podrían tener errores y no se podría

visualizar si el problema se encuentra con la función de rotación o con el dibujado de líneas.

En cada clase se llamara a la librería de dibujado de líneas, como ejemplo concreto se describe a continuación como se grafica un triangulo.

Como ya se había mencionado la clase triangulo tiene un vector llamado puntos de tamaño 6 que tendrá las coordenadas del triangulo como sigue, puntos[0]=x1, puntos[1]=y1, puntos[2]=x2, puntos[3]=y2, puntos[4]=x3, puntos[5]=y3. asi cuando se dibuje el triangulo se crea tres objetos de tipo línea como sigue

izq.setValues(puntos[0],puntos[1],puntos[2],puntos[3]);

der.setValues(puntos[2],puntos[3],puntos[4],puntos[5]);

bas.setValues(puntos[4],puntos[5],puntos[0],puntos[1]);

para después solo mandar a llamar a la función de dibujado de líneas y se dibujan las tres líneas declaras como izq,der y bas.

Para la función de traslación lo primero que se hace es definir un la matriz de traslación que es la siguiente

```
mat[0][0]=1;
mat[0][1]=0;
mat[0][2]=a;
mat[1][0]=0;
mat[1][1]=1;
mat[1][2]=b;
mat[2][0]=0;
mat[2][1]=0;
mat[2][2]=1;
```

a y b cumplen como el vector de traslación .una vez obtenida esta matriz se multiplica cada punto por dicha matriz y de esta forma se van obteniendo las coordenadas finales de la figura ya trasladada.

Para la función de escalamiento lo primero que hay que hacer es definir el punto pivote el cual es trasladado al origen restándole las coordenadas del mismo punto a todos los puntos de la figura , por

ejemplo si se toma de pivote el punto 1 entonces el vector de traslación estaría compuesto por <X1,Y1> el cual sería restado a todos los puntos, para después multiplicar cada punto por la matriz de escalamiento la cual se define de la siguiente manera:

```
mat[0][0]=r;
mat[0][1]=0;
mat[0][2]=0;
mat[1][0]=0;
mat[1][1]=r;
mat[1][2]=0;
mat[2][0]=0;
mat[2][1]=0;
mat[2][2]=1;
```

la variable r es recibida como parámetro y tiene la función de factor de escalamiento, una vez multiplicado cada punto se le suma las coordenadas <X1,Y1> y el resultado de eso serian los puntos finales de la figura.

En la función de rotación también se define un punto pivote el cual se traslada al origen restando las coordenadas del punto pivote a cada punto de la figura, para después multiplicar cada uno de los puntos de la figura por la matriz de rotación la cual se inicializa de la siguiente forma:

```
mat[0][0]=cos(r);
mat[0][1]=-sin(r);
mat[0][2]=0;
mat[1][0]=sin(r);
mat[1][1]=cos(r);
mat[1][2]=0;
mat[2][0]=0;
mat[2][1]=0;
mat[2][2]=1;
```

r es recibido como parámetro es una variable entera e indica el numero de grados que se desea girar la figura.

La función de reflexión lo primero que hace es decidir con respecto a que eje se quiere la reflexión esto se puede gracias a que la función recibe como parámetro un carácter el cual dirá como definir la matriz de reflexión, en esta función hay tres casos :

1.- si el carácter recibido es 'X' entonces la matriz de reflexión es definida de la siguiente manera:

```
mat[0][0]=1;
mat[0][1]=0;
mat[0][2]=0;
mat[1][0]=0;
mat[1][1]=-1;
mat[1][2]=0;
mat[2][0]=0;
mat[2][1]=0;
mat[2][2]=1;
```

2.- si el carácter recibido es 'Y' entonces la matriz de reflexión es definida de la siguiente manera:

```
mat[0][0]=-1;
mat[0][1]=0;
mat[0][2]=0;
mat[1][0]=0;
mat[1][1]=1;
mat[1][2]=0;
mat[2][0]=0;
mat[2][1]=0;
mat[2][2]=1;
```

```
mat[0][2]=0;
mat[1][0]=0;
```

3.- si el carácter recibido es ‘O’ entonces la matriz de reflexión es definida de la siguiente manera:

```
mat[0][0]=-1;
mat[0][1]=0;
```

```
mat[0][2]=0;
mat[1][0]=0;
```

```
mat[1][1]=-1;
mat[1][2]=0;
mat[2][0]=0;
mat[2][1]=0;
mat[2][2]=1;
```

una vez definida la matriz de reflexión lo que se hace es definir el punto pivote trasladarlo al origen restando el vector de traslación

,como ejemplo si el pivote fuera el punto 1 entonces el vector de traslación seria <X1,Y1> el cual se restaría a todos los puntos de la figura , después se procede a multiplicar cada punto por la matriz de reflexión y después sumar el vector de traslación a cada punto obteniendo así los puntos finales de la figura.

La función de deformación recibe como parámetros un carácter el cual dirá con respecto a que eje se quiere la deformación de la figura y un entero que será el factor de deformación, lo primero que se hace es revisar el carácter recibido como parámetro si el carácter es 'X' entonces indica que se requiere una deformación con respecto al eje X y por tanto la matriz de deformación queda de la siguiente manera:

```
mat[0][0]=1;
mat[0][1]=sh;
mat[0][2]=0;
mat[1][0]=0;
mat[1][1]=1;
mat[1][2]=0;
mat[2][0]=0;
mat[2][1]=0;
mat[2][2]=1;
mat[0][0]=1;
mat[0][1]=0;
```

de lo contrario si el carácter recibido es ‘Y’ entonces nos indica que se requiere una deformación con respecto al eje Y y la matriz de deformación queda de la siguiente forma:

```
mat[0][0]=1;
mat[0][1]=0;
```

```
mat[0][2]=0;
mat[1][0]=sh;
mat[1][1]=1;
mat[1][2]=0;
mat[2][0]=0;
mat[2][1]=0;
mat[2][2]=1;
```

en las dos matrices el sh representa el factor de deformación , una vez obtenida la matriz de deformación se define el punto pivote se traslada al origen restando el vector de traslación ,como ejemplo si el pivote fuera el punto 1 entonces el vector de traslación seria

<X1,Y1> el cual se restaría a todos los puntos de la figura , después se procede a multiplicar cada punto por la matriz de reflexión y después sumar el vector de traslación a cada punto obteniendo así los puntos finales de la figura.

Las funciones de la clase Cuadrilátero son similares a las de la clase triangulo solo que un objeto cuadrilátero tiene un punto de mas. Aun así las operaciones son las mismas solo que con un punto más a operar.

Para la clase Circulo la función setValues recibe como parámetro tres entero los dos primeros representan las coordenadas X,Y del centro del circulo y la tercer variable es el radio que tendrá el circulo y tiene las siguientes funciones definidas.

La función traslada recibe dos enteros y su forma de operar es la misma que la función traslada de la clase triangulo.

En su función escala recibe un entero que actúa como factor de escalamiento solo que en este caso lo único que se hace es

multiplicar el radio por el factor de escalamiento y de esta forma a la hora de graficar el circulo se obtendrá el radio ya operado y a escala.

Para la función de dibujado en este proyecto se opto por utilizar el algoritmo de bresenham para dibujado de círculos.

Conclusión.

En la práctica de estas clases se obtiene un conocimiento acerca de primitivas para graficar prácticamente desde cero , esto quiere decir que se podría migrar a cualquier otro lenguaje de programación y poder graficar con los algoritmos usados en este proyecto, también se da una mejor comprensión de las cosas que hace internamente las librerías de OpenGl para así tener un mejor manejo de dichas funciones y otra más complejas basadas en las primitivas más básicas.

Como previamente se declararon estas clases y funciones no queda más que usarlas a nuestro gusto para crear figuras en dimensión 2d , en nuestro caso se debe de realizar un escenario con figuras así que se trabajo con ellas y se obtuvo el siguiente resultado.

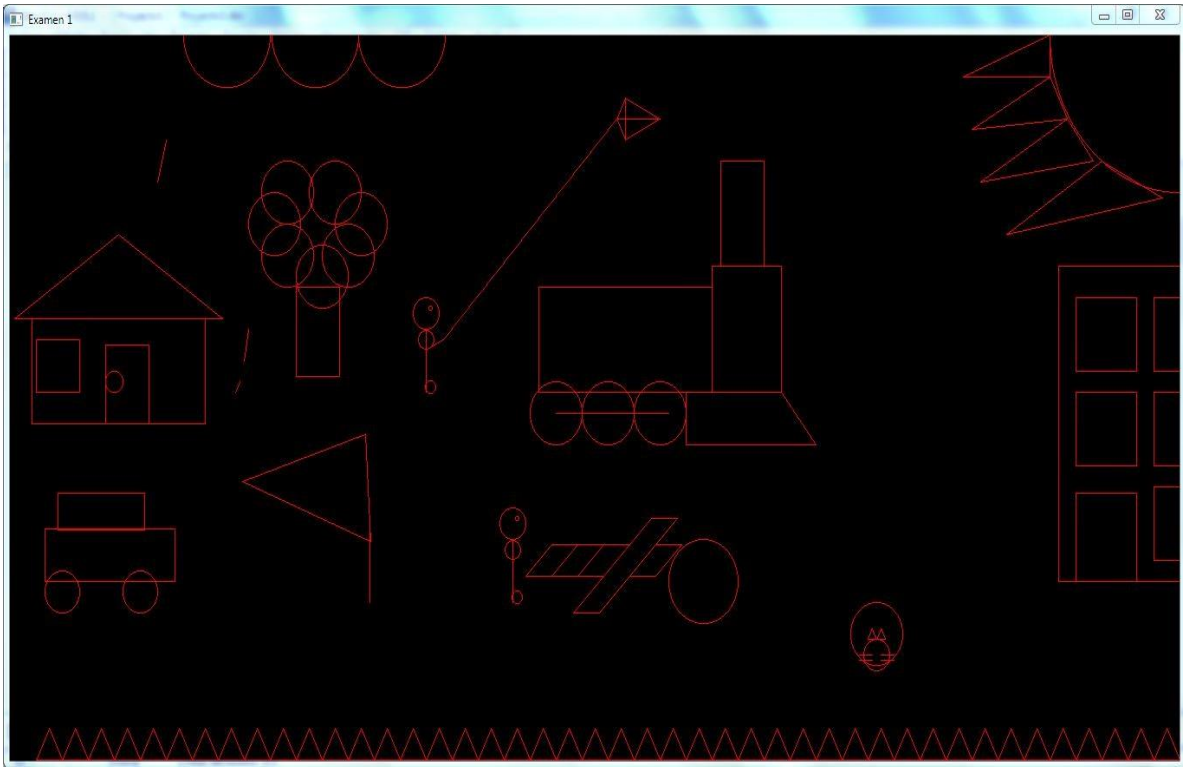


Figura1. Resultado final del proyecto.

BENEMRITA UNIVERSIDAD AUTONOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LAS COMPUTACION

MC. IVAN OLMOS PINEDA

MATERIA: GRAFICACION ALUMNO:

SANDOVAL SALAZ CONSUELO

MATRICULA: 201023593

PERIODO: PRIMAVERA 2015

Introducción

En este trabajo se hizo uso de funciones como glPushMatrix, glPopMatrix, glLoadIdentity, funciones de operadores como translación, rotación y escalamiento. Además de hacer uso de las primitivas de Opengl con la finalidad de elaborar un escenario en 2D.

Desarrollo

Declaramos las librerías de Opengl y variables a utilizar, posteriormente procedemos declarar las funciones:

Usaremos una matriz de identidad se carga de la siguiente manera glLoadIdentity(), para llevar acabo las transformaciones a realizar (rotación, escalamiento, traslación), usaremos

glPushMatrix(), glPopMatrix(), declaramos un tipo de color con la función glColor3f(), si lo deseamos declaramos un ancho de línea con glLineWidth().
Utilizando la primitiva glBegin(GL_QUADS) se pintó el suelo, pasto y demás formas de nuestro escenario.

En dicho programa se encuentran diversas figuras geométricas y para realizar una toma de agua, nubes, el sol, las llantas de la patineta y el personaje se usó la función glutSolidSphere(), con glShadeModel y el parámetro GL_FLAT indicamos a Opengl que queremos darle un color solido a nuestra figura.

En glutSolidSphere el primer parámetro es el radio, el segundo parámetro es el número de subdivisiones alrededor del eje Z, y el tercer parámetro es el número de subdivisiones a lo largo del eje Z.

Para trasladar la figura deseada usamos la función glTranslatef, donde los parámetros son los ejes x, y, z.

El escalamiento se realiza con la función glScalef, los parámetros son los ejes x, y, z.

En el escenario una patineta con el personaje principal y las nubes se mueven hasta donde aparentemente salen de la escena y regresar, para poder logar dicho movimiento (traslación) se dibuja la figura y se utiliza una variable anteriormente declarada la cual llame avanza y avanzaN las cuales se inicializan en 0.0f.

Usamos la función glTranslatef(-avanzaN,-0.0,0.0) los parámetros son la declaración de la variable, y los ejes.

En un condición if se declara digamos velocidad en la que van avanzando las cosas.

El sol rota con la función glRotatef() dentro de dicha función va un ángulo donde se especifica la velocidad en la cual va a avanzar.

La función reshape nos sirve para redimensionar la ventana y su contenido. La función Idle que en mi caso lo único que hace es dibujar lo mismo que se encuentra en la pantalla.

Dentro de la función reshape se encuentra la matriz ModelView donde se define como se transforman los objetos (es decir, traslación, rotación y escalamiento) en el marco de coordenadas.

La función keyboard que nos sirve para la interacción de nuestro programa con el teclado.

La función display se encarga de dibujar todo en pantalla.

Las funciones antes mencionadas son las proporcionadas por el profesor de la clase con una variación en la función display ya que es donde definimos las figuras a imprimir en pantalla.

Por último la función main que es el principio de nuestro código ya que es la que se encarga de llamar a todas las demás funciones que previamente se mencionaron.

Conclusión

El trabajo realizado nos ayudara a mejorar nuestras habilidades en el manejo de las funciones de Opengl, comprender de una manera más eficiente el funcionamiento de las matrices que utiliza el programa.

Además que el realizar escenas en 2D nos preparara para elaborar trabajos más complejos y ser programados en 3D.

El producto final de dicho programa es el siguiente mostrando las respectivas rotaciones escalamientos y traslaciones.

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD EN CIENCIAS DE LA COMPUTACIÓN

Examen Parcial 1

Integrantes:

García González Mauricio

Graficación

Introducción

Para la realización de la práctica correspondiente al primer parcial se propuso crear un escenario grafico mediante el cual se utilizaran todas las funciones y algoritmos vistos en clase para el desarrollo en 2D.

Desarrollo

Debido a las exigencias del proyecto se optó por dividir en varias clases los diferentes ámbitos de los cuales se encargaría el proyecto en el reporte solo colocaremos los correspondientes a la definición de las funciones para no extender tanto el texto

Clase Punto

```
#include <GL/glut.h>
#include <stdlib.h>

/*
EJEMPLO DE LA DECLARACION DE CLASES E
CURSO GRAFICACION POR COMPUTADORA, BU
IVAN OLMOS PINEDA
*/

class Punto{
    private:
        int coord[2];
    public:
        Punto();
        ~Punto();
        void setValues(int,int);
        int getX();
        int getY();
        int* getValues();
        void draw();
```

Sin mucho que agregar se puede observar que esta clase solo define al objeto punto con sus coordenadas y sus funciones correspondientes.

Clase Línea

```
/*
EJEMPLO DE LA DECLARACION DE CLASES EN C++
CURSO GRAFICACION POR COMPUTADORA, BUAP
IVAN OLMOS PINEDA
*/

#include "Punto.h"

class Linea{
    private:
        Punto p1, p2;
    public:
        Linea();
        ~Linea();
        void setValues(int, int, int, int);
        void setValues(Punto, Punto);
        void draw();
};
```

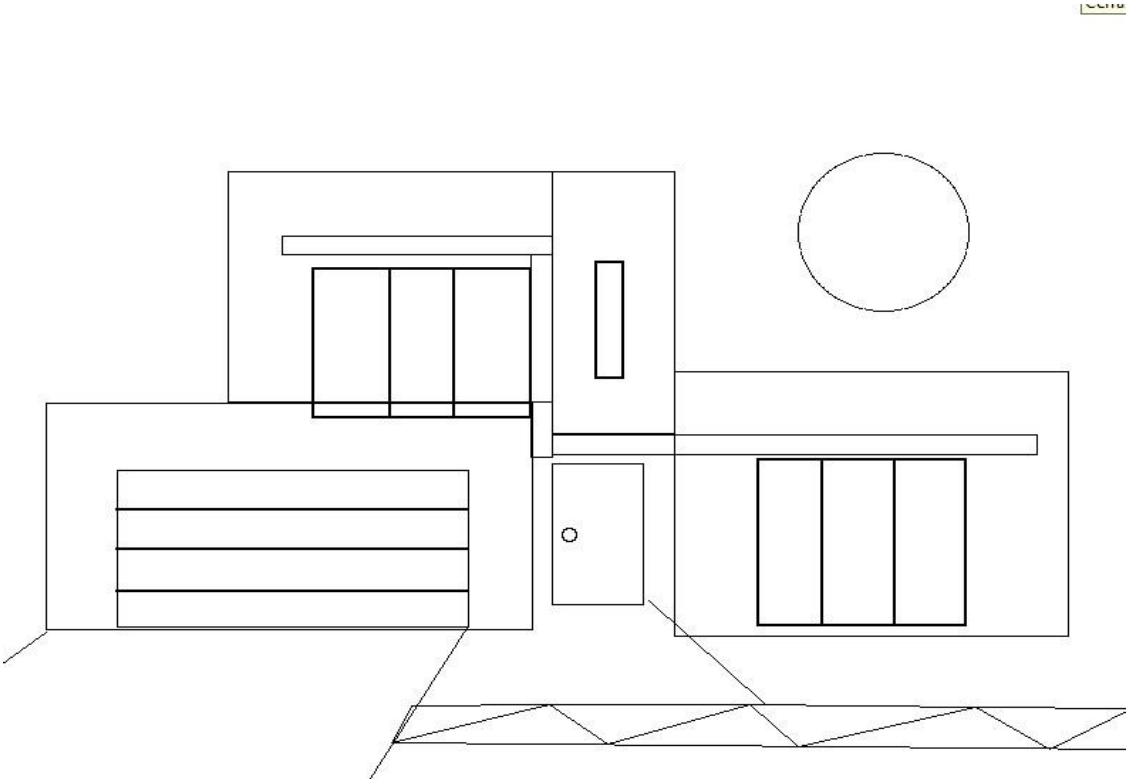
Clase Figuras

```
#include "Linea.h"
class Figuras
{
private:
    Punto p1,p2,p3;
    int puntos[6];

public:
    Figuras();
    ~Figuras();
    double radio;
    //void setRadio(double);
    //double getRadio();
    void TsetValues(int, int, int, int, int, int);
    void TsetValues(Punto, Punto,Punto, Punto);
    void Tdraw();
    void CsetValues(int, int,int,int);
    void CsetValues(Punto, Punto);
    void Cdraw();
    void CrsetValues( double,int,int);
    void CrsetValues(double, Punto);
    void Crdraw();
    int* FigetValues();
    };
```

Al igual que la primera clase aquí se definen cada uno de las funciones que va a tener dicha clase ya sea la de línea o de graficar las figura.

Función pintada de Escenario

Después de hacer las pruebas correspondientes a los dos anteriores algoritmos simplemente de procedió al llamar a las respectivas funciones para la creación de una determinada figura cuyo resultado fue el siguiente:



Observaciones

Debo mencionar que me vi en un apuro a la hora de querer implementar el algoritmo de las trasformaciones en 2D ya se traslación, rotación y escalamiento por lo que no se ve implementado en mi escenario.

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACION

ING. EN CIENCIAS DE LA COMPUTACION

GRAFICACIÓN

EXAMEN PARCIAL 2

ALUMNO: TITLA TLATELPA JOSE
DE JESÚS

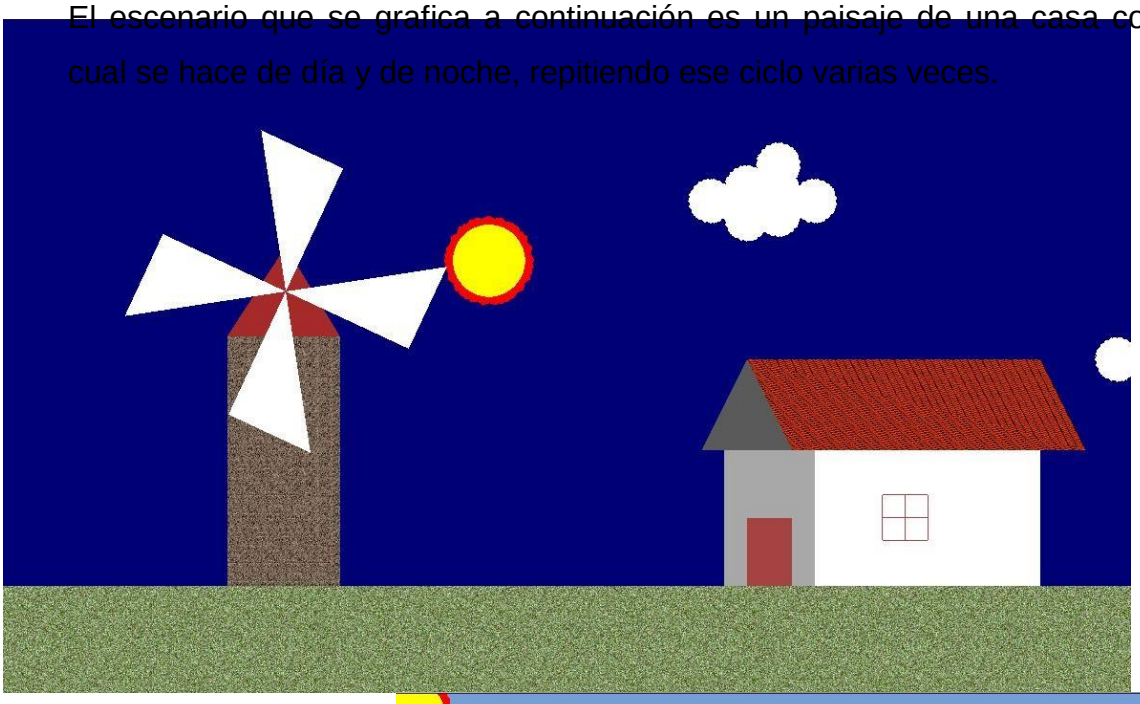
PROFESOR: OLMOS PINEDA IVAN

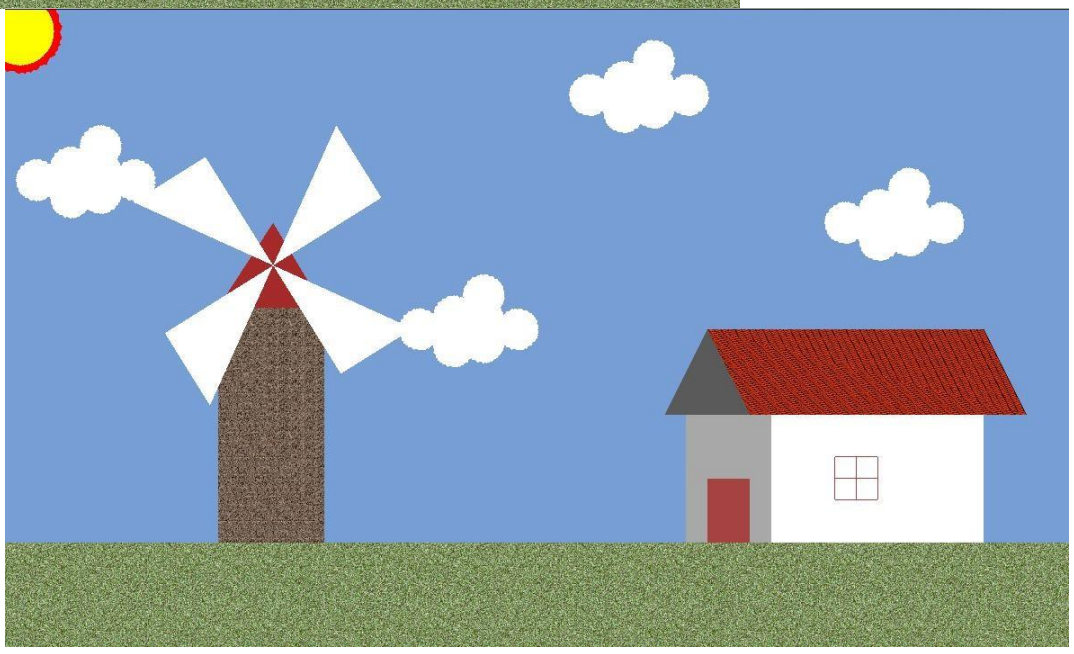
INTRODUCCIÓN

En este documento se explica cuál fue el proceso para graficar un escenario 2D usando las primitivas de OpenGL y aplicando la carga de imágenes para texturizar.

Se realizaron diversos experimentos para lograr los efectos deseados en este escenario y se usó un archivo de cabecera que contiene las funciones para graficar el escenario para que solo sean llamadas en el main.

El escenario que se grafica a continuación es un paisaje de una casa con un molino en el cual se hace de día y de noche, repitiendo ese ciclo varias veces.





DESARROLLO

Para facilitar la creación del escenario y hacer una programación mas limpia se optó por hacer la declaración de todas las un funciones en un archivo de cabecera .h y solo hacer el llamado en el main.

Al hacer esto solo se llaman la funciones en dibuja(), que es la funcion que realizara el graficado del escenario.

En las cabeceras solo se hace el llamado de draw.h que es la que contiene las funciones para dibujar el escenario.

```
#include<GL/glut.h>
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include"draw.h"
```

En draw.h se encuentran las siguientes funciones, a continuación se describe la funcion de cada una.

Cargar imágenes para hacer las texturas: int cargarTGA(char *nombre, textura *imagen)

Esta funcion es la que hace todo el proceso de conversión de texturas para poder colocarlas a un objeto, para apoyarse en dicho proceso hace uso de un struct al que llamamos **textura.**

```
typedef struct {
 GLubyte  *dibujo;       // Un puntero a los datos de la imagen
 GLuint bpp;  // bpp significa bits per pixel (bits por punto) es la calidad en palabras sencillas
 GLuint largo;                // Largo de la textura
 GLuint ancho;                // Ancho de la textura
```

```
 GLuint ID;              // ID de la textura, es como su nombre para opengl
}textura;
```

Simular el cielo: void cielo(GLfloat red, GLfloat green,GLfloat blue)

Esta funcion recibe como parámetros tres flotantes, estos flotantes se encargan de aclarar o oscurecer el color del cielo. Todo esto se controla solamente en el color de la figura.

```
glColor3f(0.74902f+red,0.847059f+green,0.847059f+blue);          //Aclarar o oscurecer.
```

Rotación del sol: void sol(GLfloat anguloma)

Esta funcion solo recibe un ángulo de rotación como parámetro, es el que se encarga del que sol avance y junto a la funcion cielo se logra el efecto día/noche de un mundo real.

Crear pasto en el suelo: void suelo()

En esta funcion no se pasa ningún parámetro, pero se cargan texturas para hacer que en el suelo se vea el pasto, un fragmento del código de las texturas es el siguiente.

```
glColor3f(1.0f,1.0f,1.0f);          //Se entinta color blanco para no alterar la textura
glEnable(GL_TEXTURE_2D); //Se habilita el uso de texturas
glBindTexture(GL_TEXTURE_2D,texturas[0].ID); //Genera texturas
    …
glDisable(GL_TEXTURE_2D);
```

Creacion de un molino: void regileteizq(GLfloat angulo)

Esta funcion al igual que las otras con rotaciones/translaciones recibe como parámetro el ángulo, pero en esta funcion, el ángulo se usa para hacer girar las aspas del molino, en el molino también se usan texturas para lograr el efecto de piedras.

Dibujado de la casa: void casa()

En esta funcion se realiza el graficado de una casa, tratando de hacer un efecto 3D sin ser

3D, esto se logra con los colores que se usan, oscureciendo los elementos mas alejados u ocultos y aclarando los mas visibles.

Nube en el cielo: void nube()

Para lograr que se dibujara una nube se crea esta funcion, la cual contiene varios círculos de color blanco sobrepuestos para simular una nube en el cielo, esta funcion solo crea una nube.

Nubes en movimiento: void cielosky()

Esta funcion no recibe parámetros explícitamente, pero usa una variable global para hacer la translación del objeto, las nubes se mueven linealmente en forma horizontal. Esta funcion

hace el llamado de nube() varias veces y con una translación se van colocando en diferentes lugares del cielo.

```
void cielosky(){ glPushMatrix();
   glTranslatef(0+angulon,110,0);
   nube();
   glPopMatrix(); glPushMatrix();
   glTranslatef(90+angulon,75,0);
   nube();
   glPopMatrix();
        …
}
```

Finalmente se llaman todas las funciones en dibuja para hacer que todos los elementos del escenario se vean juntos.

```
void dibuja(){
 glClear(GL_COLOR_BUFFER_BIT);
   cielo(red,green,blue);    //Cielo, se pasan los colores para aclarar/oscurecer
   sol(anguloma);    //Sol, se pasa el angulo para rotar
   suelo();     //Se llama a la funcion suelo
   cielosky(); //Cielo con nubes en el
   if(red<-17 || blue<-17 || green<-17){    //Saber si es de noche
      flag=1;
      cielostars();
   }

   if(red>0.04902 || blue>0.147059 || green>0.147059){       //saber si es de dia
      flag=0;
   }
   regileteizq(angulo);       //Se llama al molino
```

```
    casa();      //Se llama a la casa
        glFlush();
        if(flag==0){    //saber si es de dia
        anguloma-=.025;     red-=0.005;
    green-=0.004;    blue-=0.0002;
    angulo+=7;    angulon+=.08;}
    if(flag==1){          //Saber si es de noche
        anguloma+=.025;    red+=0.005;
    green+=0.004;    blue+=0.0002;
    angulo-=7;    angulon-=.08;}
}
```

EXPERIMENTOS

En esta ocasión se realizaron experimentos con el llamado de las funciones y con el control del día y la noche.

Posición de los objetos.

En cuanto al llamado de las funcione se concluyó que según el orden con el que se fueran llamando es el orden como se van a ir encimando, por eso es importante pensar en que va a ir al fondo, que va a ir al frente o en qué lugar van a estar colocados, haciendo esto se puede lograr que el escenario funcione de forma más realista.

Control del día y la noche.

Para controlar el color del cielo y el movimiento de los objetos según el día y la noche se hizo uso de condicionales y banderas, a continuación se explica la lógica que se siguió.

En esta funcion se comprueba que el color del cielo ya sea negro o más oscuro, si ese es el caso, la bandera se prende.

```
if(red<-17 || blue<-17 || green<-17){

    flag=1;
    cielostars();
}
```

Si la bandera está encendida, es decir, es de noche, entonces todo se hace en el orden contrario para hacer la transición noche/día.

```
if(flag==1){
   anguloma+=.025;
   red+=0.005;
   green+=0.004;
   blue+=0.0002;
   angulo-=7;
   angulon-=.08;
}
```

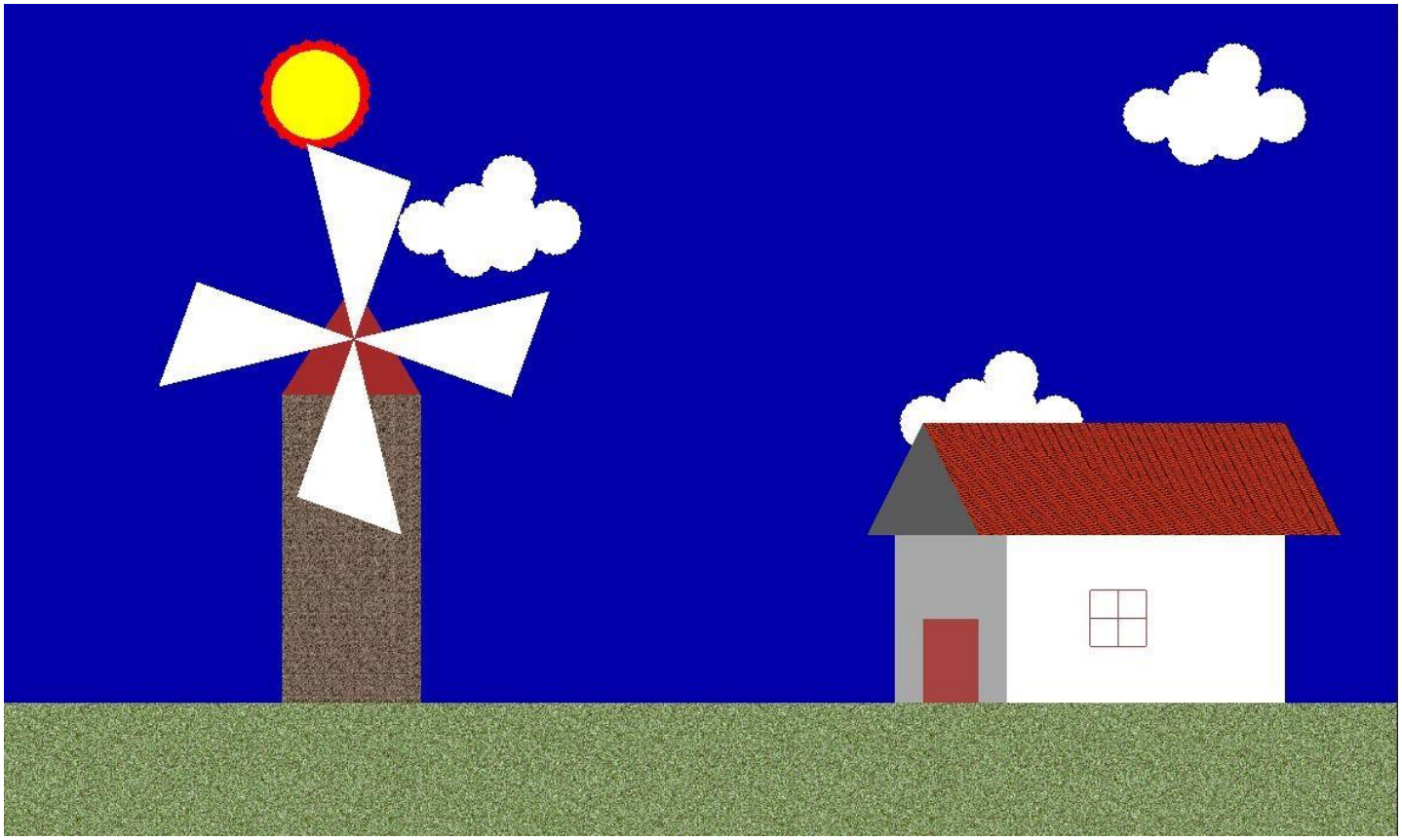
Si el color del cielo es del azul claro que se estableció al inicio, entonces se apaga la bandera y eso indica que es de día.

```
if(red>0.04902 || blue>0.147059 || green>0.147059){
    flag=0;
}
```

Si la bandera está apagada, indicando que es de día, entonces las rotaciones y actualizaciones se hacen en orden para hacer que se haga de noche.

```
if(flag==0){ anguloma-=.025;
    red-=0.005;
    green-=0.004; blue-
    =0.0002; angulo+=7;
    angulon+=.08;
}
```

Estos ciclos se van a estar realizando una y otra vez para que el ciclo día/noche no pare.




FCC
BUAP

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Ing. en Tecnologías de la Información

Asignatura:

Graficación *CCOM 259 - 101*

Examen:

Dibujar con puntos

Alumno:

Jesús García Licona 201212661

Índice

Objetivo

-Desarrollar que utilice la transición, rotación y escalamiento.

-Solo utilizar POINTS.

-Retomar los conocimientos previos de cómo obtener la matriz.

-Retomar conocimientos de Programación, Estructura de datos.

-Entender el funcionamiento de la Graficacion.

-Repasar los diferentes tipos de almacenamiento en la informática.

Estrategia de desarrollo

1- Se utilizara formula de rotación, escalamiento y traslación.

2- Se sacara el algoritmo para desarrollar una línea y el circulo con puntos

3-Se dibujara solo utilizando los algoritmos anteriormente mencionados.

Desarrollo

```
//
//  main.cpp
//  Funciones
//
//  Created by Jesus  Garcia Licona on 03/02/15.
//  Copyright (c) 2015 Jesus  Garcia Licona. All rights reserved.
//

#include <iostream>
#include <openGL/gl.h> // Xcode  syntax for  libraries, on my Mac

#include <GLUT/glut.h>

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#define PI 3.141592654

float  r1,  r2,  r3,  r4;
float  transx, transy, transx1, transy1;
float  rotx,  roty,rotx1,roty1;

void  init(void)
{
```

```
    //Establece el color de la ventana de visualizacion
    //Los tres primeros parametros corresponden al RGB
    //El cuarto parametro corresponde al valor alfa, que permite
el efecto de transparencias
    //0: objeto totalmente transparente; 1: objeto totalmente
opaco
    glClearColor(0.0, 0.0, 0.0, 0.0);

    //establece los parametros de proyeccion ortogonal
    //Se visualizara una proyeccion bidimensional de dimensiones
200 x 150
    //(0,0): esquina inferior izquierda: punto de referencia de
esta ventana
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0, 1080.0, 0.0, 1080.0);
}

//-------------------------TRASLACION-----------------------------
```

```
--

float   traslado(float x, float  y,  float  tx, float  ty){



    float  arreglo[3][3];
    float  arreglo2[3];
    float  resarreglo[2];
    float  aux,res=0;


    arreglo[0][0]=1;
    arreglo[0][1]=0;
    arreglo[0][2]=tx;


    arreglo[1][0]=0;
    arreglo[1][1]=1;
    arreglo[1][2]=ty;




    arreglo[2][0]=0;
    arreglo[2][1]=0;
    arreglo[2][2]=1;



    arreglo2[0]=x;
    arreglo2[1]=y;
    arreglo2[2]=1;




    for(int i=0;  i<2;  i++){
```

```
for(int ii=0;  i<2;  ii++){
    aux=arreglo[i][ii] * arreglo2[ii];
    res=  res+  aux;



}


resarreglo[i]=res;
```

```
    }


    x= resarreglo[0];
    y= resarreglo[1];



    return 0;







}
//-------------------------TRASLACION-------------------------------
--
//-------------------------ESCALAMIENTO-----------------------------
----

float  escalamiento (float x, float  y, float  r){




    traslado(x, y, -x, -y); // Se traslada al punto  origen


    float  arreglo[3][3];  // Definicion de la primera matriz
    float  arreglo2[3];      //Arreglo con  la cordenadas
    float  resarreglo[2];    // Arreglo Resultante de las matrizes
    float  aux,res=0;         // Variables que nos ayudaran como
auxiliares


    arreglo[0][0]=r;
    arreglo[0][1]=0;
    arreglo[0][2]=0;
```

```
arreglo[1][0]=0;
arreglo[1][1]=r;
arreglo[1][2]=0;



arreglo[2][0]=0;
arreglo[2][1]=0;
arreglo[2][2]=1;



arreglo2[0]=x;
arreglo2[1]=y;
arreglo2[2]=1;
```

```
    for(int i=0;  i<2;  i++){


        for(int ii=0;  i<2;  ii++){   //Ciclo que  nos  ayudara a
recorrer las  matrizes

            aux=arreglo[i][ii] * arreglo2[ii];

            res=  res  * aux;   // Operacion de  escalamiento



        }


        resarreglo[i]=res;



    }

    x=  resarreglo[0];
    y=  resarreglo[1];

    traslado (x,y,+x,+y); //Trasladamos a su punto  antes  definido8
    return 0;
```

```
}
//-------------------------ESCALAMIENTO-----------------------------
----

//-------------------------ROTACION---------------------------------
float  rotacion(float x, float  y, float  teta){



    traslado(x, y, -x, -y); // Se traslada al punto  origen

    float  arreglo[3][3];  // Definicion de la primera matriz
```

```
    float  arreglo2[3];       //Arreglo con  la  cordenadas

    float  resarreglo[2];     // Arreglo Resultante de  las  matrizes
    float  aux,res=0;          // Variables que  nos  ayudaran como
auxiliares


    arreglo[0][0]= cos(teta);
    arreglo[0][1]= -sin(teta);
    arreglo[0][2]=0;


    arreglo[1][0]=  sin(teta);
    arreglo[1][1]=  cos(teta);
    arreglo[1][2]=0;




    arreglo[2][0]=0;
    arreglo[2][1]=0;
    arreglo[2][2]=1;




    arreglo2[0]=x;
    arreglo2[1]=y;
    arreglo2[2]=1;






    for(int i=0;  i<2;  i++){



        for(int ii=0;  i<2;  ii++){   //Ciclo que  nos  ayudara a
recorrer las  matrizes


            aux=arreglo[i][ii] * arreglo2[ii];


            res=  res  + aux;   // Operacion de  rotacion
```

```
    }


    resarreglo[i]=res;



}

x= resarreglo[0];
y= resarreglo[1];
```

```
    traslado (x,y,+x,+y); //Trasladamos a su punto antes definido8

    return 0;




}



void dibujaLinea( double x1, double y1, double x2, double y2 )
{
    //establece que lo que se dibuja sera verde
    glBegin(GL_POINTS);
    {
        double m, y;

        m = (float)(y2- y1) / (float)(x2 - x1);
        y = y1;

        for (int x = x1; x < x2; x++)
        {

            y = ((x-x1)* m)+ y1;

            glVertex2f(x,round(y));
            y+=m;
```

```
        }
    }
    glEnd();

    //se obliga a procesar todas las instrucciones de openGL tan
rapidamente como sea posible
    glFlush();

}




//-------------------------ROTACION-------------------------------
void dibujaCirculo( float cx, float cy, float r, int num_segments
)
{
```

```
    //establece que lo que se dibuja sera verde

    glBegin(GL_POINTS);
    {


        for(int ii = 0; ii < num_segments; ii++)
        {
            float theta = 2.0f * 3.1415926f * float(ii) /
float(num_segments);//Obtiene el angulo

            float x = r * cosf(theta);//Calcula eje Y
            float y = r * sinf(theta);//Calcula eje X

            glVertex2f(x + cx, y + cy);

        }



    }
    glEnd();

    //se obliga a procesar todas las instrucciones de openGL tan
rapidamente como sea posible
    glFlush();

}
```

```
void  drawCar()


{


    glClear(GL_COLOR_BUFFER_BIT);  //borra la ventana de
visualizacion
    glColor3f(0.0, 1.0,  0.0);









//-------------------------    1 ALIEN    ------------------------

--------


    dibujaLinea( escal(60, 5),  escal(50,5),
escal(170,5),escal(50,5) );  // Chasis
```

```
    dibujaLinea(escal(60, 5),  escal(50, 5),  escal(75, 5),
escal(65, 5) );//Cofre

    dibujaLinea(escal(75, 5),  escal(65, 5),  escal(90, 5),

escal(65, 5));  // Cofre  Superior

    dibujaLinea(escal(90, 5),escal(65, 5),  escal(105, 5),
escal(80, 5));  // Vidrio trasero

    dibujaLinea(escal(105, 5),  escal(80, 5),  escal(130, 5),

escal(80, 5));// Techo

    dibujaLinea(escal(130, 5),  escal(80, 5),  escal(142, 5),
escal(65, 5));//Parabrisas

    dibujaLinea(escal(142, 5),  escal(65, 5),  escal(155, 5) ,
escal(65, 5));// Motor

    dibujaLinea(escal(155, 5),  escal(65, 5),  escal(170, 5),

escal(50, 5));




    dibujaLinea(escal(90, 5),  escal(65, 5),  escal(142, 5)

,escal(65, 5) ); // Alien

    dibujaLinea(escal(123, 5),  escal(77, 5),  escal(128, 5),
escal(80, 5));  // Antena Derecha

    dibujaLinea( escal(110, 5),  escal(80, 5),  escal(115, 5),

escal(75, 5));  // Antena Izquierda




    dibujaCirculo(escal(118, 5),  escal(72, 5),  escal(5, 5),
escal(300, 5));  //Cara Alien

    dibujaCirculo(escal(116, 5),  escal(74, 5),  escal(1.5, 5),

escal(300, 5));

    dibujaCirculo(escal(115.5, 5),  escal(75, 5),  escal(.5, 5),
escal(300, 5));//Ojo alien  Izquierdo

    dibujaCirculo(escal(118.5, 5),  escal(75, 5),  escal(.5, 5),
escal(300, 5));//Ojo alien  Derecho

    dibujaCirculo(escal(119, 5),  escal(74, 5),  escal(1.5, 5),
escal(300, 5));
```

```
    dibujaCirculo(escal(115, 5), escal(58, 5), escal(4, 5),
escal(200, 5)); //Luzes nave

    dibujaCirculo(escal(150, 5), escal(58, 5), escal(4, 5),
escal(200, 5)); //Luzes nave

    dibujaCirculo(escal(80, 5), escal(58, 5), escal(4, 5),

escal(200, 5)); //Luzes nave




    dibujaCirculo(escal(30, 5), escal(120, 5), escal(30, 5),
escal(200, 5)); // Planeta

    dibujaCirculo(escal(35, 5), escal(140, 5), escal(5, 5),

escal(200, 5)); // Crater

    dibujaCirculo(escal(35, 5), escal(140, 5), escal(5.5, 5),
escal(200, 5)); // Crater
```

```
    dibujaCirculo(escal(5, 5), escal(120, 5), escal(2, 5),
escal(200, 5)); // Crater
    dibujaCirculo(escal(5, 5), escal(120, 5), escal(2.5, 5),
escal(200, 5)); // Crater


//-------------------------    1 ALIEN    -------------------------
--------


//-------------------------    2 ALIEN    -------------------------
--------


    dibujaLinea( escal( trans(60, 100), 3), escal(trans( 50 ,
100),3), escal( trans( 170, 100) ,3),escal(trans( 50 , 100),3) );
// Chasis

    dibujaLinea(escal(trans( 60, 100), 3), escal( trans( 50 ,
100), 3), escal( trans( 75 , 100), 3), escal(trans( 65 , 100), 3)
);//Cofre

    dibujaLinea(escal(trans( 75 , 100), 3), escal(trans( 65 ,
100), 3), escal(trans( 90 , 100), 3), escal(trans( 65 , 100), 3));
// Cofre  Superior

    dibujaLinea(escal(trans( 90 , 100), 3),escal(trans( 65 , 100),
3), escal(trans( 105 , 100), 3), escal(trans( 80 , 100), 3)); //
Vidrio trasero

    dibujaLinea(escal(trans( 105 , 100), 3), escal(trans( 80 ,
100), 3), escal(trans( 130 , 100), 3), escal(trans( 80 , 100),
```

```
3));// Techo


    dibujaLinea(escal(trans( 130 , 100), 3), escal(trans( 80 ,
100), 3), escal(trans( 142 , 100), 3), escal(trans( 65 , 100),
3));//Parabrisas


    dibujaLinea(escal(trans( 142 , 100), 3), escal(trans( 65 ,
100), 3), escal(trans( 155 , 100), 3) , escal(trans( 65 , 100),
3));// Motor

    dibujaLinea(escal(trans( 155 , 100), 3), escal(trans( 65 ,
100), 3), escal(trans( 170 , 100), 3), escal(trans( 50 , 100),
3));



    dibujaLinea(escal(trans( 90 , 100), 3), escal(trans( 65 ,
100), 3), escal(trans( 142 , 100), 3) ,escal(trans( 65 , 100), 3)
```

```
); // Alien

    dibujaLinea(escal(trans( 123  , 100),  3),  escal( trans( 77  ,
100),  3),  escal(trans( 128  , 100),  3),  escal(trans(  80,  100),
3));  // Antena Derecha

    dibujaLinea( escal(trans( 110  , 100),  3),  escal(trans( 80  ,
100),  3),  escal(trans( 115  , 100),  3),  escal(trans( 75  , 100),
3));  // Antena Izquierda




    dibujaCirculo(escal(trans( 118  , 100),  3),  escal(trans( 72  ,
100),  3),  escal(5, 3),  escal(300, 5));  //Cara Alien
    dibujaCirculo(escal(trans( 116  , 100),  3),  escal(trans( 74  ,
100),  3),  escal(1.5, 3),  escal(300, 5));

    dibujaCirculo(escal(trans( 115.5  , 100),  3),  escal(trans( 75  ,
100),  3),  escal(.5, 3),  escal(300, 5));//Ojo alien  Izquierdo
    dibujaCirculo(escal(trans( 118.5  , 100),  3),  escal(trans( 75  ,
100),  3),  escal(.5, 3),  escal(300, 5));//Ojo alien  Derecho
    dibujaCirculo(escal(trans( 119  , 100),  3),  escal(trans( 74  ,
100),  3),  escal(1.5, 3),  escal(300, 5));



    dibujaCirculo(escal(trans( 115  , 100),  3),  escal(trans( 58  ,
100),  3),  escal(4, 3),  escal(200, 5));  //Luzes nave
    dibujaCirculo(escal(trans( 150  , 100),  3),  escal(trans( 58  ,
100),  3),  escal(4, 3),  escal(200, 5));  //Luzes nave
    dibujaCirculo(escal(trans( 80  , 100),  3),  escal(trans( 58  ,
100),  3),  escal(4, 3),  escal(200, 5));  //Luzes nave
```

```
    dibujaCirculo(escal(trans( 30  , 200),  3), escal(trans( 120  ,
200),  3), escal(30, 3), escal(200, 5)); // Planeta

    dibujaLinea(604, 935, 777, 935);
    dibujaLinea(604, 939, 777, 939);



    dibujaLinea(602, 950, 779, 950);
    dibujaLinea(602, 954, 779, 954);


//-------------------------    2 ALIEN   -------------------------
--------

//-------------------------    3 ALIEN   -------------------------
--------
```

```
    dibujaLinea( escal( trans(60, 185),  3),  escal(trans( 50 ,
70),3), escal( trans( 170,  185)  ,3),escal(trans( 50 , 70),3) );
// Chasis

    dibujaLinea(escal(trans(  60,  185),  3),  escal( trans( 50 ,
70),  3),  escal( trans( 75 , 185),  3),  escal(trans( 65 , 70),  3)
);//Cofre

    dibujaLinea(escal(trans( 75 , 185),  3),  escal(trans( 65 , 70),
3),  escal(trans( 90 , 185),  3),  escal(trans( 65 , 70),  3)); //
Cofre  Superior

    dibujaLinea(escal(trans( 90 , 185),  3),escal(trans( 65 , 70),
3),  escal(trans( 105 , 185),  3),  escal(trans( 80 , 70),  3)); //
Vidrio trasero

    dibujaLinea(escal(trans( 105 , 185),  3),  escal(trans( 80 ,
70),  3),  escal(trans( 130 , 185),  3),  escal(trans( 80 , 70),
3));// Techo

    dibujaLinea(escal(trans( 130 , 185),  3),  escal(trans( 80 ,
70),  3),  escal(trans( 142 , 185),  3),  escal(trans( 65 , 70),
3));//Parabrisas

    dibujaLinea(escal(trans( 142 , 185),  3),  escal(trans( 65 ,
70),  3),  escal(trans( 155 , 185),  3) , escal(trans( 65 , 70),
3));// Motor

    dibujaLinea(escal(trans( 155 , 185),  3),  escal(trans( 65 ,
70),  3),  escal(trans( 170 , 185),  3),  escal(trans( 50 , 70),  3));



    dibujaLinea(escal(trans( 90 , 185),  3),  escal(trans( 65 , 70),
```

```
3), escal(trans( 142 , 185), 3) ,escal(trans( 65 , 70), 3) ); //
Alien


    dibujaLinea(escal(trans( 123 , 185), 3), escal( trans( 77 ,
70), 3), escal(trans( 128 , 185), 3), escal(trans( 80, 70), 3));
// Antena Derecha


    dibujaLinea( escal(trans( 110 , 185), 3), escal(trans( 80 ,
70), 3), escal(trans( 115 , 185), 3), escal(trans( 75 , 70), 3));
// Antena Izquierda




    dibujaCirculo(escal(trans( 118 , 185), 3), escal(trans( 72 ,
70), 3), escal(5, 3), escal(300, 5)); //Cara Alien
    dibujaCirculo(escal(trans( 116 , 185), 3), escal(trans( 74 ,
70), 3), escal(1.5, 3), escal(300, 5));
```

```
    dibujaCircunferencia(escal(trans( 115.5  , 185),  3),  escal(trans( 75 ,
70),  3),  escal(.5, 3),  escal(300, 5));//Ojo alien  Izquierdo
    dibujaCircunferencia(escal(trans( 118.5  , 185),  3),  escal(trans( 75 ,
70),  3),  escal(.5, 3),  escal(300, 5));//Ojo alien  Derecho
    dibujaCircunferencia(escal(trans( 119  , 185),  3),  escal(trans( 74 ,
70),  3),  escal(1.5, 3),  escal(300, 5));



    dibujaCircunferencia(escal(trans( 115  , 185),  3),  escal(trans( 58 ,
70),  3),  escal(4, 3),  escal(200, 5));  //Luzes nave
    dibujaCircunferencia(escal(trans( 150  , 185),  3),  escal(trans( 58 ,
70),  3),  escal(4, 3),  escal(200, 5));  //Luzes nave
    dibujaCircunferencia(escal(trans( 80  , 185),  3),  escal(trans( 58 ,
70),  3),  escal(4, 3),  escal(200, 5));  //Luzes nave









    //--------------------------    3 ALIEN    ----------------------
------------



    //--------------------------    ESTRELLAS    -------------------
--------------



    dibujaCircunferencia(500,  500,  1,  300);
    dibujaCircunferencia(700,  800,  1,  300);
    dibujaCircunferencia(200,  300,  1,  300);
    dibujaCircunferencia(700,  700,  1,  300);
```

```
dibujaCirculo(100, 100, 1, 300);
dibujaCirculo(900, 150, 1, 300);
dibujaCirculo(100, 900, 1, 300);
dibujaCirculo(300, 1000, 1, 300);

dibujaCirculo(300, 500, 1, 300);
dibujaCirculo(400, 600, 1, 300);
dibujaCirculo(300, 800, 1, 300);
dibujaCirculo(500, 750, 1, 300);
dibujaCirculo(900, 900, 1, 300);
dibujaCirculo(400, 100, 1, 300);
dibujaCirculo(500, 150, 1, 300);
dibujaCirculo(800, 50, 1, 300);
dibujaCirculo(700, 100, 1, 300);
dibujaCirculo(950, 532, 1, 300);
dibujaCirculo(823, 832, 1, 300);
dibujaCirculo(980, 700, 1, 300);
```

```
    dibujaCirculo(850,650, 1, 300);


    //--------------------------    ESTRELLAS    --------------------

--------------

}




int main(int argc, char** argv)

{

    //se inicializa la pantalla grafica
    glutInit(&argc, argv);

    //establece el modo de visualizacion

    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);

    //Establece la posicion de la esquina superior izquierda del
grafico en la pantalla

    glutInitWindowPosition(100,100);

    //se establece el ancho y la altura de la ventana de
visualizacion

    glutInitWindowSize(800,600);

    //se crea la ventana de visualizacion
    glutCreateWindow("Mi primer dibujo :D, no me sale :(");

    //se ejecuta la funcion de inicializacion de parametros

    init();

    //se envian los graficos a pantalla
    glutDisplayFunc(drawCar);


    glutMainLoop();


}
```

```
    return 0;
```

Prueba

Conclusión

En el desarrollo del programa se pudo concluir , que no es trivial como se grafica de forma nativa sin utilizar líneas u otro tipo de bibliotecas solo con puros puntos y eso nos da una clara idea del funcionamiento interior del OPEN GL.

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACION

ING. EN CIENCIAS DE LA COMPUTACION

GRAFICACIÓN

EXAMEN PARCIAL 1

ALUMNO: TITLA TLATELPA JOSE DE JESÚS

PROFESOR: OLMOS PINEDA IVAN

INTRODUCCIÓN

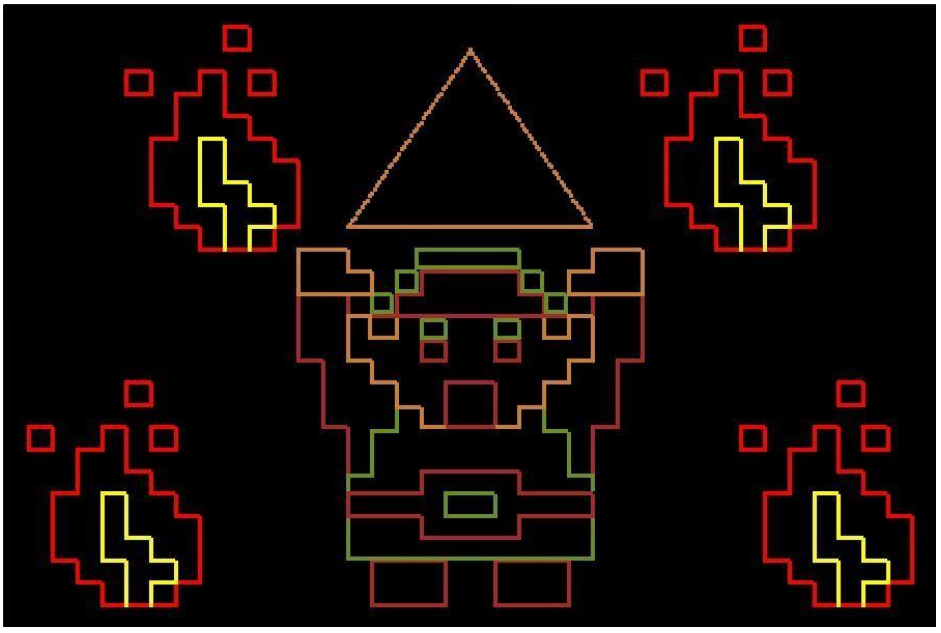
En este documento se explica cuál fue el proceso para graficar un escenario usando OpenGL y la programación orientada a objetos con C++. Empezando desde la creación de cada clase según la figura y los métodos de las misma.

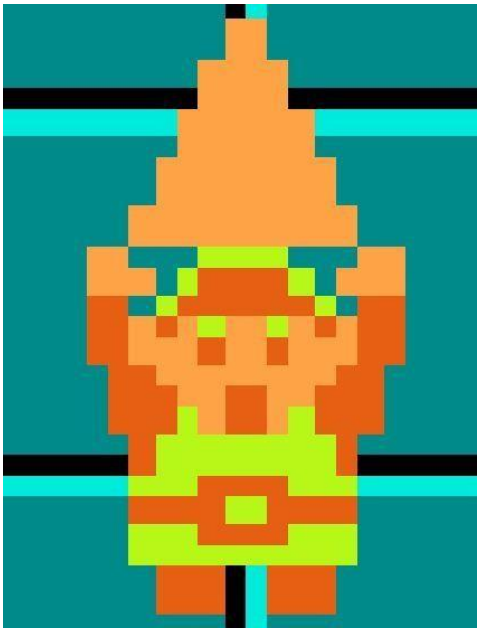
Se realizaron diversos experimentos empezando desde la creación de las clases, las definiciones y la forma en que se programaron los métodos de cada clase. Además se incluye debajo una comparación entre el dibujo original y el resultado que se obtuvo después de graficar el escenario con las clases que antes se programaron.

Algo necesario para la realización de las clases y el escenario fue conocer las fórmulas para las diversas operaciones básicas con objetos, translación, rotación, etc. Incluso se necesitó tener un poco de conocimiento en la POO.

El dibujo que decidí graficar es Link, de la serie de videojuegos The Legend of Zelda, a continuación se muestra algo de información sobre dicho personaje.

Link (リンク <Rinku>) es el nombre genérico que reciben los distintos protagonistas de la serie de videojuegos The Legend of Zelda de Nintendo, creada por Shigeru Miyamoto en 1986, una de las franquicias más exitosas de la empresa. La popularidad de Zelda ha dado lugar a varias encarnaciones de la trama original y, por ende, de Link; el personaje hizo su primera aparición en el título homónimo para la consola NES (Nintendo Entertainment System), como un héroe épico que lucha contra el mal.



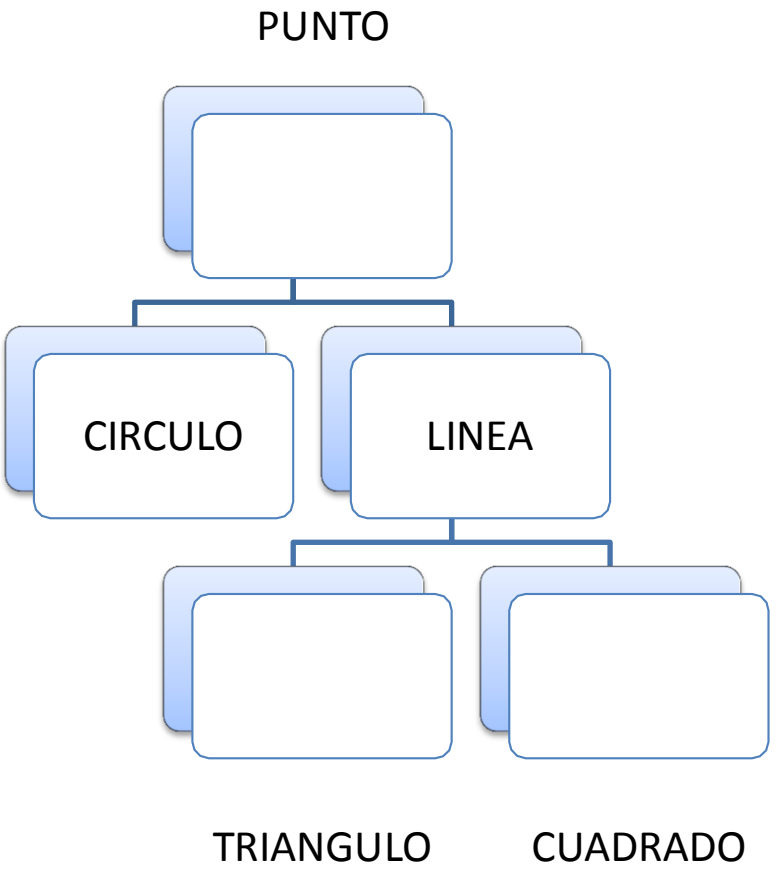


DESARROLLO

Para facilitar la creación del escenario y evitar la duplicidad del código se optó por hacer uso de la POO y programar cada figura en clases diferentes. La programación de las figuras se empezó desde la más básica hasta la más compleja, incluso para hacer una programación más limpia en un archivo .h se hizo la declaración de la clase y en el .cpp se realizó la definición de los métodos.

A continuación se presenta un diagrama de cómo se fueron relacionando los llamados de cada clase en otra clase, si es que lo realizan, desde la más básica hasta la más compleja.

*Los archivos .cpp realizan el llamado de sus correspondientes .h



El llevar esta jerarquía provoca que en el *main* solo se haga el llamado de: circulo, triangulo y cuadrado. Cada clase tiene sus respectivos métodos con las operaciones de trasladar, rotar y escalar, entre otros.

Algunos métodos especiales que se hicieron son: *void <clase>::draw()* y *void <clase>::print<inicial de la clase>().* El segundo método nos enseña las coordenadas de la figura y sirvió para hacer un testing sobre la programación de ella.

Una vez que se tienen las clases programadas se buscó algo para “dibujar”, yo escogí la imagen del personaje que les mostré en la introducción y le agregué como elemento extra el fuego. Ahora analizaremos por partes el archivo principal desde las cabeceras hasta las funciones para dibujar antes del *main.*

En las cabeceras solo se realiza el llamado de los .cpp de algunas figuras, porque esas figuras a su vez llaman otras figuras, por lo explicado anteriormente.

```
#include <GL/glut.h>
#include<iostream>
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include"Cuadrado.cpp"          //incluimos la clase cuadrado
#include"Circulo.cpp"           //incluimos la clase circulo
#include"Triangulo.cpp"         //incluimos la clase triangulo
using namespace std;            //para hacer uso de cout y cin
```

La función init no sufre gran cambio, solo por el tamaño del área a dibujar.

```
void init(void){
    glClearColor(0.0,0.0,0.0,0.0);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0, 380.0, 0.0, 280.0);
    glClear(GL_COLOR_BUFFER_BIT);
}
```

Se declaran los prototipos de funciones, como se observa reciben coordenadas (x,y), eso indica que se pueden colocar en cualquier punto.

```
void LinkNES(double x,double y);
void LinkCafe(double x,double y);
void LinkVerde(double x,double y);
void LinkPiel(double x,double y);
void Fuego(double x,double y);
```

Esta función es llamada por *glutDisplayFunc(<funcion>),* aquí hacemos el llamado de las funciones que definimos más arriba y cambiamos el tamaño de los puntos.

```
void dibujar(){
    glClear(GL_COLOR_BUFFER_BIT);
    glPointSize(4);

    Fuego(40,10);                  //Fuego izquierdo

    Fuego(80,170);

    Fuego(330,10);                  //Fuego Derecho

    Fuego(290,170);
    LinkNES(150,10);
    glFlush();

}
```

La función *main* no sufre cambios a excepción de la posición y el tamaño, después del main se definen las funciones para dibujar.

```
int main(int argc,char **argv){ glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
   glutInitWindowPosition(250,80);
   glutInitWindowSize(900,600);
   glutCreateWindow("Escenario");

   init();
   glutDisplayFunc(dibujar);
   glutMainLoop();

   return 0;

}
```

La función para el fuego usa los colores rojo, amarillo y va tomando como referencia puntos anteriores.

```
void Fuego(double x,double y){
   Linea linea1;

   Cuadrado c1;

   glColor3f(1.0, 0.0, 0.0);       /*establece que lo que se dibuja será rojo*/

   linea1.setValues(x,y,x,y+10);

   linea1.draw();

   linea1.setValues(linea1.getP2X(),linea1.getP2Y(),linea1.getP2X()-10,linea1.getP2Y());

   linea1.draw();

   <…más líneas…>

   glColor3f(1.0, 1.0, 0.0);
   linea1.setValues(linea1.getP2X()+10,linea1.getP2Y(),linea1.getP2X()+10,linea1.getP2Y()+20);
   linea1.draw();
```

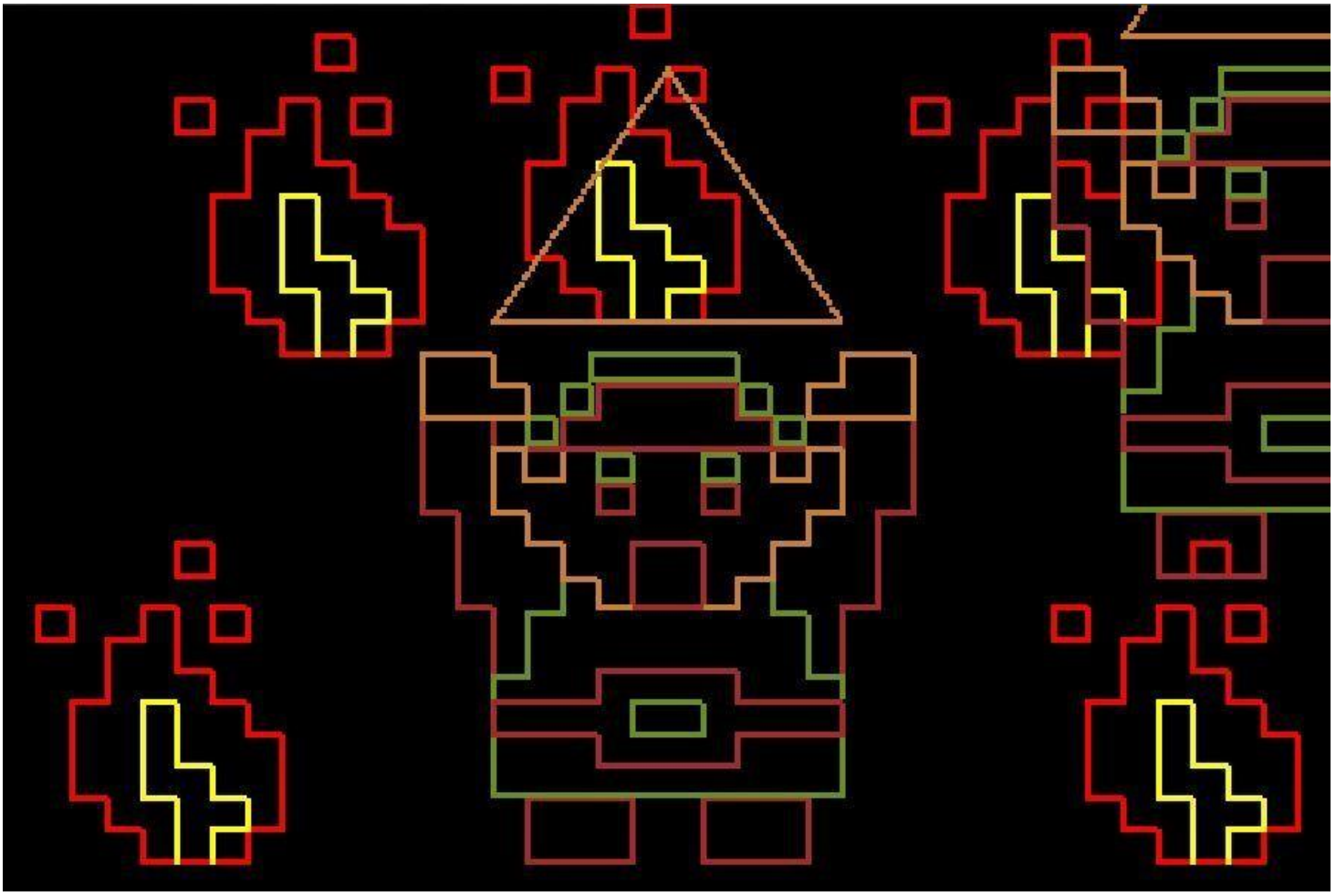
La siguiente función es especial porque en ella se hace el llamado de otras funciones, esta función es la única que se llama en la función dibujar, debajo de esta función se definen las funciones que faltan, cada una usa colores diferentes para el dibujo; El nombre de la función hace referencia a cada color usado.

```
void LinkNES(double x,double y){
   LinkCafe(x,y);

   LinkVerde(x,y);
   LinkPiel(x,y);
```

```
}


void LinkCafe(double x,double y){
    Linea l2;
    Cuadrado c2;
    glColor3f(0.64759, 0.164706, 0.164706); /*establece que lo que se dibuja será café claro*/
    <…cuerpo…>
}
```

```
void LinkVerde(double x,double y){
    Linea l3;
    Cuadrado c3;
    glColor3f(0.419608, 0.556863, 0.137255);  //establece que lo que se dibuja será verde
   <…cuerpo…>
}

void LinkPiel(double x, double y){
    Linea l4;
    Triangulo t1;
    glColor3f(0.8, 0.498039, 0.196078);       //establece que lo que se dibuja será oro
   <…cuerpo…>
}
```

Una vez definidas todas estas funciones se obtiene el dibujo deseado, se pueden dibujar tanto fuegos, como Links se deseen, siempre y cuando no se salgan del área de dibujo, o en todo caso, solo se dibujara la parte que quepa.



EXPERIMENTOS

En esta ocasión se hicieron experimentos para dibujar y para la definición de las clases, a continuación se explica rápidamente algunos de estos experimentos.

- **gluOrtho2D(0.0, 380.0, 0.0, 280.0)**: Poniendo los valores de las coordenadas en esta forma logramos trabajar con el plano en la forma tradicional, se hicieron más pruebas, desde coordenadas negativas a positivas, hasta trabajar como OpenGL nos da las coordenadas.

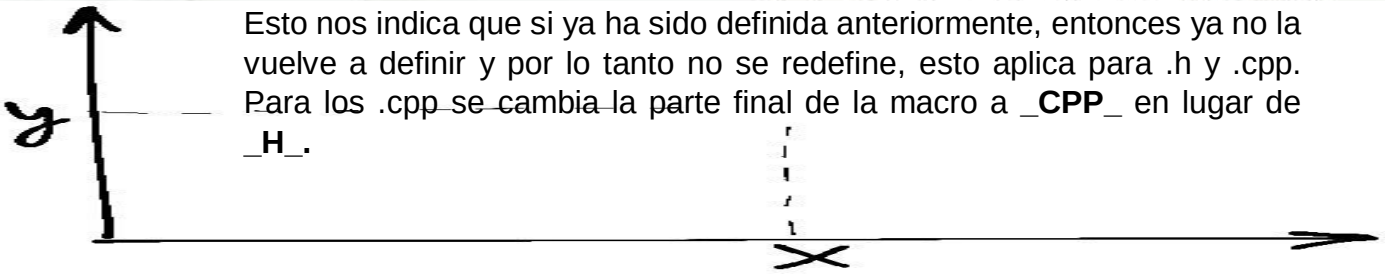
- **Archivos .h y .cpp (redefinición):** El programar las clases no tuvo mucha dificultad, en donde las cosas se pusieron extrañas fue en la inclusión de las cabeceras, como unos archivos llaman a otros, según la jerarquía arriba ilustrada, había ocasiones en que un archivo era llamado 2 o más veces por archivos diferentes y esto ocasionaba que a la hora de compilar se marcara un error de redefinición.

  Este error se solucionó fácilmente con el uso de macros, a continuación se muestra el archivo ***línea.h*** y la forma en que se usaron los macros para evitar su redefinición.

```
#include"punto.cpp"
#ifndef _LINEA_H_        //si no ha sido definida (llamada)
#define _LINEA_H_        //defínela (llámala)

class Linea{
        private:
                Punto p1,p2;
        public:
```

```
};          Linea(); ~Linea();

            void setValues(double,double,double,double);

            double getP1X(); double getP1Y();
            double getP2X(); double getP2Y();
            void trans(double x1, double x2);
            void escala(double f);

            void rotar(double grados);

            void draw();

            void printL();
```

#endif //fin de la definición



Esto nos indica que si ya ha sido definida anteriormente, entonces ya no la vuelve a definir y por lo tanto no se redefine, esto aplica para .h y .cpp. Para los .cpp se cambia la parte final de la macro a **_CPP_** en lugar de **_H_.**

CIRCULO LINEA

TRIANGULO CUADRADO

en 1

user

Documento Técnico del examen

Para este examen se pretendía utilizar texturizacion primitivas y traslación.

En mi programa utilice el uso de primitivas para la presentación del suelo use la primitiva GL_QUADS la cual proporciona un cuadrado con ella represento la calle o banqueta. Tuve problemas para que cargaran tanto los cuadrados perfectos como los polígonos no aparecía un triángulo en ellos, por o consiguiente tuve que rellenarlos con un triángulo con los vértices adecuados.

Para la nieve o simulación de ella utilice un polígono como si fuera una montaña llena de nieve en ella no tuve mayor problemas, con las matrices push y pop trasladas formas o figuras de un punto a otro.

Para el área del letrero se utilizó tanto la función de GL_QUADS como GL_LINES represente en el letrero el paso de un tren con una x y una via.

De acuerdo a los muñecos de south park o mi escenario son círculos y cuadrados utilice círculo tras círculo para tener al primer muñeco representado por el color naranja.

Use la opción glut solid sphere que te da open gl para la creación de esferas aunque a pruebas y errores descubrí como funciona puedes crear polígonos con ella como triángulos, pentágonos tú le das el número de vértices. Gl scalef con ella aumentabas el tamaño de esta opción o comando es como escalar de hecho creo que a eso se refiere o utiliza esa función

Cree un ovni con la opción de trasladar en ella tome una propuesta de un ejercicio que nos propuso en el salón de clases el ovni destella un aro de luz que son dos círculos unidos en la parte de arriba es otro circulo de color verde representando la cabina del conductor (alien)., esta nave u ovni solo sale una vez fue diseñada así puesto un ovni no regresa al mismo lugar.

A la hora de implementación de texturas tuve problemas por lo tanto no pude implementarla.

en 1

user

Introducción

Con lo visto en el curso, el proyecto consiste en desarrollar un escenario en 3D.

Coordenadas Esféricas

El sistema de coordenadas esféricas se basa en la misma idea que las coordenadas polares y se utiliza para determinar la posición espacial de un punto mediante una distancia y dos ángulos.

Para convertir las coordenadas esféricas a cartesianas se aplica las siguientes formulas

$$\begin{bmatrix} x &= r\,\mathrm{sen}\,\theta\cos\phi \\ y &= r\,\mathrm{sen}\,\theta\,\mathrm{sen}\,\phi \\ z &= r\cos\theta \end{bmatrix}$$

Transformaciones OpenGL

Traslación: glTranslatef(GLfloat x, GLfloat y, GLfloat z);

Ejemplo
Nos trasladamos 10 unidades sobre el eje X
glTranslatef(10.0f, 0.0f, 0.0f);

Rotación: glRotatef(GLfloat angulo, GLfloat x, GLfloat y, GLfloat z);

El ángulo de rotación es siempre un ángulo en sentido en contra de las agujas del reloj y medido en grados.

Ejemplo
Rotar 45 grados un objeto sobre el eje X
glRotatef(45.0f, 1.0f, 0.0f, 0.0f);

Modelos OpenGL

glutSolidCone, glutWireCone hacen un cono sólido o alambre, respectivamente.
Sintaxis:

glutSolidCone (base, altura, slices, stacks);
glutWireCone (base, altura, slices, stacks);

base: El radio de la base del cono.
altura: La altura del cono.
slices: El número de subdivisiones alrededor del eje Z.
stacks: El número de subdivisiones a lo largo del eje Z.

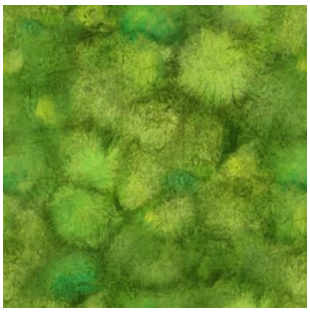
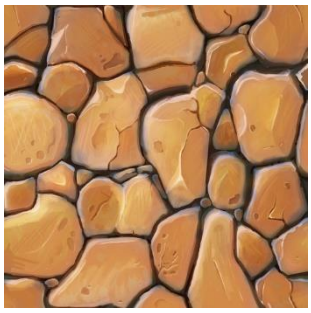
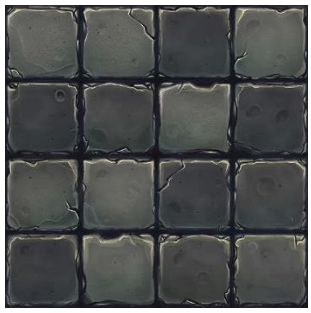
gluCylinder dibuja un cilindro orientado a lo largo del eje z
Sintaxis:

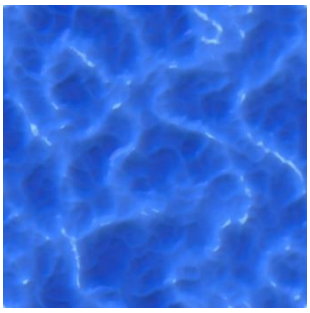
gluCylinder(quad, base, top, height, slices, stacks)

quad: Especifica el objeto de tipo quadric (creado con gluNewQuadric).

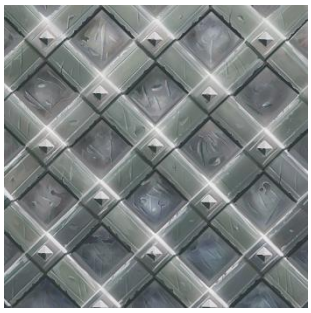
base: Especifica el radio del cilindro en z = 0.
top: Especifica el radio del cilindro en z = altura.
altura: Especifica la altura del cilindro.
slices: Especifica el número de subdivisiones alrededor del eje z.
stacks: Especifica el número de subdivisiones a lo largo del eje z.

Texturas






Desarrollo

Para crear el escenario se utilizaran tres archivos:

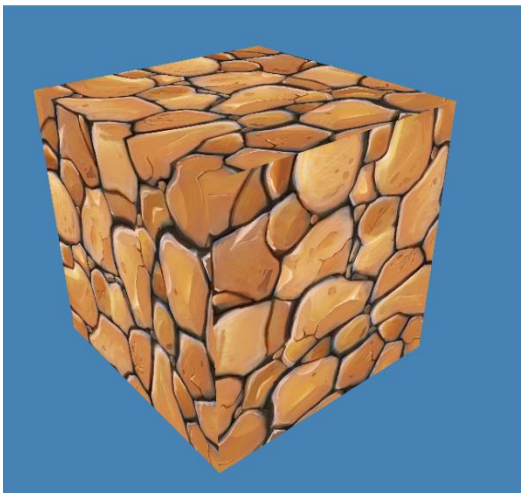
main.cpp
plataforma.h
figuras.h

figuras.h

Este archivo contiene los siguientes modelos que nos ayudaran a crear el escenario:

cubo(): es un cubo creado a partir de vértices y también se añaden las coordenadas de textura.

```
void cubo()
{
   glBegin(GL_QUADS);
// Frente
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f, 0.0f,  3.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(3.0f, 0.0f,  3.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(3.0f, 3.0f,  3.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(0.0f, 3.0f,  3.0f);
// parte de Atras
glTexCoord2f(1.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 0.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(3.0f, 0.0f, 0.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(3.0f, 3.0f, 0.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f, 3.0f, 0.0f);
// Arriba
glTexCoord2f(0.0f, 1.0f);
glVertex3f(0.0f,  3.0f, 0.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f,  3.0f, 3.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(3.0f,  3.0f, 3.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(3.0f,  3.0f, 0.0f);
// Abajo
glTexCoord2f(1.0f, 1.0f);
glVertex3f(0.0f, 0.0f, 0.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(0.0f, 0.0f, 3.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(3.0f, 0.0f, 3.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(3.0f, 0.0f, 0.0f);
//lado Derecho
glTexCoord2f(1.0f, 0.0f);
glVertex3f( 3.0f, 0.0f, 0.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f( 3.0f, 0.0f, 3.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f( 3.0f, 3.0f, 3.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f( 3.0f, 3.0f, 0.0f);
// Lado Izquierdo
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 0.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 3.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(0.0f, 3.0f, 3.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(0.0f, 3.0f, 0.0f);
glEnd();
}
```



cubo()

capa(): se realizo igual que el cubo solo que la altura es menor, este elemento nos servirá para poner una capa de pasto al terreno y también para plataformas y puentes

```
void capa()
{
glBegin(GL_QUADS);
// Frente
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f, 0.0f,  3.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(3.0f, 0.0f,  3.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(3.0f, 1.0f,  3.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(0.0f, 1.0f,  3.0f);
// parte de Atras
glTexCoord2f(1.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 0.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(3.0f, 0.0f, 0.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(3.0f, 1.0f, 0.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f, 1.0f, 0.0f);
// Arriba
glTexCoord2f(0.0f, 1.0f);
glVertex3f(0.0f,  1.0f, 0.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f,  1.0f, 3.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(3.0f,  1.0f, 3.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(3.0f,  1.0f, 0.0f);
// Abajo
glTexCoord2f(1.0f, 1.0f);
glVertex3f(0.0f, 0.0f, 0.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(0.0f, 0.0f, 3.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(3.0f, 0.0f, 3.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(3.0f, 0.0f, 0.0f);
//lado Derecho
glTexCoord2f(1.0f, 0.0f);
glVertex3f( 3.0f, 0.0f, 0.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f( 3.0f, 0.0f, 3.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f( 3.0f, 1.0f, 3.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f( 3.0f, 1.0f, 0.0f);
// Lado Izquierdo
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 0.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 3.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(0.0f, 1.0f, 3.0f);
glTexCoord2f(0.0f, 1.0f);
```

```
glVertex3f(0.0f, 1.0f, 0.0f);
glEnd();
```

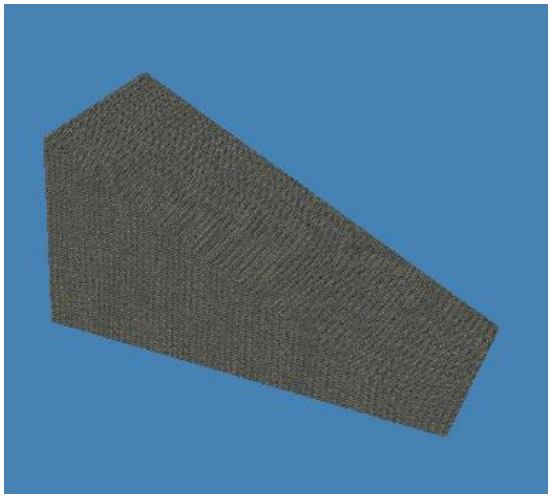
```
}
```

capa()

rampa(): este modelo como su nombre lo indica creara una rampa

```
void rampa()
{
glBegin(GL_QUADS);
// Arriba
glTexCoord2f(0.0f, 1.0f);
glVertex3f(0.0f,  3.0f, 0.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f,  3.0f, 3.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(3.0f,  1.5f, 3.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(3.0f,  1.5f, 0.0f);

glTexCoord2f(0.0f, 1.0f);
glVertex3f(3.0f,  1.5f, 0.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(3.0f,  1.5f, 3.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(6.0f,  0.0f, 3.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(6.0f,  0.0f, 0.0f);
// Abajo
glTexCoord2f(1.0f, 1.0f);
glVertex3f(0.0f, 0.0f, 0.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(0.0f, 0.0f, 3.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(3.0f, 0.0f, 3.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(3.0f, 0.0f, 0.0f);

glTexCoord2f(1.0f, 1.0f);
glVertex3f(3.0f, 0.0f, 0.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(3.0f, 0.0f, 3.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(6.0f, 0.0f, 3.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(6.0f, 0.0f, 0.0f);
// Lado Izquierdo
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 0.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 3.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(0.0f, 3.0f, 3.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(0.0f, 3.0f, 0.0f);
// Frente
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 3.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(3.0f, 0.0f, 3.0f);
glTexCoord2f(1.0f, 0.5f);
```

```
glVertex3f(3.0f, 1.5f, 3.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(0.0f, 3.0f, 3.0f);
// parte de Atras
glTexCoord2f(1.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 0.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(3.0f, 0.0f, 0.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(3.0f, 1.5f, 0.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f, 3.0f, 0.0f);

glEnd();

glBegin(GL_TRIANGLES);
// Frente
glTexCoord2f(0.0f, 0.0f);
glVertex3f(3.0f, 0.0f, 3.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(6.0f, 0.0f, 3.0f);
glTexCoord2f(0.0f, 0.5f);
glVertex3f(3.0f, 1.5f, 3.0f);
// parte de Atras
glTexCoord2f(0.0f, 0.0f);
glVertex3f(3.0f, 0.0f, 0.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(6.0f, 0.0f, 0.0f);
glTexCoord2f(0.0f, 0.5f);
glVertex3f(3.0f, 1.5f, 0.0f);
glEnd();

}
```

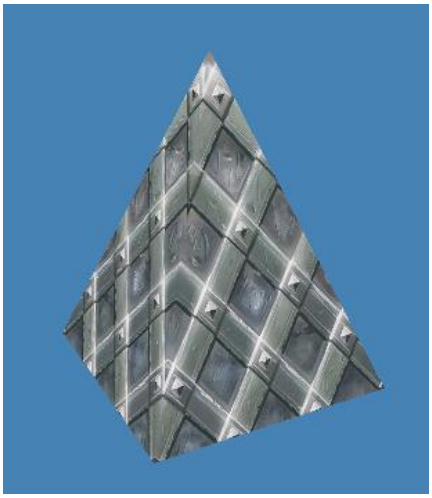


rampa()

Pica(): este modelo se utilizara en las "*trampas*"

```
void pica()
{
glBegin(GL_QUADS);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 0.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 1.5f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(1.5f, 0.0f, 1.5f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(1.5f, 0.0f, 0.0f);
glEnd();

glBegin( GL_TRIANGLES );
//atras
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 0.0f );
glTexCoord2f(1.0f, 0.0f);
glVertex3f(1.5f, 0.0f, 0.0f );
```

```
glTexCoord2f(0.5f, 1.0f);
glVertex3f(0.75f,2.0f, 0.75f);
//enfrente
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 1.5f );
glTexCoord2f(1.0f, 0.0f);
glVertex3f(1.5f, 0.0f, 1.5f );
glTexCoord2f(0.5f, 1.0f);
glVertex3f(0.75f,2.0f, 0.75f);
//lado izq
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 0.0f );
glTexCoord2f(1.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 1.5f );
glTexCoord2f(0.5f, 1.0f);
glVertex3f(0.75f,2.0f, 0.75f);
//lado der
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 0.0f );
glTexCoord2f(1.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 1.5f );
glTexCoord2f(0.5f, 1.0f);
glVertex3f(0.75f,2.0f, 0.75f);
glEnd();
}
```



pica()

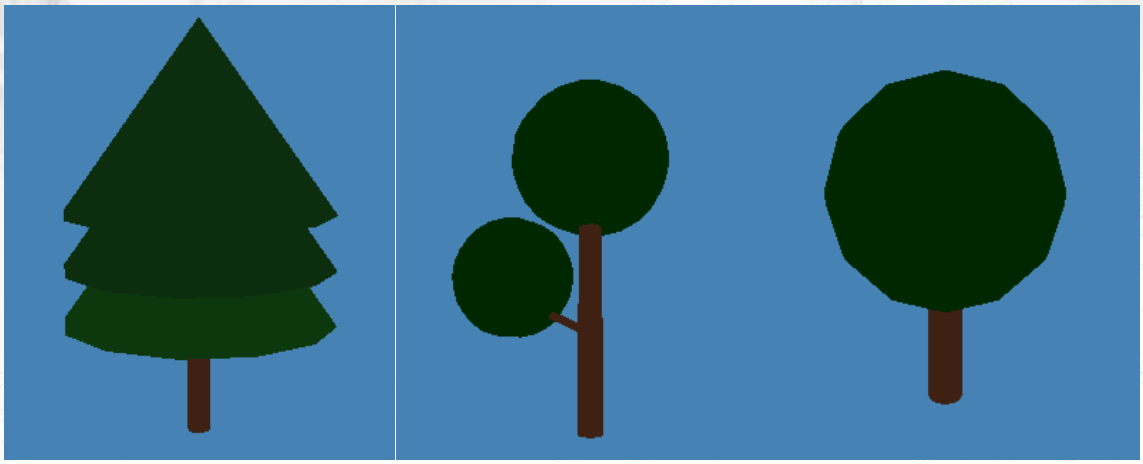
cilindro(): crea un cilindro con la ayuda de gluCylinder

```
void cilindro(double altura, double radio, int slices, int stacks)
{    GLUquadricObj* cyl;
    cyl = gluNewQuadric();
    gluQuadricDrawStyle(cyl, GLU_FILL);
    gluQuadricNormals(cyl,GLU_SMOOTH);
    glPushMatrix();
       glRotatef(90,1,0,0);
       gluCylinder(cyl, radio, radio, altura, slices, stacks);
    glPopMatrix();
}
```

cilindro()

arbol01(), arbol02(), arbol03(): 3 tipos de arboles

```
void arbol01()
{
   glColor3f(0.627, 0.322, 0.176);
   cilindro(5,0.2,12,12);
   glPushMatrix();
      glColor3f(0.133, 0.645, 0.133);
      glRotatef(-90,1,0,0);
      glTranslatef(0,0,-1);
      glColor3f(0.133, 0.445, 0.133);
      glutSolidCone(2.5,3.5,12,12);
      glTranslatef(0,0,-1);
      glutSolidCone(2.5,3.5,12,12);
      glTranslatef(0,0,-1);
      glColor3f(0.133, 0.545, 0.133);
      glutSolidCone(2.5,3.5,12,12);
      glPopMatrix();
   glColor3f(1, 1, 1);
}

void arbol02()
{
   glColor3f(0.627, 0.322, 0.176);
   cilindro(7,0.3,12,12);
   glPushMatrix();
      glRotatef(-60,1,0,0);
      glTranslatef(0,1,-4);
      cilindro(3,0.1,12,12);
   glPopMatrix();

   glColor3f(0.000, 0.392, 0.000);
   glPushMatrix();
      glutSolidSphere(2,12,12);
      glTranslatef(0,-3,-2);
      glutSolidSphere(1.5,12,12);
   glPopMatrix();
   glColor3f(1, 1, 1);
}

void arbol03()
{
   glColor3f(0.627, 0.322, 0.176);
   cilindro(5,0.3,12,12);

   glColor3f(0.000, 0.392, 0.000);
   glPushMatrix();
      glTranslatef(0,-1,0);
      glutSolidSphere(2,12,12);
   glPopMatrix();

   glColor3f(1, 1, 1);
}
```



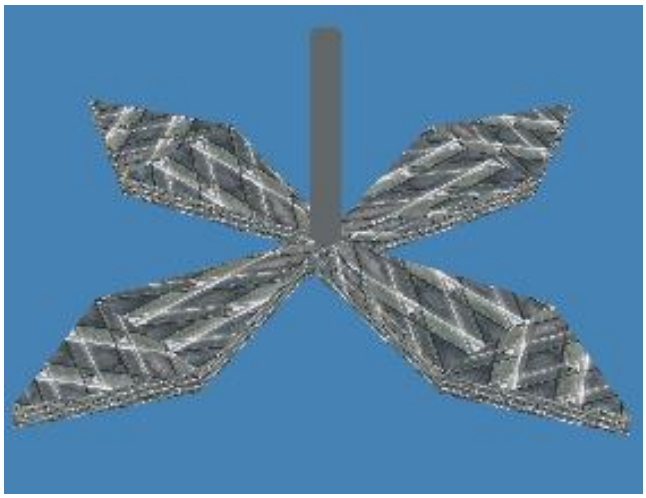
arbol01() *arbol02()* *arbol03()*

rehilete(): se colocara debajo de las plataformas que flotan

```
void elice()
{
glBegin(GL_TRIANGLES);
glTexCoord2f(0.0f, 0.5f);
glVertex3f(0.0f, -3.0f, 0.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(-1.0f, -3.0f, -3.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(1.0f, -3.0f, -3.0f);

glTexCoord2f(0.0f, 0.5f);
glVertex3f(-1.0f, -3.0f, -3.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(0.0f, -3.0f, -5.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(1.0f, -3.0f, -3.0f);

glTexCoord2f(0.0f, 0.5f);
glVertex3f(0.0f, -3.3f, 0.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(-1.0f, -3.3f, -3.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(1.0f, -3.3f, -3.0f);

glTexCoord2f(0.0f, 0.5f);
glVertex3f(-1.0f, -3.3f, -3.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(0.0f, -3.3f, -5.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(1.0f, -3.3f, -3.0f);
glEnd();

glBegin(GL_QUADS);
glTexCoord2f(1.0f, 0.0f);
glVertex3f( 0.0f, -3.0f,  0.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f( 0.0f, -3.3f,  0.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(-1.0f, -3.3f, -3.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(-1.0f, -3.0f, -3.0f);

glTexCoord2f(1.0f, 0.0f);
glVertex3f(0.0f, -3.0f, 0.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f, -3.3f, 1.5f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(1.0f, -3.3f, -3.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(1.0f, -3.0f, -3.0f);

glTexCoord2f(1.0f, 0.0f);
glVertex3f(-1.0f, -3.0f, -3.0f);
glTexCoord2f(0.0f, 0.0f);
```

```
glVertex3f(-1.0f, -3.3f, -3.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(0.0f, -3.3f, -5.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(0.0f, -3.0f, -5.0f);

glTexCoord2f(1.0f, 0.0f);
glVertex3f(1.0f, -3.0f, -3.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(1.0f, -3.3f, -3.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(0.0f, -3.3f, -5.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(0.0f, -3.0f, -5.0f);
glEnd();
```

```
}

void rehilete()
{
glPushMatrix();
cilindro(3,0.2,12,12);
elice();
glRotatef(90,0,1,0);
elice();
glRotatef(90,0,1,0);
elice();
glRotatef(90,0,1,0);
elice();
glPopMatrix();
}
```

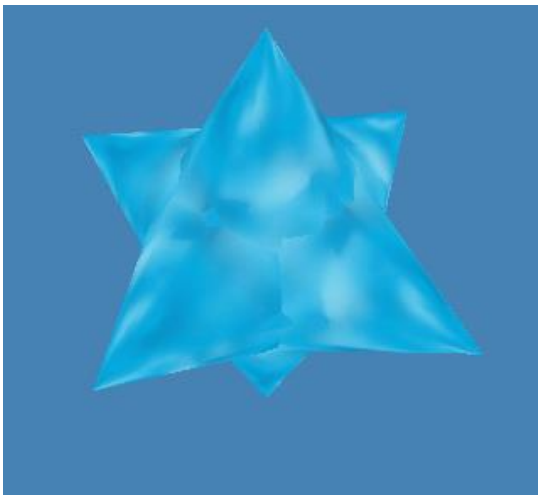


rehilete()

estrella(): se crea una estrella a partir de conos

```
void estrella()
{
GLUquadricObj* quadobj = gluNewQuadric();
gluQuadricTexture(quadobj, GL_TRUE);
glPushMatrix();
gluCylinder(quadobj, 1.0, 0, 2.0, 10, 10);
glRotatef(180,0,1,0);
gluCylinder(quadobj, 1.0, 0, 2.0, 10, 10);
glPopMatrix();
glPushMatrix();
```

```
glRotatef(90,1,0,0);
gluCylinder(quadobj, 1.0, 0, 2.0, 10, 10);
glRotatef(180,0,1,0);
gluCylinder(quadobj, 1.0, 0, 2.0, 10, 10);
glPopMatrix();
glPushMatrix();
glRotatef(90,0,1,0);
gluCylinder(quadobj, 1.0, 0, 2.0, 10, 10);
glRotatef(180,0,1,0);
gluCylinder(quadobj, 1.0, 0, 2.0, 10, 10);
glPopMatrix();
}
```



estrella()

Plataforma.h

En este archivo se crea una clase la cual estará encargada de mover la plataforma, reutilizando la mayor parte del código de la práctica de la esfera rebotando dentro de un cubo.

```
#include <stdlib.h>
#include <time.h>
#include <math.h>
class Plataforma
{
   private:
      float maxpos;
      float vel;
      float pos[3];
      float dir[3];

   public:
      Plataforma(float maxpos,float vel,int
dx,int dy,int dz)
      {
         this->maxpos = maxpos;
         this->vel = vel*0.05;

         pos[0] = 0;
         pos[1] = 0;
         pos[2] = 0;
```

```
        dir[0] = dx;
        dir[1] = dy;
        dir[2] = dz;
    }

    void valida()
    {
        if((pos[0] < 0) || (pos[0] > maxpos))
            dir[0]*=-1;

        if((pos[1] < 0) || (pos[1] > maxpos))
            dir[1]*=-1;

        if((pos[2] < 0) || (pos[2] > maxpos))
            dir[2]*=-1;

        pos[0] += dir[0]*vel;
        pos[1] += dir[1]*vel;
        pos[2] += dir[2]*vel;
    }

    void cambiaVel(float v)
    {
        this->vel=v*0.1;
    }

    float *getPos()
    {
        return pos;
    }
};
```

Main.cpp

Este archivo es el principal del escenario, se incluyen los otros dos archivos creados y para l escenario al ser 3D, la manera de visualizar será con *gluPerspective* y la posición de la cámara se manejara con *gluLookAt,* únicamente se modifica la posición de la cámara ya que siempre estará observando al punto (0,0,0).

Se utilizaran las siguientes teclas para el movimiento de cámara.

Tecla	Acción
a	Gira la cámara hacia la izquierda
d	Gira la cámara hacia la derecha
s	Gira la cámara hacia abajo
w	Gira la cámara hacia arriba
q	Acerca la cámara al escenario
e	Aleja la cámara del escenario
esc	Sale del programa

Cada vez que se presiona una tecla de movimiento, se calcula la posición de la cámara y para el caso del acercamiento o alejamiento se valida que no salga del rango de [-100,100] después de los cálculos se llama a reshape para posicionar la cámara.

```
void keyboard(unsigned char key, int x, int y)
{
   switch(key)
   {

      case 'd':
               azim -= 2;
               EYE_X=r*sin(M_PI/180*azim)*cos(M_PI/180*elev);
               EYE_Y=r*sin(M_PI/180*elev);
               EYE_Z=r*cos(M_PI/180*azim)*cos(M_PI/180*elev);
               reshape(ancho,alto);
               break;
      case 'a':
               azim += 2;
               EYE_X=r*sin(M_PI/180*azim)*cos(M_PI/180*elev);
               EYE_Y=r*sin(M_PI/180*elev);
               EYE_Z=r*cos(M_PI/180*azim)*cos(M_PI/180*elev);
               reshape(ancho,alto);
               break;
      case 'w':
               elev += 2;
```

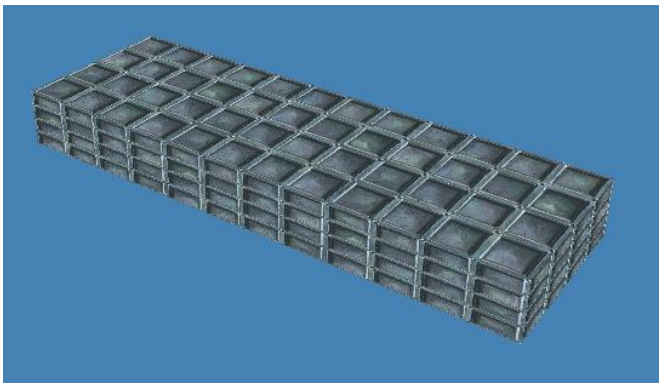
```
            EYE_X=r*sin(M_PI/180*azim)*cos(M_PI/180*elev);
            EYE_Y=r*sin(M_PI/180*elev);
            EYE_Z=r*cos(M_PI/180*azim)*cos(M_PI/180*elev);
            reshape(ancho,alto);
            break;
      case 's':
            elev -= 2;
            EYE_X=r*sin(M_PI/180*azim)*cos(M_PI/180*elev);
            EYE_Y=r*sin(M_PI/180*elev);
            EYE_Z=r*cos(M_PI/180*azim)*cos(M_PI/180*elev);
            reshape(ancho,alto);
            break;

      case 'q':
            if(r>-100)
               r--;
            EYE_X=r*sin(M_PI/180*azim)*cos(M_PI/180*elev);
            EYE_Y=r*sin(M_PI/180*elev);
            EYE_Z=r*cos(M_PI/180*azim)*cos(M_PI/180*elev);
            reshape(ancho,alto);
            break;
       case 'e':
            if(r<100)
               r++;
            EYE_X=r*sin(M_PI/180*azim)*cos(M_PI/180*elev);
            EYE_Y=r*sin(M_PI/180*elev);
            EYE_Z=r*cos(M_PI/180*azim)*cos(M_PI/180*elev);
            reshape(ancho,alto);
            break;

      case 27:    //escape
            exit(0);
            break;
   }
}
```

Para el efecto del cielo se crea una esfera y se le pone una textura de tal modo que nuestro escenario se encontrara dentro de la esfera.

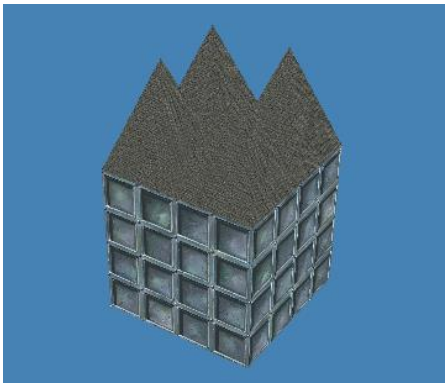
```
glBindTexture(GL_TEXTURE_2D, tex_2d[3]);
  glPushMatrix();
    gluQuadricTexture(quadObj, GL_TRUE);
    glRotatef(-90,1,0,0);
    gluSphere(quadObj,101,50,50);
  glPopMatrix();
```

Adicional a los modelos que se crearon en el archivo figuras.h se crean procesos que utilizan esos modelos y con la ayuda de transformaciones se obtienen figuras mas complejas.

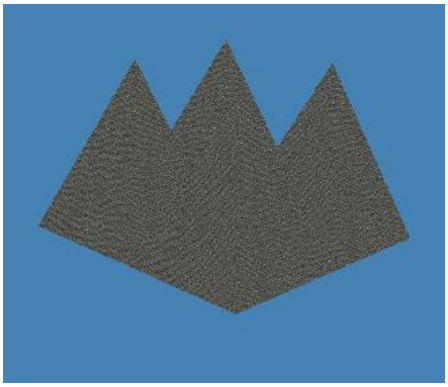
plataforma()



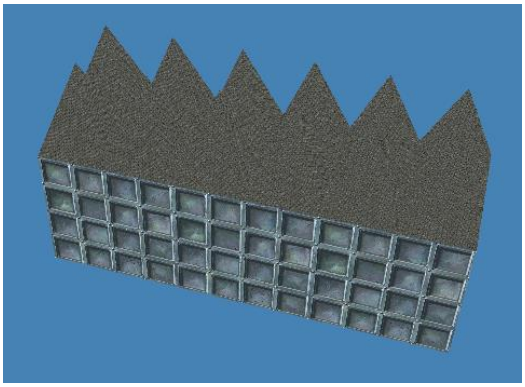
picas()



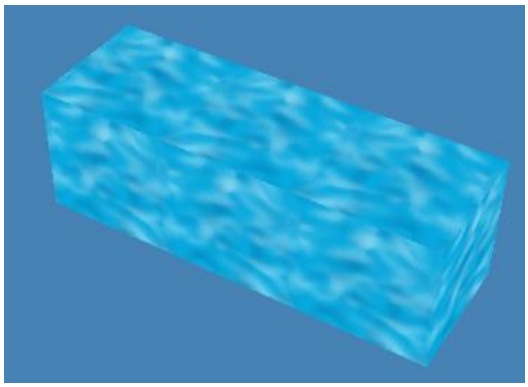
picas01()



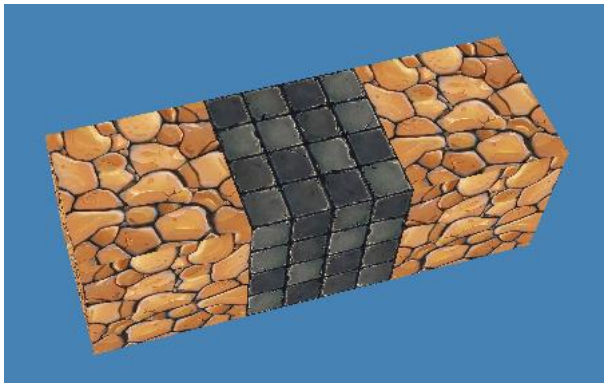
tresPicas()



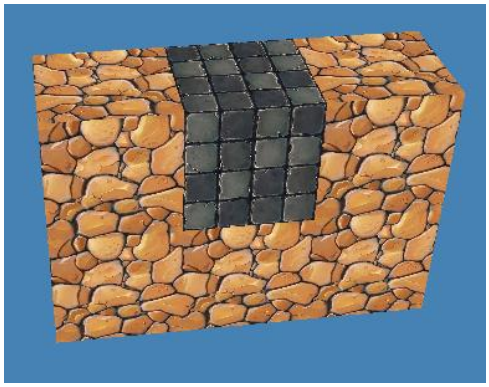
linea()



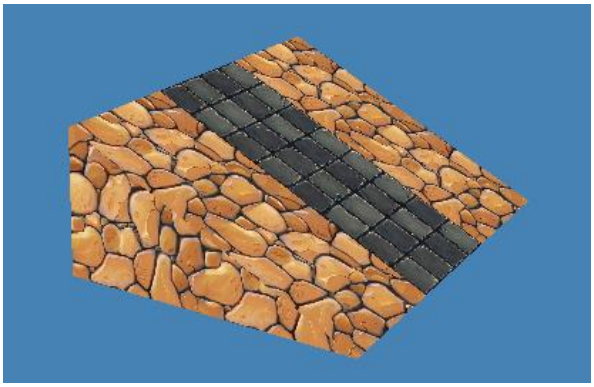
lineaCamino()



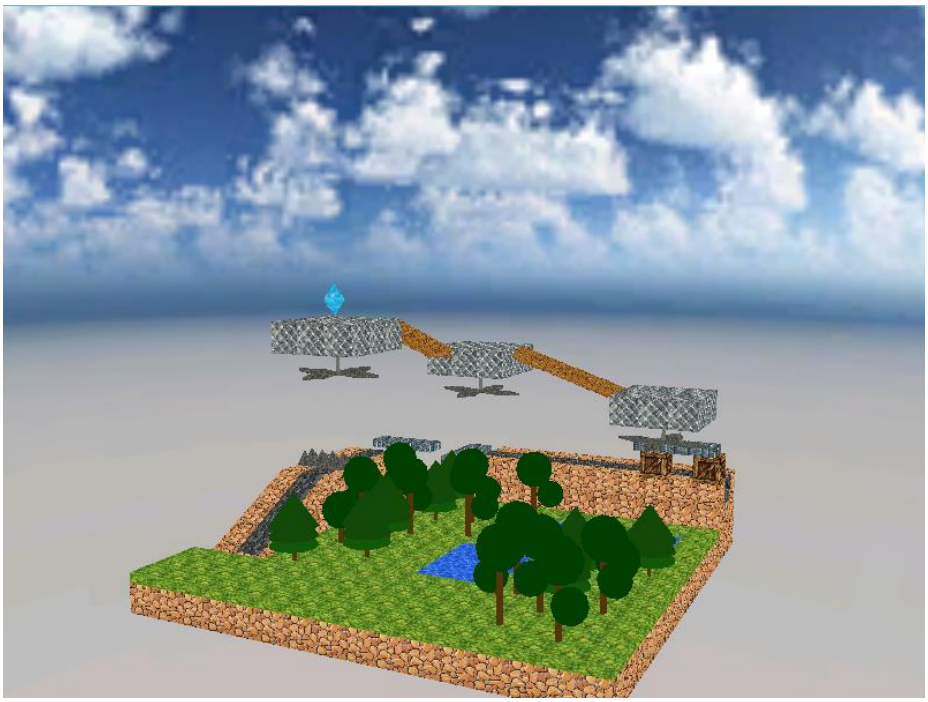
biLinea()



d_rampa()



En la función dibuja, con ayuda de transformaciones los modelos del archivo formas.h y los métodos creados en main.cpp se obtiene el escenario siguiente



Conclusión

A lo largo del curso hemos visto varios efectos, transformaciones, iluminaciones que nos permiten crear escenarios como el de este proyecto además de que nos hace ver que es fundamental el conocimiento de las bases de la Graficación por computadora para poder entender cómo es que funcionan los motores gráficos que existen y no tener mayor problema en el manejo de estos.

Referencias

HUGHES, J. F. (2014). *Computer Graphics.* Saddle River, New Jersey: Addison-Wesley.

Morales, C. C. (s.f.). *Apuntes de OpenGL y GLUT* . Obtenido de http://www.elai.upm.es/webantigua/spain/Asignaturas/InfoInd/teoria/Apuntes%20de%20OpenGL.pdf

OpenGL. (9 de Julio de 2011). *OpenGL Wiki!* Recuperado el 27 de 02 de 2015, de https://www.opengl.org/wiki/Viewing_and_Transformations

Shreiner, D., Sellers, G., Kessenich, J., & Licea-Kane, B. (2013). *OpenGL programming guide.* Michigan, Ann Arbor, Estados Unidos: Addison-Wesley.

BENEMRITA UNIVERSIDAD AUTONOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LAS COMPUTACION

MC. IVAN OLMOS PINEDA

MATERIA: GRAFICACION ALUMNO:

SANDOVAL SALAZ CONSUELO PRIMER

PARCIAL

MATRICULA: 201023593

Introducción

El propósito de este proyecto es que el alumno sea capaz de trabajar en cualquier otro lenguaje de programación a partir de una única primitiva la cual será el graficado de puntos y líneas para que a partir de ellas se pueda trabajar sin ningún problema.

El entorno en el cual se llevara a cabo dicho proyecto será el entorno Devc++

utilizando la librería de OpenGl <GL/glut.h>

El trabajo consta en crear librerías para realizar las figuras, cada librería constara de sus propios atributos y funciones de operadores que se encargan de manipular cada figura.

Desarrollo

La clase línea tendrá como atributos cuatro variables enteras las cuales contendrán x1, y1, x2, y2. También tendrá su función setValues que se encarga de recibir los valores de los puntos. Así como también su función draw la cual se encarga de graficar cada punto obtenido mediante el algoritmo de dibujado de líneas.

Para la clase Cuadrilátero su atributo será un vector de tamaño 8 el cual guardara las coordenadas del cuadrilátero.

Las funciones son las siguientes: Función setValues() que recibirá 8 entero que corresponderán a las coordenadas del cuadrilátero. Su función traslada() la cual recibe dos valores enteros que representan el vector de traslación. Su función escala() que recibe como parámetro un entero que será el factor de escalamiento. Su función rota() que recibe un entero el cual tomara su lugar como el grado en el cual queremos rotar la figura. La función reflexión() que recibe un carácter para indicar con respecto a que eje queremos la transformación de la figura, en este caso se debe ser muy cuidadoso ya que solo están definidas tres acciones la reflexión con respecto a X, la reflexión con respecto a Y y la reflexión con respecto al origen en cada caso se debe de recibir un carácter comprendido entre ‘X’,

’Y’ o bien ‘O’ este último indica que su reflexión será con respecto al origen. La función deformación() recibe como parámetros un carácter que puede ser ‘X’ o ‘Y’ para indicar con respecto a que eje se quiere la deformación, así como también recibe un entero que será el factor de deformación de la

figura. Por último se encuentra la función drawCuadri() que es la encargada de dibujar el cuadrilátero definitivo que se verá en pantalla.

En la clase círculo se tendrá como atributos un vector de tamaño 2 que será el encargado de guardar las coordenadas del punto de origen del círculo y un entero r que será el radio del círculo.

Entre las funciones están la función setValues () que recibe como parámetro tres entero los cuales representaran el X y Y del punto origen del circulo y el tercer parámetro será el tamaño del radio.

Su función traslada es análoga a la función con el mismo nombre traslada de la clase cuadrilátero previamente definida así que se omitirá en este apartado.

La función escala() debido a que un circulo se dibuja por el largo del radio, en este caso solo recibe como parámetro el factor de escalamiento el cual es multiplicado por el radio del circulo previamente definido.

Y por último la función drawCir() que se encarga del dibujado del circulo final por medio del algoritmo de bresenham.

Para el dibujado de líneas se tenía dos opciones el graficado por medio de la ecuación de la pendiente de la recta la cual es y=mx+b, donde m es la pendiente y b la intersección de la recta con el eje y. y la otra es utilizar el algoritmo de Bresenham el cual es más eficiente a la hora de graficar la línea. En este caso como se está empezando a trabajar por primera vez se optó por el algoritmo de Bresenham para estar seguros de que no se tuvieran problemas más adelante, por ejemplo a la hora de rotar una figura ya que al ser demasiado inclinada se podrían tener errores y no se podría visualizar si el problema se encuentra con la función de rotación o con el dibujado de líneas.

En cada clase se llamara a la librería de dibujado de líneas, a continuación se describe como se grafica un cuadrilátero.

Como ya se había mencionado la clase cuadrilátero tiene un vector llamado puntos de tamaño 8 que tendrá las coordenadas del cuadrilátero como sigue, puntos[0]=x1, puntos[1]=y1, puntos[2]=x2, puntos[3]=y2, puntos[4]=x3, puntos[5]=y3, puntos[6]=x4, puntos[7]=y4. Así cuando se dibuje el cuadrilátero se crea cuatro objetos de tipo línea

```
izq.setValues(puntos[0],puntos[1],puntos[2],puntos[3]);
sup.setValues(puntos[2],puntos[3],puntos[4],puntos[5]);
der.setValues(puntos[4],puntos[5],puntos[6],puntos[7]);
bas.setValues(puntos[6],puntos[7],puntos[0],puntos[1]);
```

Después solo mandar a llamar a la función de dibujado de líneas y se dibujan las cuatro líneas declaradas como izq, sup, der y bas.

Para la función de traslación define una matriz de traslación:

```
mat[0][0]=1;
mat[0][1]=0;
mat[0][2]=a;
```

```
mat[1][0]=0;
mat[1][1]=1;
mat[1][2]=b;
mat[2][0]=0;
mat[2][1]=0;
mat[2][2]=1;
```

a y b cumplen como el vector de traslación una vez obtenida esta matriz se multiplica cada punto por dicha matriz y de esta forma se van obteniendo las coordenadas finales de la figura ya trasladada.

Función de escalamiento lo primero que hay que hacer es definir el punto pivote el cual es trasladado al origen restándole las coordenadas del mismo punto a todos los puntos de la figura, por ejemplo si se toma de pivote el punto 1 entonces el vector de traslación estaría compuesto por <X1,Y1> el cual sería restado a todos los puntos, para después multiplicar cada punto por la matriz de escalamiento la cual se define de la siguiente manera:

```
mat[0][0]=r;
mat[0][1]=0;
mat[0][2]=0;
mat[1][0]=0;
mat[1][1]=r;
mat[1][2]=0;
mat[2][0]=0;
mat[2][1]=0;
mat[2][2]=1;
```

La variable r es recibida como parámetro y tiene la función de factor de escalamiento, una vez multiplicado cada punto se le suma las coordenadas <X1,Y1> y el resultado de eso serían los puntos finales de la figura.

En la función de rotación también se define un punto pivote el cual se traslada al origen restando las coordenadas del punto pivote a cada punto de la figura, para después multiplicar cada uno de los puntos de la figura por la matriz de rotación la cual se inicializa de la siguiente forma:

```
mat[0][0]=cos(r);
mat[0][1]=-sin(r);
mat[0][2]=0;
mat[1][0]=sin(r);
mat[1][1]=cos(r);
mat[1][2]=0;
mat[2][0]=0;
mat[2][1]=0;
mat[2][2]=1;
```

r es recibido como parámetro es una variable entera e indica el número de grados que se desea girar la figura.

La función de reflexión lo primero que hace es decidir con respecto a que eje se quiere la reflexión esto se puede gracias a que la función recibe como parámetro un carácter el cual dirá como definir la matriz de reflexión, en esta función hay tres casos :

1.- si el carácter recibido es ‘X’ entonces la matriz de reflexión es definida de la siguiente manera:

```
mat[0][0]=1;

mat[0][1]=0;
mat[0][2]=0;
mat[1][0]=0;
mat[1][1]=-1;
mat[1][2]=0;
mat[2][0]=0;
mat[2][1]=0;
mat[2][2]=1;
```

2.- si el carácter recibido es ‘Y’ entonces la matriz de reflexión es definida de la siguiente manera:

```
mat[0][0]=-1;
mat[0][1]=0;
mat[0][2]=0;
mat[1][0]=0;
mat[1][1]=1;
mat[1][2]=0;
mat[2][0]=0;
mat[2][1]=0;
mat[2][2]=1;
```

3.- si el carácter recibido es ‘O’ entonces la matriz de reflexión es definida de la siguiente manera:

```
mat[0][0]=-1;
mat[0][1]=0;
mat[0][2]=0;
mat[1][0]=0;
mat[1][1]=-1;
mat[1][2]=0;
mat[2][0]=0;
mat[2][1]=0;
mat[2][2]=1;
```

Una vez definida la matriz de reflexión lo que se hace es definir el punto pivote trasladarlo al origen restando el vector de traslación, como ejemplo si el pivote fuera el punto 1 entonces el vector de traslación seria <X1,Y1> el cual se restaría a todos los puntos de la figura, después se procede a multiplicar cada punto por la matriz de reflexión y después sumar el vector de traslación a cada punto obteniendo así los puntos finales de la figura.

mat[0][0]=1;

La función de deformación recibe como parámetros un carácter el cual dirá con respecto a que eje se quiere la deformación de la figura y un entero que será el factor de deformación, lo primero que se hace es revisar el carácter recibido como parámetro si el carácter es ‘X’ entonces indica que se requiere una deformación con respecto al eje X y por tanto la matriz de deformación queda de la siguiente manera:

```
mat[0][0]=1;

mat[0][1]=sh;
mat[0][2]=0;
mat[1][0]=0;
mat[1][1]=1;
mat[1][2]=0;
mat[2][0]=0;
mat[2][1]=0;
mat[2][2]=1;
```

de lo contrario si el carácter recibido es ‘Y’ entonces nos indica que se requiere una deformación con respecto al eje Y y la matriz de deformación queda de la siguiente forma:

```
mat[0][0]=1;
mat[0][1]=0;
mat[0][2]=0;
mat[1][0]=sh;
mat[1][1]=1;
mat[1][2]=0;
mat[2][0]=0;
mat[2][1]=0;
mat[2][2]=1;
```

en las dos matrices el sh representa el factor de deformación, una vez obtenida la matriz de deformación se define el punto pivote se traslada al origen restando el vector de traslación ,como ejemplo si el pivote fuera el punto 1 entonces el vector de traslación seria <X1,Y1> el cual se restaría a todos los puntos de la figura, después se procede a multiplicar cada punto por la matriz de reflexión y después sumar el vector de traslación a cada punto obteniendo así los puntos finales de la figura.

Para la clase Circulo la función setValues recibe como parámetro tres entero los dos primeros representan las coordenadas X, Y del centro del circulo y la tercer variable es el radio que tendrá el circulo y tiene las siguientes funciones definidas.
La función traslada recibe dos enteros y su forma de operar es la misma que la función traslada de la clase triangulo.

En su función escala recibe un entero que actúa como factor de escalamiento solo que en este caso lo único que se hace es multiplicar el radio por el factor de escalamiento y de esta forma a la hora de graficar el circulo se obtendrá el radio ya operado y a escala.

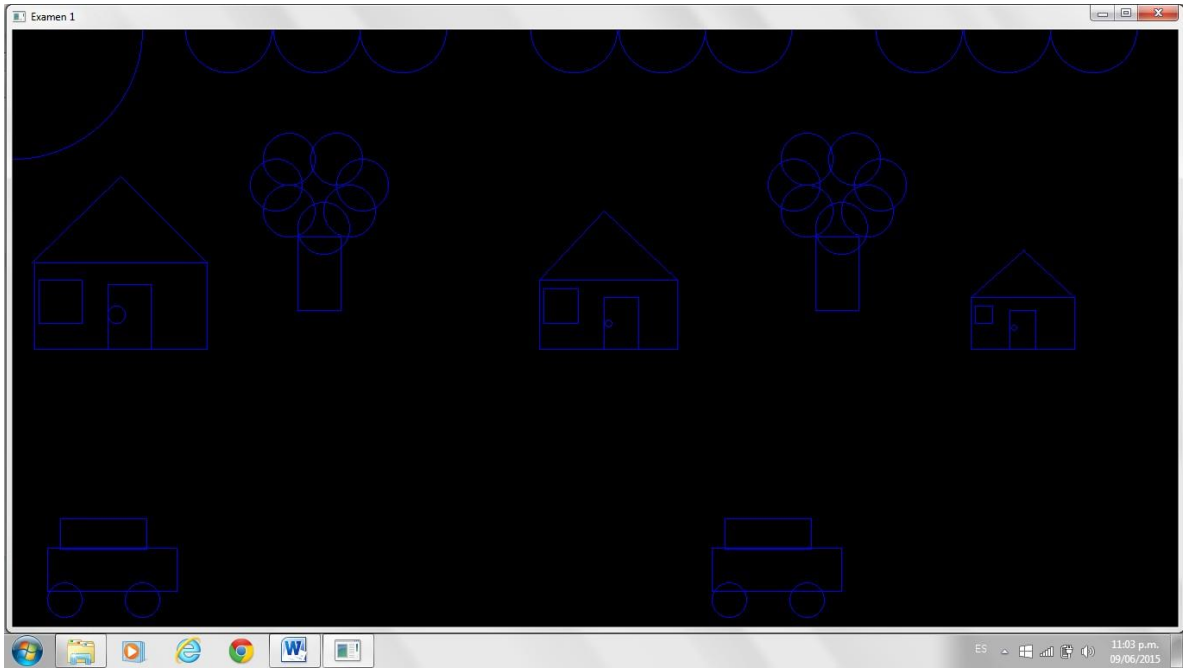
Conclusión

mat[0][0]=1;

En la práctica de estas clases se obtiene un conocimiento acerca de primitivas para graficar prácticamente desde cero, esto quiere decir que se podría migrar a cualquier otro lenguaje de programación y poder graficar con los algoritmos usados en este proyecto, también se da una mejor comprensión de las cosas que hace internamente las librerías de OpenGl

mat[0][0]=1;

El resultado obtenido es el siguiente



```
mat[0][0]=1;
```

```
mat[0][0]=1;
```

Transformaciones Lineales.

Introducción

OpenGL tiene únicamente unas pocas primitivas geométricas: puntos, líneas, polígonos. Todas ellas se describen en términos de sus respectivos vértices. Un vértice está caracterizado por 2 o 3 números en como flotante, las coordenadas cartesianas del vértice, (x, y) en 2D y (x, y, z) en 3D.

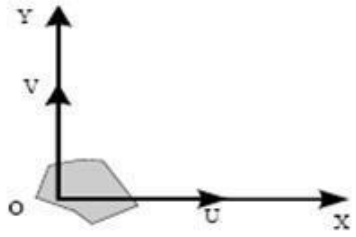
Ahora que tenemos herramientas para graficar algo más complejo en el plano 2D, haremos un escenario sencillo utilizando solo Líneas, puntos y transformaciones matriciales (traslación, escalamiento, rotación).

mat[0][0]=1;

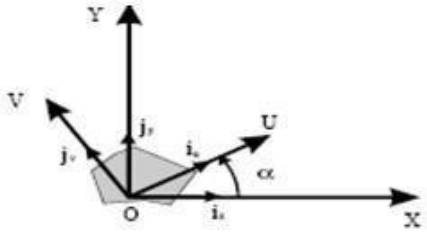
Conceptos

GL_POINTS	Nos permite iluminar un punto pixel, lo único que se necesita son para esta práctica dos parámetros, x – y, que nos representara punto de coordenadas.
	Incluiremos esta librería, para después llamar la función Sleep(), con la que podremos un intervalo de tiempo al
Traslación de Matriz	En geometría, una **traslación** es una isometría en el espacio euclídeo caracterizada por un vector $\vec{u}$, tal que, a cada punto *P* de un objeto o figura se le hace corresponder otro
Rotación de Matriz	Las matrices de rotación definer algebraicamente lo que es una rotación er un espacio 3D considerando un Angulo en el que está girando. Las matrices de importantes de notar
Escalonar Matriz	Escalonar una matriz, es prácticamente redimensionar la matriz mediante un vector dado. Esto será útil a la hora de que queramos redimensionar objetos en OpenGL
GL_LINES	Nos permite, unir dos puntos de coordenadas y así crear figuras .
glutIdleFunc()	Función de OpenGL , nos permite hacer llamado de funciones de manera automática y dentro de un bucle infinito.

Ilustración de Rotación



a)



b)

$$\mathbf{p}_{xy} = [p_x, p_y]^T = p_x \cdot \mathbf{i}_x + p_y \cdot \mathbf{j}_y$$

$$\mathbf{p}_{uv} = [p_u, p_v]^T = p_u \cdot \mathbf{i}_u + p_v \cdot \mathbf{j}_v$$

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = \mathbf{R} \begin{bmatrix} p_u \\ p_v \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} \mathbf{i}_x \mathbf{i}_u & \mathbf{i}_x \mathbf{j}_v \\ \mathbf{j}_y \mathbf{i}_u & \mathbf{j}_y \mathbf{j}_v \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} \cos\alpha & -sen\alpha \\ sen\alpha & \cos\alpha \end{bmatrix}$$

EJEMPLO DE TRASLACIÓN EN 2D

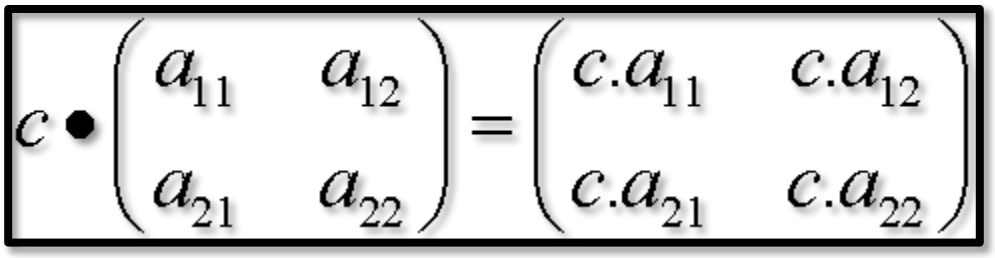
- Si queremos trasladar a dos unidades un vector en el plano obtenemos;

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Ilustración de Traslación:

$$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 0+ & 0+ & 2 \\ 0+ & 3+ & 2 \\ 0+ & 0+ & 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \\ 1 \end{bmatrix}$$

Ilustración de Escalar:



Desarrollo

Empezamos declarando una matriz global del tipo float muy importante para realizar los cálculos posteriores (float vértices[100][2]), seguido de las variables del tipo global también

, p-r-h-v-s todas inicializadas con cero.

Funciones rotación, escalamiento y traslación:

Escalamiento y traslación, son dos funciones del tipo void , dado que no necesitamos retornar ningún valor al igual que su trabajo es realizar solo cálculos para luego ser almacenadas en la matriz global vértices declarada con anterioridad. Ambas necesitan de tres parámetros "x , y , vertex" todas del tipo int . Ahora bien , en la función traslación estos parámetros serán empleados para realizar el cálculo matricial que tendremos como resultado a un x prima que será igual a la coordenadas en x (almacenada en la

Ilustración de Escalar:

matriz) + el parámetros x ; lo mismo sucede en Y. Vertex , significa vértices y sirve para condición de paro, en un ciclo For y este parámetro hace lo mismo en las tres funciones . Rotación solo necesita de 2 argumentos, un Angulo y vertex . Angulo como su nombre lo indica, es utilizado en operaciones matriciales y muy útil a la hora de rotar un objeto, es decir, cambiar en Angulo de dicho objeto.

Funciones para Plotear:

Todos los objetos que se escribieron y posteriormente se graficaron en OpenGl están estructurados por una serie de instrucciones y cada una por una función del tipo void, y ninguna de estas necesita un parámetro para trabajar. Ahora bien, todos los dibujos en sus correspondientes funciones están almacenados en la matriz global vértices , que lo único que guarda en sus filas y columnas son los vértices en el plano 2D, es por eso que la matriz se declaró de forma global y que por obvia razones estará actualizando sus valores según sea su llamado.

Entonces, para manipular los dibujos con mayor facilidad, todos los objetos se graficaron en el origen del plano cartesiano (0,0) y por decir un ejemplo, llamamos a la función pinos(), una vez sea llamada la matriz vértices actualizara sus valores donde sus valores serán los vértices en el plano, ojo , al llamar la función no graficara absolutamente nada en el programa. Para "plotear" un objeto necesitaremos un función especial llamada plot(int x) , y esta requerirá un parámetro , cual será vertex (vértices) . Prácticamente esta son las dos instrucciones que se requiere para dibujar un objeto en el plano , sin embargo, si se quiere trasladar, rotar o incluso escalar dicho objeto, antes de llamar la función plot podemos hacer el llamado de estas tres funciones con un orden jerargico que debe ser respetado, de lo contrario, no se graficara como se desea:

1. Traslación
2. Rotación
3. Escalar

Es de suma importancia tener en cuenta que la función rotación y su respectivo Angulo trabajan en contra de las manecillas del reloj, cuidado.

Animación de Objetos:

Antes de entrar en materia con la animación , cabe mencionar que la traficación de objetos puede ser empleada de forma manual, esto es, ir colocando objeto por objeto en el plano a su conveniencia, al igual que poder utilizar ciclos iterativos para realizar un dibujado de muchos objetos a la vez y al mismo tiempo manipular las transformaciones geográficas.

Bien, pasamos a la animación de objetos y lo primero que hay que tomar en cuenta son las siguientes tres funciones:

```
glutDisplayFunc(escenario);
glutIdleFunc(escenario);
glutMainLoop();
```

Esto va colocado en el main del programa, como sabemos glutDisplayFunc() crea un bucle infinito donde se estará llamando la función de manera automática que a su vez , necesita la función glutMainLoop() como regla de programación, sin embargo, en este proyecto vamos emplear la función glutIdleFunc() , esta última función nos permitirá realizar interacciones con los objetos que se explicara a continuación .

Pondremos como ejemplo la traslación del sol de este a oeste, se escribirá en los siguientes pasos:

0. La variable global r , se inicializa con cero

1. Establecemos nuestra condición de paro , en este caso if(r!=30){ -- -}
2. Llamamos al objeto circuferencia()

3. Los trasladamos al punto 17,18 en nuestro plano, que será la esquina superior derecha
4. Se hace el llamado de plot(100), el 100 indica los números de puntos que se necesita para ser graficado.
5. Incrementar r
6. Si la condición es negativa se reinicia el proceso, poniendo la variable r = 0.
7. FIN

La explicación es muy sencilla, dado que "escenario" se estará llamando en un bucle infinito por la función glutIdleFunc(escenario) las variables se actualizarán y esto permitirá que no se desborde el dibujo y cree un efecto de animación primitiva.

Configuración de Display:

En este apartado solo requerimos de tres instrucciones simples

glClearColor(1.0, 1.0, 1.0, 1.0);– Limpia la pantalla con un color blanco .

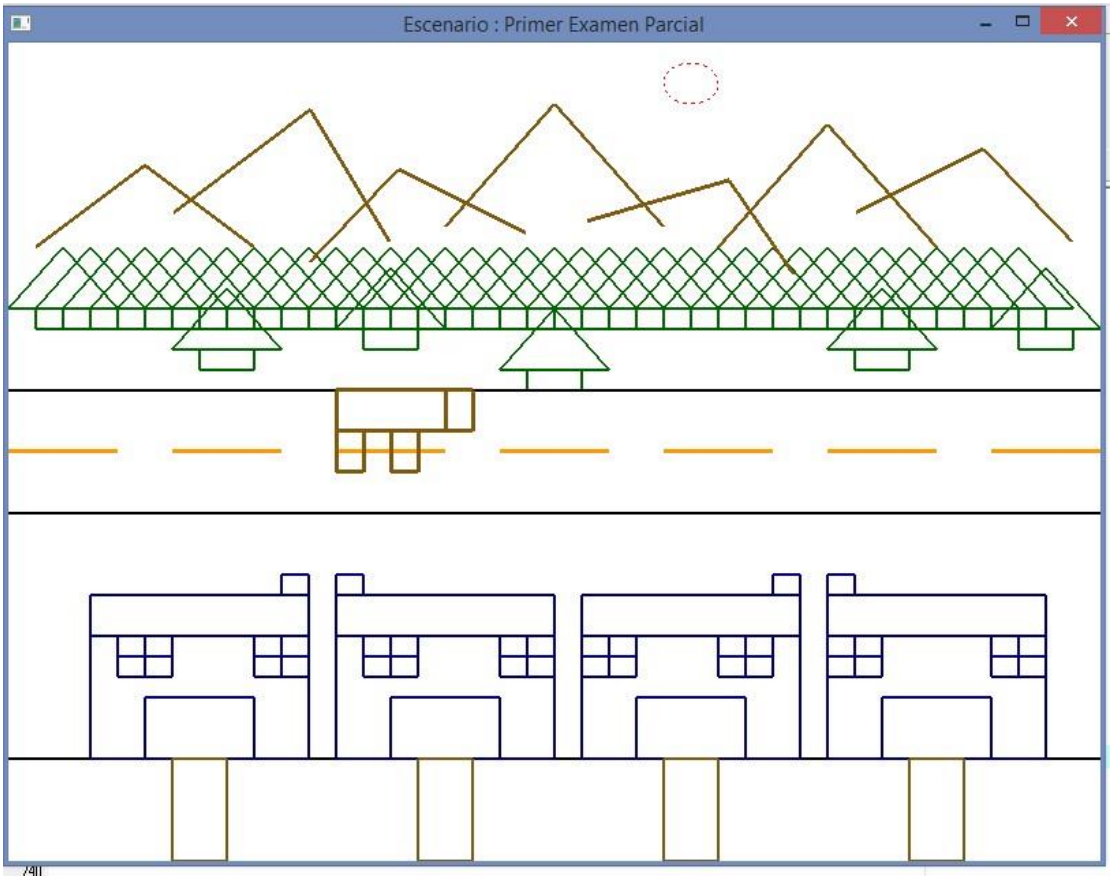
glMatrixMode(GL_PROJECTION);– Establecemos el tipo de matriz , en esta caso del tipo proyección,

gluOrtho2D(-20.0,20.0,-20.0,20.0); – Establecemos los intevalos de pantalla.

Configuracion Principal (Main) :

La configuración del Main estará dada de manera estándar, utilizamos un tamaño de 800 x 600 con glutInitWindowSize(800, 600), inicializamos el display de trabajo con glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE) , colocamos el nombre de la ventana glutCreateWindow("Dibujo de línea recta") y hacemos el llamado ala función donde se encuentra las instrucción de graficado glutDisplayFunc(escenario).

Salida de pantalla:



Bibliografía

Alcocer, R. (2001). *oocities*. Obtenido de http://www.oocities.org/valcoey/coordenadas_esfericas.html

Code, S. (09 de 06 de 2015). *lighthouse3d*. Obtenido de http://www.lighthouse3d.com/tutorials/glut-tutorial/glutpostredisplay-vs-idle-func/

Desconocido. (2013 de Enero de 26). *WikiMatematica*. Obtenido de http://www.wikimatematica.org/index.php?title=Coordenadas_Cil%C3%ADndricas_y_Esf%C3%A9ricas#Sistema_de_Coordenadas_Esfericas

E, M. R. (29 de Septiembre de 2005). *NoSoloUnix*. Obtenido de http://www.nosolounix.com/2010/05/algoritmo-bresenham-dibujar-linea.html

kegi2611. (09 de 06 de 2015). *Youtube*. Obtenido de https://www.youtube.com/watch?v=ODeiCFVh-9c

Kilgard, M. (s.f.). *https://www.opengl.org/resources/libraries/glut/spec3/node12.html*.

openboxer. (09 de 06 de 2015). *openboxer*. Obtenido de http://www.openboxer.260mb.com/asignaturas/compGrafica.php

Simon, A. (06 de 09 de 2015). *slideshar*. Obtenido de http://es.slideshare.net/karenalin/manual- de-practicas-de-opengl

Villar, M. (2009). Obtenido de http://educommons.anahuac.mx:8080/eduCommons/computacion- y-sistemas/programacion-de-graficas-computacionales/tema%202/copy_of_aprendizaje_motivacion

Wikipedia. (2015 de Mayo de 27). *Wikipedia*. Obtenido de http://es.wikipedia.org/wiki/Coordenadas_esf%C3%A9ricas

GRAFICACION

INTRODUCCIÓN

En esta ocasión el proyecto trato de la elaboración de un escenario en tercera dimensión la para la cual ahora manejamos el GluOrtho y las dimensiones en tercera dimensiones para las cuales manejamos.

Un mínimo y máximo de anchura (manejado en x)

Un mínimo y máximo de altura (manejado en y)

Un mínimo y máximo de profundidad (manejado en z)

Para una mejor observación del escenario utilizamos una función con la cual nos podemos desplazarnos de izquierda a derecha y de arriba abajo y viceversa en la proyección.

FUNCIÓN PARA DESPLAZARNOS EN LA PANTALLA

```
double rotate_y=0;
double rotate_x=0;


void specialKeys( int key, int x, int y ) {

  //  Flecha derecha: aumentar rotación 5 grados
  if (key == GLUT_KEY_RIGHT)
    rotate_y += 2;

  //  Flecha izquierda: disminuir rotación 5 grados
  else if (key == GLUT_KEY_LEFT)
    rotate_y -= 2;

  else if (key == GLUT_KEY_UP)
    rotate_x += 2;

  else if (key == GLUT_KEY_DOWN)
    rotate_x -= 2;

  //  Solicitar actualización de visualización
  glutPostRedisplay();

}
```

Para realizar esta función manejamos dos variables del tipo doublé las cuales iniciamos en 0.

Manejamos un if con sus elses con los cuales íbamos especificando el desplazamiento conforme a la tecla oprimida por ejemplo:

Izquierda -2 y Derecha +2

Arriba +2 y Abajo -2

MATRICES DE TRANSFORMACION

Rotación----------glRotate

Traslación----------glTraslated

Escalacion---------- glScaled

Las cuales nos sirvieron para dar el efecto de movimiento que en este caso yo las use para la simulación en Mario y su hermano Luigi para la simulación que se movían respecto al escenario estos dos los simule con dos cuadrados que contenían texturas adentro.

PILA DE MATRICES

Poner matriz----------PushMatrix

Quitar matriz----------PopMatrix

Para manejar correctamente las traslaciones y rotaciones independientes en cada figura fue necesario el uso de la implementación de una matriz a cada figura al cual se le aplicara una traslación o rotación para que solo se vea afectada dicha figura.

En esta ocasión se nos permitió usar funciones de opengl para la creación de figuras.

Cuadriláteros----------GLQUADS

Triángulos----------GLTRIANGLES

Líneas----------GLLINES

Predefinidos 3D

Esferas----------GLUTSPHERES

Conos---------- GLUTSOLIDCONE

Cubos---------- GLUTSOLIDCUBE

Torus---------- GLUTSSOLIDTORUS

```
GLuint textures[NTextures];

char *texturefiles[] = {
    "textura1.png",
    "textura2.png",
    "textura3.png",
    "textura4.png",
    "fondoa.png",
    "fondomm.png",
    "textura6.png",
    "textura7.png",
    "mario.png",
    "textura8.png",
    "textura9.png",
    "textura10.png",
    "luigi.png",
    "textura12.png"
};

FIBITMAP  *loadImage(const char *filename){
   FIBITMAP *dib1 = NULL;

  FREE_IMAGE_FORMAT fif = FreeImage_GetFIFFromFilename(filename);

  dib1 = FreeImage_Load(fif, filename, JPEG_DEFAULT);
  if (!dib1)
    {
      //std::cerr << "Erreur ouverture d\'image" << std::endl;
      exit (0);
    }
  return dib1;
}
```

TEXTURAS

Para el manejo de texturas maneje la librería de texturas FreeImage para la cual fue necesario el uso de una función en específico de esta librería.

Para el manejo de texturas fue necesario primero definir el tamaño del arreglo que contendría las estructuras en este caso fue NTextures.

Cada imagen que íbamos insertando la teníamos que meter dentro de dicho arreglo de manera de cadena.

En la función load image se verificaba que no hubiese ningún error en la carga ósea que se encontrara satisfactoriamente el formato y se cargara a la variable dib1.

Si no se generaba ningún error retornaba dicha variable.

Función LoadTexture

```
  height, width;

  RGBQUAD rgbquad;


   FREE_IMAGE_TYPE type;
  BITMAPINFOHEADER *header;

   type = FreeImage_GetImageType(dib1);


   height = FreeImage_GetHeight(dib1);
  width = FreeImage_GetWidth(dib1);

  header = FreeImage_GetInfoHeader(dib1);
int scanLineWidh = ((3*width)%4 == 0) ? 3*width : ((3*width)/4)*4+4;
  unsigned char * texels= (GLubyte*)calloc(height*scanLineWidh, sizeof(GLubyte));
  for (x=0 ; x<width ; x++)
    for (y=0 ; y<height; y++)
    {
      FreeImage_GetPixelColor(dib1,x,y,&rgbquad);

      texels[(y*scanLineWidh+3*x)]=((GLubyte*)&rgbquad)[2];
      texels[(y*scanLineWidh+3*x)+1]=((GLubyte*)&rgbquad)[1];
      texels[(y*scanLineWidh+3*x)+2]=((GLubyte*)&rgbquad)[0];
    }

      glGenTextures (1, &tex_id);
      glBindTexture (GL_TEXTURE_2D, tex_id);

      glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
      glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);


      glTexImage2D (GL_TEXTURE_2D, 0, GL_RGB,
            width, height, 0, GL_RGB,
            GL_UNSIGNED_BYTE, texels);
      free(texels);

  return tex_id;
```

Para la carga de varias texturas fue necesario el uso de esta función ya que la otra función por si sola generaba errores tales como:

Mala carga en los colores ya que colocaba todo en un tono verdoso fuera cual fuera la textura.

Para la función de cargar textura solo se manejaba para cargar las dimensiones de la textura por ejemplo ancho y alto, además de colocar correctamente los colores RGB.

FUNCION CARGARTEXTURAS

```
void cargarTexturas()
{
    int i;
    FIBITMAP *dib;
    for(i=0; i<NTextures; i++)
    {
        dib=loadImage(texturefiles[i]);
        textures[i]=loadTexture(dib);
    }
}
```

Para cargar las diversas texturas fue necesario de la función cargar texturas la cual consistía de un if el cual va verificando textura por textura para su correcta carga en el programa.

Para ir creando mis figuras fue necesario implementar funciones las cuales iba llamando como fuera necesario ya que cada objeto creado representaba demasiado código por eso fue por lo que decidí que era mejor colocarlo por separado en vez de todo junto para así también a la hora de

colocarlo en mi parte principal no me generara problemas y además tuviera mejor manejo de el.

Para que no tuviera problemas al importar mis funciones que contenían dichos objetos tuve que usar una nueva función la cual fue.

glDepthFunc que sirve para especificar valores en especificación de profundidad esto es para que al momento de trabajarlos no nos dé el problema de que tome representación 2d en vez de 3d además de que especificamos los parámetro en los cuales vamos a trabajar.

```
void pino(void){

    glColor3f(21.0/255.0,211.0/255.0,12.0/255.0);
    glPushMatrix();
    glRotated(-90,1,0,0);
    glutSolidCone(2,4,30,40);
    glPopMatrix();

    glColor3f(18.0/255.0,184.0/255.0,10.0/255.0);
    glPushMatrix();
    glTranslated(0,-1,0);
    glRotated(-90,1,0,0);
    glutSolidCone(2.2,4.5,30,40);
    glPopMatrix();

    glColor3f(17.0/255.0,167.0/255.0,10.0/255.0);
    glPushMatrix();
    glTranslated(0,-2,0);
    glRotated(-90,1,0,0);
    glutSolidCone(2.4,4.5,30,40);
    glPopMatrix();

    glColor3f(22.0/255.0,214.0/255.0,12.0/255.0);
    glPushMatrix();
    glTranslated(0,1,0);
    glRotated(-90,1,0,0);
    glutSolidCone(1.8,4,30,40);
    glPopMatrix();

    glColor3f(23.0/255.0,223.0/255.0,13.0/255.0);
    glPushMatrix();
    glTranslated(0,2,0);
    glRotated(-90,1,0,0);
    glutSolidCone(1.6,4,30,40);
    glPopMatrix();
```

FUNCIÓN PINO

Esta función la utilice creando varios conos para representar la parte de arriba del pino, algunas esferas para representar la parte media de que dejaban estas y por ultimo solo utilice la función glcube para deformarla y representar el tronco del pino.

```
void nube(){


    glColor3f(210.0/255.0,210.0/255.0,210.0/255.0);
    glPushMatrix();
    glTranslated(-13,10,-5);
    glutSolidSphere(1,30,40);
    glPopMatrix();

    glColor3f(210.0/255.0,210.0/255.0,210.0/255.0);
    glPushMatrix();
    glTranslated(-13,10,-5);
    glScaled(1.5,1,1);
    glutSolidSphere(1,30,40);
    glPopMatrix();

    glColor3f(210.0/255.0,210.0/255.0,210.0/255.0);
    glPushMatrix();
    glTranslated(-13,10,-5);
    glScaled(1,1,1.5);
    glutSolidSphere(1,30,40);
    glPopMatrix();

    glColor3f(210.0/255.0,210.0/255.0,210.0/255.0);
    glPushMatrix();
    glTranslated(-13,10,-5);
    glScaled(1,1.5,1);
    glutSolidSphere(1,30,40);
    glPopMatrix();

    glColor3f(210.0/255.0,210.0/255.0,210.0/255.0);
    glPushMatrix();
    glTranslated(-13,10,-5);
    glRotated(-90,0,0,1);
    glScaled(1.5,1,1);
    glutSolidSphere(1,30,40);
    glPopMatrix();

}
```

Función nube.

Para la creación de las nubes solamente fue necesario la creación de varias esferas las cuales fueron deformadas para simular las nubes

Explicación de los colores

Ahora como no pude texturizar la mayoría de mis figuras use funciones del glut pero estas ya no las pude texturizar con las funciones que ya había realizado en el proyecto anterior asi que para un mejor manejo de los colores fue necesario el uso de el circulo cromático.

Pero como opengl te los maneja en escala de 0 a 1 tienes que elegir el color y sus valores para después dividirlo entre 255 ya que son 255 varianzas.

```
void silla(void){
    glMatrixMode(GL_MODELVIEW);
    glColor3f(194.0/255.0,171.0/255.0,92.0/255.0);

    glPushMatrix();
    glTranslated(7,-3,-1.3);
    glScaled(.03,1,.03);
    glutSolidCube(4);
    glPopMatrix();

    glPushMatrix();
    glTranslated(8,-3,-1.3);
    glScaled(.03,1,.03);
    glutSolidCube(4);
    glPopMatrix();

    glPushMatrix();
    glTranslated(7,-3,-.8);
    glScaled(.05,.7,.05);
    glutSolidCube(2);
    glPopMatrix();

    glPushMatrix();
    glTranslated(8,-3,-.8);
    glScaled(.05,.7,.05);
    glutSolidCube(2);
    glPopMatrix();

    glPushMatrix();
    glTranslated(7.2,-2.2,-.8);
    glScaled(.5,.1,.3);
    glutSolidCube(2);
    glPopMatrix();

    glPushMatrix();
    glTranslated(7.45,-1,-1.3);
    glScaled(.5,.1,.1);
    glutSolidCube(2);
    glPopMatrix();
}
```

función silla

para la función silla nuevamente fue necesario el uso de glcube y deformaciones.

En este caso maneje varios gl cube uno por cada parte de la silla por ejemplo

Respaldo

Asiento

Uno por pata

Los cuales los fui deformando para que tuvieran mejores proporciones y en verdad diera el efecto de una silla.

Descansadora y Mesa

Estas dos funciones fueron creadas prácticamente con la misma lógica y los mismos pasos.

Ya que también únicamente manejamos glcubes para ellos además de que por supuesto en diferentes escalas y diferente colocación.

Iluminacion

```
void keyboard(unsigned char key, int x, int y)
{
    switch(key)
    {
    case 'p':
    case 'P':
      //inicializamos el ambiente
    GLfloat light_ambient[] = { 0.75, 0.75, 0.75, 1.0 }; //luz ambiente
    GLfloat light_diffuse[] = { 1.0, 0.0, 0.0, 1.0 }; //luz difusa
    GLfloat light_specular[] = { 1.0, 0.0, 1.0, 1.0 }; //luz especular
    GLfloat light_position[] = { 0.0, 1.0, 1.0, 0.0 }; //posicion de la luz

    //selecciona el modo de iluminacion activo
    glLightfv (GL_LIGHT0, GL_AMBIENT, light_ambient);
    glLightfv (GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv (GL_LIGHT0, GL_SPECULAR, light_specular);
    glLightfv (GL_LIGHT0, GL_POSITION, light_position);

    //activar los calculos de la iluminacion
    glEnable (GL_LIGHTING);
    glEnable (GL_LIGHT0);
    glDepthFunc(GL_LESS);
    glEnable(GL_DEPTH_TEST);

      break;


    }

}
```

Para el caso de iluminación me ocurrio un gran problema al querer iluminar un solo objeto se iluminaba todo.

Intente evitar esto con pilas de matrices y de muchos otros modos pero todo fallo.

Decidi mejor meter en el entorno una opcion en la cual se lograra ver un efecto de oscuridad al oprimir la tecla p o P para esto definimos:

Todo este código lo meti en la función keyboard donde defini:

Luz ambiente

Luz difusa

Luz especular

Y la posición de la luz.

Solo en caso de oprimir dicha tecla “p” al actualizar o mover con las SpecialKeys() se actualizara dando el ya mencionado efecto al entorno.

CONCEPTOS DESARROLLADOS

En el anterior proyecto había comentado que se había realizado como una introducción a lo que era la introducción a la realización de los videojuegos pero en esta ocasión ya fue más apegado a los videojuegos pero ahora en 3ra dimensión que es lo del día de hoy.

Aunque hay muchos problemas en lo que es cuestión de orientación ya que si ya era un tanto complicada la orientación en dos dimensiones pues ahora en tres es mucho más complicado aunque poco a poco te vas orientando y encontrándole el modo

EXPERIMENTOS

Ahora si tuve demasiados problemas por ejemplo al inicio pensaba en hacer otro escenario que pensé que era más sencillo pero al momento de empezarlo me di cuenta que era mucho más complicado porque era realizar varias pirámides las cuales no pude elaborar correctamente.

También como mejor decidí hacer un escenario sencillo y con cuadriláteros 3d tuve el gran problema de no poder texturizar ya que el programa no deja texturizar cosas en 3d predefinidas en Opengl por ejemplo en el uso de funciones como:

-Esferas

-Conos

-Cubos

-Torus

También me surgió el problema en la función del idle(); la cual es necesaria para ir actualizando la pantalla ya que por lo mismo de los botones de desplazamiento en dimensiones con la función SpecialKeys() ya que con el idle actualizaba tanto que no paraba

de girar por la misma razón me fue imposible colocar animaciones ya que alborotaba todo mi escenario.

Para la realización de la mayoría de mis figuras fue necesario la deformación de las funciones 3d aunque en el mayor caso fueron cubos creados por las función

GLUTSOLIDCUBE

CONCLUSIÓN

En este caso aprendí a manejar el entorno 3d y a orientarme en dicho entorno, además de aprender la importancia de este entorno en la animación.

Aprendí a manejar las funciones 3d de OpenGL las cuales me fueron de mucha utilidad además de como deformarlas para crear diferentes formas y figuras.

También aprendí a manejar perspectivas e iluminación.

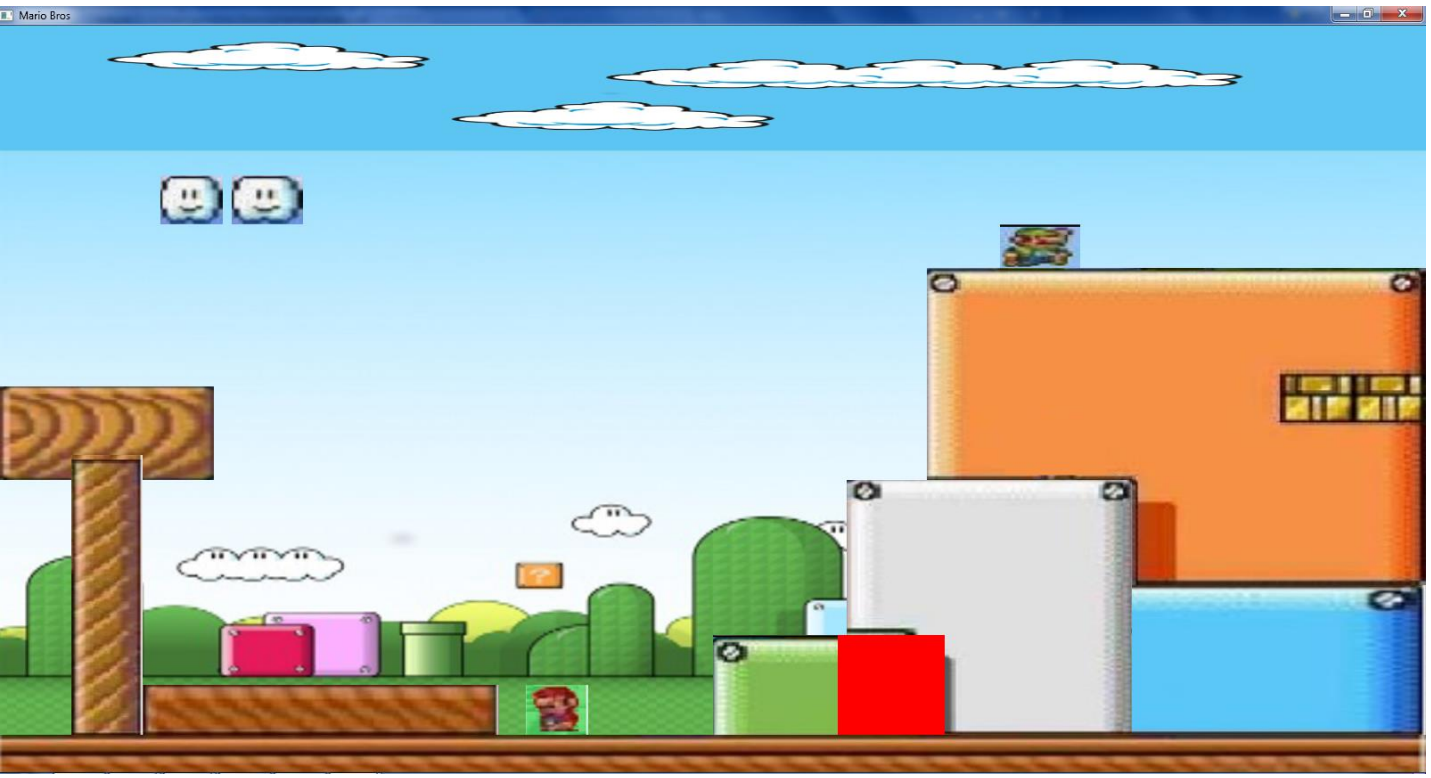
GRAFICACION

PRIMAVERA-2015

ARCOS COCONI JOSE FRANCISCO

INTRODUCCIÓN

El escenario que yo elegí fue una combinación entre varios escenarios del juego 2d “Súper Mario Bros” el cual consiste de cuadros.



Tuve que combinar varios escenarios para lograr insertar un fondo para este fondo el cual dividí en 3 partes.

La parte de arriba:

Consiste en un cielo con nubes las cuales se irán moviendo.

La parte de en medio:

Es la más extensa consiste en un fondo el cual por la parte de atrás se visualizaran plantas, tubos y cuadros típicos de este juego para simular el fondo.

La parte de abajo:

Únicamente consiste en el suelo y su texturizacion la cual es café esta únicamente con cuadriláteros los cuales son rectángulos extendidos.

Para el desarrollo de esta práctica implementamos los conocimientos desarrollados durante este parcial los cuales fueron el uso de las funciones que contienen a las matrices de.

MATRICES DE TRANSFORMACION

Rotación----------glRotate

Traslación----------glTraslated

Las cuales nos sirvieron para dar el efecto de movimiento que en este caso yo las use para la simulación en Mario y su hermano Luigi para la simulación que se movían respecto al escenario estos dos los simule con dos cuadrados que contenían texturas adentro.

PILA DE MATRICES

Poner matriz----------PushMatrix

Quitar matriz----------PopMatrix

Para manejar correctamente las traslaciones y rotaciones independientes en cada figura fue necesario el uso de la implementación de una matriz a cada figura al cual se le aplicara una traslación o rotación para que solo se vea afectada dicha figura.

En esta ocasión se nos permitió usar funciones de opengl para la creación de figuras.

Cuadriláteros----------GLQUADS

Triángulos----------GLTRIANGLES

Líneas----------GLLINES

Esferas----------GLUTSPHERES

En esta ocasión como el escenario que yo elegí estaba conformada por puros cuadros los cuales fueron creados con la función glBegin(GL_QUADS).

Implemente varios cuadros y los fui acomodando conforme al escenario para posteriormente texturizarlos.

Para facilitar el uso de estos además de obtener una mejor ubicación fue necesario el uso de paint como herramienta de coordenadas.

TEXTURAS

Para el manejo de texturas maneje la librería de texturas FreeImage para la cual fue necesario el uso de una función en específico de esta librería.

```
GLfloat angnubes=0.0f;

GLuint textures[NTextures];

char *texturefiles[] = {
    "textura1.png",
    "textura2.png",
    "textura3.png",
    "textura4.png",
    "fondoa.png",
    "fondomm.png",
    "textura6.png",
    "textura7.png",
    "mario.png",
    "textura8.png",
    "textura9.png",
    "textura10.png",
    "luigi.png",
    "textura12.png"
};

FIBITMAP  *loadImage(const char *filename){
   FIBITMAP *dib1 = NULL;

  FREE_IMAGE_FORMAT fif = FreeImage_GetFIFFromFilename(filename);

  dib1 = FreeImage_Load(fif, filename, JPEG_DEFAULT);
  if (!dib1)
    {
      //std::cerr << "Erreur ouverture d\'image" << std::endl;
      exit (0);
    }
  return dib1;
}
```

Para el manejo de texturas fue necesario primero definir el tamaño del arreglo que contendría las estructuras en este caso fue NTextures.

Cada imagen que íbamos insertando la teníamos que meter dentro de dicho arreglo de manera de cadena.

En la función load image se verificaba que no hubiese ningún error en la carga osea que se encontrara satisfactoriamente el formato y se cargara a la variable dib1.

Si no se generaba ningún error retornaba dicha variable.

Función LoadTexture

```
  height, width;

  RGBQUAD rgbquad;


   FREE_IMAGE_TYPE type;
  BITMAPINFOHEADER *header;

   type = FreeImage_GetImageType(dib1);


   height = FreeImage_GetHeight(dib1);
  width = FreeImage_GetWidth(dib1);

  header = FreeImage_GetInfoHeader(dib1);
int scanLineWidh = ((3*width)%4 == 0) ? 3*width : ((3*width)/4)*4+4;
  unsigned char * texels= (GLubyte*)calloc(height*scanLineWidh, sizeof(GLubyte));
  for (x=0 ; x<width ; x++)
    for (y=0 ; y<height; y++)
    {
      FreeImage_GetPixelColor(dib1,x,y,&rgbquad);

      texels[(y*scanLineWidh+3*x)]=((GLubyte*)&rgbquad)[2];
      texels[(y*scanLineWidh+3*x)+1]=((GLubyte*)&rgbquad)[1];
      texels[(y*scanLineWidh+3*x)+2]=((GLubyte*)&rgbquad)[0];
    }

      glGenTextures (1, &tex_id);
      glBindTexture (GL_TEXTURE_2D, tex_id);

      glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
      glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);


      glTexImage2D (GL_TEXTURE_2D, 0, GL_RGB,
            width, height, 0, GL_RGB,
            GL_UNSIGNED_BYTE, texels);
      free(texels);

  return tex_id;
```

Para la carga de varias texturas fue necesario el uso de esta función ya que la otra función por si sola generaba errores tales como:

Mala carga en los colores ya que colocaba todo en un tono verdoso fuera cual fuera la textura.

Para la función de cargar textura solo se manejaba para cargar las dimensiones de la textura por ejemplo ancho y alto, además de colocar correctamente los colores RGB.

FUNCION CARGARTEXTURAS

```
void cargarTexturas()
{
    int i;
    FIBITMAP *dib;
    for(i=0; i<NTextures; i++)
    {
        dib=loadImage(texturefiles[i]);
        textures[i]=loadTexture(dib);
    }
}
```

Para cargar las diversas texturas fue necesario de la función cargar texturas la cual consistía de un if el cual va verificando textura por textura para su correcta carga en el programa.

EJEMPLO DE CARGA DE TEXTURA

```
//fondo en medio

    glEnable(GL_TEXTURE_2D);
 glBindTexture(GL_TEXTURE_2D, textures[5]);
 glColor3d(1,1,1);
 glBegin(GL_QUADS);

 glColor3f(1.0,1.0,1.0);
 glTexCoord2f(0.0,1.0);
 glVertex2f(-40.0,20.0);

 glColor3f(1.0,1.0,1.0);
 glTexCoord2f(0.0,0.0);
 glVertex2f(-40,-28.0);

 glColor3f(1.0,1.0,1.0);
 glTexCoord2f(1.0,0.0);
 glVertex2f(40,-28.0);

 glColor3f(1.0,1.0,1.0);
 glTexCoord2f(1.0,1.0);
 glVertex2f(40.0,20.0);
 glEnd();
glDisable(GL_TEXTURE_2D);
```

Para cargar una textura es necesaria la función habilitar textura la cual es glEnable(GL_TEXTURE_2D);

Después manejamos la función que manda a pintar la textura la cual es glBindTexture(GL_TEXTURE_2D, y el arreglo con la posición de la texura);

Para la correcta carga de las texturas también necesitamos tener un buen punto de proyección el cual se va colocando en cada punto o coordenada(x,y) para la correcta visualización en la cual usamos la función

glTexCoord() la cual proyecta la textura depende a la figura.

Nuevamente una vez colocados todos los puntos y coordenadas de proyección mandamos a dibujar con la función glEnd();

Y por ultimo desactivamos la textura para que solo afecte la figura correspondiente y esto lo hacemos con la función glDisable(GL_TEXTURE_2D);

MOVIMIENTO EN LAS FIGURAS

```
//mario
glPushMatrix();
 glEnable(GL_TEXTURE_2D);
 glBindTexture(GL_TEXTURE_2D, textures[8]);
  glColor3f(1,1,1);

 glTranslated(angmario,0,0);
 glBegin(GL_QUADS);

 glColor3f(1.0,1.0,1.0);
 glTexCoord2f(0.0,1.0);
 glVertex2f(-12,-23);

 glColor3f(1.0,1.0,1.0);
 glTexCoord2f(0.0,0.0);
 glVertex2f(-12,-27);

 glColor3f(1.0,1.0,1.0);
 glTexCoord2f(1.0,0.0);
 glVertex2f(-8.5,-27);

 glColor3f(1.0,1.0,1.0);
 glTexCoord2f(1.0,1.0);
 glVertex2f(-8.5,-23);


 glEnd();
 glDisable(GL_TEXTURE_2D);
glPopMatrix();
angmario+=0.5f;
```

Para el movimiento de la figura que en este caso es el personaje principal mario fue necesario manejar una variable la cual iba incrementando hasta salir del escenario o llegar a cierto punto.

Utilice la función de trasladar la cual es glTranslated; hasta pero me genero un problema que únicamente al hacer grande y chica la pantalla repetidamente es la única manera para visualizar dicho movimiento.

También como se observa y ya había comentado fue necesario implementar dicho código dentro de una pila de matrices para no afectar las demás partes del escenario.

Al final únicamente solo se aplica el incremento a la variable de traslación la cual maneje uno diferente por cada objeto que le aplique algún movimiento.

CONCEPTOS DESARROLLADOS

Esta actividad fue como un viaje a profundidad a lo que son los principios de los videojuegos o aplicaciones desarrolladas con herramientas parecidas a esta.

Me sirvió para entender cómo interpreta OpenGl lo que el programador desea hacer y como desea interactuar con las figuras.

En texturizacion para empezar a desarrollar practica sobre este tema el cual tiene muchos aspectos a tomar interés por ejemplo tipo y atributo de la imagen y sus colores RGB.

EXPERIMENTOS

Como no sabia como colocar la perspectiva para la visualización de las texturas tuve que ir probando varias veces hasta entender como funcionaba.

En la carga de texturas tuve que ver otras fuentes sobre FreeImage para la carga de varias texturas y sus entintados.

Uno de los problemas que tuve en la realización del proyecto es que como estoy acostumbrado a usar una orientación de varios pixeles al momento de texturizar no me respetaba dichas texturas ocacionandome un efecto de deformación en la textura.

CONCLUSIÓN

Aprendí a manejar y explotar los atributos de lo visto durante el curso y más que nada durante este parcial en el cual vimos todo lo que aplique en este proyecto.

Texturizacion, traslación, rotación etc.

Me empiezo a dar cuenta sobre como desarrollan las aplicaciones graficas las personas que se dedican a esto además de los problemas que se les presenta.

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

Segundo Examen de Graficación: Crear un escenario

Usando las primitivas de OpenGL.

Profesor:

Dr. Iván Olmos Pineda

Materia:

Graficación

Alumna:

Torres Juan Amalia (201319421)

Fecha: 24-03-2015

1

Objetivo: El proposito de este segundo examen parcial es la construccion de un escenario usando las primitivas de openGL para crear texturas en objetos así como crear efectos de movimiento a través de las funciones de rotacion y traslacion, pero es este trabajo a diferencia del primer examen se realizan estos efectos usando las instrucciones glTaslate y glRotatef.

Descripción de las trayectorias: Prara describir las trayectorias de los objetos se usaron varias funciones, la funcion que describe el movimiento en linea recta para la traslacion del sol y la de un automovil. Para la traslacion de un barco se us la trayectoria de una elipse en coordenadas polares. Para esto se evaluan los puntos en X y en Y en esta funcion y se pasan como parámetros en la funcion traslate, cada vez que se dibuja el objeto, por lo que al ejecutar el programa se tiene el efecto de traslacion en linea curva.

Funciones implementadas:

Se crearon 5 funciones las cuales permiten el trazo de los objetos. En la funcion barco, se realiza el trazo del barco usando cuadrados y triangulos, además se realiz un mapeo en las distintas partes de la figura (base,mastil y velas) y las imagenes de las texturas para lograr el efecto visual de que el barco creado es de madera.

En la funcion display1 se crea la representacion de una cabaña, la cual fue construida usando las funciones glCuads y glTriangles, para el efecto de perpectiva se rotaron algunas figuras para dar el efecto de profundidad, se usaron tres texturas, las cuales permiten crear el efecto madera en las paredes de la cabaña y una textura diferente para las ventanas y la puerta. Para el techo de la cabaña se usó textura tipo paja. Se usaron las instrucciones glTraslate y glRotate y con el fin de no cambiar el punto de origen se usaron las instrucciones glPushMatrix y glPopMatrix.

Para el trazado del sol y con el fin de lograr darle volumen y una textura brillante se realizaron cambios en la funcion de inicializacion, las instruciones qusadas permiten definir un color para los tipos de luz del objeto y la posicion de la luz para darle el efecto mas brillante en la parte superior del sol y asì creal el efecto mas real de volumen.

Para la costrucción del autómovil se crearon varias funciones para crear las diferentes partes del coche las cuales usan distintas texturas. Se implement una función para el cuerpo del coche y se mapeo esta figura con una imagen de textura azul platinada, para dar el efecto

final. Para las ventanas se us una textura tipo vidrio y para el dibujo de las llantas se usó la

instruccion glSolidSphere para la figura de la llanta y glCUADS para los rayos de las llantas. La lantas del coche van rotando a una cierta velocidad cada vez que avanza el automivil, por lo cual hay una combinacion de dos movimientos, una traslacion a medida que avanza el coche y una rotación respecto al automivil en cada momento.

Para el trazo de los molinos de viento, se implement una función principal que dibuja las

aspas y la base y la parte superior, en cada una de estas partes se usó una textura diferente, para dar un efecto más realista, en la base se usó una textura tipo piedra, en la parte superior una textura tipo paja color café y en las aspas una tectura de malla.

En la función display principal se hace el llamado de estas 5 funciones y con el fin de crear efectos de movimiento diversos se modifican los parametros que describen los diversos estados del objeto, a traves de los parametros de angulos y de posicion donde se realizan incrementos constantes.

En la funcion Visualizacion se activa la matriz de proyeccion y se define el tipo de proyeccion a usar.Se carga en la matriz de proyeccion la matriz identidad, para resetearla y poder trabajar

sobre ella, se redimenciona la ventana, los parámetros con los que se trabajan son ancho y alto. Se agregaron funciones adicionales a través de la funcion mouse que indica que al pulsar el boton derecho del mouse hacer un zoom en una parte específica de la imagen y con el botón izquierdo hacer más pequeña cierta parte de la imagen.

Para controlar otras teclas adicionales se implementa la función llamada teclado donde de indica que si se pulsa las letras p o q para retroceder el objeto o para avanzar hacia adelante.

Finalmente se emple una textura para el fondo del escenario para dar el efecto de un lugar

en el campo con un río y montañas. Sobre esta textura se superponen todos los objetos creados mencionados anteriormente para lograr el efecto final que se muestra acontinuacion.

Resultado:

References

[1] Graficos por computadoras con OpenGL, Heam Baker, Prentice Hall, 2005.

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Ingeniería en Ciencias de la Computación

Graficación

Profesor: Iván Olmos Pineda

Examen 3

Alumna: Alma Amaranta Pérez Cantor

Sección 101

Marzo 2015

Para comenzar a desarrollar este trabajo lo primero que hice fue inicializar el ambiente gráfico, ya que el objetivo de este trabajo consistía en crear un escenario en 3D use en fueron instrucciones como glEnable(GL_DEPTH_TEST) para activar la profundidad en el escenario además de gluPerspective(), glMatrixMode(GL_MODELVIEW), gluLookAt() y glCullFace(GL_FRONT_AND_BACK);

Posteriormente con la primitiva GL_QUADS comencé a crear la estructura de la casa, cada pared se dibujó como un cuadrado. Después lo que hice fue crear las figuras que están en el escenario, es decir, un método para crear un auto, una casa, una cama y un sillón. Para cada uno de ellos ocupe ciertas figuras propias de OpenGL.

Para crear la casa use GL_QUADS y GL_POLYGON básicamente, después hice los métodos para dibujar cada objeto y poder manejarlos con instrucciones como glTranslated, glScalef y glRotated. Las figuras se crean de modo unitario para que su manejo sea más sencillo.

Finalmente el automóvil se crea con GL_POLYGON, a través de él lo único que hice fue calcular los puntos que serían los vértices de la figura, el problema que tuve con el auto fue crear las llantas ya que quise crearlas con glutSolidSphere() sin embargo la figura no se creaba o no se lograba visualizar en la pantalla, intente hacer pruebas primero con la esfera sola, pero también me mostraba problemas al momento de querer trasladarla. Lo que decidí hacer fue crear una clase propia que se llama esfera la cual crea un polígono con la instrucción GL_POLYGON para que pudiera estar coloreada del color que yo eligiera, los vértices de esta instrucción están dados a través de una relación entre senos y cosenos, por lo cual también use la librería math.h.

Para ir formando el escenario se tuvo que ir colocando los elementos desde el fondo hasta el elemento que está en el plano más al frente. Tuve que tener cuidado con no traslapar los elementos por ejemplo las ventanas con la pared de la casa por que de ser así entonces no se vería alguno de los elementos.

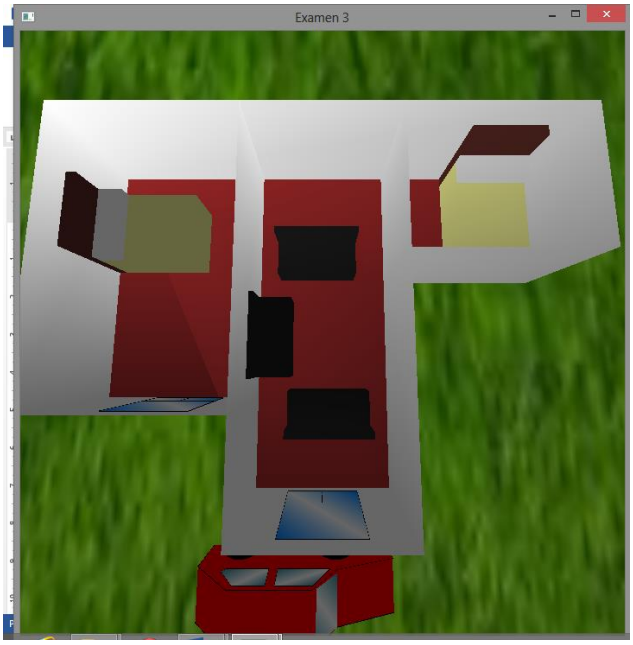
El automóvil se logran moverse a través del cambio de sus variables, en un pequeño aumento o decremento de ella y que mediante la función idle se logra guardar cada uno de esos cambios para poder lograr el efecto de movimiento.

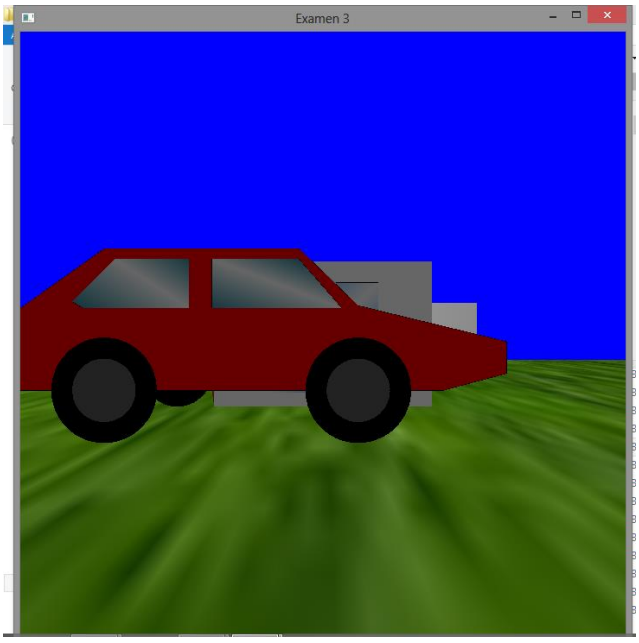
Para la carga de texturas se utilizaron la librería glaux, para lo cual utilice varias funciones necesarias para cargar las imágenes, el problema que causaron las texturas fue que me cambiaban los colores de algunas figuras por lo cual tuve que utilizar la instrucción glNormal3f() y así mantener los colores.

El piso que es el elemento que tiene la texturas se creo con un cuadrado al que se le mapearon los puntos.

Para realizar los movimientos en el escenario, ocupe la función keyboard() y reshape(). Para poder ver todo el escenario se rota con las teclas.

Resultados:





El escenario quedo de la manera anterior. Se puede visualizar desde lo alto y rotar hacia los lados.

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Ingeniería en Ciencias de la Computación

Graficación

Profesor: Iván Olmos Pineda

Examen 2

Alumna: Alma Amaranta Pérez Cantor

Sección 101

Marzo 2015

Para comenzar a desarrollar este trabajo lo primero que hice fue inicializar el ambiente gráfico, ya que el trabajo consistía en crear un escenario en 2D use gluOrtho2D(xmin, xmax, ymin, ymax) e inicialice cada uno de sus argumentos use glMatrixMode(GL_PROJECTION) y además glLoadIdentity() para realizar las transformaciones con las primitivas de OpenGL.

Posteriormente lo que hice fue crear los fondos de colores, es decir, el pasto, el cielo y el camino con la primitiva GL_QUADS. Después lo que hice fue crear las figuras que están en el escenario, es decir, un método para crear un auto, un árbol y una casa. Para cada uno de ellos ocupe ciertas figuras propias de OpenGL.

Para crear la casa use GL_QUADS, GL_POLYGON y GL_TRIANGLES básicamente, primero tuve que hace el cuadrilátero principal que sería la mayor parte de la casa y dependiendo de qué tan al fondo esta cada componente fui creándola. Los parámetros que se le pasan a esta función son tres de tipo float que sirven para establecer los colores de la casa únicamente.

La casa tiene cuatro ventanas las cuales opte por construir a través de otro método, para cuidar los detalles, y con el uso de glTranslatef() únicamente crear la ventana y trasladarla al lugar que debía en la casa.

Todas las figuras están creadas en el origen para poder manipularlas más fácilmente.

El árbol se crea muy simple a través de GL_TRIANGLES y GL_QUADS e igualmente se crea un método para poder crear muchos arboles y solo trasladarlos al lugar que se desee.

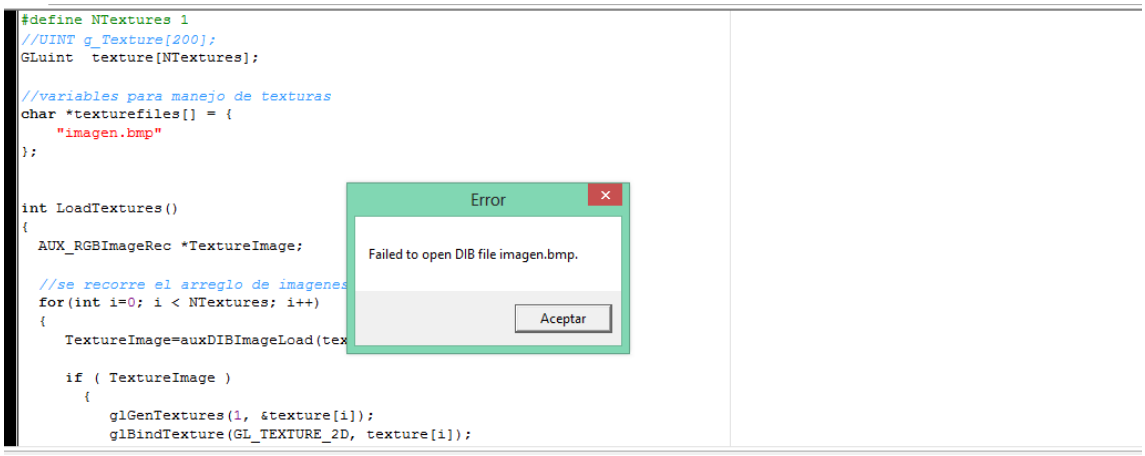
Finalmente el automóvil se crea con GL_POLYGON, a través de él lo único que hice fue calcular los puntos que serían los vértices de la figura, el problema que tuve con el auto fue crear las llantas ya que quise crearlas con glutSolidSphere() sin embargo la figura no se creaba o no se lograba visualizar en la pantalla, intente hacer pruebas primero con la esfera sola, pero también me mostraba problemas al momento de querer trasladarla. Lo que decidí hacer fue crear una clase propia que se llama esfera la cual crea un polígono con la instrucción GL_POLYGON para que pudiera estar coloreada del color que yo eligiera, los vértices de esta instrucción están dados a través de una relación entre senos y cosenos, por lo cual también use la librería math.h.

Quise hacer un único método para crear automóvil y en base a el poder direccionar el auto hacia la izquierda o derecha sin embargo no encontré ninguna primitiva que lograra hacer esto. Por esta razón cree un segundo metodo llamado coche1() el cual tiene las mismas condiciones y la misma estructura de coche() lo único que cambia son los signos de cada una de las componentes de los vértices en x para hacer de esta manera una reflexión.

Para ir formando el escenario se tuvo que ir colocando los elementos desde el fondo hasta el elemento que está en el plano más al frente.

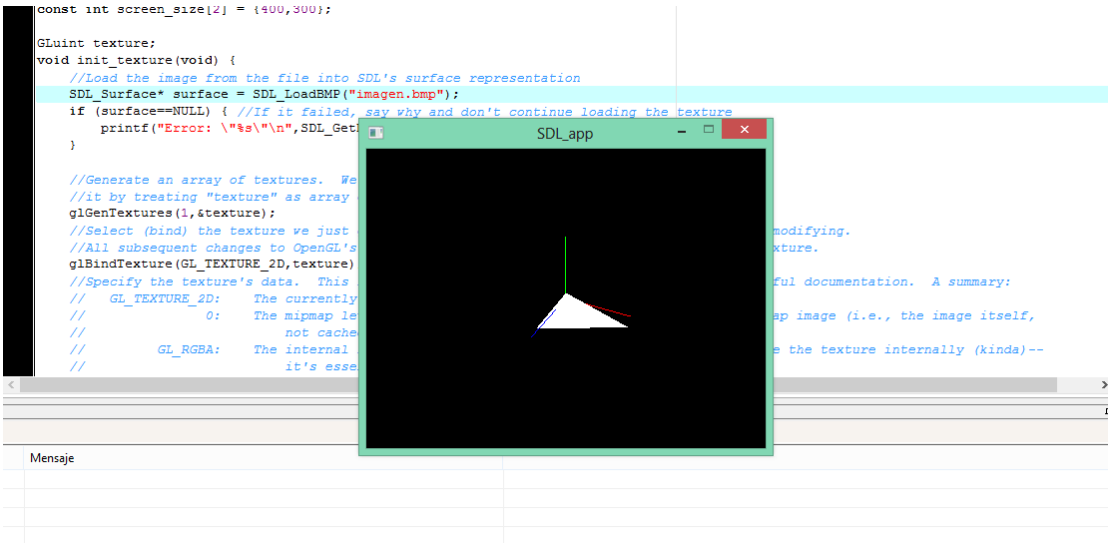
Los objetos que se mueven son el sol y los automóviles, que logran moverse a través del cambio de sus variables, en un pequeño aumento o decremento de ella y que mediante la función idle se logra guardar cada uno de esos cambios para poder lograr el efecto de movimiento.

En cuanto a las textura no las pude implementar en el escenario ya que probé con SDL, glaux y FreeImage pero ninguna me logro funcionar, ni siquiera en las pruebas. Encontré varios ejemplo que trabajan con estas librerías pero me causaban estos problemas al compilar o ejecutar:



Con glaux me marcaba este error, que al parecer la imagen no había sido cargada correctamente. Intente cambiar la versión de glaux por una anterior, pero ni aun asi funciono.

Logre, aparentemete, instalar correctamente la librería SDL pero no me podía cargar las texturas, algo que me pasaba era como esto:



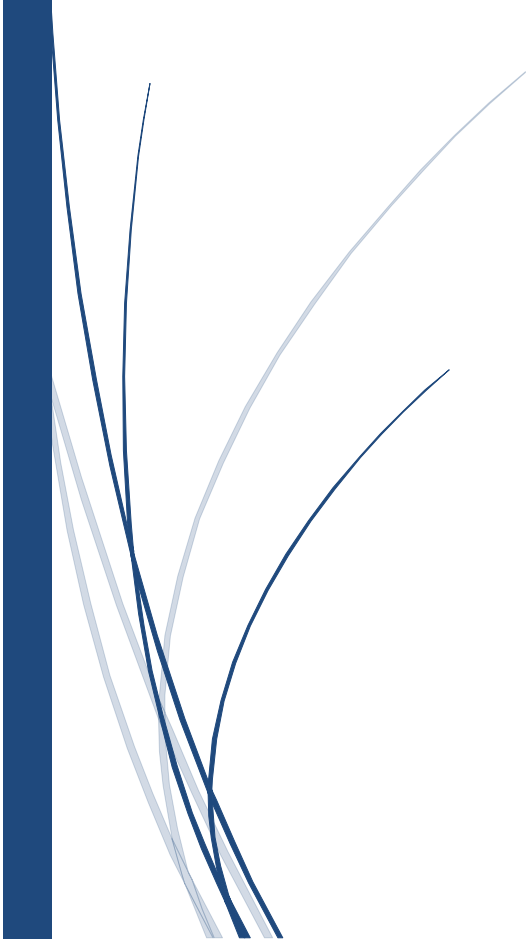
Simplemente la imagen no cargaba la textura deseada.

Finalmente este trabajo me hizo pensar en muchas soluciones para cada uno de los problemas que se me presentaron, aparentemente iba a ser más sencillo crear un escenario a través de las primitivas de OpenGL pero me di cuenta que se tiene que tomar más consideraciones además de solo dibujar puntos o líneas.

3

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

en 1 de Graficación



user

Introducción

TRANFORMACION EN 2D

En Graficación por computadora los objetos se definen mediante un conjunto de puntos. Las transformaciones son procedimientos para calcular nuevas posiciones de estos puntos, cambiado el tamaño y orientación del objeto.

OPERACIONES BASICAS

Traslación:

Se trata de una operación que desplaza un punto una distancia fija en una dirección y sentido concretos. El parámetro que se necesita es un vector (T_x, T_y). De lo que se obtiene las siguientes formulas.

$$x' = x + T_x, \qquad y' = y + T_y$$

Escalamiento:

Esta operación sirve para modificar proporcional, pero no necesariamente uniformemente, los valores que representan los puntos o vectores a través de dos factores; uno para cada dimensión. Se utiliza un factor de escala (S_x, S_y). Sus fórmulas son.

$$x' = x \cdot S_x \qquad y' = y \cdot S_y$$

Rotación:

La rotación gira los puntos de una figura alrededor de un punto fijo. De la figura se obtiene

$$x' = r\cos(\phi + \theta) = r\cos\phi\cos\theta - r\,\mathrm{sen}\,\phi\,\mathrm{sen}\,\theta$$

$$y' = r\,\mathrm{sen}(\phi + \theta) = r\,\mathrm{sen}\,\phi\cos\theta + r\cos\phi\,\mathrm{sen}\,\theta$$

REPRESENTACION MATRICIAL

Para poder representar las tres transformaciones en forma matricial como producto de matrices, es necesario representar los puntos en coordenadas homogéneas.

Estas coordenadas agregan una tercera componente a las coordenadas bidimensionales. De tal forma que, un punto (*x*, *y*) pasa a ser (*x*, *y*, z). El valor de *z* es generalmente 1.

TRASLACION

$$T\left(T_x, T_y\right) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}$$

ESCALAMIENTO

$$S\left(S_x, S_y\right) = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ROTACION

$$R(\theta) = \begin{bmatrix} \cos\theta & \operatorname{sen}\theta & 0 \\ -\operatorname{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Desarrollo

Para implementar el programa con POO, se crea una librería de clases “figuras.h” en la cual están contenidas las clases Línea, Circulo, Triangulo y Cuadrilátero.

ATRIBUTOS DE LA CLASE

Cada clase tiene como atributo una matriz de tipo *double* la cual nos sirve para realizar las transformaciones lineales de escalamiento, rotación, la traslación en este caso no se implementa con matriz.

Otro atributo que tiene cada clase son los puntos representado mediante un arreglo de tipo *int* de 3 elementos donde el tercer elemento siempre es 1, una clase puede tener varios puntos.

Se añade también un atributo de tipo *char*, esto con la finalidad de poder cambiar el color del objeto que se va a realizar, esto mediante un *switch*, solo se implementaron 5 colores que se utilizan en el escenario.

```
switch (color)
{
      case 'n':glColor3f(0.0, 0.0, 0.0);
            break;
      case 'v':glColor3f(0.2, 1.0, 0.0);
            break;
      case 'a':glColor3f(1.0, 1.0, 0.0);
            break;
      case 'c':glColor3f(0.2, 0.0, 0.0);
            break;
      case 'b':glColor3f(1.0, 1.0, 1.0);
            break;
      default: glColor3f(0.0, 0.0, 0.0);
  }
```

METODOS DE LAS CLASES

Los constructores reciben como parámetro los puntos necesarios para poder trazar la figura, el color de la figura y en el caso del circulo el radio.
El método *matrizI* se utiliza para incializar la matriz como matriz identidad.
Los métodos *escala*, *rota* inicializan la matriz para poder realizar las transformaciones, este método utiliza *matrizl* por lo que solo podemos hacer una operación a la vez.
Una vez que se realizó la inicialización de la matriz con el método *aplicarT* se realiza la multiplicación de nuestros puntos con la matriz de transformación.

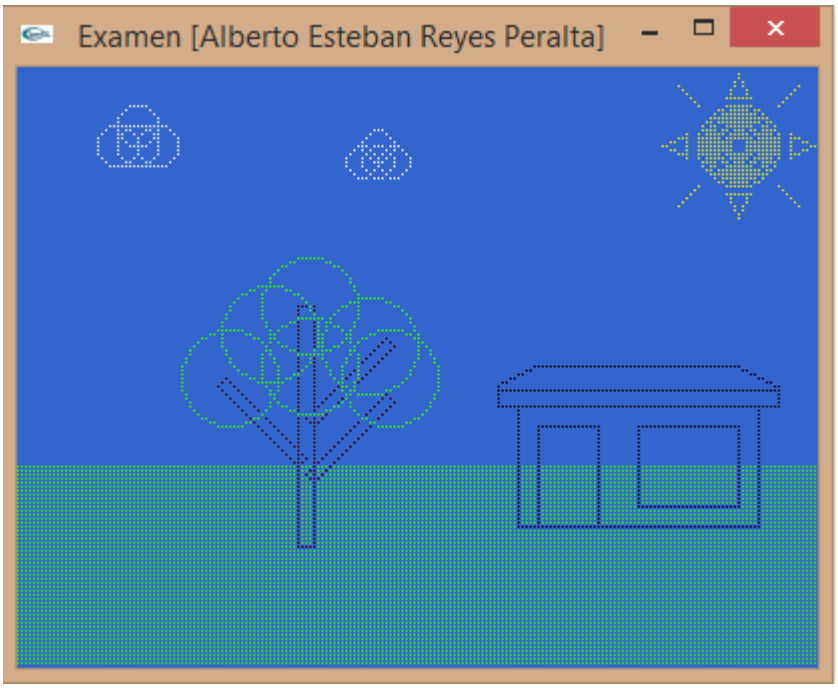
Como se realiza la traslación al origen para realizar la multiplicación, aquí se escoge mediante condiciones el valor de *x* más cercano al origen pues sobre ese punto se va a rotar la figura. Al terminar la operación los puntos de la figura ya tienen los nuevos valores.

Por último el método *dibujar* se encargara de unir los puntos para poder formar el objeto, para la línea y el circulo se utilizaron los algoritmos de Bresenham; el triángulo y cuadrilátero se realizan con líneas.

IMPLEMENTACION

Se utiliza la librería math.h, para las operaciones de seno y coseno, se realizan operaciones con arreglos y matrices mediante ciclos.

En el archivo principal, se crea el método donde se construye el escenario se crean los objetos y se operan con ellos. Para crear escenarios como este



Conclusión

El proyecto se puede mejorar, se puede implementar una clase Matriz para que no se repitan métodos en las clases, un método para la selección de color, falto por implementar transformaciones de reflexión, corte y deformación.

También se podría realizar una clase general que mediante sobrecarga de métodos dibuje cualquier figura.

Bibliografía

Anaya, H. E. (24 de Agosto de 2011). *Sitio Web de Héctor E. Medellín Anaya*. Recuperado el 17,18,19 de 02 de 2015, de http://galia.fc.uaslp.mx/~medellin/gr.htm

Broncano, P. J. (03 de 07 de 2004). *Apuntes de Gráficos*. Recuperado el 16,17,18 de 02 de 2015, de Transformaciones 2D: http://arantxa.ii.uam.es/~pedro/graficos/teoria/Transformaciones2D/Transformaciones2D.htm

Davidson, S. R. (Septiembre de 2009). *Con Clase*. Recuperado el 13 de Febrero de 2015, de http://graficos.conclase.net/curso/?cap=006#inicio

HUGHES, J. F. (2014). *Computer Graphics.* Saddle River, New Jersey: Addison-Wesley.

en 1

user

Introducción

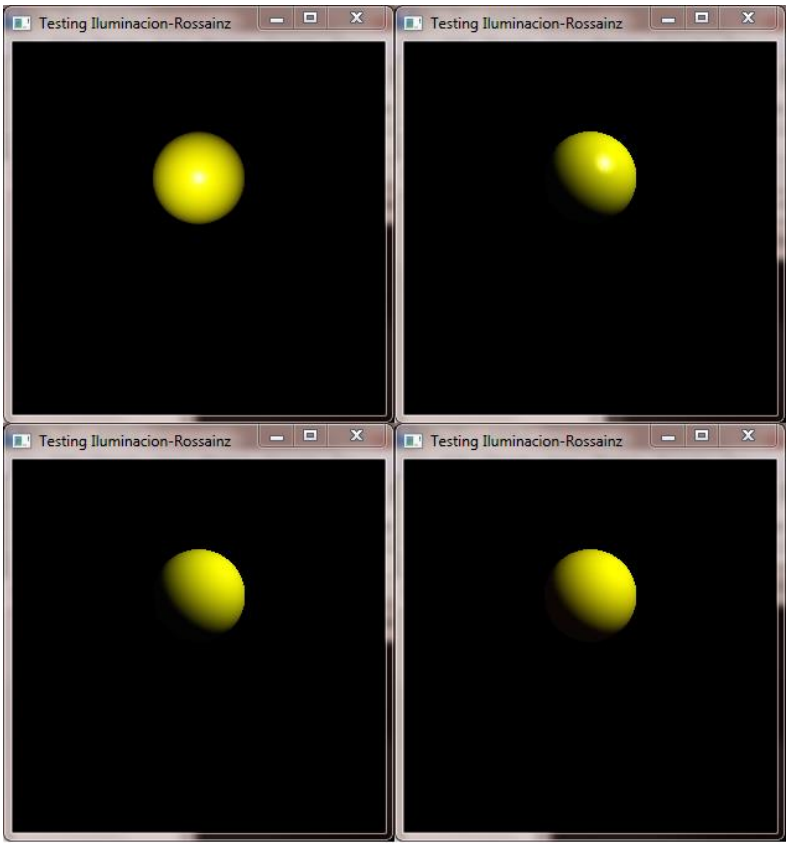
Con motivo del proyecto final de la materia graficación prepare un escenario en 3D que requería usar **texturas**, **iluminación**, y **transformaciones**. Decidí hacer algo un tanto urbano y tranquilo como he hecho con las prácticas hasta ahora y no tarde en decidirme por el **interior de un bar**, después de algunos días pensé, además, en darle una temática: "The Beatles", porque a todos nos gusta al menos una canción de la banda británica. Así nació "The Apple Bar".

Introducción

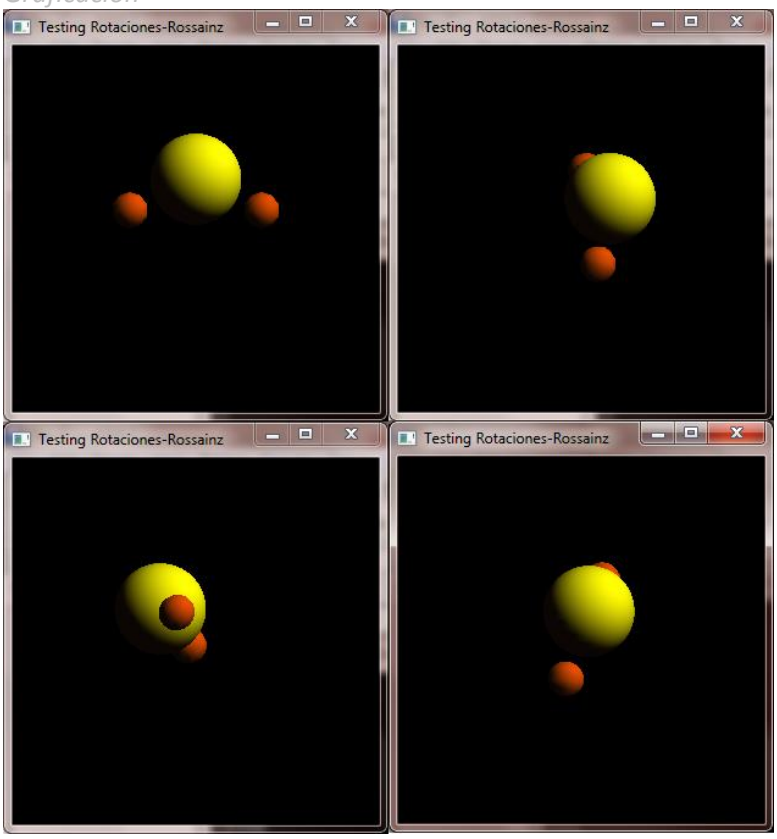
- Iluminación
- Texturas en 3D
- Transformaciones en 3D

Experimentos

Realice algunos experimentos para '*testear*' diferentes aspectos que iba a necesitar para el escenario final.



Como en el ejemplo a la izquierda en el que probé varias combinaciones de luz. Que me ayudo a dar con las combinaciones necesarias para los diferentes focos que necesitaría.



En esto otro probé varias cosas:

- Como rotar un objeto en 3D y cómo respondía a los diferentes ejes.
- Como hacer funcionar una función de teclado para rotar el un objeto deseado
- Por último, si era más sencillo mover el LookAt o rotar un el escenario completo.

Bibliografía

Material del Curso por el Profre. Ivan Olmos Pineda
http://www.cs.buap.mx/~iolmos/Curso_Graficacion.html

Apuntes de OpenGL y GLUT por Cristina Cañero Morales
http://www.elai.upm.es/webantigua/spain/Asignaturas/InfoInd/teoria/Apuntes%20de%20OpenGL.pdf

Lenguaje de Programacion: C++ GLUT Iluminación por José Luis Alonzo Velázquez
http://www.cimat.mx/~pepe/cursos/lenguaje_2010/slides/slide_48.pdf

Artículos con Clase por Javier Correa Villanueva
http://articulos.conclase.net/?tema=graficos&art=glut&pag=002

Uso de Texturas con OpenGL
http://informatica.uv.es/iiguia/AIG/docs/texturas.htm

Oocities
http://www.oocities.org/valcoey/textura.html

OpenGL.org
https://www.opengl.org/discussion_boards/showthread.php/127317-Rotating-Camera-in-GLUT

Ricardo Alejandre Trigo

Introducción

Como primer parcial el profesor se dejó hacer una libraría con las figuras de triangulo, circulo y cuadrilátero, y con las funciones de rotación, escalamiento, deformación, y reflexión, todos los objetos o figuras serian manejas mediante puntos es decir que no se podía usar las funciones de gl_LINES, gl_POLYGON ni gl_CIRCLE, mediante las coordenadas que se recibieran se trazarían puntos desde la coordenada inicial hasta la final donde el usuario definiría la densidad de la línea es decir la cantidad de puntos que se quiere entre ambas coordenadas.

Conceptos desarrollados

Para poder llevar a cabo este trabajo primero fue necesario hacer una las clases de cada objeto es decir una clase para cuadrilátero, para triangulo y para circulo cada una con sus respectivos atributos así como sus funciones, ya que sé que se tuvieran las clases yo declare una clase amiga dentro de cada clase de los objetos y la implemente dentro del archivo libreria.cpp dentro de ese mismo archivo se implementaron las funciones de las demás clases ya que en el archivo librería.hpp solo estaban las declaraciones de las mismas.

Procedimiento

Ya que cada objeto se realiza o se dibuja con líneas dentro de la creación de un cuadrilátero y de un triángulo mandaba como parámetros la cantidad de líneas necesarias, esto presento un problema ya que al mandar como parámetros objetos de tipo línea había que tener las funciones que regresen los 4 valores de cada línea, dentro de la clase línea hice las funciones que regresan los 4 puntos de la línea posteriormente al pasar como parámetro un línea ya sea en cuadrilátero como en triangulo y posteriormente en las funciones de rotar y de escalar se pasaba como parámetro un objeto ya sea cuadrado en ese caso era necesario regresar el apuntador de la primera posición del arreglo y hacer aritmética de apuntadores para encontrar los 4 valores de cada línea de las 4 líneas de un cuadrado y de la misma forma con un triángulo ya que se tuvieran los cuatro puntos (cabe mencionar que habrá 4 puntos repetidos) se podrían aplicar a la multiplicación de matrices para obtener su traslación, rotación, deformación y reflexión y dentro de la clase movimientos donde estaban todas las funciones mencionadas anteriormente y mediante el polimorfismo de funciones se le aplicaría a cada objeto que se pasara como parámetro además de otros parámetros, en el caso de rotación también se pedía los grados que rotaría, en escalamiento, el escalar en deformación si se desea en el eje x o en el eje y, más aparte el valor que se deformaría y en reflexión en que eje.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Funciones declaradas e implementadas

Linea

```
class linea{
    private:
        int punto1[2];
        int punto2[2];
    public:
        linea();
        void modificar(int,int,int,int);
        int getpunto_1x();
        int getpunto_1y();
        int getpunto_2x();
        int getpunto_2y();
        int *getpunto_1();
        int *getpunto_2();
        void dibuja(int);
    };
```

Se tenía un arreglo para guardar cada punto de la línea.

Constructor y modificador

Regresa el

Regresa apuntador hacia la primera posición del arreglo

```
void linea::dibuja(int np){
    glColor3f(1.0,1.0,1.0);
    double m,y;
    double yy= (punto2[1]-punto1[1])/n
    double xx=(punto2[0]-punto1[0])/np;
    if(punto1[0]==punto2[0]){
        glBegin(GL_POINTS);
        for(int i=punto1[1];i<=punto2[1];i++){
            glVertex2i(punto1[0],i+round(yy));
         i=i+round(yy);
        }
         glEnd();
        }
    else{
    m=(double)(punto2[1]-punto1[1])/(double)(punto2[0]-punto1[0]);
    y=punto1[1];
    glBegin(GL_POINTS);
        for(int i=punto1[0];i<=punto2[0];i++){
            glVertex2i(i,round(y));
            y=yy+y+m;
            i+=xx;

            }
    glEnd();
    }
    glFlush();
}
```

Función dibujar casi la misma solo que ahora usaba las posiciones del arreglo.

Cuadrilátero

```
class cuadro{
    linea l1;
    linea l2;
    linea l3;
    linea l4;
    int c1[4];

public:
    cuadro();
    void modificar(linea,linea,linea,linea);
    void modificar_1(int[]);
    int *get_linea1();
    int *get_linea2();
    int *get_linea3();
    int *get_linea4();
```

Las líneas como parámetros y un arreglo para regresar las coordenadas

Constructor y modificador

Modifica los valores de la líneas

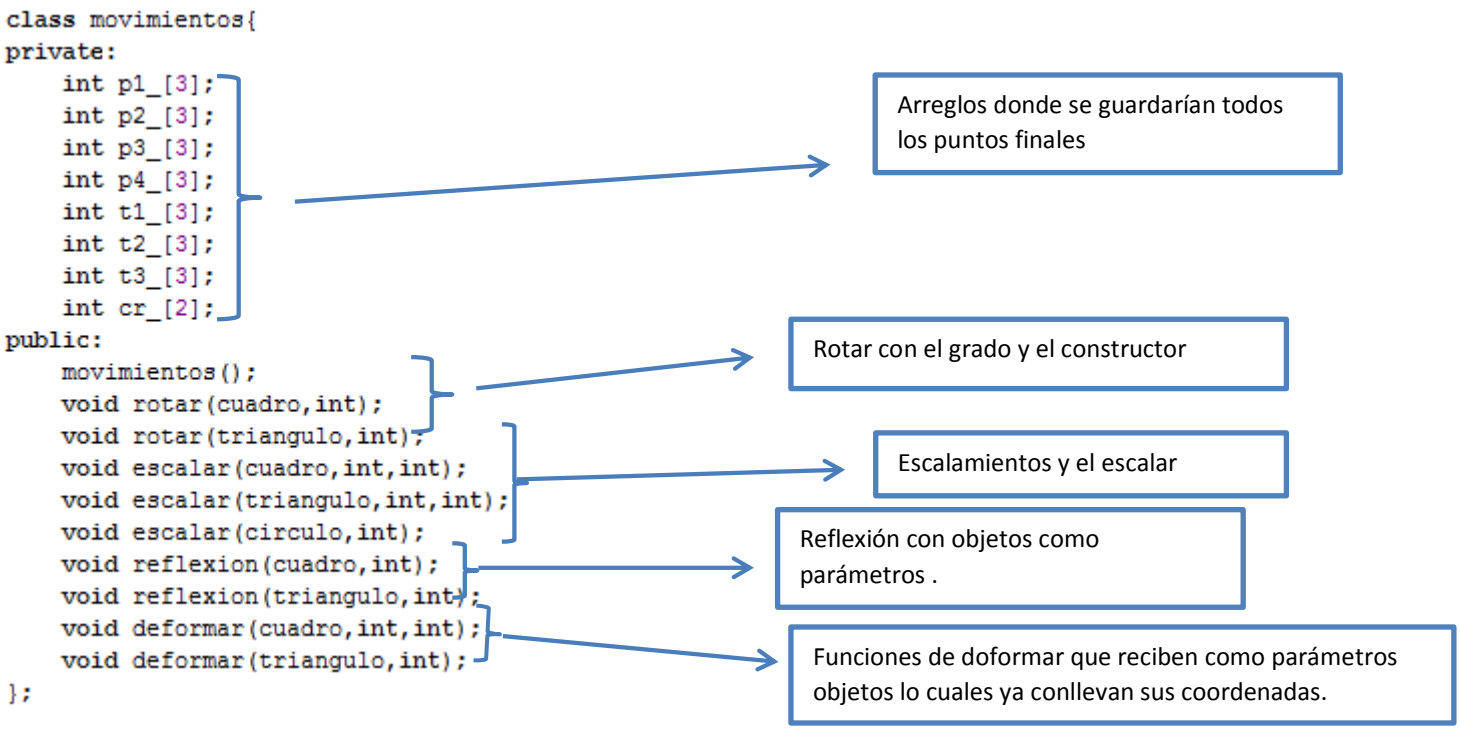
Regresa apuntador hacia la primera

Triangulo

```
class triangulo{
    private:
        linea l1;
        linea l2;
        linea l3;
        int c1[4];
    public:
        triangulo();
        void modificar(linea,linea,linea);
        int *get_linea1();
        int *get_linea2();
        int *get_linea3();
        void dibuja(int);
        friend class movimientos;

    };
```

Las mismas funciones y la misma clase amiga.

Clase amiga movimientos



Funciones de clase movimientos

```
void movimientos::rotar(cuadro c1,int n){
    int c[4],b[4];
    double rotacion[3][3];
    c[0]=*(c1.get_linea1());
    c[1]=*(c1.get_linea1()+1);
    c[2]=*(c1.get_linea1()+2);
    c[3]=*(c1.get_linea1()+3);
    b[0]=*(c1.get_linea3());
    b[1]=*(c1.get_linea3()+1);
    b[2]=*(c1.get_linea3()+2);
    b[3]=*(c1.get_linea3()+3);

    int t[2];
    t[0]=c[0]*-1;
    t[1]=c[1]*-1;

    int p1[4];
    int p2[4];
    int p3[4];
    int p4[4];
    p1[0]=c[0]+t[0];
    p1[1]=c[1]+t[1];
    p1[2]=1;

    p2[0]=c[2]+t[0];
    p2[1]=c[3]+t[1];
    p2[2]=1;

    p3[0]=b[0]+t[0
    p3[1]=b[1]+t[1
    p3[2]=1;
```

```
p4[0]=b[2]+t[0];
p4[1]=b[3]+t[1];
p4[2]=1;
for(int i=0;i<4;i++)
    p1_[i]=p2_[i]=p3_[i]=p4_[i]=0;
rotacion[0][0]=cos(n);
rotacion[0][1]=sin(n)*-1;
rotacion[0][2]=0;
rotacion[1][0]=sin(n);
rotacion[1][1]=cos(n);
rotacion[1][2]=0;
rotacion[2][0]=0;
rotacion[2][1]=0;
rotacion[2][2]=1;
for(int i=0;i<3;i++){
    for(int j=0;j<3;j++){
        p1_[i]=p1_[i]+(p1[j]*rotacion[i][j]);
        p2_[i]=p2_[i]+(p2[j]*rotacion[i][j]);
        p3_[i]=p3_[i]+(p3[j]*rotacion[i][j]);
        p4_[i]=p4_[i]+(p4[j]*rotacion[i][j]);
    }
}
t[0]=t[0]*-1;
t[1]=t[1]*-1;
p1_[0]=p1_[0]+t[0];
p1_[1]=p1_[1]+t[1];
```

```
p3_[0]=p3_[0]+t[0];
p3_[1]=p3_[1]+t[1];
p3_[2]=1;
p4_[0]=p4_[0]+t[0];
p4_[1]=p4_[1]+t[1];
p4_[2]=1;

int punto[8];
punto[0]=p1_[0];
punto[1]=p1_[1];
punto[2]=p2_[0];
punto[3]=p2_[1];
punto[4]=p3_[0];
punto[5]=p3_[1];
punto[6]=p4_[0];
punto[7]=p4_[1];

c1.modificar_1(punto);

c1.dibuja(500);
```

Conclusi

Ya que no se n bal de función amiga movimientos e s para que solo con el hecho de man el parámetro adicional ya sea el escalar e n y de cuanto, se mandara a llamar la fun s a los puntos volver a

dibujar ya que por eso se mandó el objeto para que con el mismo llame a función dibujar con los puntos modificados. En el caso de que se me dieron unos 2 días mas podría tener las demás funciones terminadas y la listas para solo crear las líneas mandarlas como parámetros después mandar el objeto y el parámetro adicional y hacer cualquiera de los movimientos.

Benemérita Universidad Autónoma de Puebla.

Facultad de Ciencias de la Computación.

Ingeniería en Ciencias de la Computación.

Graficación. Sec: 101.

Morales Márquez Luis Enrique. Mat: 201313813.

Segundo examen parcial.

Escenario construido con texturas y transformaciones.

Heroica Puebla de Zaragoza, Puebla. Marzo 2015.

Introducción.

El último examen presentado fue creado con procesos extremadamente primitivos y básicos, pues se suponían restringidas casi todas las primitivas de OpenGL, para la actual presentación se ha autorizado el uso de estas funciones, desde las de transformación hasta las más complejas de texturizado, partiendo de las practicas anteriores, es necesario conservar la misma ideología, es decir, elementos independientes en el escenario y sistemas *inteligentes* que permitan una animación atractiva, ¿qué quiero decir con esto? Simple, el uso de figuras y colores sólidos es bastante útil, sin embargo siempre resultan más llamativas las animaciones con texturas reales y movimientos poco habituales.

Desarrollo.

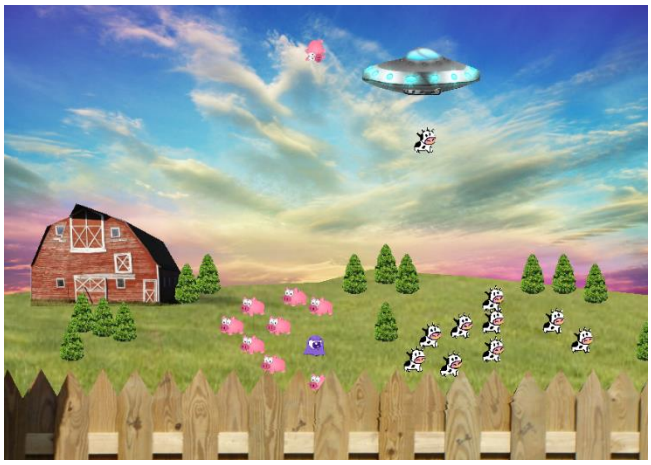
1.- Elementos nuevos.

Los escenarios mejorados requieren de un grado mayor de complejidad, como comparativa se enuncian a continuación las funciones *nuevas* que se han utilizado en la programación de esta escena.

- ***glRotatef(GLfloat, GLfloat, GLfloat, GLfloat);*** Rota el plano actual, el ángulo es el primer argumento, y el vector eje son los otros tres.
- ***glTranslatef(GLfloat, GLfloat, GLfloat);*** Traslada el plano actual según el vector indicado.
- ***glPushMatrix(); glPopMatrix();*** Funciones para regresar a sistemas cartesianos previos.
- ***FIBITMAP *loadImage(const char *filename);*** Carga a memoria una imagen cuyo *path* es especificado como argumento.
- ***GLuint loadTexture (FIBITMAP *dib1);*** Regresa la textura lista para usarse, como argumento recibe el puntero devuelto por la función anterior.

Es conveniente resaltar un detalle, la carga de texturas no se hizo a través de la librería ***glaux.h*** pues esta está deprecada, por lo cual, se recurrió a una librería externa ***FreeImage***, así pues las funciones de carga de texturas ya mencionadas son específicas para esta librería (más información al final del documento.)

2.- El escenario.



Los beneficios de utilizar funciones de más alto nivel son claros, entonces se optó por construir una escena hasta cierto punto clásica, una pequeña granja cuyos animales son raptados por naves espaciales, además de tener interacción a través de un pequeño alien que lanza animales por los aires, aún sigue siendo muy sencillo, aunque no por eso representa menor trabajo.

3.- Funciones auxiliares.

Las funciones básicas ya han sido descritas en documentos anteriores, así que aquí solo se enlistaran las funciones que representen objetos del escenario.

void stage(); Grafica el cielo y el campo.
void barn(int, int); Coloca un granero en la posición indicada.
void ufo1(); Genera el primer ovni con su recorrido.
void ufo2(); Similar a la anterior.
void cow(); Genera una vaca especial que es la vaca raptada por el ovni.
void pig(); Similar al anterior pero con otros valores y figura.
void staticCown(); Genera una vaca estándar en el escenario. (n=1 -> 9).
void staticPign(); Genera un cerdo en el escenario (n=1 -> 8).
void pine(int, int); Coloca un pino en la posición indicada.
void fence(); Grafica la cerca en el primer plano.
void alien(); Crea el alien responsable de la interacción con el programa.

4.- Interacción.

El mini juego del alien lanzando animales por el aire se debe a una función de teclado ***void keyboard(unsigned char key);*** la cual captura pulsos de teclado y modifica ciertos parámetros en el código para generar interacción.
W: Mueve al alien hacia arriba.
S: Mueve al alien hacia abajo.
A: Mueve al alien hacia la izquierda.
D: Mueve al alien hacia la derecha.
Espacio: Si el alien está en la posición de un animal, lo lanza por los aires.

Conclusión.

Como ya vimos es un tanto más complicada la generación de espacios texturizados y animados desde OpenGL, sin embargo tiene su ciencia y es perfectamente dominable, la complejidad y uso de los escenarios dependen de la creatividad y tiempo disponible del programador, pero esta pequeña muestra del funcionamiento es suficiente para mostrar el potencial que tienen estas librerías y métodos de graficación, abajo se muestran los sitios de donde se extrajo el código de apoyo para la realización de este proyecto.

Bibliografía.

Las funciones de carga de textura son propiedad del autor del blog y es usada en este proyecto sin ningún tipo de intención de lucro o conveniencia, puedes obtener el código mediante el enlace: http://www.malgouyres.org/freeimagehowto y la libreria de FreeImage es propiedad de sus autores y se utiliza bajo licencia libre, obtenida de: http://freeimage.sourceforge.net/

Benemérita Universidad Autónoma de Puebla.

Facultad de Ciencias de la Computación.

Ingeniería en Ciencias de la Computación.

Graficación. Sec: 101.

Morales Márquez Luis Enrique. Mat: 201313813.

Proyecto Final.

Escenario 3D completo
Mini Sandbox.

Heroica Puebla de Zaragoza, Puebla. Abril 2015.

Introducción.

Previamente hemos prendido todos los conceptos básicos para poder generar un escenario funcional en OpenGL, desde las primitivas elementales para puntos, hasta métodos complejos para iluminar y hacer animación *a la vieja escuela,* es cierto que se pueden utilizar programas y motores para facilitar el trabajo, sin embargo es primordial conocer cómo se construye un escenario desde un nivel *bajo* y el objetivo de esta práctica final es demostrar lo aprendido y aplicarlo a el escenario más complejo generado hasta ahora, desde texturas simples hasta trucos de cámara pueden ser apreciados en este último programa que además puede ser considerado un demo de algún futuro mini juego.

Desarrollo.

1.- Elementos a cubrir.

La escena se compone según algunos parámetros.
Objetos 3D: El *mapa* se compone de edificios y autobuses tridimensionales.
Texturas: Cada elemento dispone de un diseño a partir de una textura.
Color sólido: Algunos objetos se grafican con colores sólidos con el fin de no abusar del texturizado.
Animación: La escena es dinámica, puede moverse gracias a transformaciones elementales del escenario, además de utilizar funciones de escalado, además de que estas transformaciones pueden ser controladas a través de pulsos de teclado.
Iluminación: Se muestra iluminación básica para resaltar algunos aspectos.

2.- El escenario.

Para esta presentación, se optó por mostrar un pequeño mapa de una ciudad, en clara imitación hacia los videojuegos de exploración (lo que conocemos como *sandbox*), en el cual se puede visualizar en primera persona o bien desde una vista global (tratado más adelante), el objetivo es mostrar un pequeño ejemplo de un videojuego muy básico en un ciudad, con tan solo un par de

bloques es posible ver el funcionamiento de las primitivas y explorar un poco.

Aquí las vistas previas del mapa en tercera y primera persona respectivamente.

3.- Funciones auxiliares.

Esta pequeña ciudad se construye con algunas funciones que pueden cambiarse para generar una distinta

void roadWay(); Genera la carretera de asfalto.
void sideWalks(); Construye las aceras de los bloques.
void bank(int, int); Genera un banco en la posición x, z indicadas.
void crystalBuild(int, int); Coloca el edificio blanco en la posición indicada.
void delicatesen(int, int); Genera una salchichería en a posición indicada.
void grosery(int, int); Construye una tienda en las coordenadas dadas.
void palace (int, int); Genera un palacio amarillo en las coordenadas.
void redBuild(int, int); Coloca un rascacielos rojo en el punto indicado.
void sandBuild(int, int); Genera un edificio color arena.
void back(); Coloca el fondo de la escena.
void purpleBus(); Genera un bus purpura con su trayectoria.
void aquaBus(int, int); Coloca un bus azul a la altura y la orientación indicadas.
void greenBus(); Grafica el autobús verde de dos pisos.
void redBus(); Genera la trayectoria y el objeto del bus rojo.
void rightWheel(float, float, float, float); / void leftWheel(float, float, float, float); Esta función coloca llantas móviles en los buses, por lo tanto es llamada dentro de ellas y se especifica la posición en el bus y el tamaño escalar.

4.- Interacción.

Como ya se mencionó previamente, esta escena es interactiva, al inicio el usuario puede elegir entre una vista aérea con la cual observar todo el mapa:
Q/E: Rotación respecto al eje Y.
A/D: Rotación respecto al eje X.
W/S: Rotación respecto al eje Z.
Esc: Salir del mapa.
O bien, elegir una visión en primera persona para poder explorar el mini mapa construido como si se estuviese dentro:
Q/E: Girar la cámara a la derecha e izquierda.
W: Avanzar hacia adelante.
S: Retroceder.
A: Desplazarse a la izquierda.
D: Desplazarse a la derecha.
Esc: Salir del mapa.

Conclusión.

Habiendo concluido este curso de graficación, es ahora notorio que esos escenarios sorprendentes no resultan tan difícil e imposibles de generar, simplemente es necesario conocer el lenguaje y las librerías necesarias, además de poder lograr trasformación e escena y de cámara a base de funciones y pensamiento matemático, trigonometría en este casi simple, además de poder comprender la facilidad y la lógica para crear escenas que en un futuro puedan pulirse y mejorarse en el mundo de videojuegos, simulaciones, etc.

Bibliografía.

Las funciones de carga de textura son propiedad del autor del blog y es usada en este proyecto sin ningún tipo de intención de lucro o conveniencia, puedes obtener el código mediante el enlace: http://www.malgouyres.org/freeimagehowto y la librería de FreeImage es propiedad de sus autores y se utiliza bajo licencia libre, obtenida de: http:// freeimage.sourceforge.net/.

OpenGL, marca y sus librerías con propiedad de Silicon Graphics y es usa bajo licencia freeware.

Las imágenes de textura son propiedad de sus respectivos autores y diseñadores y se usan bajo licencia gratuita acorde a los términos de uso bajo no monetización.

en 1 de Graficación

user

INTRODUCCIÓN

TRANSFORMACIONES OPENGL

Traslación: glTranslatef(GLfloat x, GLfloat y, GLfloat z);

Ejemplo

Nos trasladamos 10 unidades sobre el eje X

glTranslatef(10.0f, 0.0f, 0.0f);

Rotación: glRotatef(GLfloat angulo, GLfloat x, GLfloat y, GLfloat z);

El ángulo de rotación es siempre un ángulo en sentido en contra de las agujas del reloj y medido en grados.

Ejemplo

Rotar 45 grados un objeto sobre el eje X

glRotatef(45.0f, 1.0f, 0.0f, 0.0f);

FUNCIONES OPENGL

OpenGL tiene una pila para las transformaciones geométricas y de la cámara llamada GL_MODELVIEW, y otra para las proyecciones denominada GL_PROYECTION. Para indicar sobre qué pila estamos trabajando se utiliza la función glMatrixMode(Nombre_Pila).

Si se quieren aplicar distintas transformaciones a distintos objetos sería necesario poder modificar el contenido de la pila. OpenGL nos ofrece 3 funciones para manejar las pilas: glLoadIdentity(), glPushMatrix() y glPopMatrix().

La función glLoadIdentity sustituye el contenido de la pila por la matriz de identidad.

La función glPushMatrix() realiza una copia de la matriz superior y la pone encima de la pila, de tal forma que las dos matrices superiores son iguales. De esta forma al llamar a la función glPushMatrix() se duplica la matriz superior y por tanto las siguientes transformaciones que se realizan se aplicarán sólo a la matriz superior de la pila, quedando la anterior con los valores que tenía en el momento de llamar a la función glPushMatrix().

La función glPopMatrix() elimina la matriz superior, quedando en la parte superior de la pila la matriz que estaba en el momento de llamar a la función glPushMatrix().

LIBRERÍA SOIL (Simple OpenGL Image Loader)

Soil es una librería que se basa en la librería stb_image. Se usa principalmente para la carga de texturas en OpenGL.

Soporta los siguientes formatos de imagen.

- BMP
- PNG
- JPG
- TGA
- DDS
- PSD
- HDR

Para instalar la librería, solo se copia el archivo SOIL.h a la carpeta de "include" de MinGW y el archivo libSOIL.a se copia a la carpeta "lib".

En los parámetros de compilación se anexa –lSOIL antes que –lglut ya que puede generar errores.

EJEMPLO DE FUNCION PARA CARGAR TEXTURAS CON SOIL

```
GLuint tex_2d;
int LoadTextures()
{
/* Carga una imagen desde archivo como una nueva textura OpenGL*/
   tex_2d = SOIL_load_OGL_texture
         (
                  "prueba.png",
                  SOIL_LOAD_AUTO,
                  SOIL_CREATE_NEW_ID,
                  SOIL_FLAG_MIPMAPS | SOIL_FLAG_INVERT_Y | SOIL_FLAG_NTSC_SAFE_RGB |
SOIL_FLAG_COMPRESS_TO_DXT
         );

/* Verifica si hubo algún error durante la carga de la textura*/
   if( 0 == tex_2d )
   {
            printf( "SOIL loading error: '%s'\n", SOIL_last_result() );
   }

}
```

Desarrollo

Crear un escenario en el cual se aplique texturizado, transformaciones con opengl.

METODOS

En el proyecto se crean estos métodos, adicionales a los que ya se tienen por default

```
void sol()
void luna()
void suelo()
void arbusto()
void cerro()
void agua()
void nube()
```

IMPLEMENTACION

Para el manejo de texturas se utiliza SOIL ya que con FreeImage la imagen que se utiliza con textura intercambia los canales de color al momento de cargarlos.

Para poder manejar nuestras texturas se crea una constante nT que será el número de texturas que se cargaran un arre de cadenas donde irán los nombres de los archivos imagen y un arreglo de tipo GLuint para guardar las imágenes que se carguen.

```
#define nT 6
GLuint tex_2d[nT];
char *nomTex[] = {
        "prueba.bmp","nube.jpg","sol.jpg","luna.jpg","arbusto.jpg","agua.jpg"
};
```

Otras variables globales que se utilizan son

```
#define KEY_ESCAPE 27
float r= 0.0;
float g= 0.6;
float b= 1.0;
float angulo=0.0;
float nubeX=25;
GLUquadric  *quadObj= gluNewQuadric();
```

Las variables r,g,b se utilizan para ir cambiando el tono de azul del fondo de la ventana para así poder dar un efecto de que el tiempo transcurre. Angulo se utiliza para la rotación de 2 objetos y nubeX para el movimiento de traslación que tendrán las nubes.

GLUquadric *quadObj= gluNewQuadric() se utilizara para generar esferas ya que no hay una función que dibuje círculos.

El método para cargar imágenes con SOIL

```
int LoadTextures()
{
  for (int iTex=0;iTex<nT;iTex++)
  {
  tex_2d[iTex] = SOIL_load_OGL_texture
        (
                nomTex[iTex],
                SOIL_LOAD_AUTO,
                SOIL_CREATE_NEW_ID,
                SOIL_FLAG_MIPMAPS | SOIL_FLAG_INVERT_Y | SOIL_FLAG_NTSC_SAFE_RGB |
SOIL_FLAG_COMPRESS_TO_DXT
        );

      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
      glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
      glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);

  if( 0 == tex_2d )
  {
          printf( "SOIL loading error: '%s'\n", SOIL_last_result() );
  }
  }
}
```

Donde en cada ciclo del for se va cargando una imagen y es importante aclarar que las imágenes iran en la carpeta donde se encuentre el ejecutable. Asi el programa empieza con un azul claro para ir oscureciendo y con las condiciones que se tienen en la función display volver a tener el color azul claro.

```
if (g <= 0.0)
    g=0.6;
  else
    g=g-0.00004;
  if (b <= 0.4)
    b=1.0;
  else
    b=b-0.00004;
```

Sol() y Luna()

Estos procesos crean dos círculos que estarán rotando en el escenario de tal manera que se traslada el punto de referencia a la parte inferior central y con el valor en el eje z para que queden por detrás de los objetos que se generen como las nubes; se mueve después nuestro objeto a la izquierda para el sol y a la derecha para la luna y se rota con respecto al centro, en este caso gluQuadricTexture(quadObj, GL_TRUE); sirve para cargar la textura en las esferas que harán de luna y sol.

```
void sol()
{

   glPushMatrix();
      glTranslatef(0,-18.0f,-5.0f);
      glPushMatrix();

        glRotatef(-1*angulo, 0.0f, 0.0f, 1.0f);
        glTranslatef(-27,0.0f,0.0f);
      glBindTexture(GL_TEXTURE_2D, tex_2d[2]);
     gluQuadricTexture(quadObj, GL_TRUE);


      gluSphere(quadObj,3.2,50,50);
      glPopMatrix();
   glPopMatrix();
}
```

Nube()

Las nubes son generadas con 2 círculos y se trasladan con respecto al eje X, en este caso se incrementa el eje en la función display, se traslada hasta salir del escenario y un vez que todas las nubes salen se regresan al punto original.

```
void nube()
{

  glPushMatrix();


    glBindTexture(GL_TEXTURE_2D,
tex_2d[1]);
        gluQuadricTexture(quadObj,
GL_TRUE);

    glPushMatrix();
      glTranslatef(nubeX,0.0f,0.0f);
        glTranslatef(0,7,0);
      gluSphere(quadObj,1.2,50,50);
      glTranslatef(1,1,0);
      gluSphere(quadObj,2,50,50);
      glTranslatef(2,0,0);
      gluSphere(quadObj,1.5,50,50);
    glPopMatrix();

    glPushMatrix();
      glTranslatef(nubeX,0.0f,0.0f);
      glTranslatef(7,2,0);
      gluSphere(quadObj,1.2,50,50);
      glTranslatef(1,1,0);
      gluSphere(quadObj,2,50,50);
      glTranslatef(2,0,0);
      gluSphere(quadObj,1.5,50,50);
    glPopMatrix();

    glPushMatrix();
      glTranslatef(nubeX,0.0f,0.0f);
      glTranslatef(14,15,0);
      gluSphere(quadObj,1.2,50,50);
      glTranslatef(1,1,0);
      gluSphere(quadObj,2,50,50);


      glTranslatef(2,0,0);
      gluSphere(quadObj,1.5,50,50);
    glPopMatrix();

    glPushMatrix();
      glTranslatef(nubeX,0.0f,0.0f);
      glTranslatef(28,0,0);
      gluSphere(quadObj,1.2,50,50);
      glTranslatef(1,1,0);
      gluSphere(quadObj,2,50,50);
      glTranslatef(2,0,0);
      gluSphere(quadObj,1.5,50,50);
    glPopMatrix();

    glPushMatrix();
      glTranslatef(nubeX,0.0f,0.0f);
      glTranslatef(35,9,0);
      gluSphere(quadObj,1.2,50,50);
      glTranslatef(1,1,0);
      gluSphere(quadObj,2,50,50);
      glTranslatef(2,0,0);
      gluSphere(quadObj,1.5,50,50);
    glPopMatrix();

    glPushMatrix();
      glTranslatef(nubeX,0.0f,0.0f);
      glTranslatef(45,6,0);
      gluSphere(quadObj,1.2,50,50);
      glTranslatef(1,1,0);
      gluSphere(quadObj,2,50,50);
      glTranslatef(2,0,0);
      gluSphere(quadObj,1.5,50,50);
    glPopMatrix();

  glPopMatrix();

}
```

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Para los métodos suelo(), arbusto(), cerro(), agua() se utilizan círculos, cuadros y polígonos en cuanto a las texturas al utilizar

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
```

Las texturas si son mas pequeñas que el objeto se estarán repitiendo hasta llenar la figura por lo que se utilizan texturas con patrones simétricos.

Bibliografía

Morales, C. C. (s.f.). *Apuntes de OpenGL y GLUT .* Obtenido de http://www.elai.upm.es/webantigua/spain/Asignaturas/InfoInd/teoria/Apuntes%20de%20OpenGL.pdf

OpenGL. (9 de Julio de 2011). *OpenGL Wiki!* Recuperado el 27 de 02 de 2015, de https://www.opengl.org/wiki/Viewing_and_Transformations

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Ing. en Tecnologías de la Información

Asignatura:

Graficacion *CCOM 259 - 101*

Practica:

2 Parcial

Alumno:

Jesús García Licona 201212661

Índice

Objetivo

-Graficar un Escenario.

-Utilizar Bibliotecas en OpenGL.

-Implementar traslaciones o rotaciones.

Estrategia de desarrollo

1- Se realizara un bosquejo previo de nuestro escenario.

2- Procederá a la colocación de las figuras.

3- Al final se colocara las traslaciones.

Desarrollo

SISTEMAS DE COORDENADAS

En OpenGL, cuando se crea una ventana sobre la que dibujar, se debe especificar el sistema de coordenadas que se desea utilizar y cómo se mapean las coordenadas especificadas a los píxeles de la pantalla física.

Uno de los sistemas de coordenadas más habituales es el sistema cartesiano, donde se define un origen del sistema en el punto x=0, y=0. La coordenada x será una medida de la posición en sentido horizontal mientras que la coordenada y será una medida en el sentido vertical. Sin embargo, en OpenGL antes de empezar a dibujar sobre una ventana, es necesario establecer la traslación entre este sistema y los píxeles que conforman dicha ventana en la pantalla física. Esto se hace definiendo la región del espacio cartesiano ocupada por dicha ventana, lo que se denomina *Clipping area*. Se corresponderá con el mínimo y máximo valor para las coordenadas x e y. La Ilustración 2 muestra dos *clipping areas*.

Dibujando líneas

La primitiva `GL_POINTS` que hemos venido utilizando hasta ahora es bastante directa, en tanto en cuanto para cada vértice especificado, dibuja un punto. El siguiente paso lógico es especificar dos vértices y dibujar una línea recta entre ellos.

Ésta operación es llevada a cabo por la primitiva `GL_LINES`. La siguiente sección de código dibuja una línea entre los puntos (0,0,0) y (20,20,20)

```
      glBegin(GL_LINES);              glVertex3f(0.0f,  0.0f,  0.0f);
glVertex3f(20.0f,  20.0f,  20.0f);


glEnd();
```

Nótese que una línea es definida únicamente por dos vértices, pero la primitiva puede contener tantas parejas de vértices como se desee, tomándose de dos en dos de manera consecutiva. En el caso de que se especifique un mayor impar de vértices, el último será ignorado.

A continuación se muestra una sección de código más compleja que muestra una serie de líneas a modo de radios de una rueda

```
//Llamar una unica vez para todos los puntos glBegin(GL_LINES);
//Todas las lineas se definen en el plano xy z = 0.0f;

for(angulo = 0.0f; angulo <= GL_PI*3.0f; angulo += 0.5f) {

//Puntos en la mitad superior de la circunferencia x = 40.0f*sin(angulo);
y = 40.0f*cos(angulo);
glVertex3f(x, y, z);

//Puntos en la mitad inferior de la circunferencia x =
40.0f*sin(angulo+3.1415f);
y = 40.0f*cos(angulo+3.1415f);
glVertex3f(x, y, z);

} glEnd();
```

Dibujar en 3D líneas, círculos y polígonos

Series de líneas y lazos

Las dos siguientes primitivas de OpenGL se basan el `GL_LINES` para permitir especificar una lista de vértices a través de los cuales de dibuja una única línea continua.

De este modo, cuando se emplea la primitiva `GL_LINE_STR`IP, se especifica que se desea dibujar una línea de un vértice al siguiente pero de manera continua, de tal manera que un vértice sea inicio y fin de un segmento (a excepción del primero –sólo inicio– y del último –solo fin).

El siguiente segmento de código dibuja dos líneas en el plano xy definidas por tres vértices.

Se muestra el resultado junto al código.

```
glBegin(GL_LINE_STRIP);         glVertex3f(0.0f,  0.0f,  0.0f);    // V0
glVertex3f(40.0f,  40.0f,  0.0f); // V1        glVertex3f(40.0f,  90.0f,
0.0f); // V2

glEnd();
```

La última primitiva para el trazado de líneas es `GL_LINE_LOOP`. Esta primitiva se comporta exactamente igual que `GL_LINE_STRIP` con la salvedad de que se dibuja un línea adicional entre el primer y el último vértice especificado. Ésta es, por lo tanto, una manera sencilla de dibujar una figura de líneas cerrada. El siguiente segmento de código muestra el mismo ejemplo que para la primitiva `GL_LINE_STRIP`, pero empleando esta nueva primitiva.

```
glBegin(GL_LINE_LOOP);        glVertex3f(0.0f,  0.0f,  0.0f);    // V0
glVertex3f(40.0f,  40.0f,  0.0f); // V1       glVertex3f(40.0f,  90.0f,
0.0f); // V2

glEnd();
```

Aproximación de curvas con líneas rectas

Obviamente, cualquier figura, y en particular, una curva, puede construirse a través de un conjunto de puntos correctamente especificados empelando la primitiva `POINTS` previamente tratada. Sin embargo, para figuras complejas, puede resultar excesivamente lento por requerir cientos o miles de puntos de tal manera que no se note la separación entre ellos.

Una manera más sencilla y válida de aproximar una curva es utilizar la primitiva `GL_LINE_STRIP` para conectar los puntos. A medida que los puntos se definen más próximos unos de otros, una curva más fina y visualmente más perfecta aparece, sin necesidad de especificar todos los puntos.

El siguiente segmento de código muestra cómo dibujar una espiral empleando esta técnica.

```
//Llamar una unica vez para todos los puntos glBegin(GL_LINE_STRIP);
z = -50.0f;
for(angulo=0.0f;  angulo<=(2.0f*GL_PI)*3.0f;  angulo+=0.1f)  {

            x=50.0f*sin(angulo);              y=50.0f*cos(angulo);
//Especificar  el  punto  y  mover  el  valor  de  z             //ligeramente
hacia arriba             glVertex3f(x,  y,  z);             z+=0.5f;

} glEnd();
```

Establecer el ancho de una línea

Del mismo modo que se pueden establecer diferentes tamaños para los puntos, también es posible especificar varios anchos de línea cuando éstas son dibujadas. Esto es llevado a cabo con la función `glLineWidth`:

```
void glLineWidth(GLfloat  width);
```

Esta función toma un único parámetro que especifica el ancho aproximado, en píxeles, de la línea dibujada. Exactamente igual que para el tamaño de los puntos, no todos los anchos son soportados, y se deberá comprobar si el ancho que se desea especificar es válido. El siguiente código muestra el rango de anchos válidos y el menor intervalo entre ellos:

```
GLfloat tamanos[2]; // Almacena el rango soportado de ancho de linea
GLfloat intervalo; // Almacena el incremento soportado para ancho de
linea


// Obtiene el rango de ancho y el incremento soportados
glGetFloatv(GL_LINE_WIDTH_RANGE,  sizes);
glGetFloatv(GL_LINE_WIDTH_GRANULARITY,  &step);
```

En este ejemplo, la variable `tamanos` contendrá dos elementos que harán referencia al menor y el mayor valor válido que la función `glLineWidth` puede tomar. Por otro lado, en la variable `intervalo` se almacenará el intervalo menor de tamaño permitido entre valores de ancho de línea.

El siguiente ejemplo muestra cómo dibujar líneas de diferentes anchos:

```
      // Llamada para redibujar una escena       void RenderScene(void)
{


GLfloat y;
GLfloat fTam[2];
GLfloat fTamAct;
...
//Almacenar el valor menor y mayor para ancho
glGetFloatv(GL_LINE_WIDTH_RANGE,fTam);  fTamAct=fTam[0];


            // Subir 20 unidades en el eje Y cada linea
for(y=-90.0f; y<90.0f; y+=20.0f)             {
                  // Establecer el ancho de linea
glLineWidth(fTamAct);                    // Dibujar la linea
glBegin(GL_LINES);
                        glVertex2f(-80.0f,  y);
                        glVertex2f(80.0f,  y);                    glEnd();
```

```
// Se incrementa el ancho fTamAct += 1.0f;
```

Polígonos

En este apartado se describirá ampliamente el uso de polígonos en OpenGL. Para ello este apartado estará dividido en dos partes: una primera dónde se presentan los aspectos teóricos y una segunda dedicada a la parte práctica mediante la visualización de ejemplos.

Triángulos

El triángulo es el polígono más simple, ya que tiene el mínimo número de vértices para definir una superficie cerrada. Como se comentó en el apartado teórico, todos los vértices de un triángulo se encuentran en el mismo plano, evitando así los problemas ya mencionados.

El triángulo es la primitiva más recomendada para programar en OpenGL, ya que cualquier otra forma poligonal puede descomponerse en triángulos. También podemos aproximar cualquier forma tridimensional cerrada utilizando triángulos de forma similar a la aproximación de curvas por medio de rectas que vimos en el apartado correspondiente.

Además, la mayoría del hardware de aceleración 3D está altamente optimizado para el dibujo de triángulos, de hecho varios bancos de prueba 3D están medidos en triángulos.

Para el dibujo de triángulos utilizaremos, como siempre, `glBegin()` y `glEnd()`. En este caso la macro del argumento para `glBegin` será `GL_TRIANGLES`. En el interior especificaremos un número de vértices que deberá ser múltiplo de tres. OpenGL cogerá los vértices de tres en tres para dibujar tantos triángulos como le sea posible. Si el número de puntos especificados no fuera múltiplo de tres, los últimos vértices definidos serían ignorados.

Por ejemplo, si usamos los vértices V_1, V_2, V_3, V_4, V_5, V_6 y V_7 tendríamos el resultado de la Ilustración 13.

Para muchas aplicaciones se necesitan conjuntos de triángulos conectados. La primitiva `GL_TRIANGLE_STRIP` nos permite crear una tira de triángulos con aristas unidas.

Los tres primeros vértices que se le especifiquen se utilizan para crear un triángulo. A partir del cuarto, toma el nuevo vértice y los dos últimos del triángulo anterior para dibujar otro triángulo que compartirá una arista con el anterior. OpenGL adapta el orden de los vértices de los triángulos de la tira para que todos tengan el mismo encaramiento que el primero.

La utilización de tiras de triángulos en lugar de especificar los triángulos separadamente tiene dos claras ventajas. Primero, tras especificar los tres primeros vértices para el triángulo inicial, sólo se necesita especificar un único punto para cada triángulo adicional. Esto ahorra mucho tiempo y espacio de datos cuando se dibujan muchos triángulos.

La segunda ventaja es que así podemos componer objetos o superficies utilizando triángulos lo cual, como hemos visto reporta beneficios.

Además de tiras de triángulos, se puede usar la primitiva `GL_TRIANGLE_FAN` para producir un grupo de triángulos conectados que se sitúan en torno a un punto central. El primer vértice forma el origen de giro, es decir, el punto central alrededor del cual se agruparán los triángulos. Después de que se usen los tres primeros vértices para dibujar el triángulo inicial, todos los vértices subsiguientes se emplean con el origen y el vértice que lo precede inmediatamente para formar el siguiente triángulo.

Cuadriláteros:

Una vez vistos los triángulos, el resto de polígonos no tiene ningún secreto. Tan sólo existen macros especiales para cuadriláteros además de para triángulos.

Hay una macro `GL_QUADS` y otra `GL_QUAD_STRIP`, con comportamiento análogo a

las dos macros de triángulos:

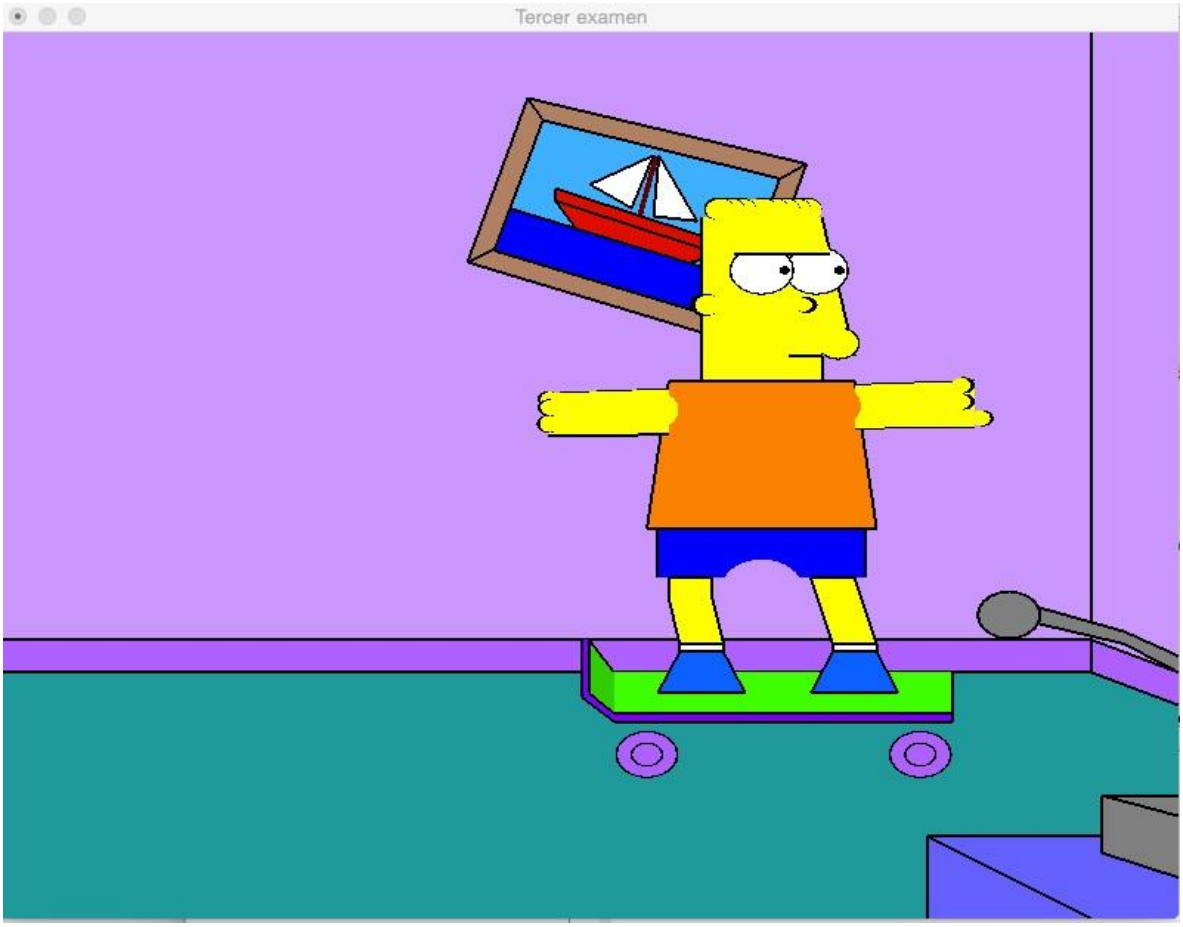
Si tenemos nueve vértices $V_1,...,V_9$:

La Ilustración 27 es el resultado con `GL_QUADS`, V_9 se ignora por no ser suficiente para crear un nuevo cuadrilátero.

Ahora vemos la tira que crea `GL_QUAD_STRIP`, que necesita cuatro vértices para el primer cuadrilátero y a partir de ahí pares de puntos que definan la nueva arista a unir. Así, V_9 es ignorada, pero si tuviéramos un V_{10} sería suficiente para definir otro cuadrilátero .

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Resultados



Alberto Esteban Reyes Peralta
Matricula: 201316680

Conclusion

Gracias al desarrollo de este segundo examen parcial se parendio a utilizar todos los elementos de openGL desde puntos hasta traslaciones o rotaciones

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Graficación

2 Parcial: Manejo de texturas, traslación y rotación

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Andrea Estephany Sánchez Hernández 201225072

Iván Olmos Pineda

Primavera 2015

Contenido

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Introducción

Para realizar este proyecto utilizaremos: curvas de Bézier, carga de texturas, rotación y traslación (En push y pop Matrix) para la creación de un escenario animado con texturas.

Conceptos Desarrollados

Texturas

La carga de textura se refiere a cargar una imagen en un polígono con el fin de personalizar el polígono lo cual nos permitirá obtener imágenes mucho más realistas.

Para aplicar una textura a un objeto 2D deberemos seguir los siguientes pasos:

1.- Creación de la textura

2.- Definimos las condiciones en las que se va a aplicar la textura

3.- Habilitar la aplicación de texturas

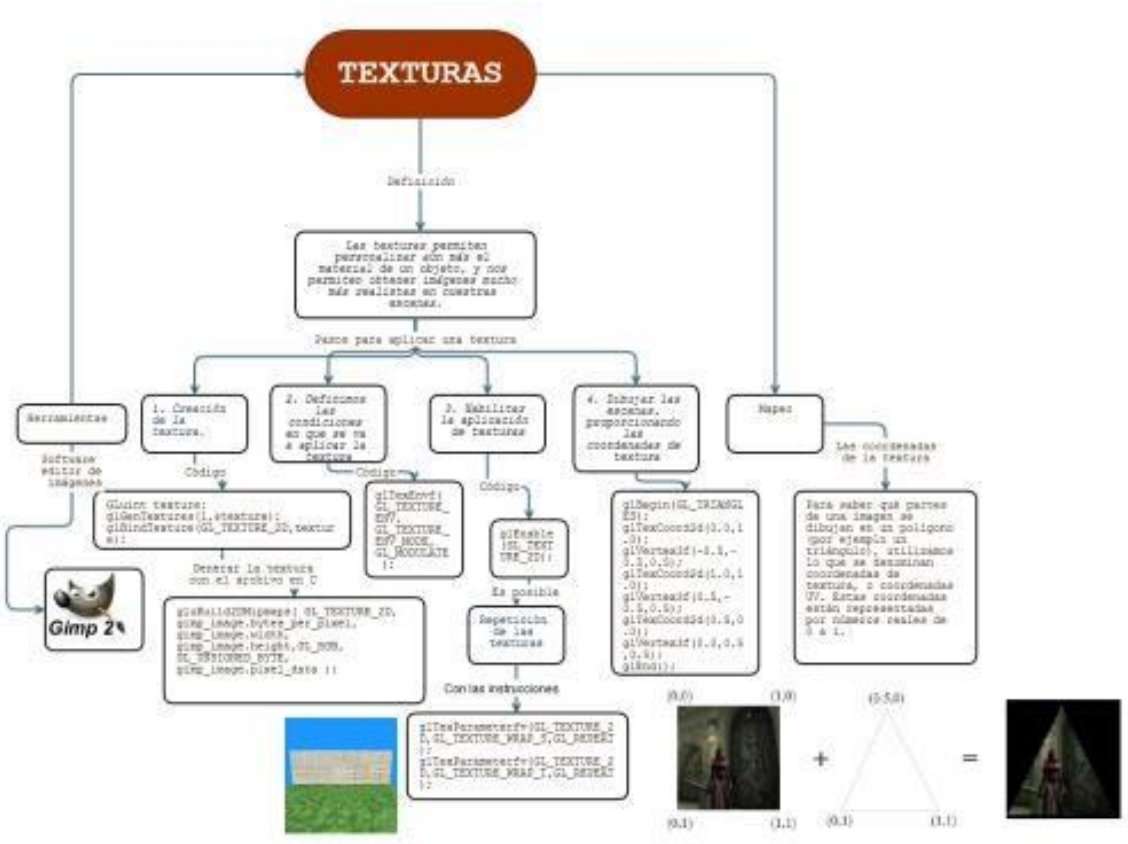
4.- Dibujar las escenas, proporcionando las coordenadas de textura.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Las coordenadas de textura

Para saber qué partes de una imagen se dibujan en un polígono (por ejemplo un triángulo), utilizamos lo que se denominan coordenadas de textura, o coordenadas UV. Estas coordenadas están representadas por números reales de 0 a 1.

A continuación se muestra un diagrama el cual representa los pasos a seguir para cargar texturas:



DEL LA TEXTURA DEL DIAGRAMA TENEMOS LA TEXTURA:

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

*Crear textura:

Int texture;

*Condicionoes de textura

Se define la cantidad de texturas que se manejaran

#define NTextures 2

```
GLuint  texture[NTextures];
Variables para manejo de texturas
char *texturefiles[] = {
        "pastito.bmp","canasta.bmp"
};
*Activar una textura:
glBindTexture(GL_TEXTURE_2D,texture);



glEnable(GL_TEXTURE_2D);
glBegin(GLTRIANGLES);
glTextCoord2d(0.0,1.0);
glVertex3f(-0.5,-0.5,0.5);
glTexCoord2d(1.0,1.0);
glVertex3f(0.5,-0.5,0.5);
glTexCoord2d(0.5,1.0);
glVertex3f(0.0,0.5,0.5);
```

Uso de glPushMatrix()

La función glPushMatrix() realiza una copia de la matriz superior y la pone encima de la pila, de tal forma que las dos matrices superiores son iguales. En la figura 1 se observa la pila en la situación inicial con una sola matriz, al llamar a la función glPushMatrix() se duplica la matriz superior. Las siguientes transformaciones que se realizan se aplican sólo a la matriz superior de la pila, quedando la anterior con los valores que tenía en el momento de llamar a la función glPushMatrix().

Uso de glPopMatrix()

La función glPopMatrix() elimina la matriz superior, quedando en la parte superior de la pila la matriz que estaba en el momento de llamar a la función glPushMatrix().

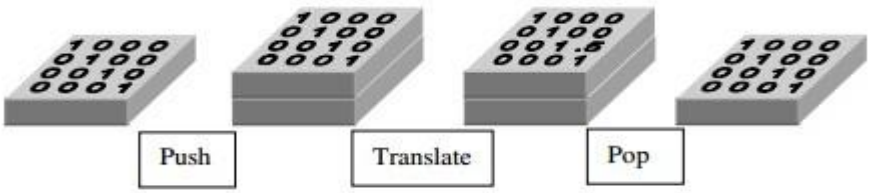


Figura 1 PushMatrix y PopMatrix

En la función display() al llamar a la función glPushMatrix() se realiza una copia de la matriz actual. La traslación en el eje Z se realiza en la matriz

superior de la pila, es decir, en la copia de la matriz, de tal forma que al llamar a la función glPopMatrix(), como se muestra en la figura 1, se elimina la matriz superior, que es la que tenía el efecto de esta transformación, quedando la matriz que estaba en el momento de llamar a glPushMatrix(). Al descomentar las llamadas a las funciones glPushMatrix() y glPopMatrix() las transformaciones realizadas entre ambas no afectan al resto de la aplicación.

Curvas de Bézier

Las curvas de Bézier son un algoritmo para trazar curvas polinomiales de manera eficiente, usando puntos de control.

El polinomio que describe la posición (en dos o en tres dimensiones) de cada punto que conforma una curva de Bézier para los n puntos de control que la definen es:

$$P(u) = \sum_{k=0}^{n} p_k B_{k,n}(u)$$

Donde $B_{k;n}(u)$ es la k-ésima función de combinación para n + 1 puntos de control (p0; p1; … ; pn) y están definidas como:

$$B_{k,n}(u) = \binom{n}{k} u^k (1-u)^{n-k}$$

Por ejemplo, para 4 puntos de control, tendríamos las cuatro funciones de combinación siguientes:

$B_{0,3}(u) = (1 \text{ i } u)^3$

$B_{1,3}(u) = 3u(1 \text{ i } u)^2$

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

$B_{2,3}(u) = 3u^2(1 \text{ i } u)$

$B_{3,3}(u) = u^3$

Estas funciones se muestran en la figura 2.3, en la esquina superior izquierda se muestra $B_{0,3}$, en la esquina superior derecha se muestra $B_{1,3}$, en la esquina inferior izquierda se muestra $B_{2,3}$ y en la esquina inferior derecha se muestra $B_{3,3}$.

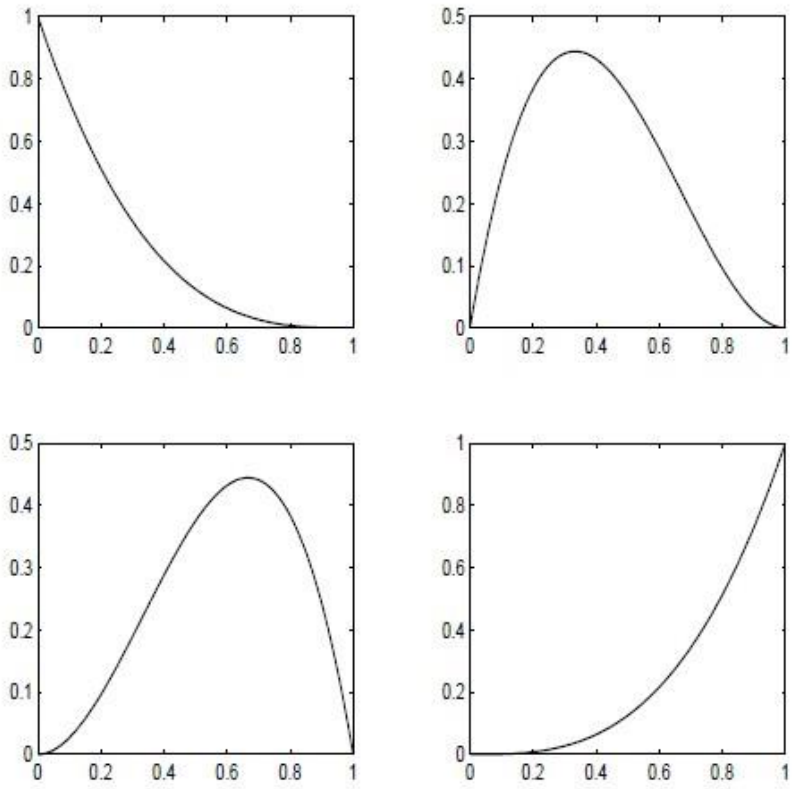


Figura 2.3: Funciones de Combinación de Bezier para 4 puntos de control

El siguiente código muestra la implementación de las curvas de bezier:

```
void na(int x1,int y1,int x2,int y2,int x3,int y3,int x4,int y4)

{ Float
PuntosdeControl[4][3]={{x1,y1,0.0},{x2,y2,0.0},{x3,y3,0.0},{x4,y4,0

.0}};//Define los puntos de control

//Se activa las curvas de Bezier con:
glMap1f(GL_MAP1_VERTEX_3,0.0,1.0,3,4,*PuntosdeControl);
glEnable(GL_MAP1_VERTEX_3);

//Cantidad de sub_intervalos

glLineWidth(4);
glMapGrid1f(100,0.0,1.0);
glColor3f(1,1,1);

//Evaluamos y mostramos la curva de Bezier
glEvalMesh1(GL_LINE,0,100);

//glEvalMesh1(GL_POINT,0,100);

//Se desactivan las curvas de Bezier
glDisable(GL_MAP1_VERTEX_3);

}
```

Nota importante:

Tómese en consideración que el código fue realizado en c++ usando el compilador wxdev c++ y las librerías opengl y freeimage.

Código.

//Andrea Estephany Sánchez Hernández FCC BUAP PRIMAVERA 2015

//Incluimos las librerías con las que se trabajaran (la librería freeimage debe estar dentro de la carpeta ya que no está configurada en el compilador, de lo contrario no funcionara el código)

```
#include <GL/gl.h>

#include<stdlib.h>

#include <iostream>

#include <math.h>

#include <GL/glut.h>

#include <stdlib.h>

#include "FreeImage.h"
```

//Definimos los máximos y mínimos para nuestro plano donde dibujaremos

```
#define xmin -200

#define xmax 400

#define ymin -200

#define ymax 200
```

```
//gluLookAt(EYE_X,EYE_Y,EYE_Z,CENTER_X,CENTER_Y,CENTER_Z,UP_X
,UP_Y,UP_Z)

float EYE_X=0.0;
float EYE_Y=0.0;
float EYE_Z=5.0;
float CENTER_X=0;
float CENTER_Y=0;
float CENTER_Z=0;
float UP_X=0;
float UP_Y=1;
```

```
float UP_Z=0;
GLint ancho, alto;
int hazPerspectiva=0;
```

//creamos los ángulos que ocuparemos para rotar nuestros objetos

```
GLfloat angulo=0.0f, anguloS=0.0f;
```

//se define la cantidad de texturas que se manejaran

```
#define NTextures 2
GLuint      texture[NTextures];
```

//variables para manejo de texturas

```
char *texturefiles[] = {
        "pastito.bmp","canasta.bmp"
};
```

//configuración de freeimage

```
bool readImage()
{
        //image format
        FREE_IMAGE_FORMAT fif = FIF_UNKNOWN;
        //pointer to the image, once loaded
```

```
FIBITMAP *dib(0);

//pointer to the image data

BYTE* bits(0);

//image width and height
unsigned int width(0), height(0);

//check the file signature and deduce its format

fif = FreeImage_GetFileType(texturefiles[0], 0);

//if still unknown, try to guess the file format from the file extension
if(fif == FIF_UNKNOWN)
```

```
        fif = FreeImage_GetFIFFromFilename(texturefiles[0]);

//if still unkown, return failure
if(fif == FIF_UNKNOWN)
        return false;



//check that the plugin has reading capabilities and load the file
if(FreeImage_FIFSupportsReading(fif))
        dib = FreeImage_Load(fif, texturefiles[0]);

//if the image failed to load, return failure
if(!dib)
        return false;

// ** AGREGADO ** (convierto la imagen a una de 24bits) //

dib = FreeImage_ConvertTo24Bits(dib);

//retrieve the image data

bits = FreeImage_GetBits(dib);

//get the image width and height
width = FreeImage_GetWidth(dib);
height = FreeImage_GetHeight(dib);

//if this somehow one of these failed (they shouldn't), return failure
if((bits == 0) || (width == 0) || (height == 0))
```

```
            return false;



        //generate an OpenGL texture ID for this texture
        glGenTextures(1, &texture[0]);
//      glGenTextures(1, &texture[1]);

        //bind to the new texture ID

        glBindTexture(GL_TEXTURE_2D, texture[0]);
```

```
//      glBindTexture(GL_TEXTURE_2D, texture[1]);

        //store the texture data for OpenGL use

        // ** AGREGADO ** (filtros para la textura y condicional de
mipmap) //

        glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,
GL_REPEAT);

        glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,
GL_REPEAT);

        glTexParameteri(GL_TEXTURE_2D,     GL_TEXTURE_MAG_FILTER,
GL_LINEAR);

        glTexParameteri(GL_TEXTURE_2D,     GL_TEXTURE_MIN_FILTER,
GL_LINEAR);

        glTexImage2D(GL_TEXTURE_2D, 0, 3, width, height, 0, GL_RGB,
GL_UNSIGNED_BYTE, bits);



        //Free FreeImage's copy of the data

        FreeImage_Unload(dib);



        //return success

        return true;
```

```
}

//configuración de nuestra pantalla

void reshape(int width, int height)

{

glViewport(0,0,width, height);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
if(hazPerspectiva)
```

```
{

gluPerspective(60.0f, (GLfloat)width/(GLfloat)height,1.0f,80.0f);

glMatrixMode(GL_MODELVIEW);

gluLookAt(EYE_X, EYE_Y, EYE_Z, CENTER_X, CENTER_Y, CENTER_Z,UP_X, UP_Y, UP_Z);



}

else

{
gluOrtho2D(xmin,xmax,ymin,ymax);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,0,0,0,-1,0,1,0);
}

ancho=width;

alto=height;

}

void keyboard(unsigned char key, int x, int y)

{
```

```
switch(key)

{

case 'p': case 'P':
hazPerspectiva=1;
reshape(ancho,alto);
break;
case 'o':

case 'O':
```

```
    hazPerspectiva=0;
    reshape(ancho,alto);
    break;



case 27: //escape
    exit(0);
    break;
}
}
```

//tipo de proyección y configuración de color

```
void init()
{ int r;
    //Establece el color de la ventana de visualizacion
    //Los tres primeros parametros corresponden al RGB
    //El cuarto parametro corresponde al valor alfa, que permite el efecto de transparencias
    //0: objeto totalmente transparente; 1: objeto totalmente opaco
    glClearColor(0.22,0.69,0.87,0);
}
```

```
    //establece los parametros de proyeccion ortogonal

    //Se visualizara una proyeccion bidimensional de dimensiones 200 x
150

    //(0,0): esquina inferior izquierda: punto de referencia de esta
ventana

    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(-200,200,-200,200);
    readImage();
    //std::cout <<









}
```

//Función para dibujar un triangulo

```
void triangulo(int x1, int y1, int x2, int y2, int x3, int y3)

{
glBegin(GL_TRIANGLE_FAN);
glVertex2i(x1,y1);
   glVertex2i(x2,y2);

   glVertex2i(x3,y3);

glEnd();

}
```

//Función para dibujar Círculos

```
void circulo(int x, int y, int r)

{



      glBegin(GL_POINTS);









}
```

```
        for (int i=0;i<=360; i++){ glVertex2f(x+sin(i) * r, y+ cos(i) * r);}

        glEnd();

}
```

//Función para crear líneas

```
void linea(int x1, int y1, int x2, int y2)

{

   glBegin(GL_LINES);
   glVertex2i(x1,y1);
   glVertex2i(x2,y2);

glEnd();









}
```

//Función para crear rectángulos/cuadrados

```
void rectangulo(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4)

{
glBegin(GL_QUADS);
glVertex2f(x1,y1);
glVertex2f(x2,y2);
glVertex2f(x3,y3);
glVertex2f(x4,y4);
glEnd();



}
```

//Creación de la flor

```
void flor()

{

triangulo(4,0,17,-3,17,3);
triangulo(0,-4,-3,-17,3,-17);
triangulo(-17,3,-17,-3,-4,0);
triangulo(-3,17,0,4,3,17);



}
```

//Función para las curvas de Bézier

```
void na(int x1,int y1,int x2,int y2,int x3,int y3,int x4,int y4)

{



float

PuntosdeControl[4][3]={{x1,y1,0.0},{x2,y2,0.0},{x3,y3,0.0},{x4,y4,0
.0}};//Define los puntos de control
```

```
//Se activa las curvas de Bezier con:
glMap1f(GL_MAP1_VERTEX_3,0.0,1.0,3,4,*PuntosdeControl);
glEnable(GL_MAP1_VERTEX_3);

//Cantidad de sub_intervalos

//glLineWidth(4);
glMapGrid1f(100,0.0,1.0);
glColor3f(1,1,1);

//Evaluamos y mostramos la curva de Bezier
glEvalMesh1(GL_LINE,0,100);

//glEvalMesh1(GL_POINT,0,100);

//Se desactivan las curvas de Bezier
glDisable(GL_MAP1_VERTEX_3);



}
```

//Uso de curvas de Bézier para crear las nubes

```
void nubes(void)

{

//nube1
na(100,100,110,110,120,110,140,100);
na(130,105,140,106,170,110,170,90);
na(170,95,170,60,120,40,120,75);
na(122,65,110,55,90,45,90,70);
na(105,105,70,115,60,95,90,65);
```

```
//nube2

na(-170,95,-170,60,-120,40,-120,75);

na(-100,100,-110,110,-120,110,-140,100);

na(-122,65,-110,55,-90,45,-90,70);
```

```
na(-130,105,-140,106,-170,110,-170,90);

na(-105,105,-70,115,-60,95,-90,65);

//nube3
na(300,100,310,110,320,110,340,100);
na(330,105,340,106,370,110,370,90);
na(370,95,370,60,320,40,320,75);
na(322,65,310,55,290,45,290,70);
na(305,105,270,115,260,95,290,65);
}
```

//Uso de funciones para dibujar un globo aerostático

```
void globo()

{

//glColor3f(0.65,0.49,0.24);

//rectangulo(-180,-130,-170,-130,-170,-120,-180,-120);

glColor3f(0,0,0);

rectangulo(-180,-102,-170,-102,-170,-92,-180,-92);

glLineWidth(.5);

linea(-180,-120,-180,-102);
linea(-170,-120,-170,-102);
glLineWidth(4);
glColor3f(0,0,1);
```

```
linea(-188,-88,-161,-88);

glColor3f(1,1,0);

linea(-195,-76,-155,-76);

glColor3f(0,1,0);

linea(-195,-72,-155,-72);

glColor3f(0.52,0.39,0.39);
```

```
linea(-195,-68,-155,-68);
glColor3f(0.678431,0.917647,0.917647);
linea(-193,-63,-158,-63);
glColor3f(0,0,1);
linea(-188,-56,-161,-56);
glColor3f(0.55,0.09,0.09);
glPointSize(2.7);
circulo(-175,-72,20);



}
```

//Uso de funciones para dibujar los carritos de la rueda de la fortuna

```
void carritos()

{

//carrito1
glColor3f(1,1,1);
circulo(-15,80,1);
glColor3f(0.847059,0.74902,0.847059);

circulo(-15,80,5);
glColor3f(0.737255,0.560784,0.560784);
circulo(-15,80,10);
//carrito2
glColor3f(1,1,1);
circulo(75,-45,1);
```

```
glColor3f(0.71,0.65,0.26);
circulo(75,-45,5);
glColor3f(0.72,0.45,0.20);
circulo(75,-45,10);
```

```
//carrito3
glColor3f(1,1,1);
circulo(-75,-45,1);

glColor3f(0.67,0.38,0.98);
circulo(-75,-45,5);
glColor3f(0.73,0.16,0.96);
circulo(-75,-45,10);

//carrito4



glColor3f(1,1,1); circulo(75,45,1);
glColor3f(0.498039,1.0,0);
circulo(75,45,5);
glColor3f(0.196078,0.8,0.196078);
circulo(75,45,10);

//carrito5
glColor3f(1,1,1);
circulo(-75,45,1);

glColor3f(0.2,0.5,0.847059);
circulo(-75,45,5);
glColor3f(0.196078,0.196078,0.8);
circulo(-75,45,10);

//carrito6
```

```
glColor3f(1,1,1);
circulo(15,-80,1);
glColor3f(1,0.75,0);
```

```
circulo(15,-80,5);
glColor3f(1,0.5,0);
circulo(15,-80,10);
}
```

//Uso de funciones para dibujar el sol

```
void sol()

{   glColor3f(1,1,0);
    circulo(180,180,5);
    circulo(180,180,10);
    glColor3f(1,0.95,0);
    circulo(180,180,15);
    glColor3f(1,0.85,0);
    circulo(180,180,20);
    glColor3f(1,0.75,0);
    circulo(180,180,25);
    glColor3f(1,0.5,0);
    circulo(180,180,30);
    glColor3f(1,0.25,0);
    circulo(180,180,35);
    glColor3f(1,0.15,0);
     circulo(180,180,40);
}

glColor3f(1,1,1);
```

//Crear los columpios Giratorios

```
void cc()

{

   //carrito1









glColor3f(1,1,1);
```

```
circulo(-75,45,1);
glColor3f(0.2,0.5,0.847059);
circulo(-75,45,5);
glColor3f(0.196078,0.196078,0.8);
circulo(-75,45,10);
linea(-75,55,-75,90);

//linea(-75,35,-75,10);

//carrito2
glColor3f(1,1,1);
circulo(-15,45,1);
glColor3f(0.847059,0.74902,0.847059);
circulo(-15,45,5);
glColor3f(0.737255,0.560784,0.560784);
circulo(-15,45,10);
linea(-15,55,-15,90);

//linea(-15,35,-15,10);

//carrito3 glColor3f(1,1,1);
circulo(75,45,1);
glColor3f(0.498039,1.0,0);
circulo(75,45,5);



glColor3f(1,1,1);
```

```
glColor3f(0.196078,0.8,0.196078);
circulo(75,45,10);
linea(75,55,75,90);
//linea(75,35,75,10);

//carrito4
```

```
glColor3f(1,1,1);
```

```
circulo(35,45,1);
glColor3f(1,0.75,0);
circulo(35,45,5);
glColor3f(1,0.5,0);
circulo(35,45,10);
linea(35,55,35,90);
//linea(35,35,35,10);




}
void carrusel()
{
rectangulo(150,-180,350,-180,350,-165,150,-165);
glLineWidth(5);
linea(165,-180,165,-150);
linea(335,-180,335,-150);
linea(165,-150,165,-60);
linea(335,-150,335,-60);
linea(165,-60,335,-60);
glPushMatrix();
glTranslatef(250,-150,0);
```

```
drawEllipse(85,15);
glPopMatrix();


glPushMatrix();

glTranslatef(250,-150,-15);
```

```
glRotatef(angulo1,0,1,0.0f);

//glRotatef(angulo1,0,1,-1.0f);

cc();
glPopMatrix();
angulo1+=1;

}
```

//Uso de funciones para crear la rueda de la fortuna

```
void rueda()

{
glColor3f(0.196078,0.8,0.6);
glPointSize(12.0);
circulo(0,0,20);
glPointSize(4.0);

rectangulo(-100,-160,-100,-180,100,-180,100,-160);
rectangulo(-80,-150,-80,-160,80,-160,80,-150);
circulo(0,0,90);

linea(-1,-20,-1,-90);

linea(1,-20,1,-90);
linea(-1,20,-1,90);
linea(1,20,1,90);
linea(20,-1,90,-1);
linea(20,1,90,1);
linea(-20,-1,-90,-1);
linea(-20,1,-90,1);
```

```
glLineWidth(18);

linea(-60,-165,-8,-15);
```

```
linea(-50,-165,0,-15);
linea(60,-165,8,-15);
linea(50,-165,0,-15);
glColor3f(1,1,1);
glLineWidth(.5);
linea(10,11,69,60);
linea(10,9,68,58);
linea(-10,11,-69,60);
linea(-10,9,-68,58);
linea(-10,-11,-69,-60);
linea(-10,-9,-68,-58);
linea(10,-11,69,-60);
linea(10,-9,68,-58);
glLineWidth(.5);
}
```

//Función de textura

```
void textura()
{
```

//activa la textura

```
   glEnable(GL_TEXTURE_2D);
   glBindTexture(GL_TEXTURE_2D, texture[0]);
       glBegin(GL_QUADS);
       //CUADRADO
//     glColor3f(0.647059,0.164706,0.164706);
```

```
glTexCoord2d(0.0,0.0);
glVertex2f(-200,-200.0);
glTexCoord2d(0.0,10.0);
```

```
glVertex2f(200.0,-200.0);
glTexCoord2d(50.0,10.0);
glVertex2f(200.0,-180.0);
glTexCoord2d(50.0,0.0);
glVertex2f(-200.0,-180.0);


   glEnd(); glEnable(GL_TEXTURE_2D);
   glBindTexture(GL_TEXTURE_2D, texture[1]);
        glBegin(GL_QUADS);
        //CUADRADO
        glColor3f(0.65,0.49,0.24);
glTexCoord2f(-180.0f,-130.0f);
                glVertex2f(-180.0f, -130.0f);


                //glColor3f(0.0f,0.0f,1.0f);
        glTexCoord2f(-170.0f,-130.0f);
                glVertex2f(-170.0f,-130.0f);


                //glColor3f(0.0f,0.0f,1.0f);
        glTexCoord2f(-170.0f,-120.0f);
```

```
                glVertex2f( -170.0f,-120.0f);



        //glColor3f(0.0f,0.0f,1.0f);

        glTexCoord2f(-180.0f,-120.0f);

                glVertex2f( -180.0f,-120.0f);

glEnd();
```

```
}

//Funcion para dibujar

void display()

{

   glClear(GL_COLOR_BUFFER_BIT);

//Mandamos a llamar a las funciones que dibujan las figuras

   globo(); sol();
   glLineWidth(4);
   glColor3f(1,1,1);
   nubes();
    rueda();

//Utilizamos Push y Pop Matrix para que los carritos giren en la rueda de la fortuna

//carritos glPushMatrix();
glTranslatef(0,0,0);
glRotatef(anguloS,0.0f,0.0f,1.0f);
carritos();
glPopMatrix();

anguloS+=0.1f;

//Columpios
glColor3f(0.8,0.498039,0.196078);
carrusel();
```

```
glLineWidth(.5);

//Llamamos nuestra función de textura

 //pasto
glColor3f(0.419608,0.556863,0.137255);
```

```
    textura();
```

//Generamos un ciclo para repetir las flores, como tenemos un plano de -200 hasta 200 (400) y se incrementa en 40, 400/40=10 flores, por lo cual nuestro contador i será menor igual a 20 para agregar una flor más

```
    //flores
glColor3f(1.0, 0.25,0 );
int a=0;
for(int i=0; i<=20;i++)
{
linea(-200+a,-200,-200+a,-165);
circulo(-200+a,-161,4);
glPushMatrix();
glTranslatef(-200+a,-161,0);
glRotatef(angulo,0.0f,0.0f,1.0f);
flor();
glPopMatrix();
angulo+=0.01f;
a+=40;
}
   glEnd();
    glFlush();
}
```

```
void idle()

{

display();

}
```

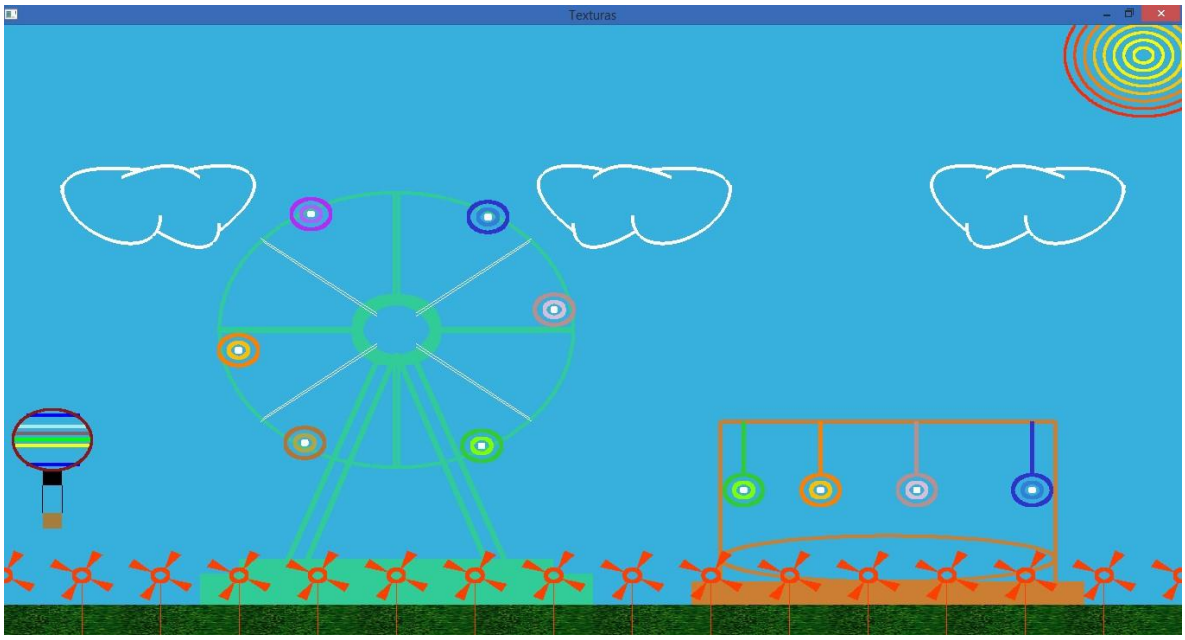
//Finalmente colocamos los parámetros en la función main donde llamamos a nuestra función display, colocamos la posición de la pantalla, el tamaño y cargamos las funciones que no son parte del dibujo ni de las texturas

```
int main(int argc, char **argv)

{

   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
   glutInitWindowPosition(100,100);
   glutInitWindowSize(1000, 600);
   glutCreateWindow("Texturas");
   //if(!LoadTextures()) return 0;

   init();

   glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutIdleFunc(idle);
glutKeyboardFunc(keyboard);
   glutMainLoop();

   return 0;

}
```

Alberto Esteban Reyes Peralta
Matricula: 201316680

Dibujo

Esta es la gráfica que mostrará el programa al ser compilado.



Conclusiones

En esta ocasión agregamos las curvas de Bézier y la carga de texturas a nuestros conocimientos básicos para crear un escenario más completo en

2D.

Bibliografía

1.- Jorge García (Bardok). (2003). Texturas 2D. 20/03/2015, de Wordpress Sitio web: https://graficacion11030440.wordpress.com/texturas-2d/

2.- Jorge García (Bardok). (2003). Curso de introducción a OpenGL (v1.0). En Manual opengl (47-50). http://www.elai.upm.es/webantigua/spain/Asignaturas/InfoInd/teoria/Manual-opengl.pdf: ESIDE Ghost.

3.- Jose Antonio Camarena Ibarrola. (2010). Notas de Graficación. En Notas de Graficación (31). Univ. Michoacana de San Nicolás de Hgo.: Facultad de Ingeniería Eléctrica.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

°

GRAFICACIÓN VERANO 2015

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Graficación

Dr. Iván Olmos Pineda

Verano 2015

Alumno: Mario Alejandro Cortez Salinas 201302537

EN 1

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Introducción

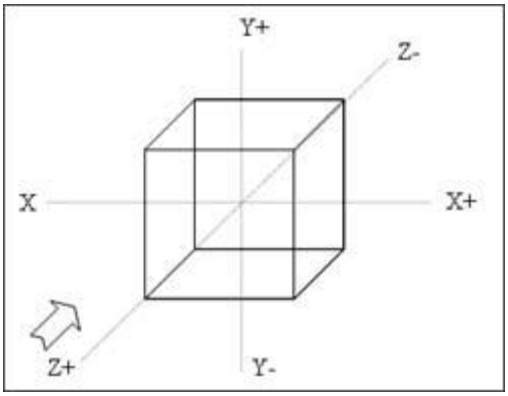
Durante este periodo vimos la manera de dibujar en el espacio 3D, esto implica que consideremos nuevos aspectos como el eje Z y que ahora nuestras matrices extendidas serán de 4*4.

El trabajo consiste en dibujar un escenario en el que se hagan operaciones de transformación a objetos 3D también aplicar rellenos de color a nuestros dibujos y trabajar con las perspectivas del escenario.

Conceptos

El dibujo 3D en OpenGL se basa en la composición de pequeños elementos, con los que se va construyendo la escena deseada. Estos elementos se llaman primitivas. Todas las primitivas de opengl son objetos de una o dos dimensiones, abarcando desde simples puntos y líneas, a polígonos complejos. Las primitivas se componen de vértices, que no son más que puntos 3D.

La Ilustración muestra un eje de coordenadas inmerso en un volumen de visualización sencillo, que se utilizará para definir y explicar el espacio en el que se va a trabajar. Este volumen se correspondería con una perspectiva ortonormal, haciendo una llamada a glOrtho(). Como se puede observar en la figura, para el punto de vista, el eje de las x sería horizontal y crecería de izquierda a derecha; el eje y, vertical y crece de abajo hacia arriba y, por último, el eje z, que sería el de profundidad, crecería hacia nuestras espaldas, por tanto, en la dirección del punto de vista, cuanto más lejos de la cámara esté el punto, menor será su coordenada z.



Una primitiva es simplemente la interpretación de un conjunto de vértices dibujados de una manera específica en pantalla. Hay diez primitivas distintas en opengl, las más comunes son: puntos (GL_POINTS), líneas (GL_LINES), triángulos (GL_TRIANGLES) y cuadrados (GL_QUADS).También las primitivas GL_LINES_STRIP, GL_TRIANGLE_STRIP y GL_QUAD_STRIP, utilizadas para definir “tiras” de líneas, triángulos y de cuadrados respectivamente.

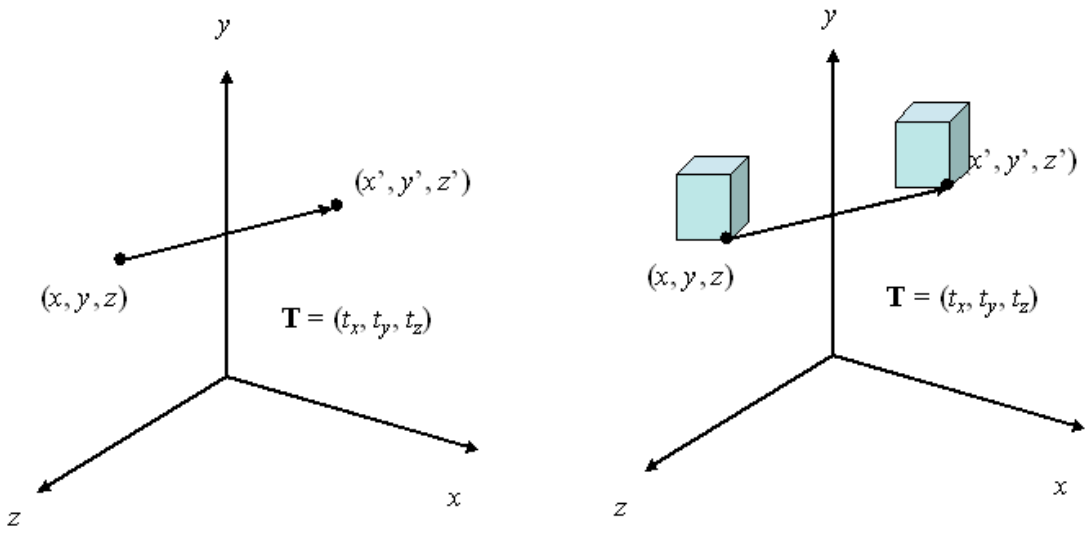
En la creación de objetos sólidos, el uso de puntos y líneas es insuficiente. Se necesitan primitivas que sean superficies cerradas, rellenas de uno o varios colores que, en conjunto, modelen el objeto deseado. En el campo de la representación 3D de los gráficos en computación, se suelen utilizar polígonos (que a menudo son triángulos) para dar forma a objetos “semisólidos” (ya que en realidad son superficies, están huecos por dentro).

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Traslación

Mueve el objeto a una nueva posición. La representación matricial es la siguiente:

$$T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

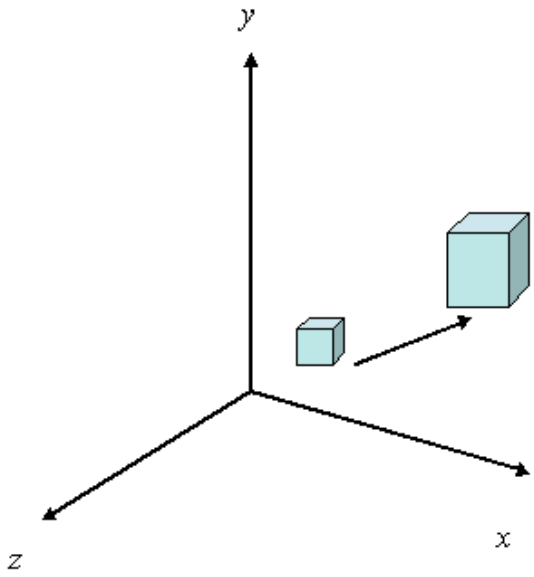


Escalamiento

El escalamiento c ambia el tamaño del objeto y al mismo tiempo desplaza el objeto a una nueva posición.

$$\mathbf{S}(S_x, S_y, S_z) = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación



Escenario

Se utiliza la función de “lookAt” para cambiar el punto desde el cual estamos viendo nuestro eje de coordenadas.

```
void init (void)
{
    gluLookAt(2,2,2,0,0,0,0,1,0);
```

Asignamos los valores de las matrices como son las de traslación, y las que definen los puntos de nuestros objetos a dibujar.

```
traslacionnave2[3][0]=1;
traslacionnave2[3][1]=0;
traslacionnave2[3][2]=0;
traslacionnave2[3][3]=1;

traspiramide[0][0]=1;
traspiramide[0][1]=0;
traspiramide[0][2]=0;
traspiramide[0][3]=0;
```

Creamos una función para que haga todos los rellenos de nuestras figuras, para hacer esto tenemos la función GL_POLYGON en la que tenemos que especificar los vértices del polígono al que le queremos asignar relleno.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

```
void dibujar_relleno (void)
{

    glColor3f(0, 1, 0);
    glBegin(GL_POLYGON);//Pasto
    glVertex3f(-20, 0, 20);
    glVertex3f(20, 0, 20);
    glVertex3f(20, 0, -20);
    glEnd();
```

Esta función dibuja todas las líneas que delimitan nuestros objetos

```
void dibujar_trazos (void)
{
    glColor3f(0, 0, 0);
       glBegin(GL_LINES);
        glVertex3i(0,30,0);
        glVertex3i(0,-30,0);

        glVertex3i(30,0,0);
        glVertex3i(-30,0,0);

        glVertex3i(0,0,15);
        glVertex3i(0,0,-15);
```

Se utiliza una función para controlar la traslación de nuestra nave, en esta se hacen los cálculos y se vuelven a dibujar los puntos en las nuevas posiciones.

```
void tras_nave (void)
{
    for(int i=0;i<=29;i++)
        {
        nave[i][0]=traslacionnave2[0][0]*nave[i][0]
        nave[i][1]=traslacionnave2[0][1]*nave[i][0]
        nave[i][2]=traslacionnave2[0][2]*nave[i][0]
        nave[i][3]=traslacionnave2[0][3]*nave[i][0]
        }
```

Nuestra función escenario es donde se calculan las operaciones de transformación y se multiplican por los puntos que definen los objetos. También llamamos a las funciones que dibujan nuestro relleno y el contorno de nuestros objetos.

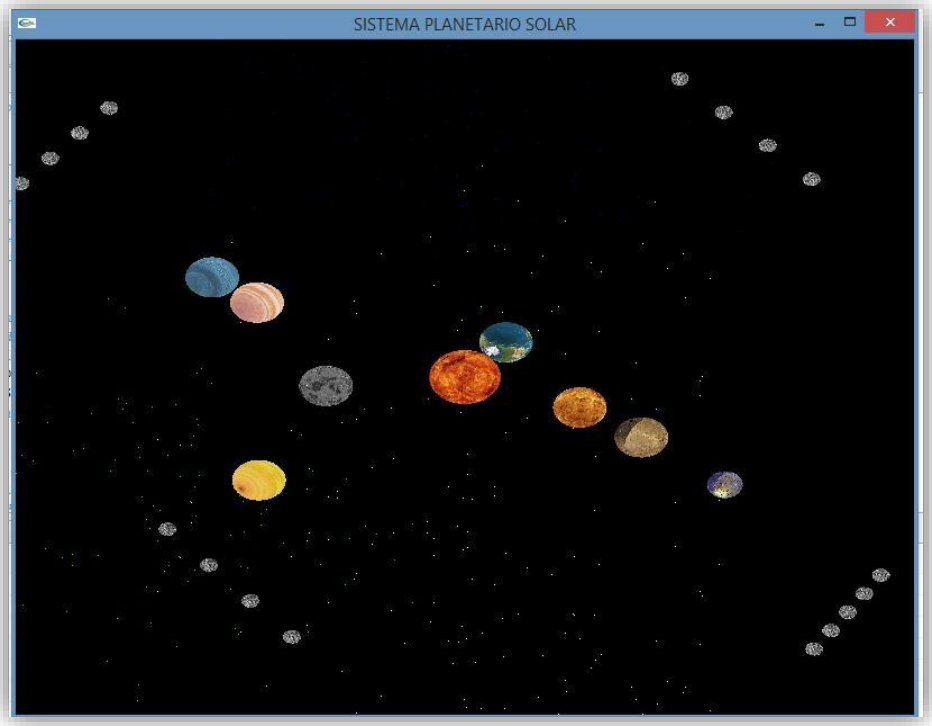
```
void escenario (void)
{

    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f(0.0, 0.0, 0.0);

 for(int i=0; i<=40;i++)
 {
     glClear (GL_COLOR_BUFFER_BIT);
     dibujar_trazos();

     dibujar_relleno();
     tras_nave();

     dibujar_trazos();
```

Bibliografía

Graficos por computadora con OpenGL: Donald Baker, Tercera Edicion, Prentice Hall 2006.

http://portales.puj.edu.co/objetosdeaprendizaje/Online/OA04/Sistemas%20de%20coordenadas.htm

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación



GRAFICACIÓN : PROYECTO FINAL

Descripción breve

GR
AFIC
ACI
ÓN :
PRO
YEC

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

TO FINAL *Sistema Planetario Solar*

Descripción breve

El proyecto desarrollado muestra de un sistema planetario en 3D utilizando distintas funciones de openGl y sus librearías.

Eduardo Carrera Huerta

03/07/15

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

INTRODUCCIÓN

Hemos llegado al final del curso de Traficación por computador y ahora nos encontramos con el desarrollo de un escenario en tercera dimensión utilizando todo el potencial posible que nos ofrece OpenGl y C++.

Se eligió como proyecto final un sistema planetario solar, ya que podemos emplear muchas funciones y librerías, así como manipular un cámara de observador y esta es muy importante si se quiere ver propiedades o características de cualquier objeto.

En el transcurso del proyecto actual, se comentara algunas observaciones y curiosidades que se encontraron en el transcurso del desarrollo del escenario.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

CONCEPTOS

FreeImage.h	FreeImage es una librería que nos ofrece el lenguaje C y es invocada para cargar
GL_BGR	Es invocada para estables un canal de Azul ,Verde y Rojo. Es empleada a la hora de aplicar filtro a una textura .
time.h	Biblioteca del lenguaje C, es utilizada cuando necesitamos una pausa de ejecuacion en cualquier parte del programa.
GLfloat	Es un nuevo tipo de declaraciones para variables, es empleada por la simplicidad y rendondeo en las variables flotantes.
stdio.h	Librería del lenguje de C, es utilizada para entrada y salida en pantalla.
stdlib.h	Biblioteca estándar del lenguaje C, nos permite declarar variables entre otras cosas.

GLUTIDLEFUNC()	Función de OpenGL , nos permite hacer llamado de funciones de manera automática y dentro de un bucle infinito
SLEEP()	Función de OpenGL, nos permite hacer una pausa en milisegundos según los parámetros que le pasemos.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

GLBEGIN(GL_TRIANGLES)	Funcion de OpenGL, nos permite dibujar un triangulo con solo tres vértices, además de poder agregarle color a cada lado
RAND()	Función del lenguaje C, la utilizamos a la hora de plotear puntos de manera aleatoria.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

DESARROLLO

Empezamos el desarrollo incluyendo una seria de libreras muy importantes para trabajar sin ellas, por mencioar las de mayor interés:

- #include <FreeImage.h>
- #ifndef GL_BGR
- #define NTextures 11

mayor trabajo en el program
en los conceptos, es una libı
imágenes o texturas para lu

Estas tres prácticamente reciben el
a, **FreeImage** como ya se mencionó
ería que nos facilita la carga de

ego manipularlas y finalmente

L_BGR , es un canal de colores, es
n filtro a todas las imágenes leídas

para dejarla es sus texturas original y por ultimo **NTextures** con una valor de 11, es como una variable donde 11 es el número de texturas que se emplea en el proyecto.

Continando con el desarrollo nos encontramos con la siguiente declaración **GLuint texture[NTextures] ,** Es solo un arreglo donde más adelante salvara las ID'S de las texturas, mismas que se emplearan de acuerdo a su identificación , ahora en la

declaración y asignación de **char*texturefiles[]** , nos encontramos con las texturas con sus respectivos nombres y formatos que se encuentran en la carpeta donde está almacenado el proyecto y sus archivos.

Seguido de una serie de variables globales, muy útiles al momento de realizar las animaciones de cada objeto correspondiente por mencionar algunas y más importantes:

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

FUNCIONES

texture_sphere() : Prepara una esfera para luego ser insertada una textura , de gran utilidad ya que OpenGl no ofrece de manera nativa Texturizar esferas.

readImage(int i): Función especial de la librería **FreeImage,** carga imágenes almacenadas con sus respectivos nombres y formatos en tu computador mas bien dentro de tu carpeta del proyecto .

cargar_imagenes() : Contiene un ciclo For y su principal función es cargar todas las imágenes que contiene el arreglo **Textures_Files[]** . Llamando a su vez la función **readimage(int i)** donde i , es el número de iteración y de imágenes a cargar.

OBJETOS : Cada función representa un objeto y además comparte la características que en todas de ellas llama a la **función texture_sphere() , glPushMatrix (),glEnable(GL_TEXTURE_2D),glBindTexture(GL_TEXTURE_2D,texture[i]),glDisable(GL_TEXTURE_2D) Y glPopMatrix()**

- sol()
- mercurio()
- venus()
- tierra()
- marte()
- jupiter()
- saturno()
- urano()

- neptuno()
- universe()
- rocks() ---- 1,2,3,4
- estrellas()

display(): Contiene solo las funciones de los objetos anteriormente mencionados, así como también los parámetros como **gluLookAt, glLoadIdentity** y por su puesto **glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)**, esta ultima nos limpiar el buffer antes de entrar en cada iteración,

key(unsigned char key, int x, int y): Contiene instrucciones específicas para realizar una acción si se presiona una tecla programada y organizada por un Switch case.

idle() : Se encuentra en un loop infinito y es la función responsable para que en cada iteración se haga de manera correcta.

Configuración de resize:

glMatrixMode(GL_PROJECTION);– Establecemos el tipo de matriz , en esta caso del tipo proyección, gluOrtho(20.0,20.0,-20.0,20.0,20.0,-20.0); – Establecemos los intervalos de pantalla.

glLoadIdentity(); Carga la matriz identidad para encadenar operaciones matriciales .

Configuracion Principal (Main) :

La configuración del Main estará dada de manera estándar, utilizamos un tamaño de 800 x 600 con glutInitWindowSize(800, 600) pero esta no será vista en la dimensión mencionada al menos que se comente la función de glutFullScreen, inicializamos el display de trabajo con

glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH) , colocamos el nombre de la ventana glutCreateWindow("SISTEMA PLANETARIO SOLAR ") y hacemos

el llamado ala función donde se encuentra las instrucción de graficado glutDisplayFunc(escenario).

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

glutIdleFunc(display); Hace un loop infinito, que estará llamando de manera automática la función display, indispensable para realizar las animaciones.

glutSwapBuffers(); Empleada al final del display , ya que se necesita dos buffers para el dibujado de un escenario complejo.

Bibliografía

Anonimo. (03 de 07 de 2015). .*opengl-tutorial.org*. Obtenido de http://www.opengl-tutorial.org/beginners- tutorials/tutorial-5-a-textured-cube/

Anonimo. (03 de 07 de 2015). *crodrigp.wordpress.com*. Obtenido de https://crodrigp.wordpress.com/2008/11/23/esferas-que-rotan-en-su-propio-eje/

MathiasVP, #. (03 de 07 de 2015). *dreamincode*. Obtenido de http://www.dreamincode.net/forums/topic/253361- texture-mapping-and-glutsolidsphere/

msi_333. (03 de 07). *codemiles*. Obtenido de 2015: http://www.codemiles.com/c-opengl-examples/draw-3d-cube- using-opengl-t9018.html

opengl. (03 de 07 de 2015). *opengl*. Obtenido de https://www.opengl.org/sdk/docs/man2/xhtml/glTexImage2D.xml

Orta, A. E. (03 de 07 de 2015). *aerodevelop.blogspot*. Obtenido de http://aerodevelop.blogspot.mx/2012/02/sistema-solar-opengl_17.html

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

FACULTAD

CIENCIAS DE LA COMPUTACIÓN

MATERIA

GRAFICACIÓN

DOCENTE

IVAN OLMOS PINEDA

TRABAJO

PROYECTO FINAL

ESTUDIANTE

BRAYAN USIEL TOZCANO JIMÉNEZ.

MATRICULA

201317966

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

PUEBLA, PUEBLA
3 DE JULIO DEL 2015

Introducción.

En este trabajo vamos a implementar las funciones de OpenGL que se utilizan para las operaciones de objetos. Las operaciones servirán para realizar translaciones, rotaciones y escalamiento. Pero en este trabajo solo utilizaremos rotación y traslación. También aplicaremos texturización para darle un aspecto más real a nuestro escenario.

Conceptos desarrollados

Para comenzar a desarrollar nuestro trabajo es necesario que sepamos los conceptos acerca de nuestra matriz de MODEL_VIEW. Esta matriz al momento de inicializarla tiene la matriz identidad. Una vez realizado esto realizaremos un glPushMatrix(); Esta operación es para almacenar en una pila una serie de matrices. Esto es con el fin de no perder ninguna matriz y en caso de requerirlo poder volver a utilizarla. Cada vez que aplicamos na operación se multiplicara una matriz que genera la función por la matriz identidad. Lo que hacen las operaciones de Opengl es mover el centro de nuestro eje y así dará la impresión de que se está realizando la operación deseada.

Para lograr cargar las texturas utilizaremos una librería auxiliar llamada FreeImage, con esta podremos cargar imágenes de distintos formatos y poder aplicarlas a un polígono mediante una

función que será glTexCoord2f(). A continuación mostrare un ejemplo:

```
glBindTexture(GL_TEXTURE_2D, texture[2]);
glBegin(GL_QUADS);
glVertex3f(-30,0,-5);
glTexCoord2f(0,0);
glVertex3f(-100,0,-5);
glTexCoord2f(0,1);
glVertex3f(-100,0,-175);
glTexCoord2f(1,1);
glVertex3f(-30,0,-175);
glTexCoord2f(0,1);
glEnd();
```

La función del inicio sirve para indicar que textura se va a estar trabajando. Las texturas que vallamos cargando las iremos guardando en un arreglo que llamamos texture[ ].

Procedemos a realizar una rotación que va aumentando un grado. Esto se hace para que se logre visualizar el escenario desde cualquier rincón.

Se procede a dibujar el escenario con los polígonos correspondientes y cambiando de texturas para aplicar las que nosotros deseamos, debemos de tener cuidado con la forma de aplicar texturas, ya que debe ser con la misma orientación con que se dibuja nuestro polígono, de lo contrario la imagen se puede distorsionar o colocar con una orientación no deseado, por ejemplo verticalmente en vez de horizontal.

Un aspecto muy importante es que al momento de aplicar texturas tiene un filtro y es necesario quitarlo, si no lo hacemos nuestros colores originales de nuestra textura van a cambiar. Para poder realizarlo se define lo siguiente y se aplica de la siguiente manera.

#ifndef GL_BGR

#define GL_BGR 0x80E0

#endif

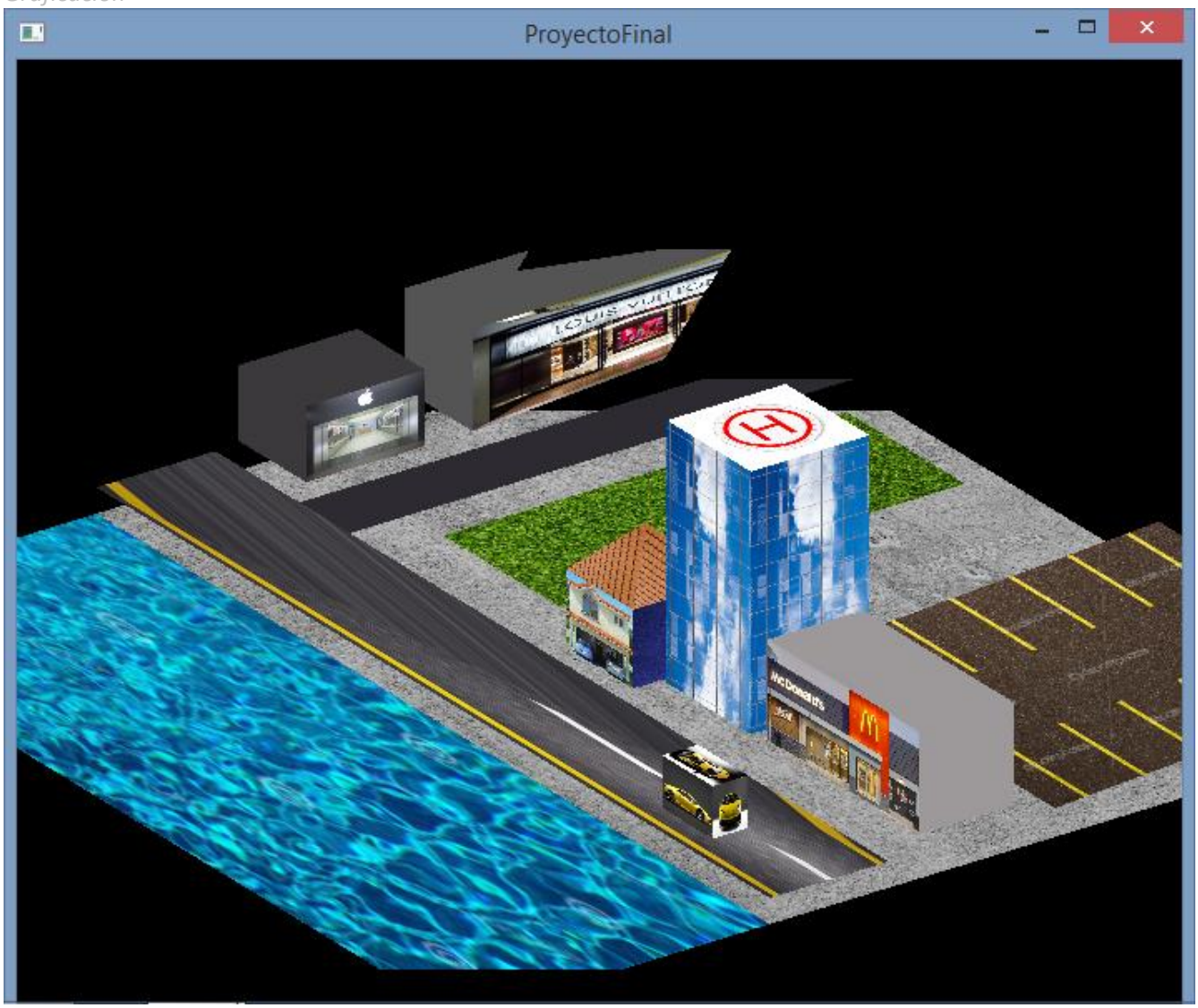
Y se va a poner asi la siguiente línea de código: glTexImage2D(GL_TEXTURE_2D, 0, 3, width, height, 0, GL_BGR, GL_UNSIGNED_BYTE, bits);

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

En algunas ocasiones, al momento de realizar la rotación nuestros objetos dibujados se traslapan, esto se debe a que OpenGL no detecta la profundidad por defecto, para esto activaremos un buffer conocido como z-buffer. La función del buffer es identificar que objeto está más cerca respecto a un punto observador y acomodar los objetos para que no se encimen.

En el proyecto se puede percibir a un coche y a una luna siendo trasladadas, esto lo realizamos por medio de un glTranslatef(traS,0,0).El argumento traS es una variable declarada globalmente , su valor ira cambiando dependiendo de las iteraciones del gluLoopMain(); el valor inicial será de -361 e ira disminuyendo. Esto se debe a que queremos que nuestro objeto se mueva hacia la izquierda. Para el coche se utiliza la misma variable pero la como parámetro será la variable tras multiplicada por -1. Esto se debe a que ahora el movimiento es hacia la derecha.

Una vez que se realiza 360 veces esta repetición, se vuelve a inicial la variable con los valores iniciales, esto para que regresen a su posición original y se vuelva a realizar el mismo proceso de translación. A continuación mostrare una captura de mi trabajo ya finalizando:



Conclusiones.

Las operaciones en 3D mediante OpenGL son muy útiles y prácticas de manejar, te ayudan a ahorrar mucho tiempo ya que no tenemos que estar calculando matrices, gracias a esto ya no debemos preocuparnos por operaciones demasiado complejas. Con el manejo de las texturas logramos darle un aspecto más profesional a nuestros proyectos, solo que debemos tener cuidado al momento de cargarlas y aplicarlas. Todo lo aprendido en el curso me ha ayudo a tener una idea general de la potencia de los gráficos en la computadora, me di cuenta que es un mundo enorme y que con mucho empeño, dedicación y una gran imaginación se pueden lograr grandes cosas. También se puede observar como las matemáticas se aplican a problemas ya

consolidados y nos ayudan a resolverlos y también a mejorarlos para que funcionen de una manera mas eficiente.

Bibliografía: Gráficos por computadora con OpenGL. Heam Baker, Pretice Hall.2005.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Graficación

Dr. Iván Olmos Pineda

Verano 2015

Alumno: Mario Alejandro Cortez Salinas 201302537

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

EXAMEN PRIMER PARCIAL

ESCENARIO CON TRANSFORMACIONES.

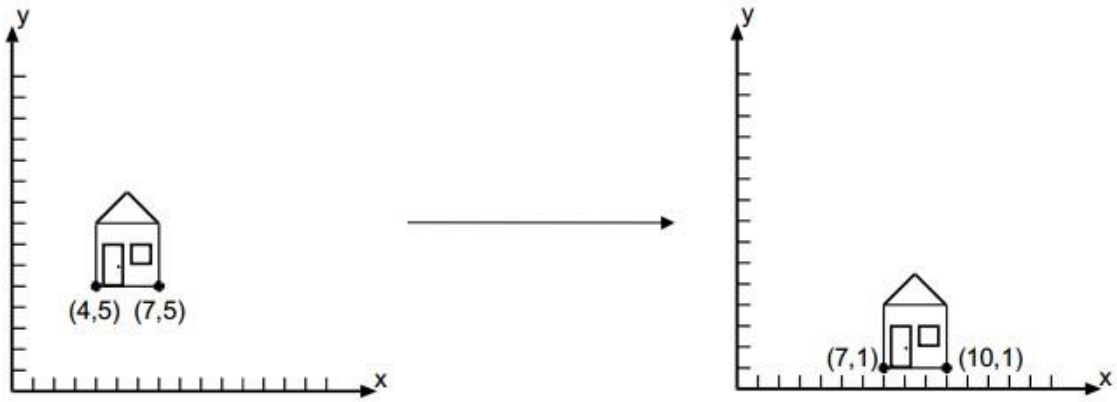
Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Introducción

Durante las primeras semanas de clase estuvimos trabajando con las primitivas de líneas, puntos y transformaciones, con todo lo aprendido durante ese tiempo de hará el desarrollo de un escenario que contenga esos elementos para que podamos ver con más claridad como funciona cada uno de ellos.

Conceptos

- Traslación

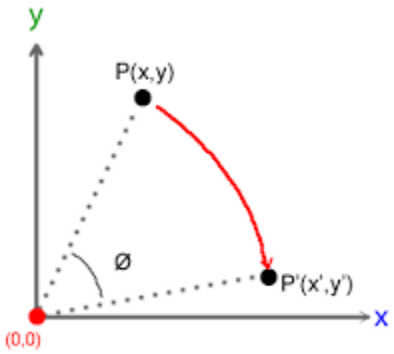


Traslación de un objeto

La traslación es un tipo de ***transformación de sólido-rígido*** que mueve objetos sin deformarlos. Esto es, cada punto de un objeto es trasladado en la misma medida. Un segmento en línea recta es trasladado mediante la aplicación de una ecuación de transformación a cada uno de los puntos finales de la línea y redibujando la línea entre los dos nuevos puntos lineales.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

- Rotación

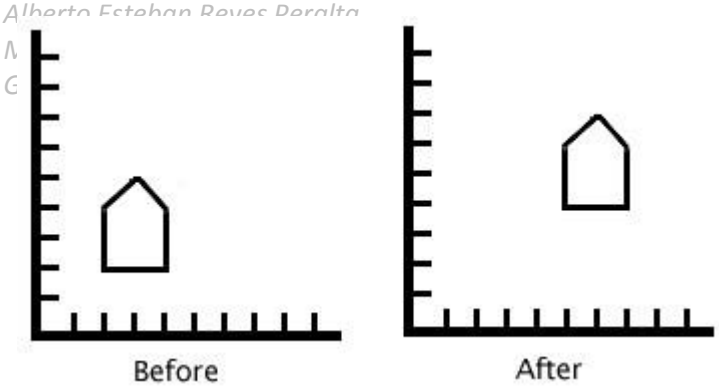


Una rotación bidimensional de un objeto se obtiene mediante la recolocación del objeto a lo largo de una trayectoria circular sobre el plano *xy.* En este caso, se está rotando el objeto sobre un eje de rotación que es perpendicular al plano (paralelo al eje de coordenadas *z).* Los parámetros para la rotación bidimensional son el ángulo de rotación

0, y una posición *(xr yr)* llamada **punió de rotación** (o **punto de pivote)** sobre los cuales

el objeto va a ser rotado. El punto de pivote es la posición de intersección entre el eje de coordenadas y el plano *xy.* Un valor positivo para el ángulo *0* define una rotación en sentido contrario a las agujas del reloj sobre el punto de pivote, como en la , y un valor negativo rota objetos en el sentido de las agujas del reloj.

- Escalamiento



Para alterar el tamaño de un objeto, aplicamos transformaciones **de escala.** Una simple operación de cambio de escala bidimensional se lleva a cabo multiplicando las posiciones de los objetos *(x y)* por los **factores de escala *sx y sy*** para producir las coordenadas transformadas (.x',y’*)***.**

Podemos controlar la localización de un objeto cambiado de escala eligiendo una posición, llamada punto fijo, que debe permanecer sin cambios después de la

transformación de escala. Las coordenadas para el punto fijo, son a menudo elegidas de la posición de algún objeto, tal como su centro, aunque puede elegirse cualquier otra

posición espacial.

Dibujando

Para comenzar a trabajar en nuestro escenario se determinaron las coordenadas de los puntos que forman nuestras figuras y las dibujamos en el origen para que de esta manera se mas fácil operarlas de manera matricial.

```
float k2=-20,k3=-10;
float ym=0;
int montanas[20][3];
int nave [20][3];
int casa [20][3];
int base [20][3];
int sol  [380][3];
int humo  [380][3];
int reloj [360][3];
int traslacion[3][3];
```

```
nave[0][0]=1;//Nave
nave[0][1]=1;
nave[0][2]=1;

nave[1][0]=5;
nave[1][1]=6;
nave[1][2]=1;

nave[2][0]=1;
nave[2][1]=1;
nave[2][2]=1;

nave[3][0]=20;
nave[3][1]=1;
nave[3][2]=1;
```

De esta manera especificamos cada uno de los objetos de nuestro escenario y también cada una de las operaciones que estas tendrán.

```
traslacion[0][0]=1;
 traslacion[0][1]=0;
 traslacion[0][2]=0;
```

```
for(int i = 0 ; i < 361; i++){
```

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Para el caso especial de la rotación primero debemos pasar nuestros grados a radianes.

```
ang2=(ang*3.1416)/180;
```

```
rotacion[0][0]=cos(10);
rotacion[0][1]=sin(10);
rotacion[0][2]=0;

rotacion[1][0]=-sin(10);
rotacion[1][1]=cos(10);
rotacion[1][2]=0;

rotacion[2][0]=0;
rotacion[2][1]=0;
rotacion[2][2]=1;
```

Para poder tener el control del tiempo en el que se vuelven a calcular los dibujos y se mandan a pantalla usamos una función “clock”.

```
seg1=clock();
seg2=clock();
if((seg2-seg)==aux2)
{
    aux2=aux2+1000;
```

Una vez que tenemos el control de nuestros objetos y sus operaciones procedemos a operar (realizar multiplicación de matrices), esto incluye: rotar, trasladar o escalar.

```
for(int i=0; i<=360;i++)
{
    humo[i][0]=escalamientos[0][0]*humo[i][0]+escalamientos[1][0]*humo[i][1]+escalamientos[2][0]*humo[i][2];
    humo[i][1]=escalamientos[0][1]*humo[i][0]+escalamientos[1][1]*humo[i][1]+escalamientos[2][1]*humo[i][2];
    humo[i][2]=escalamientos[0][2]*humo[i][0]+escalamientos[1][2]*humo[i][1]+escalamientos[2][2]*humo[i][2];
```

Ahora que está todo listo podemos mandar a dibujar todo a pantalla.

```
glBegin(GL_LINES);
    for (int x=0; x<=17; x++)//Dibujar nave
    {
            glVertex2i(nave[x][0], nave[x][1]);
    }
    for(int i=0;i<=19;i++)//Dibujar casa
    {
        glVertex2i(casa[i][0], casa[i][1]);
    }
```

Bibliografía

Graficos por computadora con OpenGL: Donald Baker, Tercera Edicion, Prentice Hall 2006.

Alberto Esteban Reyes Peralta
Matricula: 201316680

BENEMRITA UNIVERSIDAD AUTONOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LAS COMPUTACION

MC. IVAN OLMOS PINEDA

MATERIA: GRAFICACION ALUMNO:

SANDOVAL SALAZ CONSUELO

PERIODO: VERANO 2015

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Introducción

En el siguiente proyecto se presenta un escenario 3D en el cual se hará uso de todas las primitivas de Opengl tales como glrotatef, glscalef, gltranslatef, glquads entre otras para lograr dibujar unas casitas, arboles, automóviles, bancas, el sol, nubes, la luna y estrellas.

Desarrollo

Primero que nada se usan las funciones que en todo el curso se han ido utilizando como la función Init que se encarga de inicializar las variables y entornos del proyecto como la pantalla, el color, el tamaño el tipo de proyección entre otras cosas.

La función rashape fue utilizada para redimensionar la ventana y su contenido. La función keyboard que nos sirve para la interaccion de nuestro programa con el teclado. La función display que es la que se encarga de dibujar todo en pantalla. Por último la función main que es el principio de nuestro código ya que es la que se encarga de llamar a todas las funciones anteriormente mencionadas.

En el código del proyecto para dibujar cada objeto se crearon funciones, posteriormente solo se mandarían a llamar a las funciones para dibujar n-objetos como se deseara es decir si queremos dibujar un automóvil solo mandamos a llamar a la función carro y le mandamos las coordenadas donde queremos colocar el carro, cada función contiene una estructura como se descirbe a continuación.

```
Void nombre ([parametros]){
glPushMatrix();
código de objeto a dibujar;
}//fin de la función
```

Siguiendo dicha estructura se dibujaron las siguientes funciones:

Void escenario(): Esta funcion no recibe parámetros de entrada porque solo dibuja el piso de la escena y tres muros a los lados y en el fondo, al mandar a llamar a las demás funciones se dibujaran sobre esta.

Void nube (float X,float Z):Aquí se dibuja una nube sin movimiento, los parámetros que recibe son las coordenadas en donde se ubicara el objeto nube en el escenario.

Void sol (float X, float Z): Se encarga de dibujar una esfera con un tamaño predeterminado que simula el sol, los parámetros son las coordenadas en donde se desea ubicar el sol, este objeto tiene además traslación.

Void luna (float X, float Z): La función dibuja una esfera de tamaño predeterminado, los parámetros recibidos son las coordenadas donde se ubicara el objeto, dicho objeto tiene traslación e iluminación.

Void estrellas (float x, float y, float z): Cuando desaparece el objeto sol el cielo se torna de color negro y aparecen unos objetos que simulan las estrellas, las cuales se realizaron con cuatro conos colocados en modo que simulen picos, sus parámetros son las coordenadas de los ejes x, y, z.

Void casa (float X, float Z, float C): Esta función recibe como parámetros las coordenadas en los ejes x y z, el tercer parámetro es para indicar el color de cada casa mediante un valor entero(1 morado, 2 rosa, 3 verde). Las casas se visualizaran al fondo de la pantalla.

Void árbol (float X, float Z): Las coordenadas recibidas son para dibujar un árbol de tamaño

predeterminado en dicha función se usa GLUquadricObj* que selecciona un estilo de dibujo para formas cuadráticas en mi caso es para dibujar un tipo cilindro que simulara el tronco del árbol y una esfera simulara la copa del árbol.

Void carro (float X, float Z, float C): Los parámetros que recibe son las coordenadas en el eje x y z, el tercer parámetro es un número entero que indica el color del carro (1 azul, 2 rojo y 3 verde).
Void banca (float X, float Z): Es la funcion que se encarga de dibujar las bancas en las coordenadas especificadas en los ejes x y z.

Por último en la función display se invocan todas las funciones que dibujan cada objeto, se actualizan las variables globales camina, EYE_X y CENTER_X también se crean las condiciones para las variables camina, EYE_X y CENTER_X estas condiciones son que cuando cada variable llegue a

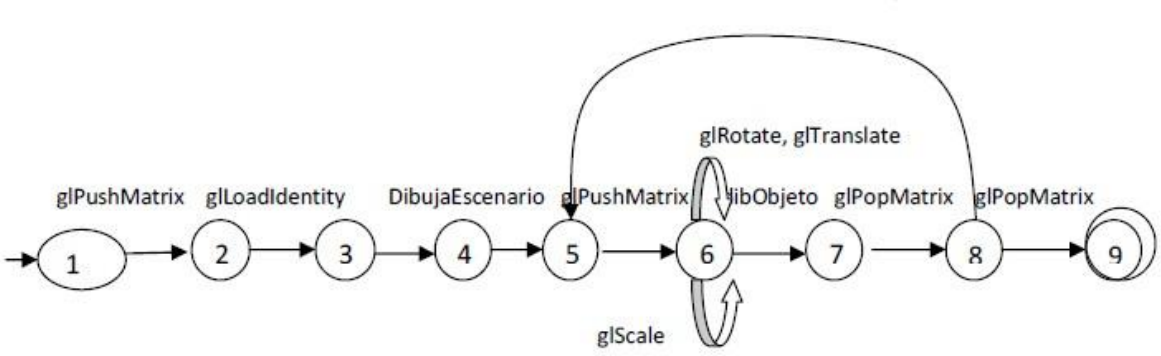
-300 la variable camina se actualiza en 300 y la variable move cambie de signo permitiendo así el

cambio de dirección de la cámara.

Cabe mencionar que cada objeto contiene la invocación a la función lookAt que es la encargada de mover la cámara y su dirección así como también la inclinación de la misma, también cada objeto contiene al principio la llamada de la función que activa la iluminación una vez se dibuje, es decir antes de salir de dicha función se desactiva el modo de iluminación para que de esta manera no afecte a los demás objetos en escena.

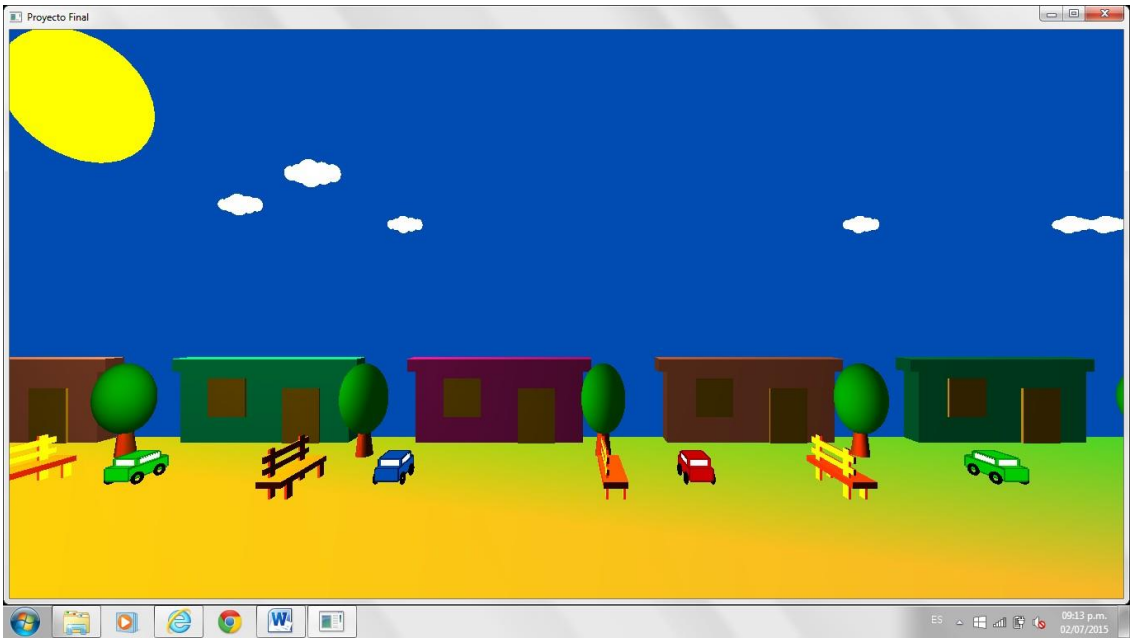
Para la iluminación se uso Ambient: representa el color que va a tener la zona oscura, Diffuse: es el color del objeto, Specular: indica el color que va a tener la zona de maxima luz del objeto, Shininess: Define la cantidad de puntos luminosos y su concentración. Digamos que variando este parámetro podemos conseguir un objeto más o menos cercano al metal.

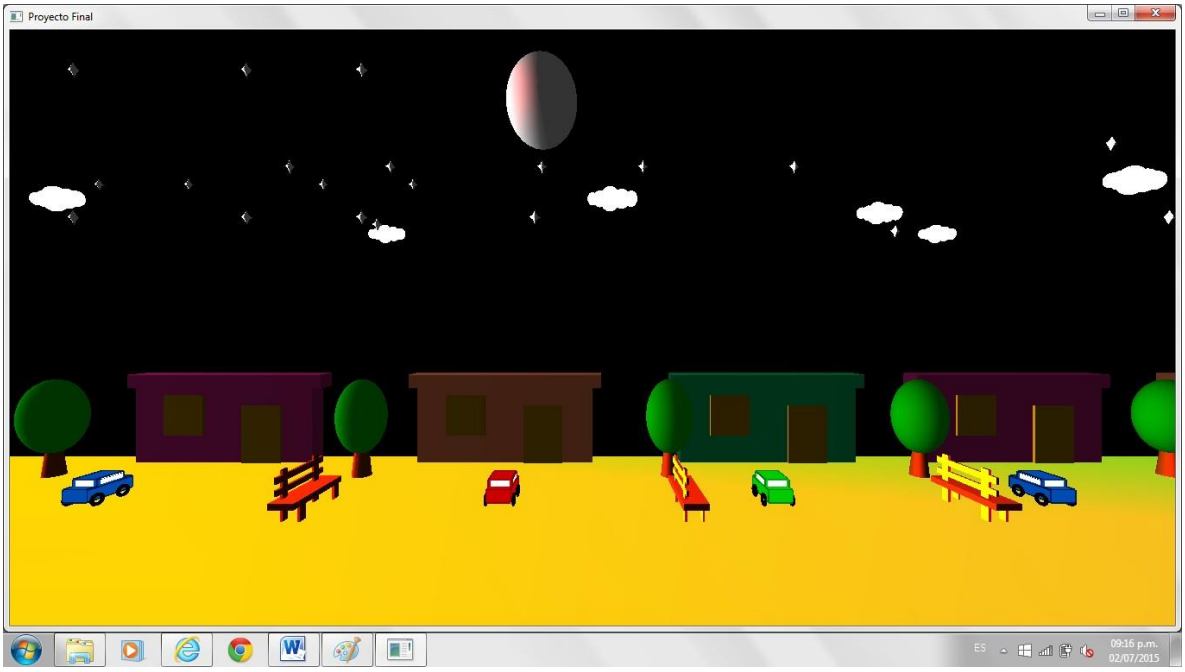
Ya que Opengl funciona como una máquina de estados se puede mostrar mediante un diagrama como se realizó el proyecto.



Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Conclusión





Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Primer Parcial

Graficación Verano 2015 Benemérita Universidad Autónoma de Puebla Facultad de Ciencias de la Computación Alumno: Celso Reyes Gonzalez
11/junio/2015
Profesor: Ivan Olmos Pineda

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

FACULTAD

CIENCIAS DE LA COMPUTACIÓN

MATERIA

GRAFICACIÓN

DOCENTE

IVAN OLMOS PINEDA

TRABAJO

Parcial 1.

ESTUDIANTE

BRAYAN USIEL TOZCANO JIMÉNEZ.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

MATRICULA

201317966

PUEBLA, PUEBLA

09 DE JUNIO DE 2015

Introducción.

A continuación veremos un escenario donde se lograra percibir distintos movimientos como son: Traslación, escalamiento y Rotación. Todo fue construido en base a funciones matemáticas y con ayuda de las funciones de OpenGL llamadas GL_LINES y GL_POINTS.

Conceptos desarrollados.

Para comenzar explicare acerca de cómo forme el escenario. Esto se llevó a cabo con algunas funciones que cree como son: rectángulo (), triangulo () y circulo (). Esto se debe a que el escenario está construido por puras formas geométricas, entonces para no ir creando figura con figura solo es necesario mandar puntos de referencia que son necesarios para crear cada figura. A continuación mostrare las 3 funciones creadas:

```
void rectangulo(float x, float y,float xa,float ya)
{
 glBegin(GL_LINES);
    glVertex2f(x,y);
    glVertex2f(x,ya);
    glVertex2f(x,ya);
    glVertex2f(xa,ya);
    glVertex2f(xa,ya);
    glVertex2f(xa,y);
    glVertex2f(xa,y);
    glVertex2f(x,y);
    glEnd();
}
```

Esta función recibe como parámetros 4 coordenadas. Las primeras dos son un punto (x,y). Las restantes son otras coordenadas que generan el rectángulo. Solo recibe este número de coordenadas ya que es número mínimos para poder formar un rectángulo.

```
void triangulo(float x, float y,float xa, float ya,float xb,float yb)
{
    glBegin(GL_LINES);
    glVertex2f(x,y);
    glVertex2f(xa,ya);
    glVertex2f(xa,ya);
    glVertex2f(xb,yb);
    glVertex2f(xb,yb);
    glVertex2f(x,y);
    glEnd();
}
```

Esta función recibe tres pares de coordenadas, todas estas son de tipo flotante. Al igual que con el rectángulo tiene un número mínimo de puntos para poder saber y formar un triangulo. Pero para esta figura geométrica es necesario 3 puntos.

```
void circulo(int r, int a, int b)
{
    int p,p1;
    glBegin(GL_POINTS);
    for(int i=0;i<=360;i++)
    {
        p=(a+(r*cos(i)));
        p1=(b+(r*sin(i)));
        glVertex2i(p,p1);
    }
    glEnd();
```

Por ultimo veremos la función para generar un círculo con centro en cualquier coordenada. Esta función recibe como parámetros el tamaño del radio(r) y los puntos (x,y) donde queramos que se encuentre el centro de nuestra circunferencia. También es necesario incluir la librería Math.h, esto se debe a que ocuparemos de las funciones sin() y cos() para generar la circunferencia. El ciclo que se encuentra es para ir incrementado el número del ángulo para así lograr los 360 grados de una circunferencia.

Conclusiones.

Gracias a las Traslaciones, escalamientos y Rotaciones podemos realizar grandes cosas, desde hacer mover un objeto simple hasta realizar animaciones con estas tres operaciones. Mientras más controlemos estas funciones lograremos realizar trabajos más profesionales e impactantes. Afortunadamente la librería de OpenGL ya nos ofrece estas funciones si es que nosotros no las queremos realizar por completo o simplemente para facilitarnos la vida.

Bibliografía: Gráficos por computadora con OpenGL. Heam Baker, Pretice Hall.2005.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

FACULTAD

CIENCIAS DE LA COMPUTACIÓN

MATERIA

GRAFICACIÓN

DOCENTE

IVAN OLMOS PINEDA

TRABAJO

PARCIAL 2

ESTUDIANTE

BRAYAN USIEL TOZCANO JIMÉNEZ.

MATRICULA

201317966

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

PUEBLA, PUEBLA
23 DE JUNIO DEL 2015

Introducción.

Como ya vimos en el ámbito 2D existe diferentes operaciones sobre los puntos para poder hacer con ellos distintas cosas como: Traslaciones, Rotaciones o escalamientos. En el siguiente documento explicaremos como se realiza una rotación, traslación y escalamiento de algunos objetos sobre un escenario.

Conceptos desarrollados.

Rotación:

Para comenzar necesitamos establecer los puntos del objeto a rotar y los almacenaremos en una matriz, así será más fácil manipularlos y los tendremos de una forma más ordenada saber que lograr realizar una traslación es necesario antes trasladar, para ello tendremos definida una matriz que será:

1 0 0 xt

0 1 0 yt

0 0 1 zt

0 0 0 1

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Antes de empezar a rotar nuestra figura deberemos definir un eje de rotación, esto se debe a que dependiendo de este será más larga o más corta a la rotación. Después de tener nuestro eje procederemos a sacar el vector unitario. Este lo obtendremos restando los Puntos x,y,z de un extremo del eje de rotación menos x1,y1,z1 del otro extremo. Esto no es todo, a continuación realizaremos la operación siguiente y ahora si tendremos nuestro vector unitario:$\sqrt{x2^2 + y2^2 + z2^2}$ donde x2, y2, z2 es la diferencia de los extremos del eje de rotación. Al resultado de esta operación la llamaremos “va”.

Una vez obtenido deberemos definir los grados que vamos a necesitar rotar a esta variable la llamaremos alfa. Debemos considerar que en caso de ser necesario deberemos convertir a grados por medio de la siguiente operación: ((pi/180)*ángulo). En mi caso los angulos iran incrementando para dar una apariencia de animación. Para continuar deberemos definir unas matrices que vamos a multiplicar para obtener una matriz resultante, esa matriz solo se necesita multiplicar por mis puntos de la figura para obtener la rotación deseada. A continuación muestro como definí mis matrices para poder realizar mis operaciones.

```
float MatrizTraslacion[4][4]={{1,0,0,-eje[0][0]},{0,1,0,-eje[0][1]},{0,0,1,-eje[0][2]},{0,0,0,1}};
float MatrizAnguloA[4][4]={{1,0,0,0},{0,c/va,-1*(b/va),0},{0,b/va,c/va,0},{0,0,0,1}};
float MatrizAnguloB[4][4]={{va,0,-a,0},{0,1,0,0},{a,0,va,0},{0,0,0,1}};
float MatrizRotacion[4][4]={{cos(alfa),-1*(sin(alfa)),0,0},{sin(alfa),cos(alfa),0,0},{0,0,1,0},{0,0,0,1}};
float MatrizTraslacionR[4][4]={{1,0,0,eje[0][0]},{0,1,0,eje[0][1]},{0,0,1,eje[0][2]},{0,0,0,1}};
float MatrizAnguloAR[4][4]={{1,0,0,0},{0,-1*(c/va),b/va,0},{0,-1*(b/va),-1*(c/va),0},{0,0,0,1}};
float MatrizAnguloBR[4][4]={{-va,0,a,0},{0,1,0,0},{-a,0,-va,0},{0,0,0,1}};
float MatRes[4][4];
```

Donde a= x2/va, b=y2/va y c=z2/va.

Para la multiplicación de matrices realice una función que recibe 2 matrices de 4x4 cualquiera y las multiplica. La función es la siguiente:

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

```
void MultiplicaMat(float mat1[][4],float mat2[][4])
{
     for(int i=0;i<=3;i++)
    {
        for(int j=0;j<=3;j++)
        {
            MatRes[i][j]=mat1[i][0]* mat2[0][j]+mat1[i][1]*mat2[1][j]+mat1[i][2]*mat2[2][j]+mat1[i][3]*mat2[3]

        }

    }

}
```

Una vez obtenida la matriz resultante (MatRes) la multiplicaremos por los puntos y obtendremos otros nuevos. Estos los guardaremos en una matriz. Para finalizar solo deberemos dibujar los nuevos puntos.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Traslación:

La Traslación es la operación más fácil desde mi punto de vista, esta se realiza multiplicando los puntos de un objeto a trasladar por una matriz que será de la siguiente manera:

1 0 0 tx

0 1 0 ty

0 0 1 tz

0 0 0 1

Donde tx,ty,tz serán el vector con el cual vamos a trasladar. Si queremos darle efecto de animación solo es necesario actualizar los datos de nuestra figura y volver a pintar. Afortunadamente no necesitamos meterlo en un ciclo, esto se debe a que OpenGL ya tiene un ciclo infinito.

Escalamiento:

Para el escalamiento utilizaremos el mismo método que en 2D, deberemos trasladar un punto pivote al origen, después aplicaremos la escala y por ultimo regresaremos al punto de origen. En este caso vamos a realizarlo por medio de una operación matricial que se define así:

*sx 0 0 (1-sx)*Px*

*0 sy 0 (1-sy)*Py*

*0 0 sz (1-sz)*Pz*

0 0 0 1

El valor de sx, sy y sz será el número de la escala, El valor de Px, Py y Pz serán las coordenadas de punto pivote que elegimos. Solo falta multiplicar por los puntos y obtendremos unas nuevas coordenadas, estas serán las que ya se estarán escaladas, solo hace falta dibujarlas.

Conclusiones.

Las operaciones en 3D son de un mayor grado de complejidad pero podremos hacer cosas mucho más profesionales. Cabe mencionar que esta operación ya las trae OpenGL nativamente. Con este proyecto logre comprender que es lo que traen las funciones ya establecidas y así poco a poco voy comprendiendo y sobre todo aprendiendo más cosas en el tema de la programación mediante OpengGL.

Bibliografía: Gráficos por computadora con OpenGL. Heam Baker, Pretice Hall.2005.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

BENEMRITA UNIVERSIDAD AUTONOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LAS COMPUTACION

MC. IVAN OLMOS PINEDA

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

MATERIA: GRAFICACION ALUMNO:

SANDOVAL SALAZ CONSUELO

PERIODO: VERANO 2015

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Introducción

En este trabajo se realizara un escenario 3D con rotaciones, escalamientos y traslaciones.

Desarrollo

En este proyecto se dibujara un escenario en 3D el cual es un salón de química, con un laboratorista, unas mesas, sillas y matraces.

Para comenzar el código declaramos las librerías de Opengl y variables a utilizar, posteriormente procedemos declarar las funciones funciones para dibujar diferentes objetos en el escenario, se usaron las funciones:

Void pata (void), void superficie (void), void manija (void), fueron llamados para la función void mesa (void).

Void descanso (void), void tubsill (void), void tubpies (void), void cruz(void), fueron llamados para la función void silla(void).

Void cajonlab(void), void cajaagua(void), void manijalab (void), void suplab (void), void llaves (void), void mezclador (void), fueron llamados para la función void lavabo (void).

Void través (void), void cristal (void), void soportes (void), fueron llamados para la función void estante (void).

Void cuarto (void) es la función que dibujara el cuarto de nuestro escenario.

Void cuello (void), void pecho (void), void pecho (void), void pierna (void), void brazo (void), void ojos (void), void boca (void), fueron llamados en la función void laboratorista ().

Void matraz y void matrazc (double x, double y, double z) dibujaran matraces en las mesas con unas bolitas saliendo de ellas.

Opengl funciona como una máquina de estados los cuales se realizar diversas operaciones o cambios tales como cambiar colores, dibujar polígonos, esferas, trasladar, rotar y escalar dichas figuras.

En la función void conjunto (void) (es la función Display) donde se dibujara nuestro escenario.

La funciónes se usara glTranslated(), se llama a la función mesa y se cierra la matriz. Se activa de nuevo la matriz glPushMatrix(), se procede a rotar, trasladar y escalar con las funciones glRotated, glTranslated y glScalef se llama a la función predefinida lavabo para dibujar 3 diferentes lavabos en el escenario se cierra la matriz.

Procedemos a activar nuevamente la matriz, realizamos una traslación con glTranslated llamamos a la función silla y dibujamos 9 sillas en el escenario se cierra la matriz.

Ahora activamos la matriz y rotamos, trasladamos y escalamos con las funciones glRotated, glTranslated, glScalef y llamamos a la función estante para dibujar 3 estantes y cerramos la matriz.

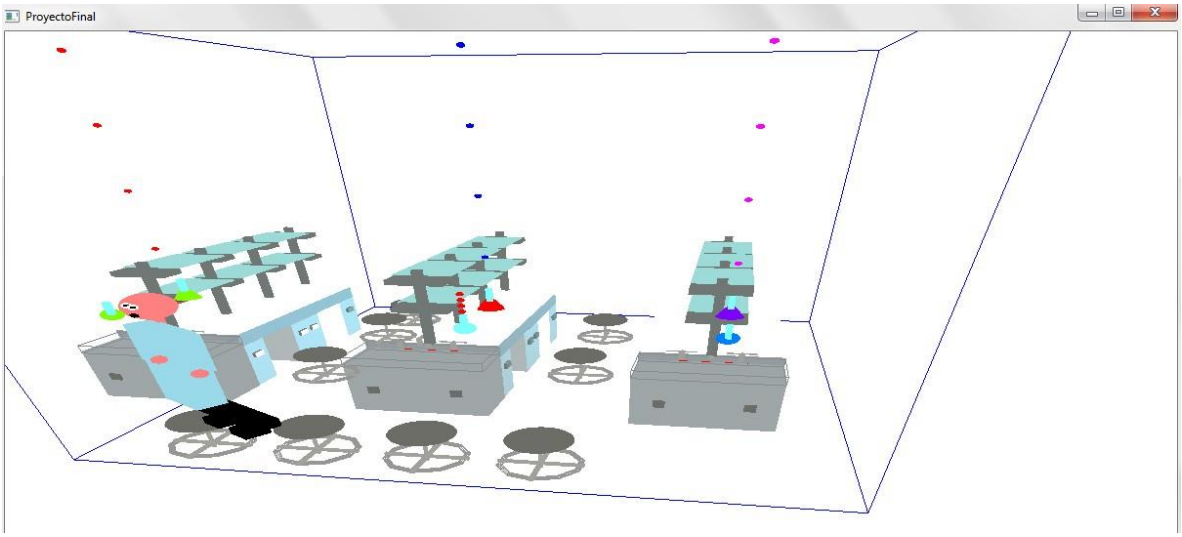
Finalmente declaramos glPushMatrix, trasladamos con glTranslated, llamamos a

la función laboratorista y dibujara a un personaje en el esenario cerramos la matriz con glPopMatrix().

Para lograr el movimiento del personaje, su sombra y las bolitas que salen del matraz se utiliza un for que se inicia de 1 hasta 7 es el número de veces que aparecerán dibujadas en el escenario logrando el efecto de movimiento.

Conclusión

El trabajo realizado nos ayudara a mejorar nuestras habilidades en el manejo de las funciones de Opengl, comprender de una manera más eficiente el funcionamiento de las matrices que utiliza el programa.



Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA

MATERIA:

GRAFICACION

DOCUMENTO TECNICO:

PRIMER EXAMEN PARCIAL

ALUMNO

ANGEL JESUS PACHECO DE LA LUZ

VERANO 2015

MIERCOLES 10 DE JUNIO DE 2015

INTRODUCCION

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

OpenGL y GLUT nos ofrecen varias funciones que nos sirven para realizar manipulaciones, por ejemplo traslaciones, rotaciones y escalamientos, esto es precisamente lo que nos ha interesado en esta ocasión. En esta ocasión nos interesó realizar nuestras propias funciones que realicen las operaciones de rotación, traslación y escalamiento. Una vez que tengamos implementadas estas funciones haremos uso de ellas en una pequeña animación para ponerlas en prueba.

DESARROLLO

Para facilitar un poco la manipulación de los objetos que dibujaremos en la pantalla se realizó una clase llamada “Punto”. Pues según este autor un objeto está compuesto por una serie de puntos con coordenadas “x” e “y” que definen su posición en el plano cartesiano. Así pues la clase Punto tendría como atributos de clase dos variables del tipo entero que almacenarían los valores de las coordenadas “x” e “y”. Como un objeto es una serie de puntos, estos podrían ser almacenados en un arreglo del tipo “Punto”. Dentro del proyecto existe una clase que es una pequeña variante de la clase “Punto” llamada “PuntoF”, siendo la única diferencia el tipo float de las variables que almacenaran los valores los atributos “x” e “y”. El motivo de esta clase es tener valores más precisos al trabajar con rotaciones.

Una vez aclarado como se almacenaran las figuras que se dibujaran en pantalla toca discutir como implementaremos las operaciones de traslación, rotación y escalamiento. Para la implementación nos auxiliaremos de unas matrices. Estas matrices llamadas matrices de operación, serán definidas en nuestro main como matrices de alcance global. Estas matrices tienen algunas de sus componentes ya definidas y solo necesitan de uno o dos argumentos para tener nuestra matriz de operación lista. Dentro del programa llamado main las funciones que se encargan de inicializar ya las matrices con sus valores finales son: la función “escalar()”, la función “trasladar()” y la función “rotar()”. Cabe resaltar que estas funciones solo realizan la operación para un solo tipo de movimiento.

```
void traslacion(int tx, int ty, Punto p[], int tam)
{
    int r[3] = {0};
    traslada[0][2] = tx;
    traslada[1][2] = ty;
    for(int j=0; j<tam; j++)
    {
        int v[3] = {0, 0, 1};
```

```
        v[0] = p[j].dax();
        v[1] = p[j].day();
        r[0] = 0;
        r[1] = 0;
        r[2] = 0;
        for(int i=0; i<3; i++)
        {
            for(int k=0; k<3; k++)
            {
                r[i] = r[i] + ( v[k] * traslada[i][k] );
            }
        }
        p[j].actx(r[0]);
        p[j].acty(r[1]);
    }
    traslada[0][2] = 0;
    traslada[1][2] = 0;
}

void escalar(int ex, int ey, Punto p[], int tam)
{
    int nv[3] = {0};

    escala[0][0] = ex;
    escala[1][1] = ey;
    for( int k=0; k<tam; k++ )
    {
        int v[3] = {0, 0, 1};
        v[0] = p[k].dax();
        v[1] = p[k].day();
        for( int i=0; i<3; i++ )
        {
            for(int j=0; j<3; j++)
```

```
            {
                nv[j] = nv[j] + ( escala[i][j] * v[j] );
            }
        }
        p[k].actx(nv[0]);
        p[k].acty(nv[1]);
        nv[0] = 0;
        nv[1] = 0;
        nv[2] = 0;
    }
    escala[0][0] = 0;
    escala[1][1] = 0;
}

void rotacion(int grados, PuntoF p[], int tam)
{
    int v[3] = {0, 0, 1};
    float nv[3] = {0, 0, 0};

    rotar[0][0] = coseno(grados);
    rotar[0][1] = seno(grados) * -1;
    rotar[1][0] = seno(grados);
    rotar[1][1] = coseno(grados);
    for( int k=0; k<tam; k++ )
    {
        v[0] = p[k].dax();
        v[1] = p[k].day();
        for(int i=0; i<3; i++)
        {
            for(int j=0; j<3; j++)
            {
                nv[i] = nv[i] + ( rotar[i][j] * v[j] );
            }
```

```
        }
        p[k].actx(nv[0]);
        p[k].acty(nv[1]);
        nv[0] = 0;
        nv[1] = 0;
        nv[2] = 0;
    }
    rotar[0][0] = 0;
    rotar[0][1] = 0;
    rotar[1][0] = 0;
    rotar[1][1] = 0;
}
```

Ahora lo único que resta decir es que falto la parte de poder encadenar en una sola matriz las operaciones que quiera realizar. Esto con el fin de optimizar un poco más.

CONCLUSIONES

Honestamente, implementar y sobre todo entender las operaciones de manipulación de figuras que realiza ya opengl no es tarea sencilla. Este autor aun no comprende bien porque utilizar matriz extendida, y mejor como implementar sin de algún tipo de error inesperado.

Tambien comentare que la animacion que se presenta en el programa principal no es totalmente satisfactoria para mi pues no pude realizar funciones del todo genericas, es decir, funciones que solo tomen los puntos de cualquier figura y relizen su manipulacion sin problemas. Tampoco se logro realizar la multiplicacion de matrices.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Colegio: Benemérita Universidad Autónoma de Puebla

Nombre: Juan Pablo Pavón Robledo

Matricula: 201304414, 201325916, 201320395 y 201243571

Profesora: Ivan Olmos Pineda

Materia: Graficación

Trabajo: Documento técnico

Fecha: 03/07/2015

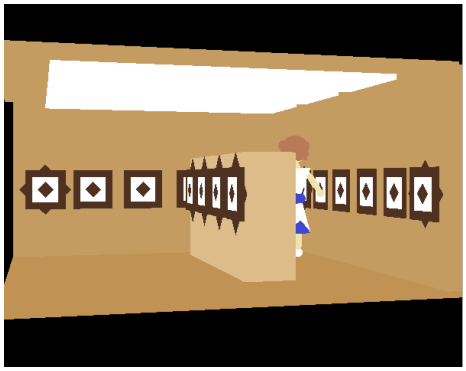
Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

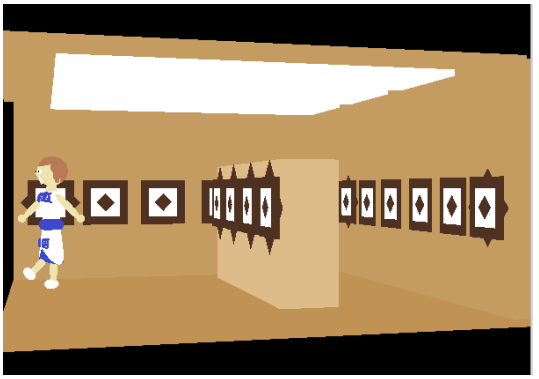
"Documento técnico"

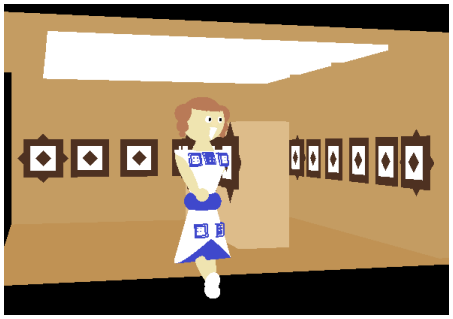
El objetivo no lo logre las texturas no se dejaron debido a que se corrompían los archivos, intente con 2 manejadores gráficos que son el FreeImage y el glaux desde una versión más antigua pero no se dejó, lo intente compensar con el uso básico de luz basado en el libro: beginnin opengl gamer programming.

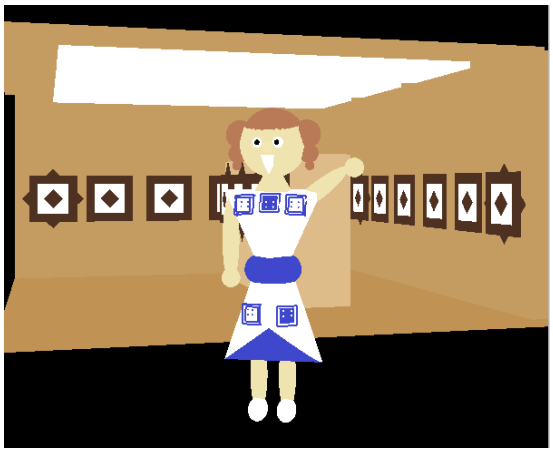
El escenario esta realizado en un cubo donde una muñeca realizada en 3d usando lo que asemeja un vestido mexicano (con motivos de talavera) recorre un segmento de museo donde se demuestra lo visto en el curso como son las rotaciones, translaciones, escalamiento y el uso de perspectiva; para después acercarse al espectador y despedirse.

Los resultados son los siguientes:









Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

VERANO 2015

GRAFICACION

DOCUMENTO TECNICO

PROYECTO

ANGEL JESUS PACHECO DE LA LUZ

03 de julio de 2015

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

INTRODUCCION

Como práctica final del curso de graficacion se pidió elaborar una escena en 3d usando cualquier recurso disponible en OpenGL, primitivas de OpenGL para dibujar figuras geométricas básicas, etc.

DESARROLLO

Pasemos a explica cómo funciona brevemente la escena. Como en esta ocasión no fue necesario el uso de matrices no habrá tanto código que explicar. La escena principal la componen una casa, un árbol, unos muros que rodean la propiedad de la casa y un ambiente semihumedo.

Para dibujar la casa se usó la primitiva de Opengl “GL_QUADS”. Esto principalmente para construir los muros y poder aplicar las texturas con mayor facilidad. Explicaremos un poco mejor lo de las texturas más adelante. Para la parte del techo de la casa se usaron las primitivas “GL_QUADS” y “GL_TRIANGLES” esto por la forma de la chimenea. Finalmente para la parte de la chimenea de la casa se usó nuevamente las primitivas “GL_QUADS” para poder aplicar con facilidad las texturas.

Ahora expliquemos como se construyeron los muros. Los muros no tienen mayor dificultad pues nuevamente nos hemos apoyado en el uso de la primitiva “GL_QUADS” y como anteriormente ya mencione, se elige esta forma de construir para tener facilidad a la hora de aplicar texturas. La pared realmente no tiene mucha importancia solo es para delimitar la escena.

La ambientación de la escena es un clima lluvioso, y como graficar eso tomaría mucho pero mucho tiempo, nos apoyaremos en el uso de las texturas. He aquí una gran utilidad de las texturas permiten dar mayor estética a una escena sin invertir tanto tiempo en el modelado de un objeto en específico. Para la ambientación en esta escena usamos las imágenes “nube y nube3” y las aplicamos directo aun plano alejados de los límites de la escena, esto da la ilusión de ambiente húmedo.

Hablemos del árbol, realmente la construcción de este fue muy práctico, pues solo fue necesario situar un cubo deformado en algún lugar de la escena y colocarle encima de este una esfera del color que uno quiera y un numero de esferas más chicas que la primera esfera para simular frutos. Lo único que hay de nuevo en su construcción fue el uso de las funciones “glPushMatrix()” y “glPopMatrix()”, pero entender el uso de esto es relativamente fácil. “glPushMatrix” almacena la matriz sobre la que estamos operando actualmente y la

almacena en una pila de matrices que maneja Opengl, y “glPopMatrix” saca la última matriz que se almaceno en la pila de matrices.

Hablemos del último objeto que se muestra en pantalla, la lluvia. Para dibujar la lluvia se utilizó una clase llamada “lluvia”. La clase lluvia no es tan extensa, lo que facilita muy rápido su comprensión. La clase lluvia consta de los métodos constructores por defecto y uno con argumentos, el método de dibujar gota, y un método llamado destrucción. Los métodos constructores son los que construyen una gota, especificando o no argumentos a la hora de construir la gota. Los métodos constructores se utilizan para inicializar los miembros de la clase. Si se especifican valores, deben especificarse 4 valores, 3 para las coordenadas de “x”, “y”, “z” y un cuarto argumento para especificar la velocidad de caída de la gota. El método llamado dibuja gota como su nombre los indica solo dibuja la gota, esta función se apoya de la primitiva “GL_TRIANGLES” para su dibujado en pantalla. Y finalmente el método destrucción tiene el objetivo de saber cuándo la gota ha tocado suelo y devolver la gota al cielo con unos nuevos valores “x”, “y” y “z”.

Lo que es tema nuevo en este proyecto es el uso de texturas. Para comenzar a usar una textura en Opegnl debemos tener una imagen en formato bmp, además debemos asegurarnos de que el ancho y alto de la imagen estén en potencia de 2. Una vez tengamos nuestras imágenes debemos cargarlas a Opengl, esto lo realiza la función readImage(). La funcion “readImage()” recibe como parámetros un apuntador de tipo char que contiene el nombre de la imagen, incluyendo su extensión, y un entero que será el identificador que le asignaremos a la textura. La función realiza sus operaciones necesarias para cargar la textura a opengl, entre ellas están las de obtener el formato de la imagen, el ancho y alto, transformarla a imagen de 24 bits de profundidad, cargarlas en memoria, obtener los bits cargados en memoria, especificar como le asignaremos las coordenadas a la textura a la hora de aplicarla y que filtros usara. Si la función tiene éxito la textura estará cargada en opengl con su respectivo identificador lista para usarse. Para usar la textura, en especial si son muchas, nos apoyaremos de la función “glBindTexture”, esta función especifica que textura usaremos en ese momento.

Resumiendo, si queremos usar texturas en nuestro programa seguiremos estos pasos:

1. Cargar la textura en Opengl. Esto para obtener el identificador de la textura.
2. Habilitar las texturas en opengl mediante la función “glEnable(GL_TEXTURE_2D)”.
3. Especificar las coordenadas de la textura que tomaremos cada que se dibuje un punto. Esto lo interpreto como tomar solo un pedazo de la textura que me interesa, por ejemplo, en el caso de tener una textura con varios dibujos en ella.
4. Una vez terminado de aplicar la textura deshabilitarla con la función “glDisable(GL_TEXTURE_2D)”.

Básicamente es lo que necesitamos saber para el uso de texturas.

CONCLUSION

Durante el desarrollo de esta práctica, y todo el curso en general, he comprendido lo difícil y laborioso que resulta realizar este tipo de motores gráficos, también me llevo algunas lecciones de como cargar y usar una textura.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Alberto Esteban Reyes Peralta
Matricula: 201410180
Graficación

BENEMÉRITA UNIVERSIDAD

AUTÓNOMA DE PUEBLA

FACULTAD EN CIENCIAS DE LA

COMPUTACIÓN

2° do. Parcial

“TRANSFORMACIONES DE TRASLACION, ESCALA Y ROTACION EN 3D”

MATERIA: GRAFICACIÓN

PROF: IVAN OLMOS PINEDA

ALUMNO: LUIS DAVID MORALES ALCÁNTARA

23/06/15

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

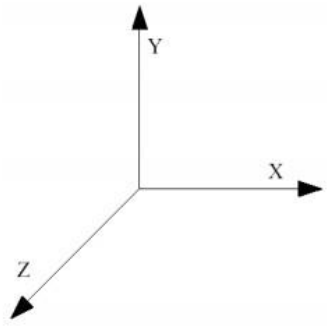
INTRODUCCION

En el siguiente documento hablare los temas de traslación, escalamiento y rotación en 3D que son temas que se vieron clase y explicaremos un poco sobre ello; el tratar un poco sobre el color de polígonos de las diferentes figuras creadas en nuestro escenario y poder manipular distintos objetos mediante la ayuda de las matrices para realizar cada transformación mencionado con su respectiva función.

CONCEPTOS DE DESARROLLO

OpenGL: Representación de Objetos 3D

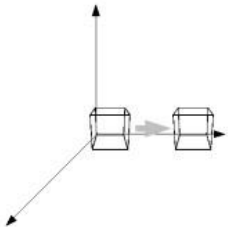
- **OpenGL utiliza una representación de objetos 3D a partir de un espacio en cuatro dimensiones**
  - El espacio 3D se representa a través de un sistema 3D ortonormal, donde los ejes son perpendiculares y cada unidad en cada eje esta representado por un vector de módulo 1
  - La cuarta coordenada se utiliza para representar la perspectiva



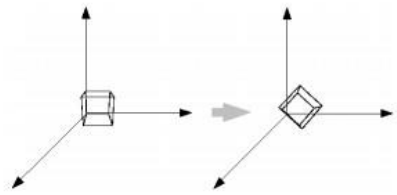
Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

OpenGL: Transformaciones de Objetos

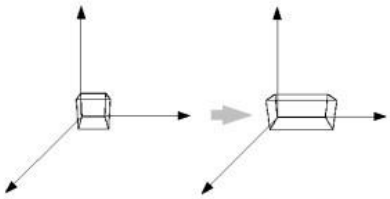
- Existen 3 operaciones básicas para transformar un objeto:
  - Traslación: desplazamiento de un objeto en el espacio



  - Rotación: rotar un objeto a partir de su centro de giro



  - Escalado: alterar el tamaño de un objeto



OpenGL: transformaciones de objetos

- Toda transformación construye una matriz de cuatro dimensiones, que es multiplicada por la matriz original de coordenadas
- Por ejemplo, para trasladar un objeto 2 unidades en el eje X

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- A partir de la matriz anterior, dibujar un punto en las coord (1,0,0) será:

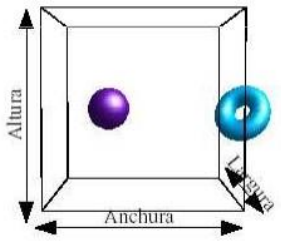
$$\begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

- Algunas formas de dibujado
  - GL_POINTS: se dibujan vértices separados
  - GL_LINES: cada par de vértices definen una linea
  - GL_POLYGON: todos los vértices definen el contorno de un polígono
  - GL_TRIANGLES: cada triplete de vértices definen un triángulo
  - GL_QUADS: cada cuarteto de vértices se interpreta como un cuadrilátero

OpenGL: proyección

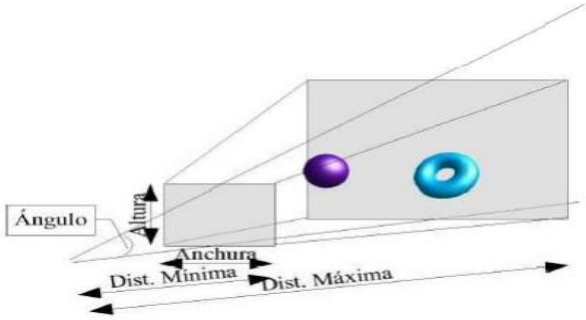
- Existen dos formas de proyectar objetos
  - Proyección ortográfica: permite visualizar todo lo que se encuentre dentro de un cubo, delimitado por los parámetros de la función glOrto



Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

OpenGL: proyección

- **Proyección perspectiva**
  - Delimita un volúmen de visualización dado por un ángulo de cámara y una relación alto/ancho. La distancia al observador delimitará el tamaño con el que un objeto se visualiza



```
gluLookAt(2,1,2,0,0,0,0,1,0);

glClearColor (1.0, 1.0, 1.0,0.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-60.0, 60.0, -30.0,30.0,-20.0,40.0);
```

Ocupamos la función de “lookat”, para poder colocar al observador en una posición en el espacio vectorial viendo hacia un punto en nuestro escenario

```
trasbote[0][0]=2;
trasbote[0][1]=0;
trasbote[0][2]=0;
trasbote[0][3]=0;

trasbote[1][0]=0;
trasbote[1][1]=2;
trasbote[1][2]=0;
trasbote[1][3]=0;

trasbote[2][0]=0;
trasbote[2][1]=0;
trasbote[2][2]=2;
trasbote[2][3]=0;
```

Creamos una matriz de traslación , de escalamiento y traslación las cuales serán utilices para cualquier comportamiento de nuestras figuras.

```
oid color (void)

   glColor3f(.5,1,.5);
   glBegin(GL_POLYGON);//Relleno de pastito
   glVertex3f(-100,0,0);
   glVertex3f(-100,0,-100);
   glVertex3f(100,0,-100);
   glVertex3f(100,0,0);
   glVertex3f(-100,0,0);
   glEnd();

   glColor3f(0,0,1);
   glBegin(GL_POLYGON);//Relleno de agua
   glVertex3f(-100,0,0);
   glVertex3f(-100,0,-20);
   glVertex3f(100,0,-20);
   glVertex3f(100,0,0);
   glVertex3f(-100,0,0);
   glEnd();
```

Creamos una funcion la cual tendra la tarea de rellenar las figuras con los colores que deseemos utilizar.

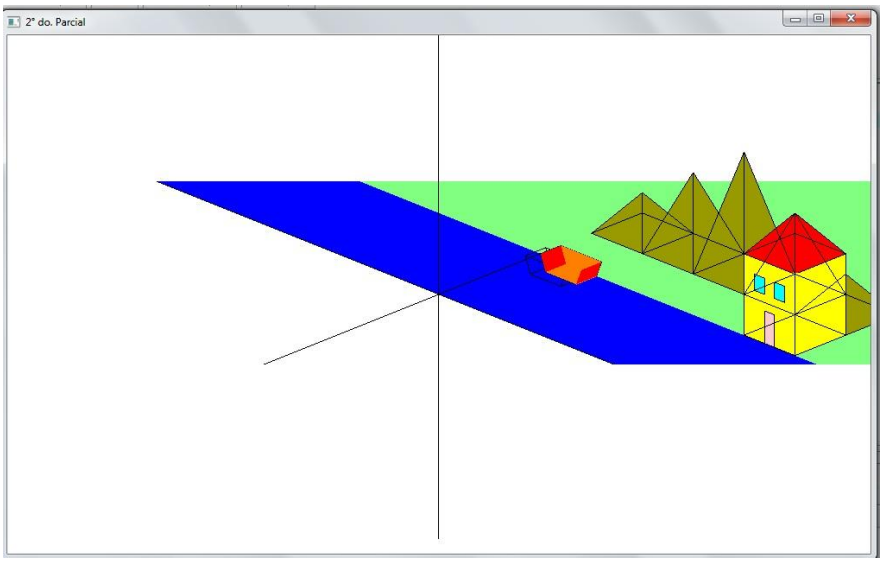
```
  glColor3f (0.0, 0.0, 0.0);
  glClear (GL_COLOR_BUFFER_BIT);
  color();
 // bote();

  glBegin(GL_LINES);

  for(int i=0; i<6;i++)
  {

      glVertex3i(ejes [i][0], ejes [i][1], ejes [i][2]);

  }

  ///////////////

  for(int i=0; i<16;i++)
  {

      glVertex3i(mont [i][0], mont [i][1], mont [i][2]);

  }
```

Con esta función hacemos que se dibujen las líneas de nuestras figuras.

CONCLUSIONES

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Mediante la investigación sobre la función de Sleep () y el manejo correcto de dicha función al igual que el resultados de operaciones logramos obtener un escenario con traslación y escalamiento con los conceptos aprendidos en el curso.



BIBLIOGRAFIA

Gráficos por computadora con OpenGL,09/06/15, http://www.opengl.org/sdk/docs/man/

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA

GRAFICACION

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

SEGUNDO EXAMEN PARCIAL

ANGEL JESUS PACHECO DE LA LUZ

22 DE JUNIO DE 2015

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

INTRODUCCION

La actividad a realizar en este segundo examen parcial es la de implementar las operaciones fundamentales de manipulación para objetos en un ambiente de 3 dimensiones. Las operaciones que se implementaron fueron las de traslación, rotación y escalado.

DESARROLLO

Pasemos a explicar la implementación de las operaciones con ayuda de las matrices extendidas. Tal y como en las notas del curso se explica para implementar las operaciones se hará utilizando matrices extendidas.

Primero explicaremos como se realizó la operación de traslación. En el programa se definió una matriz de tipo flotante que almacenara los valores, estos valores servirán para mover a un objeto a través del espacio. Esto lo realiza la función llamada “traslada”, la función toma como argumentos 3 números de tipo entero que representaran el desplazamiento en “x”, “y” y “z” respectivamente. El procedimiento para aplicar el operador de traslación a un objeto se realiza de manera directa, pues solo será necesario asignar a nuestra matriz de traslación los valores que se tomaron en la función como argumentos quedándonos la matriz de la siguiente forma:

$$M_{traslacion} = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Con la matriz actualizada solo nos queda multiplicar la matriz por el arreglo que contiene los valores de los puntos que representan a una figura en el espacio tridimensional. La operación de multiplicar la matriz por el arreglo de puntos la realiza la función llamada “multiplicap”. Una vez realizada la multiplicación de la matriz de traslado por el arreglo que contiene los valores de nuestros puntos solo nos queda mandar a llamar a nuestra función que dibuje en pantalla los puntos ya modificados.

En segundo lugar explicaremos la función de traslación. Según la teoría explicada la traslación se realiza multiplicando 3 matrices. Estas matrices serían las de traslado, rotación y nuevamente la de traslado. La primera matriz seria para trasladar el objeto al origen del sistema de coordenadas, la segunda aplicaría la operación de escalado y la última aplicaría el traslado a la posición original que tenía el objeto. Afortunadamente en lugar de multiplicar 3 matrices (traslación, escalado y nuevamente la de traslado) se

asigna directamente los valores que deseamos se escalen y multiplicamos por los puntos. La función llamada "escala" es la encargada de dejar la matriz de escala en forma de matriz identidad (1's en la diagonal) después asignamos los valores que recibe como argumentos a la matriz de escalado, dejándonos la matriz de la siguiente manera:

$$Mescala = \begin{bmatrix} [sx] & [0] & [0] & [1-sx * px] \\ [0] & [sy] & [0] & [1-sy * py] \\ [0] & [0] & [sz] & [1-sz * pz] \\ [0] & [0] & [0] & [1] \end{bmatrix}$$

Una vez actualizados los puntos almacenados en el arreglo lo único que resta es mandar a dibujar los puntos.

Finalmente queda explicar la última operación de manipulación de objetos, la rotación. Según la teoría vista para aplicar una rotación libre, es decir en cualquier sentido, habrá que implementar una rotación general. Implementar una rotación general significa multiplicar 7 matrices (traslación a 0,0,0, rotación en el eje x, rotación en eje y, rotación en z, rotación inversa a la hecha anteriormente al eje y, rotación inversa a la hecha anteriormente en el eje x, traslación inversa para devolver objeto a su posición inicial). El método que realiza todas las multiplicaciones antes mencionadas se llama rotación. Expliquemos un poco mejor como trabaja la función traslación. Antes que nada hay que explicar que la función trabajara con 4 matrices llamadas matrizrx (matriz para rotar en x), matrizry (matriz para rotar en y), matrizrz (matriz para aplicar la rotación final al objeto), la matriz de traslación (para mover al punto "0,0,0" y devolver al origen la figura) y una matriz que acumulara los resultados. Lo primero que hará la función rotar es vaciar la matriz acumulada. Esto lo hace la función llamada "vaciarmatriz" recibe como argumento una matriz. Seguidamente calculara los componentes del vector de traslación, después calculara los componentes del vector unitario. Tal y como se mencionó anteriormente será necesario multiplicar 7 matrices en un orden especifico, pero antes de mandar a multiplicar nuestras primeras matrices, inicializamos la matriz de traslación con la función "cargaidentidad" y le asignamos los valores necesarios para trasladar el punto p1 al origen. Quedándonos de la siguiente manera:

$$Mtraslacion = \begin{bmatrix} [1] & [0] & [0] & [-x] \\ [0] & [1] & [0] & [-y] \\ [0] & [0] & [1] & [-z] \\ [0] & [0] & [0] & [1] \end{bmatrix}$$

La segunda matriz que necesitaremos preparar será la matriz de rotación en x. Cargamos la matriz identidad en la matriz de rotación en x y le asignamos el coseno y el seno de ángulo formado entre el vector unitario respecto al plano xz. Quedándonos la matriz de rotación en x de la siguiente manera:

$$Mrotacionx = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sen(\theta) & 0 \\ 0 & \sen(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Una vez cargada nuestra matriz con los valores necesarios mandamos a multiplicar las matrices con la función “multiplicam”. Esta función multiplica dos matrices y el resultado lo almacena en una tercera matriz, esta tercera matriz será la matriz acumulada. La siguiente matriz que se necesita multiplicar es la matriz de rotación en y, pero antes de eso primero se carga con la identidad con la función “cargaidentidad” asignamos sus valores correspondientes al nuevo ángulo que forma el vector con el eje z, quedándonos una matriz con estos valores:

$$Mrotaciony = \begin{bmatrix} \cos(\theta) & 0 & -\sen(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sen(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Una vez multiplicada la matriz de rotación “mrotaciony” con la matriz “matriz acumulada” solo queda preparar la última matriz, la matriz de rotación en z. El procedimiento es el mismo, cargamos con la matriz identidad y calculamos sus valores que corresponderán a los grados que queramos rotar a nuestro objeto, tal como se muestra abajo. Una vez calculados sus valores la mandamos a multiplicar con la matriz acumulada.

$$Mrotacionz = \begin{bmatrix} \cos(\theta) & -\sen(\theta) & 0 & 0 \\ \sen(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Las últimas tres matrices son la inversa de la matriz de rotación en y, la inversa de la matriz de rotación en x y la inversa de la matriz de traslación. Calcular la inversa de las matrices es simplemente multiplicar por el

escalar -1 a toda la matriz. Por ultimo mandaríamos a multiplicar cada una de esas 3 matrices por la matriz acumulada y finalmente tendríamos nuestra matriz lista para multiplicar por el arreglo que contiene nuestros puntos.

El resto de las funciones son fáciles de entender porque realizan tareas muy pequeñas por ejemplo están las que inicializan opengl y glut, las que mandan a imprimir la figura en pantalla, las que multiplican matrices y tienen como argumento 3 o 2 matrices siempre guardando el resultado en su último argumento, las que calculan el seno y coseno de un ángulo dado, la que carga la identidad en una matriz que recibe como argumento, la que vacía una matriz aunque realmente no vacié sino que solo pone a 0 todos los valores de la matriz, etc.

CONCLUSION

Solo como comentario y como un aprendizaje, en esta práctica se entendió mejor cómo funcionan los motores gráficos desde lo más “bajo” prácticamente, como realizan las operaciones de traslado, rotación y escalado.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Facultad Ciencias de la Computación

Ingeniería en Ciencias de la Computación

Graficación

Primer Parcial

Fernando Ángel García Parra

Profesor: Dr. Iván Olmos Pineda

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Transformaciones en dos dimensiones

Los objetos se definen mediante un conjunto de puntos. Las transformaciones son procedimientos para calcular nuevas posiciones de estos puntos, cambiado el tamaño y orientación del objeto.

Las operaciones básicas de transformación son

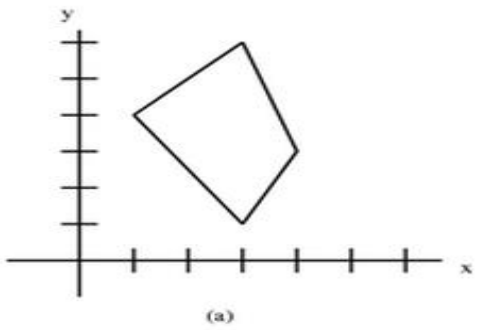
- Traslación
- Escalamiento
- Rotación.

Traslación

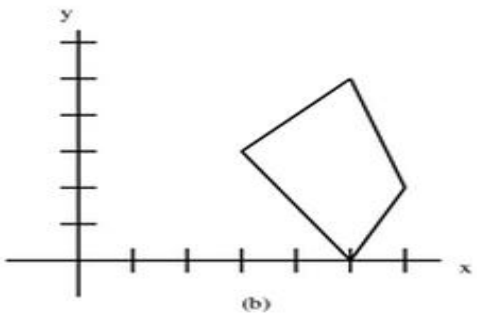
Las coordenadas (x, y) de un objeto se transforman a (x', y') de acuerdo a las fórmulas:

$$x' = x + T_x, \qquad y' = y + T_y$$

El par (T_x, T_y) se conoce como vector de traslación.



(a)



(b)

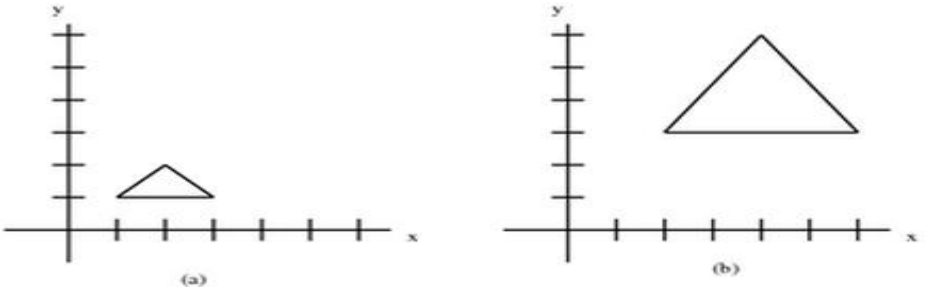
Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Escalamiento

El escalamiento modifica el tamaño de un polígono. Para obtener este efecto, se multiplica cada par coordenado (x, y) por un factor de escala en la dirección x y en la dirección y para obtener el par (x', y').

Las fórmulas son

$$x' = x \cdot S_x \qquad y' = y \cdot S_y$$



Rotación

La rotación gira los puntos de una figura alrededor de un punto fijo. De la figura se obtiene

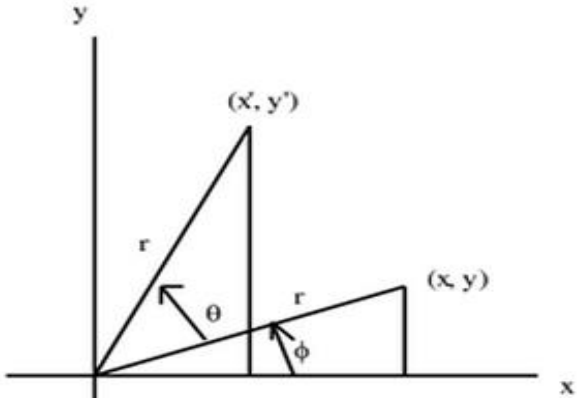
$$x' = r\cos(\phi + \theta) = r\cos\phi\cos\theta - r\,\text{sen}\,\phi\,\text{sen}\,\theta$$

$$y' = r\,\text{sen}(\phi + \theta) = r\,\text{sen}\,\phi\cos\theta + r\cos\phi\,\text{sen}\,\theta$$

Simplificando

$$x' = x\cos\theta - y\,\text{sen}\,\theta$$

$$y' = y\cos\theta + x\,\text{sen}\,\theta$$



Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Representación matricial de traslaciones

Haciendo uso de coordenadas homogéneas la traslación puede representarse como:

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}$$

En forma abreviada la transformación se representará por $T(T_x, T_y)$

$$T(T_x, T_y) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix} \qquad P' = P \cdot T(T_x, T_y)$$

Representación matricial de escalamientos

Haciendo uso de coordenadas homogéneas el escalamiento puede representarse como:

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

En forma abreviada la transformación se representará por $S(S_x, S_y)$

$$S(S_x, S_y) = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad P' = P \cdot S(S_x, S_y)$$

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Representación matricial de rotaciones

Haciendo uso de coordenadas homogéneas la rotación puede representarse como:

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \begin{bmatrix} \cos\theta & \text{sen}\,\theta & 0 \\ -\text{sen}\,\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

En forma abreviada la transformación se representará por $R(\theta)$

$$R(\theta) = \begin{bmatrix} \cos\theta & \text{sen}\,\theta & 0 \\ -\text{sen}\,\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad P' = P \cdot R(\theta)$$

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

NOMBRE: Esteban Dávila Muñoz

CICLO: VERANO 2015

MATEIRA: Graficación

INTRODUCCION

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

En el presente trabajo se pone en práctica el aprendizaje de las primitivas básicas de opengl las cuales son: glpoints y glines en sus diferentes opciones que nos ofrecen.

También la utilización de matrices las cuales nos ayudaran a realizar los movimientos clásicos de graficación los cuales son: traslación, rotación y escala, para ello se emplearan las siguientes matrices según el movimiento que se desee:

DESARROLLO

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

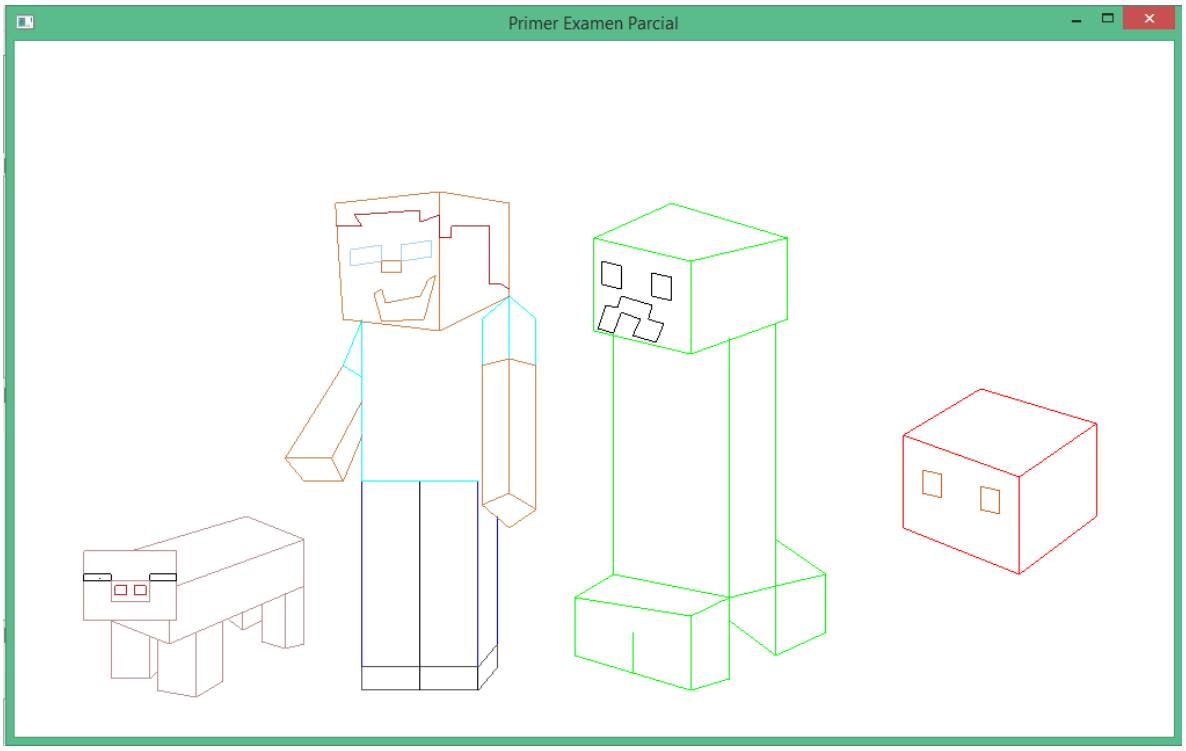
Como se mencionó anteriormente el objetivo del examen es aplicar lo aprendido con anticipación y esto se llevara a cabo mediante la creación de un pequeño escenario en el cual solo se podrán usar las primitivas de puntos y líneas, además se deberá realizar los movimientos (escala, translación o rotación).

Una vez planeado el escenario que se realizara se procede a dibujar línea por línea cada uno de los elementos que serán necesarios para el escenario, en esta ocasión decidí basarme en los personajes de un videojuego llamado “Minecraft”

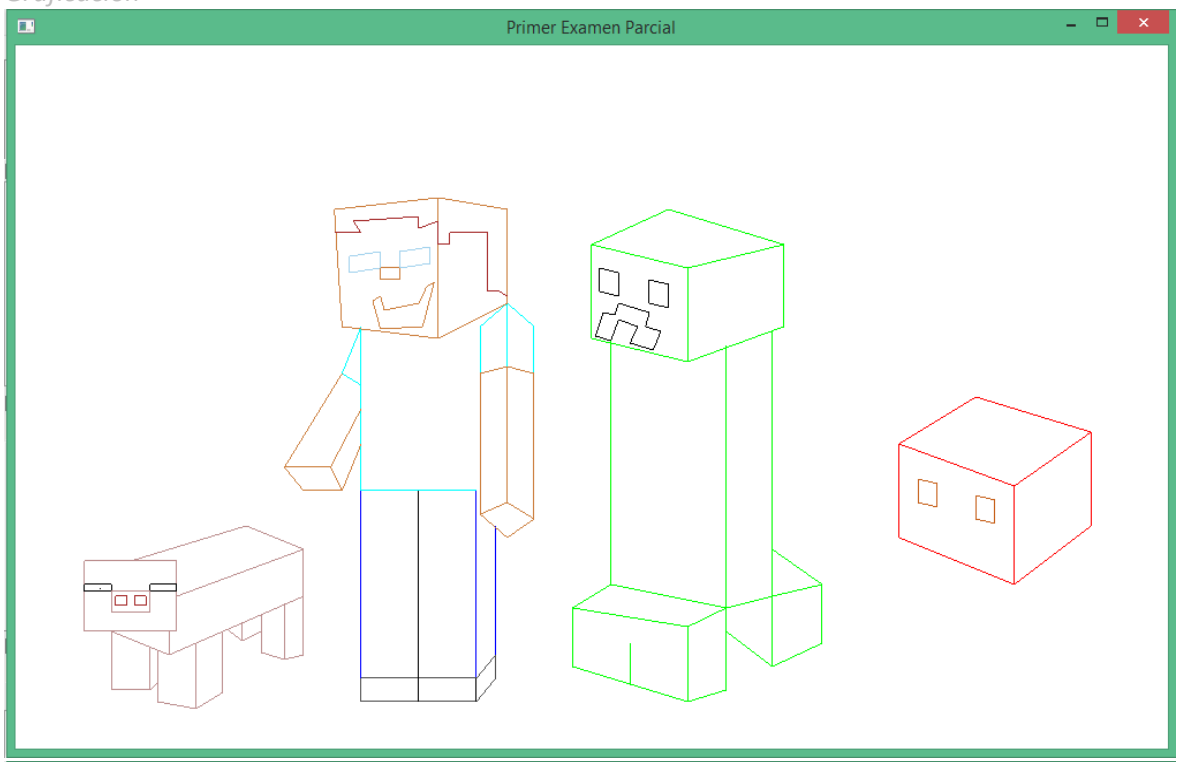
Viendo los personajes se me ocurrió que le podría dar un poco de ilusión para que los personajes se vean como si fuera 3D, lo único complicado para llevarlo a cabo era tener una buena perspectiva de cómo se verían las figuras con cierto ángulos de inclinación, pero todo salió bien en ese asperto o eso espero.

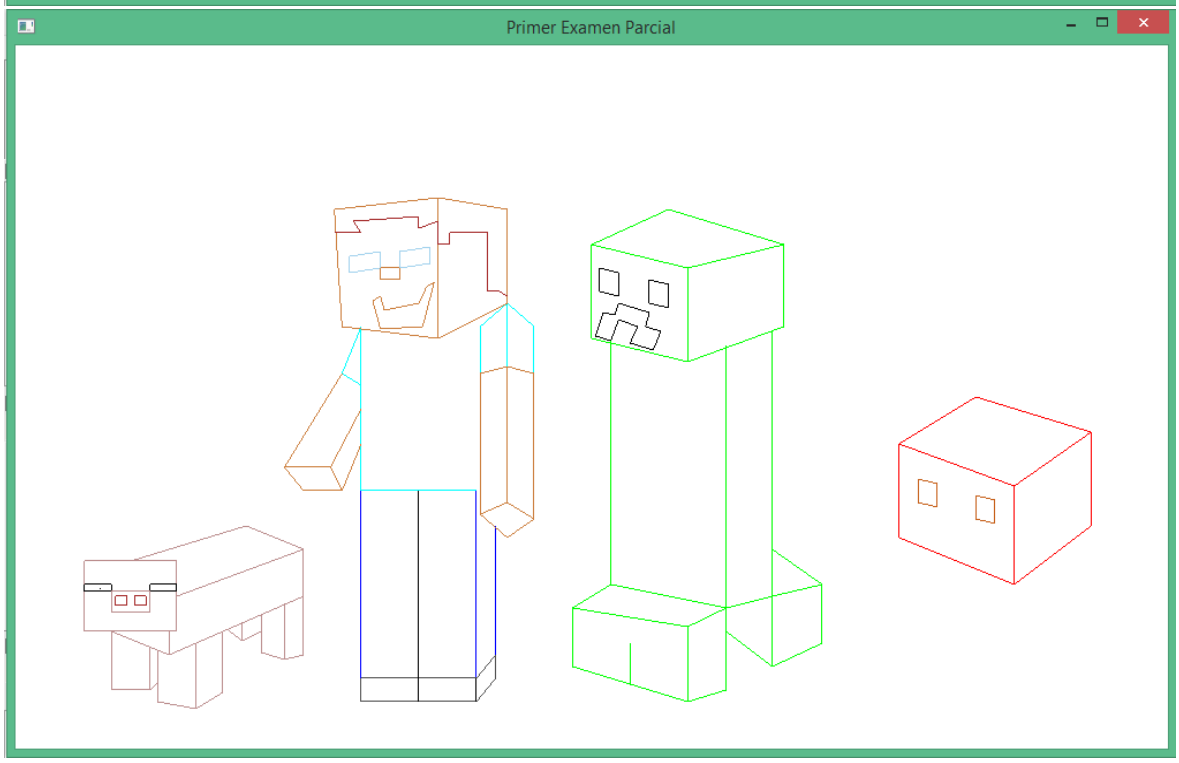
El problema más grande que se me presento fue los movimientos ya que no supe aplicarlos de la manera correcta ya que como se mencionó en clases pasadas opengl tiene un ciclo interno el cual sirve para dibujar las imágenes y quería utilizar dicho ciclo para poder los movimiento sin embargo no lo logre.

Así que el resultado final es el siguiente



Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación





Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Benemérita Universidad Autónoma de Puebla

Facultad Ciencias de la Computación

Ingeniería en Ciencias de la Computación

Graficación

Segundo Parcial:
Operaciones con matrices - 3d

Fernando Ángel García Parra

Profesor: Dr. Iván Olmos Pineda

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Introducción

En la siguiente práctica se tratara de desarrollar un programa en el cual se implementen las operaciones de rotación traslación y escalamiento, dadas en clase las cuales relacionan los operadores de matrices y de vectores. Se debe desarrollar un escenario de manera libre en un ambiente 3D.

Desarrollo

La aplicación que se desarrollo fue el tratar de hacer un cubo de rubik el cual girara en sentido horario de los tres ejes x, y, z. También se trata de implementar es escalamiento del cubo, sin que el cubo sufra alguna deformación después de cada una de las operaciones.

El desarrollo se comenzó por la definición de las matrices para cada una de las operaciones que se podían realizar así como de las variables que se estarían usando en los procesos a ejecutar durante la activación del programa.

Las matrices se declararon de manera global para facilitar el llenado y actualización de los datos.

Una vez que todo estaba declaro se comenzó con la operaciones las cuales se realizaban de manera estructurada y esto dificultaba las operaciones que tenía que realizar el procesador por lo cual el programa tardaba un poco en mostrar los nuevos resultados.

Así que se decidió adquirir e implementar una nueva forma de operar los matices, la cual es de la siguiente manera.

```
void rotacionz(float v[4][3],int an)
{
    int i,j,k;
    float vec[100];
    float tmp;
    matrizidentidad();
    float ang = an*M_PI/180.0;
    ////////////////rotacion z
```

```
    RZ[1][1] = cos(ang);
    RZ[1][2] = -sin(ang);
    RZ[2][1] = sin(ang);
    RZ[2][2] = cos(ang);
    ////////////////////////////////////////////////multiplicacion
    for(k = 1;k<=4;k++){
    vec[1] = v[k][1];
    vec[2] = v[k][2];
    vec[3] = v[k][3];
    vec[4] = 1;
////////////////////////////////
    for(i=1; i<=4; i++){
        tmp=0;
        for(j=1; j<=4; j++){
            tmp += (RZ[i][j]*vec[j]);
        }
        if(i==1)
        v[k][1] = tmp;
        if(i == 2)
        v[k][2] = tmp;
        if(i == 3)
        v[k][3] = tmp;
    }
    }
}
```

En esta estructura se pasan como parámetros la matriz con la que se desea trabajar ya sea rotación traslación, escalamiento y se le pasa el ángulo o el tipo de escala que se desea realizar.

Esto nos permite que el código no sea tan extenso y que facilite la comprensión del mismo ya que cada vez que se necesite solo se tiene que llamar a la función.

```
    rotacionx (vectorc, angulox);
```

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

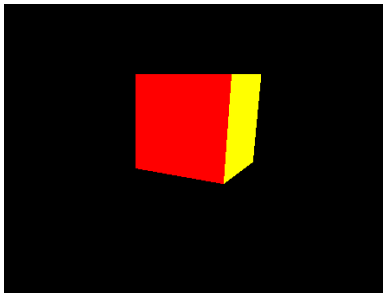
Otras funciones que se implementaron son las siguientes:

llenado();

Esta función nos permite llenar los vectores de cada una de las figuras en las coordenadas originales para que estas no se pierdan y poder utilizarlas en el momento que se requieran.

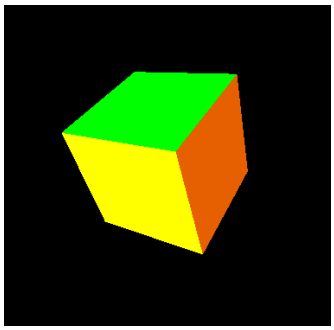
Keyboard();

Función dada por opengl que nos permite la lectura de datos mediante el teclado, se implementó ya que como se deseaba trabajar con un cubo de rubik, el usuario es que intentara mover el cubo como le convenga.

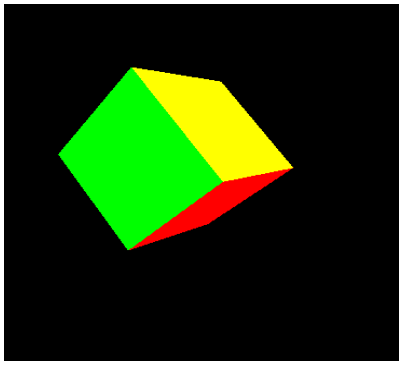


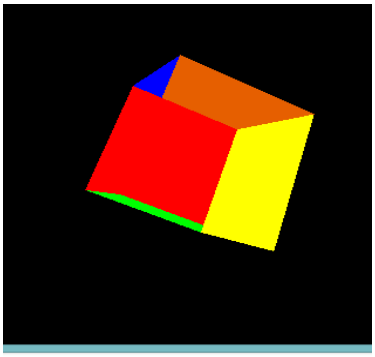
Caracolor()

Nos permite manipular el color de la cara con la que se quiere trabajar, esta de decidió utilizar ya que al rotar el cubo las caras de cruzaban y no nos permitían visualizar de una manera clara cada color del cubo.



Ese problema se intentó resolver mediante la manipulación del dibujado de las caras sin embargo no se pudo y el problema se sigue teniendo.





Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

3er Parcial Escenario Texturas 3D

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

GRAFICACIÓN
VERANO 2015

ULISES VÁZQUEZ COCOLOTL 201105989

DR IVÁN OLMOS PINEDA

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Introducción.

El siguiente trabajo tiene como objetivo crear un escenario 3D en opengl, usando las funciones qué nos ofrece el mismo, así como la implementación de texturas en las figuras para dar un mayor realismo al escenario.

Transformaciones

Las transformaciones son las que hacen posible la proyección de coordenadas 3D sobre superficies 2D. También son las encargadas de mover, rotar y escalar objetos. En realidad, estas transformaciones no se aplican a los modelos en sí, si no al sistema de coordenadas, de forma que si se quiere rotar un objeto, no lo se le rota, sino que se rota el eje sobre el que se sitúa.

Para poder llevar a cabo todas las transformaciones de las que se acaba de hablar, deben modificarse dos matrices: la matriz del Modelador y la matriz de Proyección. OpenGL proporciona muchas funciones de alto nivel que hacen muy sencillo la construcción de matrices para transformaciones. Éstas se aplican sobre la matriz que este activa en ese instante. Para activar una de las dos matrices utilizamos la función glMatrixMode. Hay dos parámetros posibles:

glMatrixMode(GL_PROJECTION); activa la matriz de proyección, y

glMatrixMode(GL_MODELVIEW); activa la del modelador.

Translación

Ogl nos ofrece la función glTranslate, que crea la matriz de transformación y la multiplica por la matriz que esté activa en ese instante (en este caso debería ser la del modelador, GL_MODELVIEW).

glTranslatef(10.0f, 0.0f, 0.0f);

La “f” añadida a la función indica que se usarán flotantes. Los parámetros de glTranslate son las unidades a desplazar en el eje x, y y z, respectivamente. Pueden ser valores negativos, para trasladar en el sentido contrario.

Rotación

Para rotar, tenemos también una función de alto nivel que construye la matriz de transformación y la multiplica por la matriz activa, glRotate. Lleva como parámetros el ángulo a rotar (en grados, sentido horario), y después x, y y z del vector sobre el cual se quiere rotar el objeto. Una rotación simple, sobre el eje y, de 10º sería

glRotatef(10, 0.0f, 1.0f, 0.0f);

Si quisiéramos rotar con sentido al eje x ponemos en 1 este eje y los otros dos en 0.

Escalado

Una transformación de escala incrementa el tamaño de nuestro objeto expandiendo todos los vértices a lo largo de los tres ejes por los factores especificados. La función glScale lleva como parámetros la escala en x, y y z, respectivamente. El valor 1.0f es la referencia de la escala, de tal forma que la siguiente línea:

glScalef(1.0f, 1.0f, 1.0f);

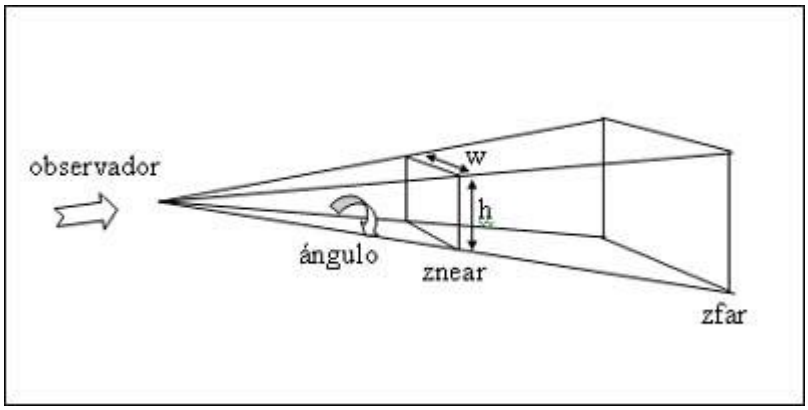
Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Proyecciones perspectivas

Una proyección en perspectiva reduce y estirar los objetos más alejados del observador. Es importante saber que las medidas de la proyección de un objeto no tienen por qué coincidir con las del objeto real, ya que han sido deformadas.

Void gluPerspective(angulo, aspecto, znear, zfar);

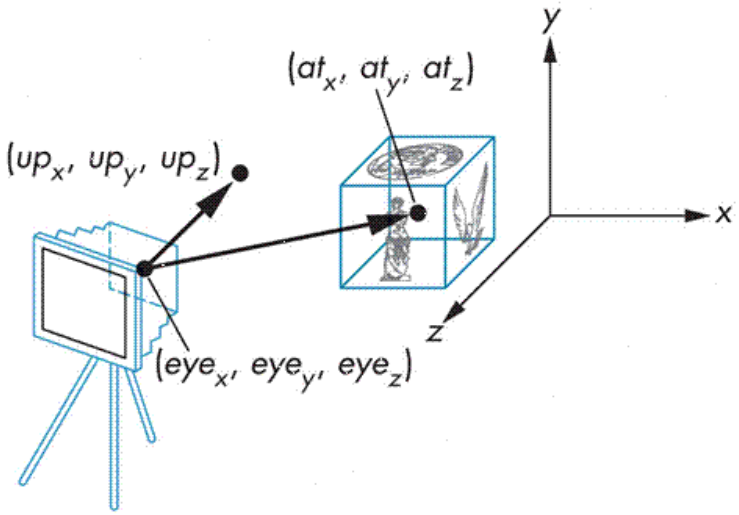
Los parámetros de gluPerspective son flotantes y definen las características mostradas en la ilustración 4.7, el ángulo para el campo de visión en sentido vertical, el aspecto que es la relación entre la altura (h) y la anchura (w) y las distancias znear y zfar de los planos que acotan el fustrum al observador. Los valores de znear y zfar no son las posiciones de esos planos en el espacio 3D, representan la distancia desde el centro de proyección, con valor positivo hacia delante y negativo hacia atrás.



Mirar hacia un punto

La función en la librería de utilidades denominada *gluLookAt()* sitúa el punto de vista respecto al punto de atención, la sintaxis de la función es la siguiente:

gluLookAt(eye.x,eye.y,eye.z,center.x,center.y,center.z,up.x,up.y,up.z)



Texturas en OpenGL.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Aplicación jerárquica de transformaciones

A menudo, la definición de una escena requiere la construcción de objetos complejos que se crean a partir de otros más simples. En estos casos es necesario poder aplicar transformaciones de forma jerárquica a los objetos. Veamos un ejemplo. Supongamos que estamos construyendo el modelo de un coche: podemos definir un modelo simple de una tuerca y utilizarla para crear una rueda que utiliza cuatro tuercas. Una vez definida la rueda, el coche la utilizará cuatro veces en su definición. En la construcción del coche, cada pieza básica definida en su sistema de coordenadas local se transforma a un nuevo sistema externo, el cual a su vez puede transformarse nuevamente.

Para permitir la aplicación jerárquica de transformaciones, OpenGL utiliza una *pila de matrices* en la que puede almacenar estados anteriores del proceso. La pila de matrices se maneja mediante las ordenes OpenGL *glPushMatrix* y *glPopMatrix*, cuya sintaxis se muestra a continuación:

```
void glPushMatrix(void);

void glPopMatrix(void);
```

La orden *glPushMatrix* mueve todas las matrices en la pila una posición hacia abajo, dejando en la cabeza de la pila una copia de la matriz cabeza de la pila anterior. Por tanto, después de ejecutar una orden *glPushMatrix*, la primera y segunda matriz de la pila contienen los mismos valores. Si se efectúan más ordenes *glPushMatrix* de las permitidas, se produce un error.

La orden *glPopMatrix* elimina la matriz cabeza de la pila, moviendo el resto de matrices almacenadas en la pila una posición hacia arriba. Si la pila contiene una única matriz, al ejecutar *glPopMatrix* se produce un error.

El uso conjunto de *glPushMatrix* y *glPopMatrix* permite almacenar estados intermedios de la transformación en la pila y recuperarlos posteriormente. El efecto de la orden *glPushMatrix* puede entenderse como “recuerda dónde estás” y el de *glPopMatrix* como “vuelve a dónde estabas”.

TEXTURAS:

Las texuras son imágenes que aplicadas a objetos de un modelo de OpenGL hace que el modelo sea mucho más real sin aumentar el número de polígonos.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Se aprenderá lo básico de las imágenes en OGL, como aplicar una textura y formas de hacerlo, filtrados para evitar imágenes pixeladas, mejoras al introducir mapas MIP y problemas deribados del uso de texturas. Aunque el objetivo es tratar texturas de una y dos dimensiones, se hará una breve introducción a las texturas 3D.

Todos los códigos ejemplos hacen referencia a estas librerías.

COORDENADAS

Las texturas tienen coordenadas que nos ayudarán a aplicarlas a objetos, estas coordenadas asociarán que parte de la imagen va con que vértice. Las coordenadas se pueden especificar a mano, aunque OpenGl también puede generarlas.

Cargar textura

El primer paso para aplicar una textura es cargar la imagen obteniendo su identificador.

Luego ese usará para cargar los datos de la textura y convertirla en la textura actual.

Aplicar textura

Las texturas no se pueden aplicar a los elementos geométricos a no ser que se encuentre activado el estado de textura apropiado.

Para activar y desactivar el estado se usa glEnable y glDisable con las máscaras correspondientes a cada textura. Por ejemplo para activar la textura en 2D:

```
glEnable(GL_TEXTURE_2D);
```

Los datos de la textura cargados se dirigen a través del mismo canal de píxel y procesamiento de la imagen, lo que significa que las operaciones realizadas como el zoom se aplican a la textura.

Para aplicar la imagen surgen las coordenadas de textura, estas se debe especificar el vértice al que se aplican.

Las coordenadas de textura son flotantes que van desde el 0.0 al 1.0 y se denominan **s**, **t**, **r** y **q** (de forma similar a las coordenadas del vértice **x**, **y z** y **w**). La coordenada q se corresponde con la coordenada geométrica w y es el factor de escala.

Ajuste de la textura

Normalmente las coordenadas de textura se especifican con valores entre 0.0 y 1.0.

Si las coordenadas se encuentran fuera de este rango, OpenGL actúa en consecuencia a como haya sido ajustada la textura.

Una vez hecho el glEnable estamos preparados para usar la función glTexImage3D. La manera de crear una textura 3D es similar a una 2D:

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

1. Cargar ó generar los texels
2. Crear un nombre para la textura
3. Asociar ese nombre a una textura 3D
4. Especificar el filtro y otros parametros
5. Finalmente llamar a glTexImage3D

```
//(generate texel code omitted)
unsigned int texname;
glGenTextures(1, &texname);
glBindTexture(GL_TEXTURE_3D, texname);
glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_R, GL_REPEAT);
glTexImage3D(GL_TEXTURE_3D, 0, GL_RGB8, WIDTH, HEIGHT, DEPTH, 0, GL_RGB,
        GL_UNSIGNED_BYTE, texels);
```

Para hacer una textura activa (la que nuestra geometría debe usar) usamos glBindTexture igual que en 2D, pero ahora se le especifica GL_TEXTURE_3D.

Desarrollo.

El primer paso que se realizó en este proyecto fue cargar las texturas con el manejador FreeImage por su fácil uso y manejo de las texturas. Para esto instalamos el paquete que ya nos ofrece freeimage y se instala directo en en DevC++ una vez instaldo el paquete la secuencia en el código para cargar las texturas es la siguiente.

```
#include <GL/glut.h>
#include <stdlib.h>
#include <FreeImage.h>
#include <GL/glext.h>

//se define la cantidad de texturas que se manejaran
#define NTextures 2
GLuint  texture[NTextures];
//variables para manejo de texturas
char *texturefiles[] = {
    "manos.jpg","imagen2.bmp"
};

int i=0;
bool readImage()
{
    for(i=0; i < NTextures; i++)
  {
      //image format
    FREE_IMAGE_FORMAT fif = FIF_UNKNOWN;
    //pointer to the image, once loaded
    FIBITMAP *dib(0), *dib_copy(0);
    //pointer to the image data
    BYTE* bits(0);
    //image width and height
    unsigned int width(0), height(0);
```

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

```
//check the file signature and deduce its format
fif = FreeImage_GetFileType(texturefiles[i], 0);
//if still unknown, try to guess the file format from the file extension
if(fif == FIF_UNKNOWN)
    fif = FreeImage_GetFIFFromFilename(texturefiles[i]);
//if still unkown, return failure
if(fif == FIF_UNKNOWN)
    return false;

//check that the plugin has reading capabilities and load the file
if(FreeImage_FIFSupportsReading(fif))
    dib = FreeImage_Load(fif, texturefiles[i], BMP_DEFAULT);
//if the image failed to load, return failure
if(!dib)
    return false;
//TEXTURE GENERATION

//retrieve the image data
bits = FreeImage_GetBits(dib);
//get the image width and height
width = FreeImage_GetWidth(dib);
height = FreeImage_GetHeight(dib);
//if this somehow one of these failed (they shouldn't), return failure
if((bits == 0) || (width == 0) || (height == 0))
    return false;
```

```
    height = FreeImage_GetHeight(dib);
    //if this somehow one of these failed (they shouldn't), return failure
    if((bits == 0) || (width == 0) || (height == 0))
        return false;

    //generate an OpenGL texture ID for this texture
    glGenTextures(1, &texture[i]);
    //bind to the new texture ID
    glBindTexture(GL_TEXTURE_2D, texture[i]);
    //store the texture data for OpenGL use
    // ** AGREGADO ** (filtros para la textura y condicional de mipmap) //
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexImage2D(GL_TEXTURE_2D, 0, 3, width, height, 0, GL_BGR_EXT, GL_UNSIGNED_BYTE, bits);

    //Free FreeImage's copy of the data
    FreeImage_Unload(dib);
    }

    //return success
    return true;

}
```

Con este primer paso la textura ya está cargada en memoria y lista para usarse en nuestro proyecto.

En la función de init en los parámetros de la función gluLookAt sumamos variables en los tres primeros ejes que es a donde está “parado el observador” esto lo hacemos con el objetivo de mover la cámara definiendo unas teclas.

```
void init(void){

glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(100,1,0.01,200);
readImage();
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
   glClear(GL_COLOR_BUFFER_BIT);
   glEnable(GL_TEXTURE_2D);
gluLookAt(15+bb,3+yy,14.0+h
             ,0.0,1.0,-20.0
             ,0.0,0.5,0.0);
glClearColor(0.74902,0.847059,0.847059,0);
glEnable(GL_DEPTH_TEST);
}
```

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

En esta función hacemos el aumento y decremento de las variables para mover la cámara.

```
void keyboard(unsigned char key, int x, int y)
{
switch (key) {
case 'a':
bb-=0.5;
break;
case 'd':
bb+=0.5;
break;
case 'w':
h-=0.5;
break;
case 's':
h+=0.5;
break;
case 'q':
yy+=0.5;
break;
case 'e':
yy-=0.5;
break;
case '0':
yy=0;
bb=0;
h=0;
```

Sólo basta con presionar las teclas indicadas para obtener dicho efecto.

Definimos nuestra función que va a generar un cubo, para posteriormente escalarlo, rotarlo y trasladarlo, pero aquí recibimos un parámetro qué nos indica el índice del arreglo de las texturas para poder usar una y cubrir con esta las 6 caras del cubo.

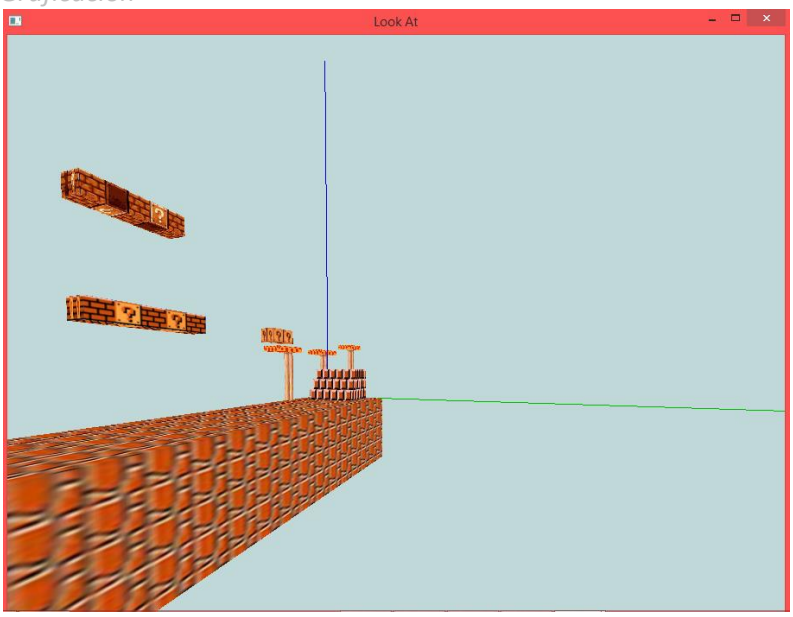
```
void cubo(int x)
{
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, texture[x]);
    glColor3f(1.0f, 1.0f, 1.0f); //rojo
    glBegin(GL_QUADS);        //cara de abajo
        glTexCoord2f(0.0f,1.0f);
        glVertex3f(cuboauxi[0][0],cuboauxi[0][1],cuboauxi[0][2]);
        glTexCoord2f(0.0f,0.0f);
        glVertex3f(cuboauxi[1][0],cuboauxi[1][1],cuboauxi[1][2]);
        glTexCoord2f(1.0f,0.0f);
        glVertex3f(cuboauxi[7][0],cuboauxi[7][1],cuboauxi[7][2]);
        glTexCoord2f(1.0f,1.0f);
        glVertex3f(cuboauxi[6][0],cuboauxi[6][1],cuboauxi[6][2]);
    glEnd();
```

Como observamos en la figura anterior la textura ya está mapeada en la figura, en este caso en una de las caras del cubo, el mismo proceso se hace con las demás caras del cubo. Cuando acabamos de hacer el mapeado desactivamos la textura con el siguiente comando glDisable(GL_TEXTURE_2D);

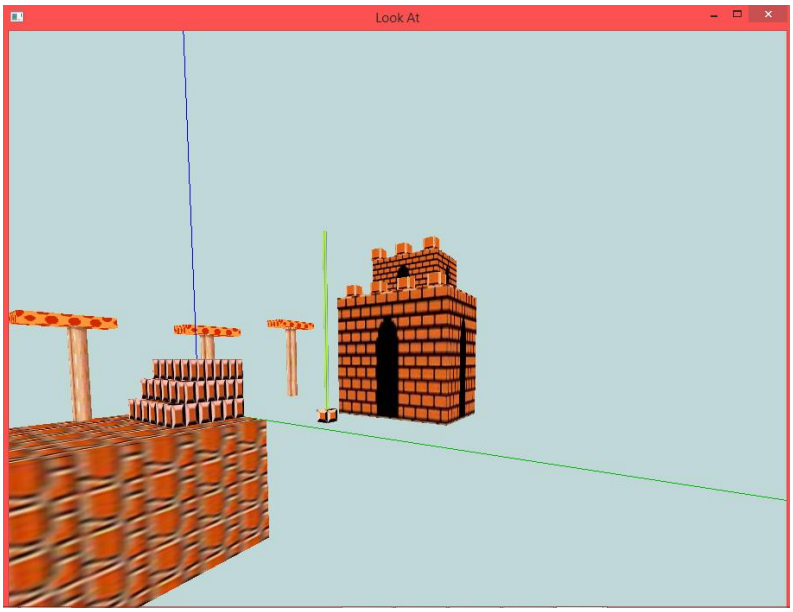
```
void ladrillo1()
{
    glPushMatrix();
    glScalef(1.0f, 1.0f, 4.0f);
    glTranslatef(1.0f, 10.0f, 4.0f);
    cubo(1);
    glPopMatrix();
}
```

Empezamos a hacer uso de las funciones definidas por opengl para armar nuestro escenario a partir de puros cubos, sólo hace falta manipular el escalado y trasladado de los cubo y ubicarlos en las coordenadas deseadas para obtener nuestras figuras ya con texturas.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación



Vemos que en el escenario sólo hay cubos modificados con la función de escalado que nos ofrece opengl.



Creamos el castillo y el asta de la bandera.

Creamos la bandera y le aplicamos la textura que deseamos.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

```
void bandera2()
{

    glPushMatrix();
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, texture[11]);
    glTranslatef(1.5f, 9.0f, -8.0f);
    glBegin(GL_TRIANGLES);        //cara trasera
    glTexCoord2f(0.0f,0.0f);
    glVertex3f( 0.0f, 0.0f, 0.0f);
    glTexCoord2f(0.0f,1.0f);
    glVertex3f(0.0f, 2.0f, 0.0f);
    glTexCoord2f(1.0f,1.0f);
    glVertex3f(2.0f,  1.0f, 0.0f);
    glEnd();
    glDisable(GL_TEXTURE_2D);
    glPopMatrix();
```

De igual manera que el castillo y los bloques, creamos la superficie de nuestro escenario, y al mismo tiempo se crea el fondo al que se le aplica una textura para hacer llamativo el escenario.

Una vez creado el castillo aplicamos operaciones de rotación, escala y traslada para generar replicas de diferente tamaño en diferente posición, el resultado es el siguiente.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Agregamos una esfera animada entre el castillo y la bandera.

```
if (anguloEsfera<360)
{esfera();
anguloEsfera += 15;}
else anguloEsfera = 0;
```

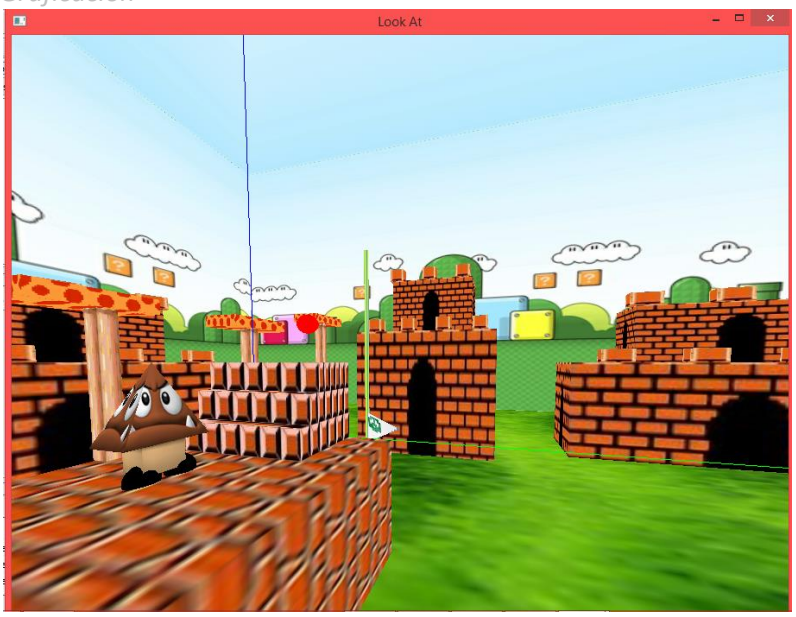
Hacemos que la esfera se mueve incrementando el ángulo y llevando un control para que la variable no llegue a su tope.

```
void esfera()
{
    glTranslatef(1.0f, 5.0f, -5.0f);
    glRotatef(anguloEsfera, 0.0f, 1.0f, 0.0f);
    glTranslatef(0.0f, 0.0f, 2.0f);
    glColor3f(1.0f, 0.0f, 0.0f);
    glutSolidSphere(0.5f,18,8);
}
```

Ahora procedemos a agregar un hongo a partir de la función que crea pirámides, al igual que la del cubo ya con la textura mapeada.

Dicho hongo se moverá en el eje z, haciendo el efecto de caminar y retroceder, apretando las teclas “k” y “l” respectivamente.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación



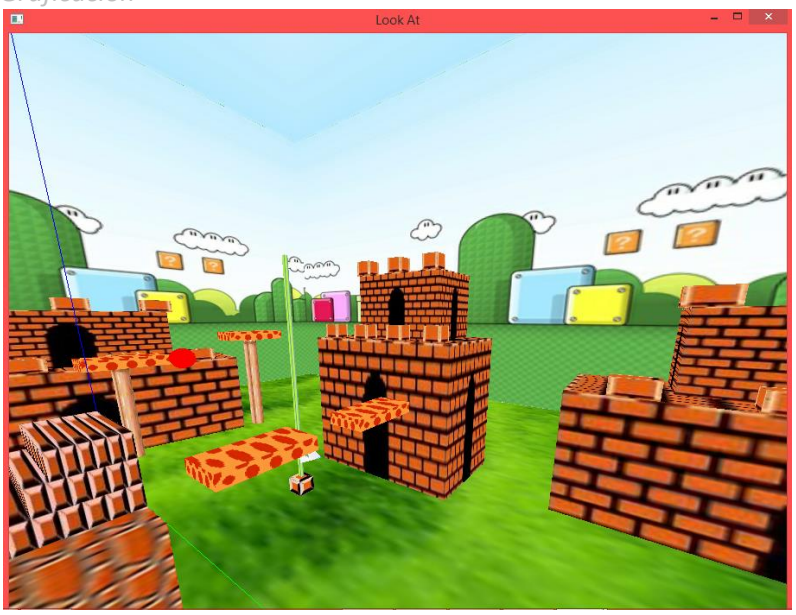

```
glPushMatrix();
glTranslatef(0.0,0.0,m);
hongo();
glPopMatrix();
```

```
void hongo()
{
    glPushMatrix();
    glScalef(1.0f, 2.0f, 2.0f);
    glTranslatef(1.0f, 0.5f, 2.0f);
    piramide(16);
    glScalef(0.6f, 0.5f, 0.5f);
    glTranslatef(0.3f, -0.8f, 0.5f);
    cubo(17);
    glColor3f(0.0f, 0.0f, 0.0f);
    glTranslatef(0.7f, -0.1f, 0.0f);
    glutSolidSphere(0.4f,18,8);
    glTranslatef(0.0f, 0.0f, 0.9f);
    glutSolidSphere(0.4f,18,8);
    glPopMatrix();
}
```

Está creado con una pirámide, el cuerpo es un cubo y las patas son dos esferas.

Finalmente agregamos dos barras que se irán desplazando en el escenario.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación




```
void desplaza()
{glPushMatrix();
   if(mueve2 < 6 && control==0)
    {glTranslatef(10.0,-2.0,mueve2);
    poste22();
    mueve2+=0.3;
    }
   if(mueve2 >6) control=1;
   //regreso
  if(mueve2 > 0 && control==1)
  {glTranslatef(10.0,-2.0,mueve2);
     poste22();
     mueve2-=0.3;
  }
  if (mueve2 < 0) control=0;

   glPopMatrix();
}
```

En esta parte se debe llevar el control con dos variables para que la figura vaya y venga en el escenario.

Finalmente este es el resultado.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Bibliografía

[1] http://sabia.tic.udc.es/gc/Tutorial%20OpenGL/tutorial/cap4.htm

[2] http://sabia.tic.udc.es/gc/Tutorial%20OpenGL/tutorial/cap4.htm#_Toc535127350

[3] http://stackoverflow.com/questions/13781874/glulookat-first-person-view

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

2do Parcial

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

GRAFICACIÓN
VERANO 2015

ULISES VÁZQUEZ COCOLOTL 201105989

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

DR IVÁN OLMOS PINEDA

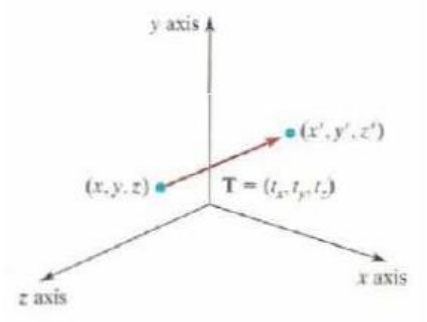
Introducción.

Cuando nos introducimos al mundo 3D, hay que considerar, el factor de profundidad, las combinaciones que se pueden generar sobre 3 ejes, la perspectiva de observación, etc.

Por lo tanto los operadores se ven afectados en diferente medida.

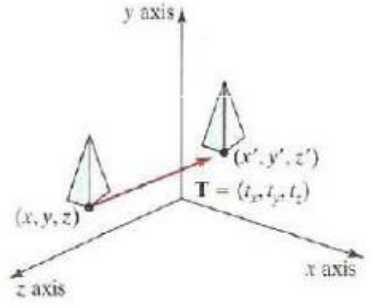
Translación 3D

Así como en el espacio 2D, la traslación se define a partir de un vector, ahora con 3 componentes



El operador de traslación se puede definir a través de una matriz de la siguiente forma:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Para rotar un objeto 3D con un eje de rotación paralelo a un eje:

Primero se mueve el eje de rotación al eje de rotación definido para trabajar (uno de los 3 ejes del plano 3D)

- Se aplica la rotación que se desea aplicar

- Se regresa el eje de rotación a su posición original

Matricialmente, considere que un punto P=(x,y,z) será rotado con respecto al eje X. Las operaciones a realizar serán las siguientes:

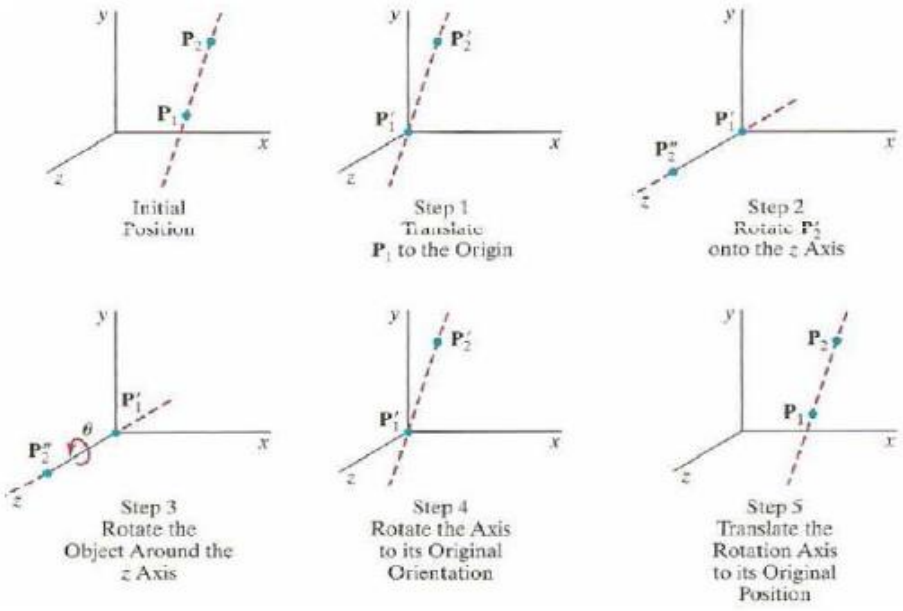
$$P' = T^{-1} \cdot R_x(\theta) \cdot T \cdot P$$

Donde:

- P' es el punto resultante de la rotación
- R_x es la matriz se rotación con respecto al ángulo especificado
- T es la matriz de translación al eje
- T^{-1} es la matriz inversa de T de traslación al eje

Cuando el eje de rotación de un objeto no es paralelo a uno de los ejes, se tiene que proceder de la siguiente manera:

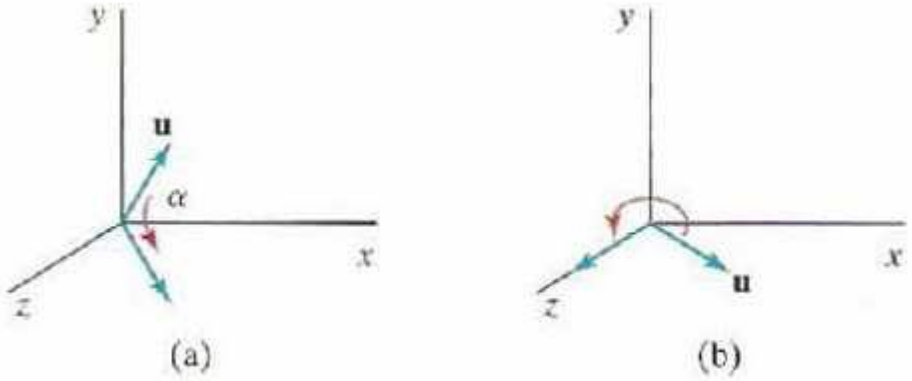
1. Trasladar el objeto de tal forma que el eje de rotación pase a través de la coordenada de origen
2. Rotar el objeto de tal forma que el eje de rotación coincida con alguno de los ejes de coordenadas
3. Realizar la rotación especificada sobre el eje de coordenadas seleccionado
4. Aplicar la rotación inversa para regresar el eje de rotación a su orientación original
5. Aplicar la translación inversa para regresar el eje de rotación a su posición espacial original



Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Con la notación anterior, los pasos para la rotación libre son los siguientes:

1. Se define la matriz de traslación al origen (tomando a P1)
2. Se realizan las transformaciones para colocar el eje de rotación sobre uno de los ejes del sistema (este paso se puede realizar de diferentes formas)
   a. Se rota a U sobre X para colocarlo en el plano XZ
   b. Se rota a U sobre Z para colocarlo en el plano YZ



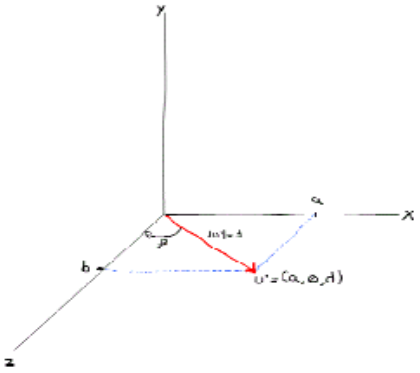
Proyectar a u sobre el plano XZ, requiere rotar a dicho vector sobre
X, por lo que la matriz de rotación a utilizar es:

$$R_x(\theta)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\theta & -\sin\theta & 0\\ 0 & \sin\theta & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$R_x(\alpha)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & c/d & -b/d & 0\\ 0 & b/d & c/d & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Equivale a tener

- El siguiente paso consiste en calcular la matriz de rotación del vector *u'* proyectado sobre el plano XZ para colocarlo sobre el eje positivo Z



$$|u'|=\sqrt{a^2+d^2}=\sqrt{a^2+b^2+c^2}=1$$

De la figura, se puede observar que:

$$\sin(\beta)=\frac{a}{\sqrt{a^2+d^2}}=a,\quad \cos(\beta)=\frac{d}{\sqrt{a^2+d^2}}=d$$

Aplicando la matriz de rotación sobre Y:

$$R_y(\beta)=\begin{bmatrix}\cos\beta & 0 & -\sin\beta & 0\\ 0 & 1 & 0 & 0\\ \sin\beta & 0 & \cos\beta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}d & 0 & -a & 0\\ 0 & 1 & 0 & 0\\ a & 0 & d & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Por tanto: $$R_y(\beta)u'=\begin{bmatrix}d & 0 & -a & 0\\ 0 & 1 & 0 & 0\\ a & 0 & d & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}a\\ 0\\ d\\ 1\end{bmatrix}=\begin{bmatrix}0\\ 0\\ 1\\ 1\end{bmatrix}$$

3. Con las matrices de transformación ya expuestas, se realiza la rotación del vector u de acuerdo al ángulo tetha:

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

En general, la matriz de rotación para cualquier eje se expresa como:

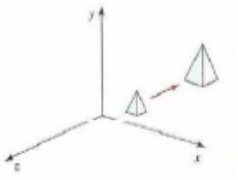
$$R(\theta) = T^{-1} \cdot R_x^{-1}(\alpha) \cdot R_y^{-1}(\beta) \cdot R_z(\theta) \cdot R_y(\beta) \cdot R_x(\alpha) \cdot T$$

Escalado 3D

Escalar un punto P=(x,y,z) con respecto al origen es una extensión directa del caso 2D

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Punto P' Matriz de escalado S Punto P



Notas

Si los valores de sx, sy, sz son diferentes, se cambiará el aspecto general de la imagen
Para escalar objetos 3D (al igual que en el caso 2D) se debe elegir un punto de referencia del mismo

Desarrollo.

Lo principal fue hacer la declaración de nuestras matrices globales que irán almacenando valores parciales para nuestras figuras y poder conseguir un escenario en 3D.

```
float traslacion [4][4] = {1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1};
float escala [4][4] = {1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1};
float rotacion [4][4] = {1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1};
float cubop[8][4] = {0,0,0,1,1,0,0,1,0,1,0,1,1,0,1,0,1,1,1,1,1,1,0,0,1,1,1,0,1,1};
float cuboauxi[8][4] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
float cubomov[8][4] = {2,0,8,1,5,0,8,1,2,3,8,1,5,3,8,1,2,3,11,1,5,3,11,1,2,0,11,1,5,0,11,1};
float NT[8][4],NE[8][4],NR[8][4],NTP[5][4],NEP[5][4],NRP[5][4],NTM[8][4];
float piramide[5][4]={{0,0,0,1},{0,0,1,1},{1,0,1,1},{1,0,0,1},{0.5,1,0.5,1}};
float piramide3d[5][4]={{0,0,0,1},{1,0,0,1},{1,0,1,1},{0,0,1,1},{0.5,1,0.5,1}};
float piraauxi[5][4]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
float PA[1][3]={{5,-3,-1}};  //PUNTO A
float PB[1][3]={{5,-3,1}}; //PUNTO B
float PAB[1][3]; //VECTOR DE A a B
float a,b,c,d; //componentes del vector
float RxA[4][4],RyB[4][4],iRxA[4][4],iRyB[4][4],RzT[4][4],T[4][4],iT[4][4];
float M[4][4]; //matriz final M
float NueP[5][4]; // nuevos puntos
```

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Se definen los métodos que pondrán los valores en las matrices de rotación, traslación y escalamiento respectivamente,

```
void Traslada(float x,float y,float z){
     traslacion[0][3]=x;
     traslacion[1][3]=y;
     traslacion[2][3]=z;
}
```

```
void Escala(float x,float y,float z){
     escala[0][0]=x;
     escala[1][1]=y;
     escala[2][2]=z;
}
```

En el caso de la rotación simple se tendrán tres matrices cada una definiendo el eje respecto con el que se quiere rotar la figura.

```
void RotaX(float grado)
{
    float gradocos,gradosen,gradosin;
    gradocos = cos(grado);
    gradosen = -sin(grado);
    rotacion[1][1] = gradocos;
    rotacion[1][2] = gradosen;
    rotacion[3][1] = (gradosen*-1);
    rotacion[3][2] = gradocos;

}

void RotaY(float grado)
{
    float gradocos,gradosen,gradosin;
    gradocos = cos(grado);
    gradosen = -sin(grado);
    rotacion[0][0] = gradocos;
    rotacion[2][0] = gradosen;
    rotacion[0][2] = (gradosen*-1);
    rotacion[2][2] = gradocos;

}
```

```
void RotaZ(float grado)
{
    float gradocos,gradosen,gradosin;
    gradocos = cos(grado);
    gradosen = -sin(grado);
    rotacion[0][0] = gradocos;
    rotacion[0][1] = gradosen;
    rotacion[1][0] = (gradosen*-1);
    rotacion[1][1] = gradocos;

}
```

En el desarrollo del programa se cuenta con dos figuras base, el cubo y la pirámide.

Se hace el dibujo de ambas figuras con medida unitaria para facilitar el manejo en cada una de las operaciones que se le requiera hacer a dicha figura.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

```
void cubo()
{

    glBegin(GL_QUADS);
        glVertex3f(cuboauxi[0][0],cuboauxi[0][1],cuboauxi[0][2]);
        glVertex3f(cuboauxi[1][0],cuboauxi[1][1],cuboauxi[1][2]);
        glVertex3f(cuboauxi[7][0],cuboauxi[7][1],cuboauxi[7][2]);
        glVertex3f(cuboauxi[6][0],cuboauxi[6][1],cuboauxi[6][2]);

        glVertex3f(cuboauxi[4][0],cuboauxi[4][1],cuboauxi[4][2]);
        glVertex3f(cuboauxi[5][0],cuboauxi[5][1],cuboauxi[5][2]);
        glVertex3f(cuboauxi[7][0],cuboauxi[7][1],cuboauxi[7][2]);
        glVertex3f(cuboauxi[6][0],cuboauxi[6][1],cuboauxi[6][2]);

        glVertex3f(cuboauxi[4][0],cuboauxi[4][1],cuboauxi[4][2]);
        glVertex3f(cuboauxi[2][0],cuboauxi[2][1],cuboauxi[2][2]);
        glVertex3f(cuboauxi[0][0],cuboauxi[0][1],cuboauxi[0][2]);
        glVertex3f(cuboauxi[1][0],cuboauxi[1][1],cuboauxi[1][2]);

        glVertex3f(cuboauxi[4][0],cuboauxi[4][1],cuboauxi[4][2]);
        glVertex3f(cuboauxi[5][0],cuboauxi[5][1],cuboauxi[5][2]);
        glVertex3f(cuboauxi[3][0],cuboauxi[3][1],cuboauxi[3][2]);
        glVertex3f(cuboauxi[2][0],cuboauxi[2][1],cuboauxi[2][2]);

        glVertex3f(cuboauxi[2][0],cuboauxi[2][1],cuboauxi[2][2]);
        glVertex3f(cuboauxi[3][0],cuboauxi[3][1],cuboauxi[3][2]);
```

```
void drawpiramide()
{   glBegin(GL_QUADS);
        glVertex3f(piraauxi[0][0],piraauxi[0][1],piraauxi[0][2]);
        glVertex3f(piraauxi[1][0],piraauxi[1][1],piraauxi[1][2]);
        glVertex3f(piraauxi[2][0],piraauxi[2][1],piraauxi[2][2]);
        glVertex3f(piraauxi[3][0],piraauxi[3][1],piraauxi[3][2]);
    glEnd();

        glBegin(GL_TRIANGLES);
        glVertex3f(piraauxi[0][0],piraauxi[0][1],piraauxi[0][2]);
        glVertex3f(piraauxi[1][0],piraauxi[1][1],piraauxi[1][2]);
        glVertex3f(piraauxi[4][0],piraauxi[4][1],piraauxi[4][2]);
        glEnd();

        glBegin(GL_TRIANGLES);
        glVertex3f(piraauxi[1][0],piraauxi[1][1],piraauxi[1][2]);
        glVertex3f(piraauxi[2][0],piraauxi[2][1],piraauxi[2][2]);
        glVertex3f(piraauxi[4][0],piraauxi[4][1],piraauxi[4][2]);
        glEnd();

        glBegin(GL_TRIANGLES);
        glVertex3f(piraauxi[2][0],piraauxi[2][1],piraauxi[2][2]);
        glVertex3f(piraauxi[3][0],piraauxi[3][1],piraauxi[3][2]);
        glVertex3f(piraauxi[4][0],piraauxi[4][1],piraauxi[4][2]);
        glEnd();
```

El cubo está formado por 6 cuadrados mientras que la pirámide consta de 4 triángulos y un cuadrado.

Tenemos las operaciones respalda y limpia, que se encargan de pasar los puntos originales de las figuras a matrices auxiliares para operar con estas matrices y no perder los originales.

```
void respalda()
{
    int i,j;
    //  printf("matriz cubo auxiliar\n");
    for (i=0;i<=7;i++)
    {  for (j=0;j<=3;j++)
        {
            cuboauxi[i][j]=cubop[i][j];
        //   printf(" %f ",cuboauxi[i][j]);
        }
        //printf("\n");
    }
}
```

```
void limpia()
{
    int i,j;
    //  printf("matriz cubo auxiliar limpia\n");
    for (i=0;i<=7;i++)
    {  for (j=0;j<=2;j++)
        {
            cuboauxi[i][j]=0;
          //  printf(" %f ",cuboauxi[i][j]);
        }
       // printf("\n");
    }
```

Como las figuras son trazadas en el origen, nos ahorramos el paso de regresarlas a este punto, por lo que sólo aplicamos la traslación directamente dependiendo los puntos x, y y z que deseemos.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

```
void OperTras(float traslacion[4][4], float cubop[8][4],float x, float y, float z)
{
    Traslada(x,y,z);
    int i,j,k;

    for (i=0;i<=7;i++)
    {  for (j=0;j<=3;j++)
        { NT[i][j]=0;
           for (k=0;k<=3;k++)
            {
             NT[i][j]=NT[i][j]+(cuboauxi[i][k]*traslacion[j][k]);
            }
        }
    }

    for (i=0;i<=7;i++)
    {
        for (j=0;j<=3;j++)
        {cuboauxi[i][j] = NT[i][j];
        }
    }
}
```

Lo mismo sucede con la operación de escalado, sólo hay que aplicar las operaciones directas sin regresar la figura al origen, lo único diferente, es que después de esta operación hay que trasladar la figura usando la función anterior.

```
void OperEsca(float escala[4][4], float cubop[8][4], float x, float y, float z)
{

   Escala(x,y,z);
    int i,j,k;
   for (i=0;i<=7;i++)
    {  for (j=0;j<=3;j++)
        {  NE[i][j]=0;
           for (k=0;k<=3;k++)
            {
             NE[i][j]=NE[i][j]+(cuboauxi[i][k]*escala[j][k]);
            }
        }
    }
   for (i=0;i<=7;i++)
    {
        for (j=0;j<=3;j++)
        {cuboauxi[i][j] = NE[i][j];
        }
    }

}
```

La operación de rotado tiene las misma lógica, es decir no hace falta regresar la figura ala origen pues es ahí donde se genera, lo único que hay que hacer es aplicar operaciones directamente y después trasladar la figura al lugar donde deseemos que aparezca.

```
void OperRotaCubo(float rotacion[4][4], float cubop[8][4], float g)
{
     //respalda();
     RotaX(g);
     int i,j,k;

    //aplicando operaciones de rotacion
    for (i=0;i<=7;i++)
    {  for (j=0;j<=3;j++)
        {  NR[i][j]=0;
           for (k=0;k<=2;k++)
            {
             NR[i][j]=NR[i][j]+(cuboauxi[i][k]*rotacion[j][k]);
            }
        }
    }

    //actualizando cubo
    for (i=0;i<=7;i++)
    {
        for (j=0;j<=3;j++)
        {cuboauxi[i][j] = NR[i][j];
        }
    }
}
```

La siguiente función se encarga de darle movimiento al objeto en el display

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

```
void OperTrasMov(float traslacion[4][4], float cubomov[8][4],float x, float y, float z)
{
    Traslada(x,y,z);
    int i,j,k;
    printf("original cubo mov\n");
    for (int i=0;i<=7;i++)
    {
        for (int j=0;j<=3;j++)
        {printf(" %f ",cubomov[i][j]);
        }
       printf("\n");
    }
    for (i=0;i<=7;i++)
    {  for (j=0;j<=3;j++)
        { NT[i][j]=0;
           for (k=0;k<=3;k++)
            {
             NT[i][j]=NT[i][j]+(cubomov[i][k]*traslacion[j][k]);
            }
        }
    }
    for (i=0;i<=7;i++)
    {
        for (j=0;j<=3;j++)
        {cubomov[i][j] = NT[i][j];
        }
    }
```

Irá actualizando los puntos guardados en una matriz auxiliar que es la que se manda a pintar en el display, se aplican las operaciones de traslación.

También está implementada la rotación libre, descrita en el trabajo de Tarea4.

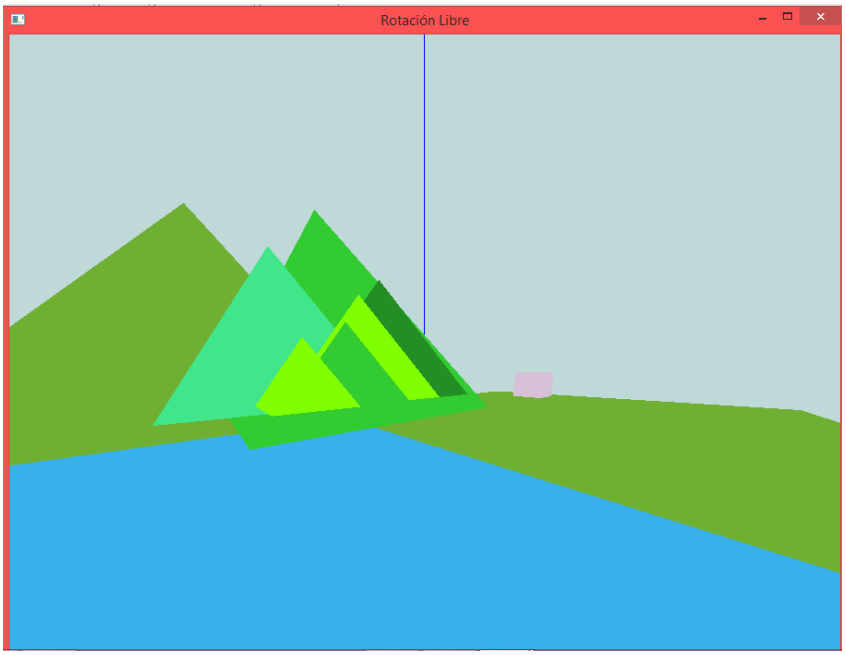
```
// Matriz de rotacion sobre y
void Rotaybeta(){
    int i,j;
     for(i=0; i<4; i++){
         for(j=0; j<4; j++){
             if(i==j) RyB[i][j]=1;
             else RyB[i][j]=0;
                  }
              }
RyB[0][0]=d;RyB[0][2]=-a;RyB[2][0]=a;RyB[2][2]=d;
}

//inversa de matriz y
void invRotaybeta(){
    int i,j;
     for(i=0; i<4; i++){
         for(j=0; j<4; j++){
             if(i==j) iRyB[i][j]=1;
             else iRyB[i][j]=0;
                  }
              }
iRyB[0][0]=d;iRyB[0][2]=a;iRyB[2][0]=-a;iRyB[2][2]=d;
}

//rotando matriz tetha
void Rotazteta(int grad){
     int i,j;
     float grad2;
```

Una vez creados los métodos a aplicar en las figuras, procedemos a crear cada una de ellas en esta imagen está la función que genera las montañas.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

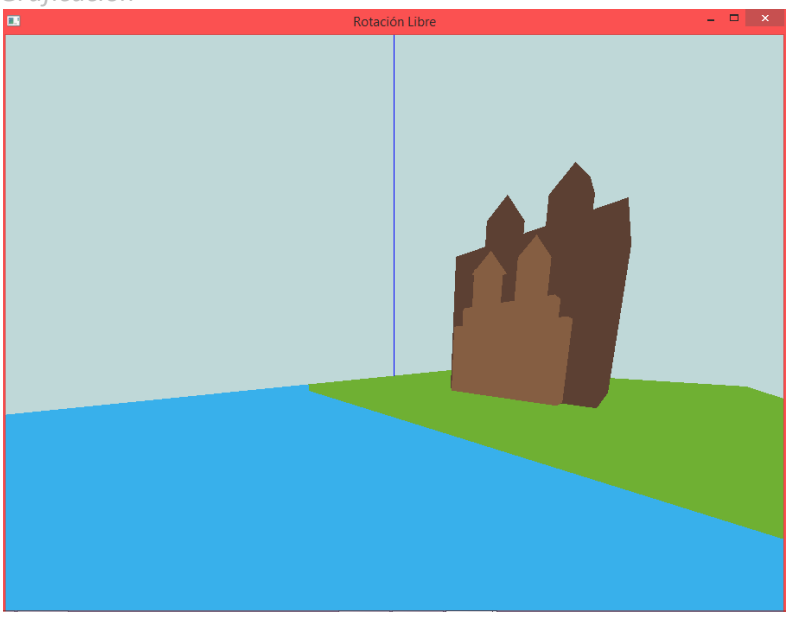
```
void monta()
{
   respaldapira();
   OperEscaPi(escala,piraauxi,20,15,15);
   OperTrasPi(traslacion,piraauxi,-18,0,6);
   drawpiramide();
   limpiapira();
   glColor3f( 0.196078,0.8,0.196078);
   respaldapira();
   OperEscaPi(escala,piraauxi,25,18,15);
   OperTrasPi(traslacion,piraauxi,-20,0,-2);
   drawpiramide();
   limpiapira();
   glColor3f(0.13725,0.556863,0.137255);
   respaldapira();
   OperEscaPi(escala,piraauxi,15,15,15);
   OperTrasPi(traslacion,piraauxi,-18,0,-10);
   drawpiramide();
   limpiapira();
   glColor3f(0.255,0.9,0.54);
   respaldapira();
   OperEscaPi(escala,piraauxi,15,15,15);
   OperTrasPi(traslacion,piraauxi,-18,0,1);
   drawpiramide();
   limpiapira();
   glColor3f(0.498039,1.0,0.0);
```

Como se puede apreciar, se aplica la operación de escalar para generar distintos tamaños de las pirámides y cada una de ellas se va trasladando en el lugar deseado.

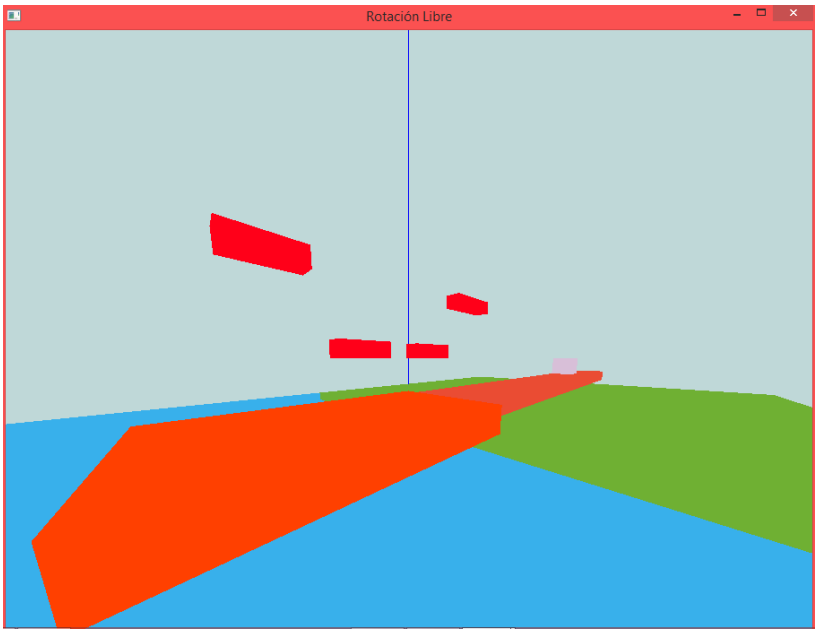


Lo mismo sucede con la estructura del castillo, es creado a partir de diferentes escalados y traslados tanto de cubos como cubos de modo que generemos la estructura deseada.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

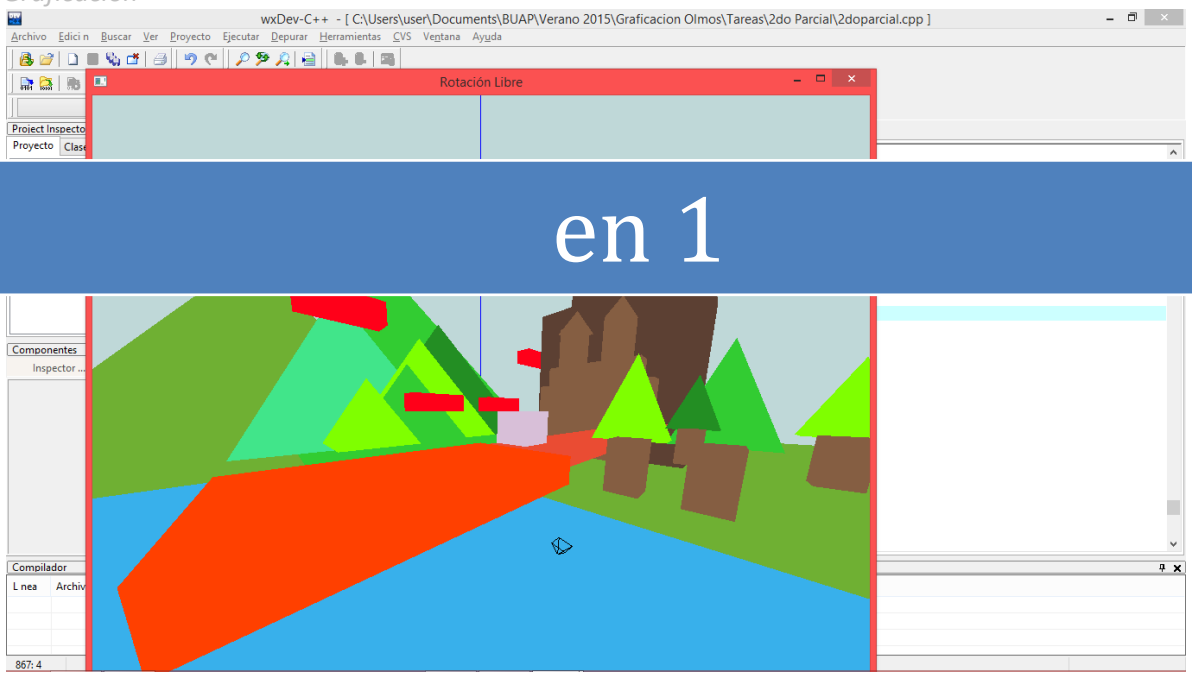


Lo siguiente simula ser un camino hacia el castillo y un puente rumbo al mismo, acompañado de bloques como los del videojuego de mario Bross,



De igual manera se generan las demás figuras en el escenario obteniendo como resultado el siguiente escenario.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación



en 1

Bibliografía

[1] http://sabia.tic.udc.es/gc/Tutorial%20OpenGL/tutorial/cap4.htm

[2] http://grafi-ricardo.blogspot.mx/2012/04/glulookat.html

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

GRAFICACIÓN
VERANO 2015

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

ULISES VÁZQUEZ COCOLOTL 201105989

DR IVÁN OLMOS PINEDA

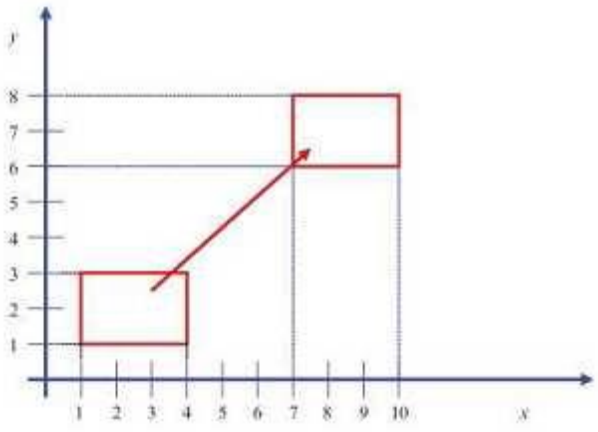
Introducción.

Para el mundo gráfico en 2D existen transformaciones geométricas que permiten desplazar una figura, rotar o cambiar su tamaño.

TRASLACIÓN

Una de las operaciones básicas consiste en trasladar una figura de una posición (x,y) a una nueva posición (x',y'). Formalmente, una translación se representa a través de un vector T = <tx,ty>, tal que si se aplica a un punto P = (x,y), entonces P' = P + T

x' = x + tx
y' = y + ty

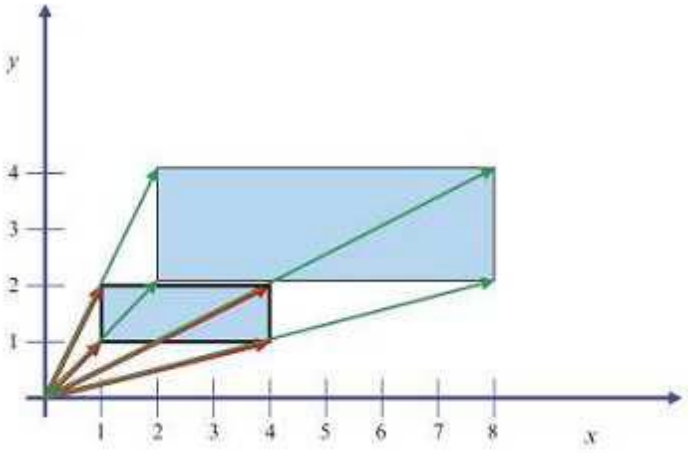


Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

ESCALA

Otra operación básica consiste en alterar el tamaño de una figura (aplicable desde un punto hasta figuras completas). Para ello se requiere un factor de escalado tanto para el componente en X como para el componente en Y

x' = x * sx
y' = y * sy

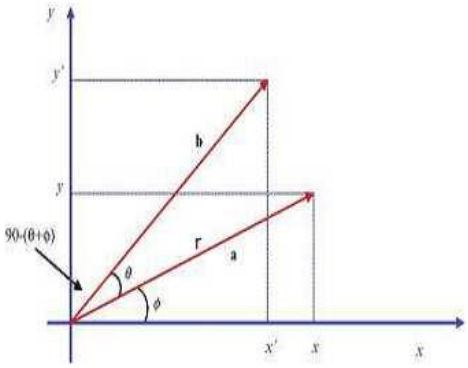


ROTACIÓN

Para girar o rotar una figura, se debe de tener en cuenta dos aspectos importantes:
- Ángulo en el cual se desea rotar la figura
- Punto o pivote de rotación

Si consideramos que la rotación se realiza con respecto al punto origen (0,0) y la rotación será de θ° grados, la operación queda de la forma:

x' = x cos θ - y sen θ
y' = x sin θ - y cos θ



Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Para llevar a cabo estas operaciones tenemos que tener definidas nuestras matrices de cada operación respectivamente definidas de la siguiente manera

$$T = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \qquad E = \begin{bmatrix} E_x & 0 & 0 \\ 0 & E_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} \cos(\alpha) & -\text{sen}(\alpha) & 0 \\ \text{sen}(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Si sólo contamos con rotación centrada en el origen y queremos efectuar una rotación con respecto a otro punto tenemos que:

1.- trasladar el objeto al origen a partir de un punto pivote

2.- aplicar operaciones en este caso rotación pero bien pudiera ser escala

3.- trasladar el objeto de vuelta a su posición original

Ejemplo de la rotación.

$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} \cos\theta & -sen\theta & 0 \\ sen\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

Traslación rotación traslación al origen

En la traslación hacia el origen, se tiene que sustituir en la matriz de traslación los puntos inversos del pivote, una vez efectuadas las operaciones se debe regresar la figura a su posición original.

Si quisiéramos hacer más de una operación en una misma figura seguiríamos el mismo algoritmo, trasladar figura al origen, efectuar operaciones y regresar figura al origen.

Dibujo de líneas (GL_LINES)

En el dibujado de puntos, la sintaxis era muy cómoda: cada vértice es un punto. En las líneas, los vértices se cuentan por parejas, denotando punto inicial y punto final de la línea. Si se especifica un número impar de vértices, el último de ellos se ignora.

Acuérdate de añadir la librería math.h en un include al principio del código para que compilen las funciones se seno y coseno.

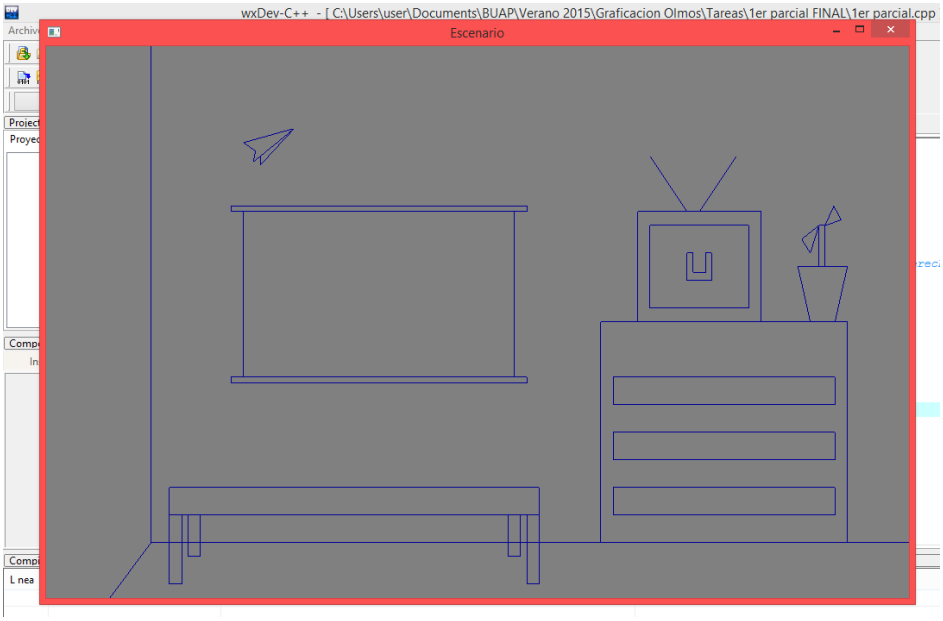
Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Si en vez de GL_LINES utilizásemos GL_LINE_STRIP, ogl ya no trataría los vértices en parejas, si no que el primer vértice y el segundo definirían una línea, y el final de ésta definiría otra línea con el siguiente vértice y así sucesivamente, definiendo un segmento continúo.

Mencionar, por último, la primitiva GL_LINE_LOOP que funciona igual que GL_LINE_STRIP pero, además, une el último vértice con el primero, creando siempre una cuerda cerrada.

Desarrollo.

El primer paso en este proyecto fue dibujar a partir de pura línea y haciendo uso de la función proporcionada por opengl GL_LINE_STRIP, GL_LINE_LOOP Y GL_LINES, un escenario que simula una habitación.



En el escenario podemos ver figuras tales como una televisión, una ventana, un avión de papel, una cama, un ropero o mueble, y un rehilete.

Las figuras a animar serán:

Avión (traslación)

Letra u en la televisión (Escala)

Aspas del rehilete (rotación)

En estas dos últimas habrá que llevar a cabo la operación de traslación.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

Definimos todas las matrices que vamos a utilizar, en este caso la mayoría de ellas son matrices auxiliares para llevar a cabo las múltiples operaciones.

```
float traslacion [3][3] = {1,0,0,0,1,0,0,0,1};
float escala [3][3] = {1,0,0,0,1,0,0,0,1};
float rotacion [3][3] = {1,0,0,0,1,0,0,0,1};
float avion [7][3] = {100,270,1,60,265,1,70,262,1,68,258,1,101,270,1,74,257,1,74,260,1};
float NT[7][3], NE[9][3], NT2[9][3],NE2[9][3],NT3[4][3],NR[4][3], NR2[4][3],NR3[4][3],NR4[4][3],NT4[4][3];
float NT5[4][3], NT6[4][3],NR5[4][3], NR6[4][3], NR7[4][3],NR8[4][3];
float u[9][3] = {425,225,1,420,225,1,420,215,1,440,215,1,440,225,1,435,225,1,435,218,1,425,218,1,425,225,1 };
float aspa[4][3] = {527,235,1,513,230,1,520,225,1,527,235,1};
float aspa2[4][3] = {532,235,1,539,242,1,545,237,1,532,235,1};
float aspa3[4][3] = {527,235,1,513,230,1,520,225,1,527,235,1};
float aspa4[4][3] = {532,235,1,539,242,1,545,237,1,532,235,1};
```

Tenemos las matrices identidad de traslación, rotación y escala, además de que guardamos en una matriz los puntos de las figuras a las que vamos a animar, en este caso el avión, la letra u y las aspas del rehilete.

Definimos nuestras funciones de traslación, escala y rotación.

```
void Traslada(float x,float y){
     traslacion[0][2]=x;
     traslacion[1][2]=y;
}

void Escala(float x,float y){
     escala[0][0]=x;
     escala[1][1]=y;
}

void Rota(float grado)
{
    float gradocos,gradosen,gradosin;
    gradocos = cos(grado);
    gradosen = -sin(grado);
    rotacion[0][0] = gradocos;
    rotacion[0][1] = gradosen;
    rotacion[1][0] = (gradosen*-1);
    rotacion[1][1] = gradocos;

}
```

Donde lo que se hace es asignar en las posiciones de cada una de ellas los respectivos valores que corresponden como vimos en la introducción de este trabajo.

Traslación.

Tenemos la función que hace las operaciones para trasladar la figura definida de la siguiente manera.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

```
void OperTras(float traslacion[3][3], float avion[7][3])
{
    Traslada(30,2);
    int i,j,k;

    for (i=0;i<=6;i++)
    {  for (j=0;j<=2;j++)
        { NT[i][j]=0;
           for (k=0;k<=2;k++)
            {
             NT[i][j]=NT[i][j]+(avion[i][k]*traslacion[j][k]);
            }
        }
    }
    //actualizando puntos del avion
    for (i=0;i<=6;i++)
    {
        for (j=0;j<=2;j++)
        {avion[i][j] = NT[i][j];
        }
    }


}
```

Recibe dos matrices como parámetro, la matriz identidad y la matriz de puntos de la figura

Se asignan los valores que queremos trasladar la figura (x,y) y procedemos a hacer los cálculos con la multiplicación de matrices.

Finalmente actualizamos los puntos de la figura.

Rotación.

```
void OperRota(float rotacion[3][3], float aspa[4][3])
{
     Rota(5);
     int i,j,k;

    float inverso1, inverso2;
    inverso1 = aspa[0][0]*-1;
    inverso2 = aspa[0][1]*-1;

    Traslada(inverso1,inverso2);
    //Llevando figura al origen
   for (i=0;i<=3;i++)
    {  for (j=0;j<=2;j++)
        {  NT3[i][j]=0;
           for (k=0;k<=2;k++)
            {
             NT3[i][j]=NT3[i][j]+(aspa[i][k]*traslacion[j][k]);
            }
        }
    }
```

Recibimos como parámetros la matriz identidad de rotación y los puntos de la figura en este caso un aspa del rehilete.

Asignamos los grados que queremos rotar dicha figura, calculamos el vector inverso y trasladamos la figura al origen.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

```
    //aplicando operaciones de rotacion
   for (i=0;i<=3;i++)
   {  for (j=0;j<=2;j++)
       {  NR[i][j]=0;
          for (k=0;k<=2;k++)
           {
            NR[i][j]=NR[i][j]+(NT3[i][k]*rotacion[j][k]);
           }
       }
   }
   //Regresando figura al origen
   Traslada((inverso1*-1),(inverso2*-1));
   for (i=0;i<=3;i++)
   {  for (j=0;j<=2;j++)
       {  NR2[i][j]=0;
          for (k=0;k<=2;k++)
           {
            NR2[i][j]=NR2[i][j]+(NR[i][k]*traslacion[j][k]);
           }
       }
   }
```

Aplicamos la operación de rotación y posteriormente regresamos la figura a su lugar de origen.

```
    //actualizando aspa
    for (i=0;i<=3;i++)
    {
        for (j=0;j<=2;j++)
        {aspa[i][j] = NR2[i][j];
        }
    }
}
```

Finalmente actualizamos los valores de la figura para que se pueda ir dibujando en cada iteración.

Escala

```
void OperEsca(float escala[3][3], float u[9][3])
{
    float inverso1, inverso2;
    inverso1 = u[0][0]*-1;
    inverso2 = u[0][1]*-1;

    //printf(" %f ",inverso1);
    //printf("\n %f \n",inverso2);

    Traslada(inverso1,inverso2);
    Escala(1.04,1.04);
    int i,j,k;

   //Llevando figura al origen
   for (i=0;i<=8;i++)
    {  for (j=0;j<=2;j++)
        {  NT2[i][j]=0;
           for (k=0;k<=2;k++)
            {
             NT2[i][j]=NT2[i][j]+(u[i][k]*traslacion[j][k]);
            }
        }
    }
```

Tenemos la función para escalar una figura, recibimos como parámetro la matriz identidad de escala y la matriz de puntos de la figura en este caso la letra u en la televisión

Hacemos el traslado de la figura al origen a partir del vector inverso del pivote.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

```
//aplicando operaciones de escala
for (i=0;i<=8;i++)
{  for (j=0;j<=2;j++)
    {  NE[i][j]=0;
       for (k=0;k<=2;k++)
        {
         NE[i][j]=NE[i][j]+(NT2[i][k]*escala[j][k]);
        }
    }
}

//Regresando figura al origen
Traslada((inverso1*-1),(inverso2*-1));
for (i=0;i<=8;i++)
{  for (j=0;j<=2;j++)
    {  NE2[i][j]=0;
       for (k=0;k<=2;k++)
        {
         NE2[i][j]=NE2[i][j]+(NE[i][k]*traslacion[j][k]);
        }
    }
}
```

Aplicamos la operación correspondiente para escalar y obtener los nuevos puntos y regresamos la figura a su lugar de origen.

```
//actualizando letra u
for (i=0;i<=8;i++)
{
    for (j=0;j<=2;j++)
    {u[i][j] = NE2[i][j];
    }
}
```

Finalmente actualizamos lo valores de la letra u para que se dibuje en cada iteración.

En la función DibujaLinea tenemos los puntos que dibujan las figuras

```
void DibujaLinea()
{

    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.0,0.0,0.6);     /

    //CAMA
    glBegin(GL_LINE_LOOP);
       glVertex2i(0,140);
       glVertex2i(300,140);
       glVertex2i(300,130);
       glVertex2i(0,130);
    glEnd();

    glBegin(GL_LINE_STRIP);
       glVertex2i(0,130);
       glVertex2i(0,105);
       glVertex2i(10,105);
       glVertex2i(10,130);
    glEnd();
    glBegin(GL_LINE_STRIP);
       glVertex2i(15,130);
       glVertex2i(15,115);
       glVertex2i(25,115);
       glVertex2i(25,130);
    glEnd();
```

Y en las figuras que vamos a animar las trazamos respecto a su matriz de puntos.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

```
//pintado del avion
glBegin(GL_LINE_STRIP);
   for (int i=0; i<=6; i++)
   {
        for (int j=0; j<=0; j++)
        {
        glVertex2i(avion[i][j],avion[i][j+1]);
        }
   }
glEnd();

//pintado letra u
glBegin(GL_LINE_STRIP);
   for (int i=0; i<=8; i++)
   {
        for (int j=0; j<=0; j++)
        {
        glVertex2i(u[i][j],u[i][j+1]);
        }
   }
glEnd();
```

Hacemos uso de la función GL_LINE_STRIP para un manejo más fácil.

```
Sleep(250);
//TRASLADO DEL AVIÓN
if(avion[0][0]<580)
{
OperTras(traslacion,avion);
}

if(u[1][0]>416){
OperEsca(escala,u);}
OperRota(rotacion,aspa);
OperRota2(rotacion,aspa2);
```

Finalmente hacemos un retraso con la función Sleep(); y checamos los rangos del primer punto en el caso del avión para que no salga de la pantalla.

Como última observación respecto al código del programa hacemos uso de la función

glutIdleFunc(); para generar iteraciones constantes y ver el movimiento de la animación.

```
int main(int argc, char **argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE
    glutInitWindowSize(1000,640);
    glutInitWindowPosition(50, 50);

    glutCreateWindow("Escenario");
    /*******************/


    init(); //funcion init
//    trasladaPoli();
    glutDisplayFunc(DibujaLinea);

    glutIdleFunc(DibujaLinea);
    glutMainLoop();


    return 0;
}
```

Conclusión.

Alberto Esteban Reyes Peralta
Matricula: 201316680
Graficación

En este proyecto se pudo hacer la animación de 3 figuras en 2D a partir de operaciones con matrices, si bien pudimos utilizar las funciones proporcionadas por opengl se trababa y se logró hacerlo “a mano” como se describió en cada uno de los pasos del desarrollo del proyecto.

Bibliografía

[1] http://sabia.tic.udc.es/gc/Tutorial%20OpenGL/tutorial/cap3.htm#_Toc535127330

[2] http://arantxa.ii.uam.es/~pedro/graficos/teoria/Transformaciones2D/Transformaciones2D.htm

[3] ***users.dsic.upv.es/~jlinares/grafics/processing_spa_4.pdf***

Alberto Esteban Reyes Peralta
Matrícula: 221470780
Graficación

BENEMÉRITA UNIVERSIDAD

AUTÓNOMA DE PUEBLA

FACULTAD EN CIENCIAS DE LA

COMPUTACIÓN

1er Parcial

“TRANSFORMACIONES DE TRASLACION, ESCALA Y ROTACION”

MATERIA: GRAFICACIÓN

PROF: IVAN OLMOS PINEDA

ALUMNO: LUIS DAVID MORALES ALCÁNTARA

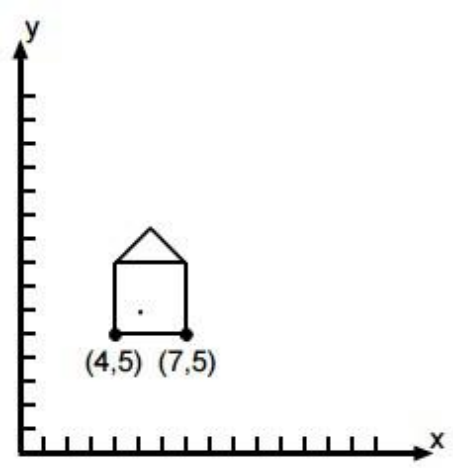
10/06/15

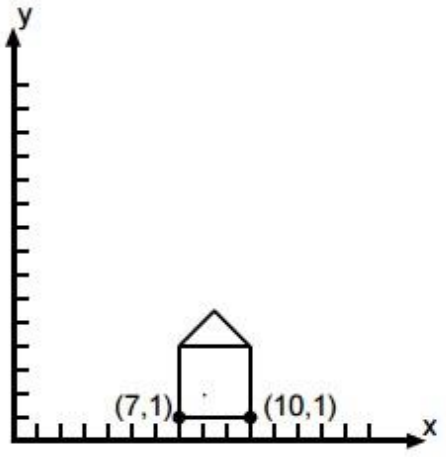
INTRODUCCION

En el siguiente documento hablare sobre la realización de las transformaciones de traslación, escalamiento y rotación para distintos objetos mediante la ayuda de las matrices para realizar cada transformación mencionada.

CONCEPTOS DE DESARROLLO

TRASLACION





INTRODUCCION

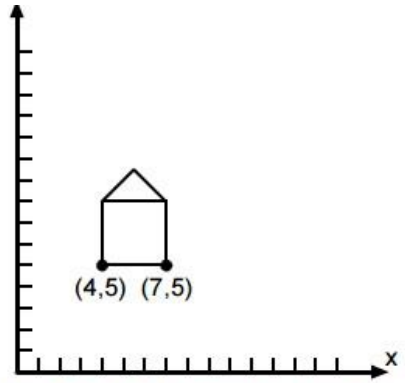
$$x' = x + d_x \qquad y' = y + d_y$$

– Vectorialmente:

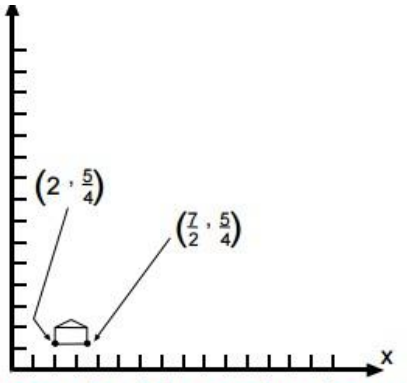
$$P = \begin{bmatrix} x \\ y \end{bmatrix}, P' = \begin{bmatrix} x' \\ y' \end{bmatrix}, T = \begin{bmatrix} d_x \\ d_y \end{bmatrix}, \qquad P' = P + T$$

Un movimiento de translación es cuando un objeto se mueve, se traslada de un lado a otro (cambio de posición) respecto a una referencia; por ejemplo un auto moviéndose sobre una calle o una persona caminando en cualquier sentido (x,y).

ESCALA



Antes del escalamiento



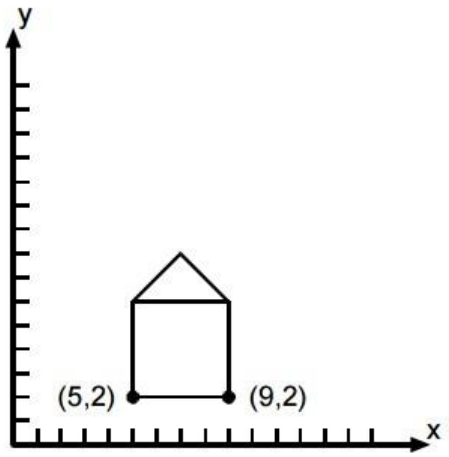
Después del escalamiento

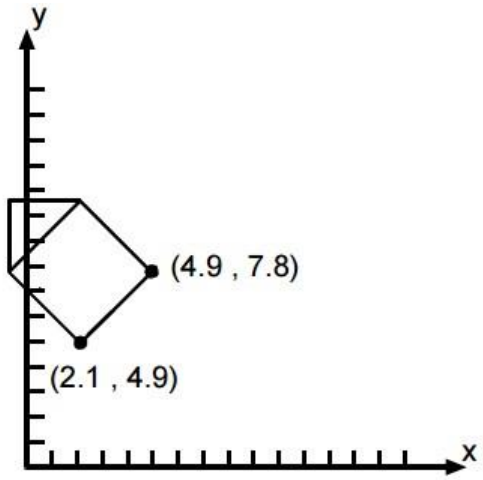
Es la proporción de aumento o disminución que existe entre las dimensiones reales y las dimensiones representadas de un objeto. En efecto, para representar un objeto de grandes dimensiones, deben dividirse todas sus medidas por un factor mayor que uno, en este caso denominado **escala de reducción**; y para representar objetos de pequeñas dimensiones, todas sus medidas se multiplican por un factor mayor que uno, denominado **escala de ampliación**. La escala a utilizar se determina entonces en función de las medidas del objeto y las medidas del papel en el cual será representado. El dibujo hecho a escala mantendrá de esta forma todas las proporciones del objeto representado, y mostrará una imagen de la apariencia real del mismo.

ROTACION

– Matricialmente

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -sen\theta \\ sen\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \Leftrightarrow P' = R \cdot P$$



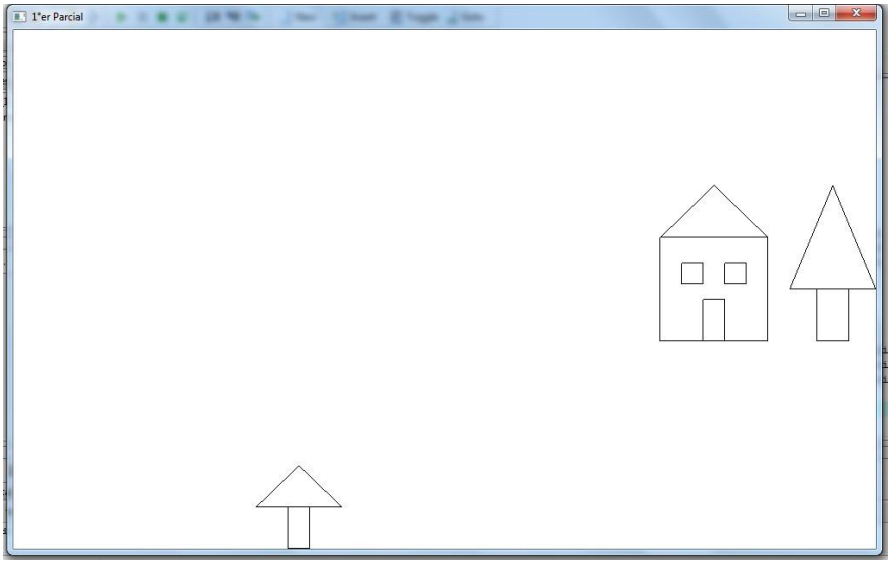


Es el movimiento de cambio de orientación de un cuerpo rígido de forma que dado un punto cualquiera del mismo, este permanece a una distancia constante de un punto fijo y tomando en cuenta la apertura que piensa rotar en cuestión de grados.

CONCLUSIONES

Mediante la investigación sobre las matrices con respecto a cada función y el manejo correcto de la multiplicación de las mismas podemos obtener los resultados de traslación y rotación un poco sencillas; y comparación con la

rotación hay que convertir grados a radianes para obtener los resultados correctos a cada operación a realizar. Y así lograr el resultado que buscamos desde un principio.


1°er Parcial

BIBLIOGRAFIA

Gráficos por computadora con OpenGL,09/06/15,

https://ingenieriayeducacion.files.wordpress.com/2013/12/graficosporcomputadora yopengl.pdf

Graficación

Gráfica con uso de clases-1er Parcial.

Andrea Estephany Sánchez Hernández 201225072

Iván Olmos Pineda

Primavera 2015

Contenido

Introducción

Una clase define un nuevo tipo de dato que especifica la forma de un objeto. Una clase incluye los datos y el código que operará sobre esos datos. Además, una clase enlaza datos y código. En conjunto con operaciones matriciales obtendremos un proyecto más limpio, menos saturado y optimizado.

Conceptos Desarrollados

Clases

Una clase es creada con la palabra reservada class. Por Ejemplo:

```
class clase1 {
private:
[Tipo de dato] variables;
public:
clase();
~clase();
[Tipo de dato] métodos;

};
```

En private se deberán colocar las variables que se usaran dentro de las operaciones, mientras que en public se colocaran los métodos (nótese que su estructura es igual que en un diagrama de clases). (Nota: es necesario ; al final de cada clase) Para la implementación de clases se recomienda separar la declaración de clases de la implementación de los métodos, para lo cual llamaremos al ejemplo anterior clase1.h y crearemos un nuevo archivo llamado clase1.cpp el cual tendrá la siguiente

estrucutra:

```
clase::clase()
{
//inicializar variables
[Tipo de dato] variables;
}
clase::~clase(){}
[Tipo de dato] clase::métodos()
{
```

```
//operaciones
}
```

Operaciones Matriciales

Para realizar operaciones matriciales utilizaremos la matriz identidad (la cual es una matriz diagonal en la que los elementos de la diagonal principal son iguales a 1).

Matriz identidad:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Matriz trasladar.

Para la matriz trasladar tendremos que realizar una multiplicación entre la matriz identidad y los puntos [x,y] que se desean trasladar para obtener:

[x+v1]

[y+v2]

[1]

Como el punto ya trasladado, para lo cual debemos modificar la matriz identidad con los puntos [v1,v2] para obtener esos resultado (Recordando la multiplicación entre una matriz de (i,j) x por una matriz de (j,k) obtendremos una matriz de (i,k)). Así que finalmente tendremos:

$$\begin{bmatrix} 1 & 0 & v1 \\ 0 & 1 & v2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} X+v1 \\ Y+v2 \\ 1 \end{bmatrix}$$

Matriz escalar.

Utilizando [r1, r2] como nuestra escala y el mismo procedimiento para la matriz trasladar tendremos:

[r1 0 0] [X] [r1*X]

[0 r1 0] [Y]= [r2*Y]

[0 0 1] [1] [1]

Matriz rotar.

Utilizando [ang] como el ángulo con el cual se rotara y utilizando el procedimiento anterior tendremos:

[cos ang –sen ang 0] [X] [cos ang*X - Y*sen ang]

[sen ang cos ang 0] [Y]= [cos ang*Y + X*sen ang]

[0 0 1] [1] [1]

Matriz Reflexión.

Para trasladar de un lado a otro tendremos 3 casos, así tomando un eje de simetría utilizaremos la matriz reflexión de la siguiente manera:

Reflexiones

Eje x →[x,-y]

Eje y →[-x,y]

Eje xy→[-x,-y]

Con las matrices de la siguiente manera:

Eje x:

[1 0 0] [X] [x]

[0 -1 0] [Y]= [-y]

[0 0 1] [1] [1]

Eje y:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} -x \\ y \\ 1 \end{bmatrix}$$

Eje xy:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} -x \\ -y \\ 1 \end{bmatrix}$$

Matriz Deformación

En esta matriz también tendremos 3 casos. Utilizando los mismos pasos de los procedimientos anteriores tendremos:

En eje x:

$$\begin{bmatrix} 1 & sh & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} x+sh*y \\ y \\ 1 \end{bmatrix}$$

En eje y:

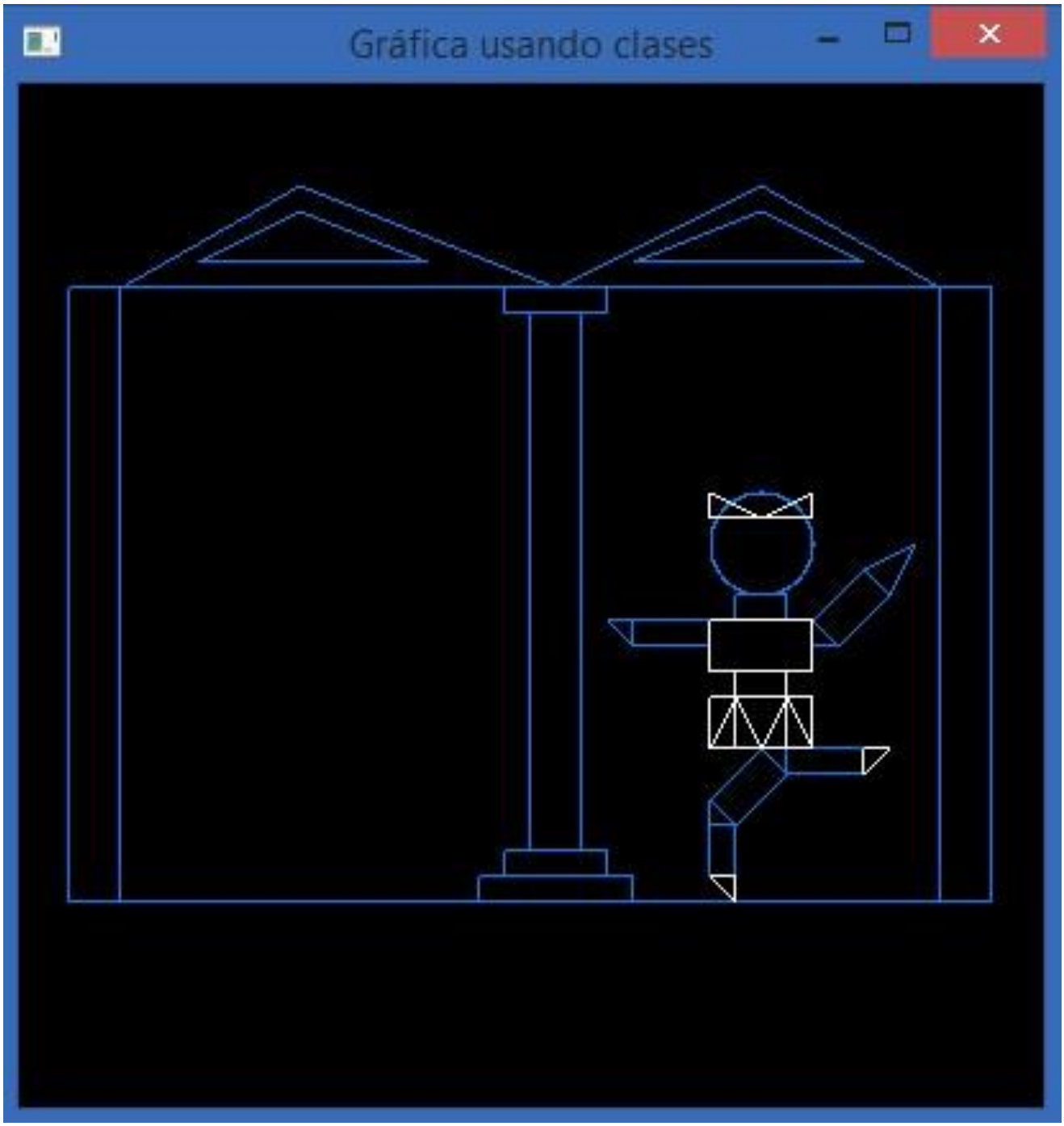
$$\begin{bmatrix} 1 & 0 & 0 \\ sh & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y+x*sh \\ 1 \end{bmatrix}$$

En eje xy:

$$\begin{bmatrix} 1 & sh & 0 \\ sh & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} x+sh*y \\ y+x*sh \\ 1 \end{bmatrix}$$

Dibujo

Esta es la gráfica que mostrará el programa al ser compilado.



cil el
s de
ciales
s, de
que

web:

Sitio

en 1

user

Introducción

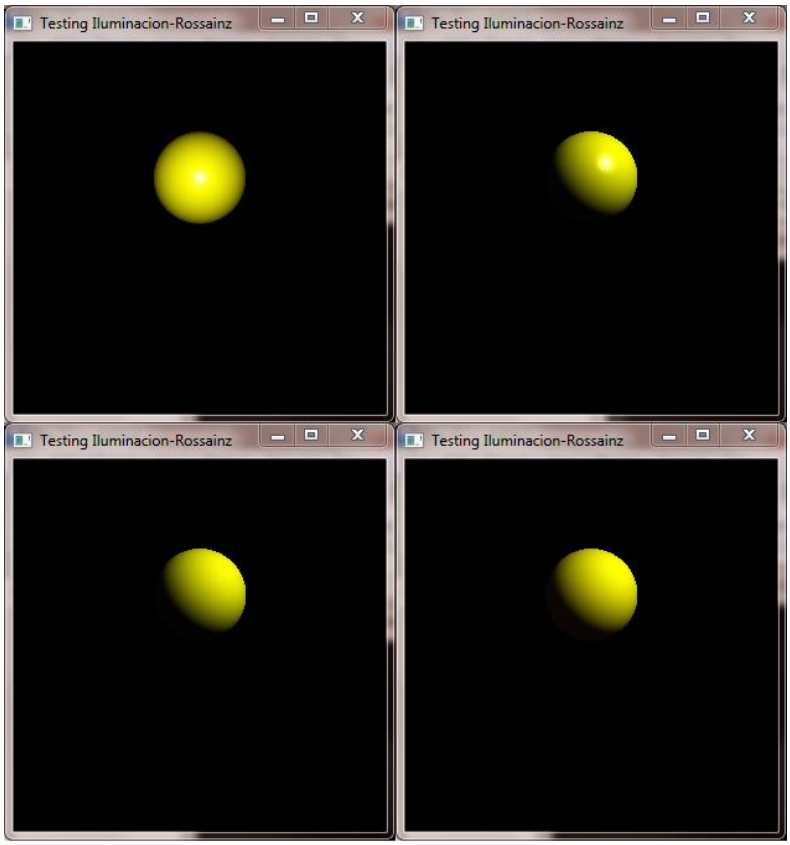
Con motivo del proyecto final de la materia graficación prepare un escenario en 3D que requería usar **texturas**, **iluminación**, y **transformaciones**. Decidí hacer algo un tanto urbano y tranquilo como he hecho con las prácticas hasta ahora y no tarde en decidirme por el **interior de un bar**, después de algunos días pensé, además, en darle una temática: “The Beatles”, porque a todos nos gusta al menos una canción de la banda británica. Así nació “The Apple Bar”.

Introducción

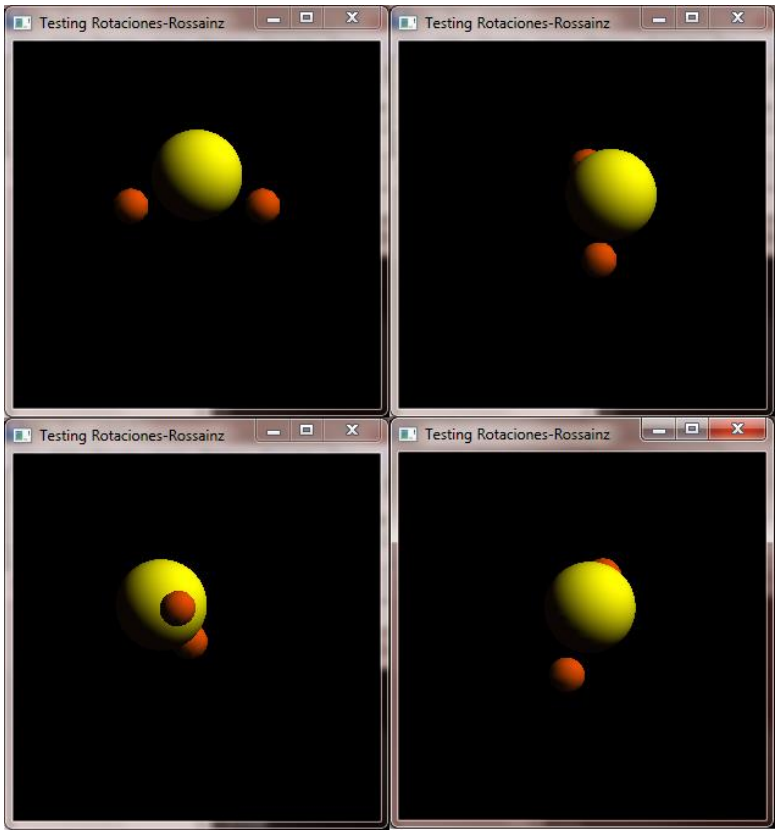
- Iluminación
- Texturas en 3D
- Transformaciones en 3D

Experimentos

Realice algunos experimentos para ‘*testear*’ diferentes aspectos que iba a necesitar para el escenario final.



Como en el ejemplo a la izquierda en el que probé varias combinaciones de luz. Que me ayudo a dar con las combinaciones necesarias para los diferentes focos que necesitaría.



En esto otro probé varias cosas:

- Como rotar un objeto en 3D y cómo respondía a los diferentes ejes.
- Como hacer funcionar una función de teclado para rotar el un objeto deseado
- Por último, si era más sencillo mover el LookAt o rotar un el escenario completo.

Bibliografía

Material del Curso por el Profre. Ivan Olmos Pineda
http://www.cs.buap.mx/~iolmos/Curso_Graficacion.html

Apuntes de OpenGL y GLUT por Cristina Cañero Morales
http://www.elai.upm.es/webantigua/spain/Asignaturas/InfoInd/teoria/Apuntes%20de%20OpenGL.pdf

Lenguaje de Programacion: C++ GLUT Iluminación por José Luis Alonzo Velázquez
http://www.cimat.mx/~pepe/cursos/lenguaje_2010/slides/slide_48.pdf

Artículos con Clase por Javier Correa Villanueva
http://articulos.conclase.net/?tema=graficos&art=glut&pag=002

Uso de Texturas con OpenGL
http://informatica.uv.es/iiguia/AIG/docs/texturas.htm

Oocities
http://www.oocities.org/valcoey/textura.html

OpenGL.org
https://www.opengl.org/discussion_boards/showthread.php/127317-Rotating-Camera-in-GLUT

Manual técnico
Visor de imágenes
en Java

(Parcial I)

Miguel Ángel Del Rello González 200710495

Introducción

El profesor de la clase de Procesamiento Digital de imágenes impartida en la Facultad de Ciencias de Computación de la Benemérita Universidad Autónoma de Puebla nos ha pedido el desarrollo de una aplicación en Java que nos permita primeramente abrir imágenes tipo: .jpeg, .bmp; y visualizarlas en pantalla. Esta aplicación ira evolucionando conforme sea requerida en clase.

Al momento de escribir este manual, la aplicación es capaz de:

- Abrir una imagen y mostrarla
- Guardar una imagen.
- Reconocer si se trata de una imagen en color, escala de grises o blanco y negro.
- Mostar el histograma de la imagen abierta (ya sea de 3 o un solo canal según sea el caso).
- Aplicación de la función identidad a la imagen leída.
- Conversión negativa de la imagen.
- Realizar un proceso de segmentación mediante la especificación de un rango de color para aislar (utilizado para la segmentación de una imagen de una célula)
- Aplicación de filtro de corrección Gamma.
- Aplicación de filtros de aclarado: Función Logarítmica, Función Seno y Función Exponencial por rangos.
- Función de oscurecimiento cosenoidal.
- Aplicación de contraste con Filtro Sigmoide.

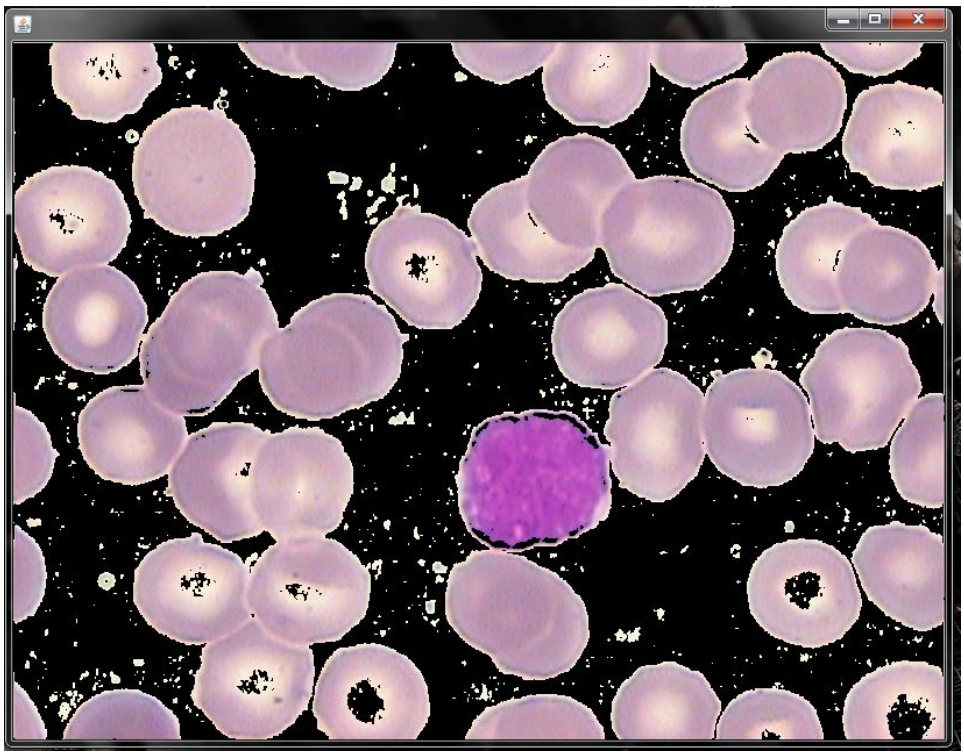
Objetivo

El objetivo del desarrollo de esta aplicación en Java es el poder observar la implementación de los algoritmos estudiados en clase.

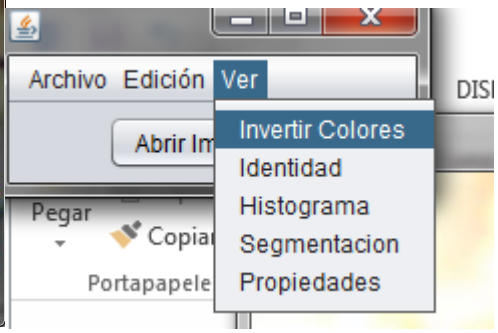
El objetivo de este manual técnico es el de proporcionar una guía de del desarrollo e implementación de la aplicación, su forma de instalación y utilización de la misma.

Funciones Implementadas en el sistema

- Leer una imagen.
- Mostrar una imagen.
- Guardar imagen.
- Reconocer si se trata de una imagen en color, escala de grises o blanco y negro.
- Mostrar histograma de la imagen
    - Muestra el histograma de la imagen abierta, ya sea de 3 o un solo canal según sea el caso.
- Reconocimiento y Segmentación de una imagen de células.
    - Se realizó un análisis previo de la imagen a segmentar, y se colocaros los valores importantes del histograma de la imagen en la aplicación para realizar la segmentación.
    - Los valores se colocaron de la siguiente forma:
        - ri=149; rf=253;
        - gi1=85; gf1=130; gi2=150; gf2=200;
        - bi1=107; bf1=150; bi2=207; bf2=230;
    - De esta forma se leen los valores RGB de cada uno de los pixeles que conforman la imagen, y si no están dentro de dicho rango se convierten en pixeles en color negro como se muestra a continuación:



- Funciones punto sobre la imagen (Operación Identidad y Negativo de una imagen)
    - Se realiza una lectura pixel a pixel de la imagen y se vuelve a colocar el mismo valor en la imagen guardada en el buffer y vuelve a imprimirse en el frame, el resultado es la imagen exactamente igual a la vista pero que ha sido re insertados todos sus valores RGB.
    - Para la Operación Negativo se lee el valor RGB de cada uno de los pixeles que conforman la imagen, se declaró un Color negro con la clase “Color” con valores RGB, después se le resta el valor del color negro “255” al valor del color leído en la imagen (valor complemento a 255) y es éste el que se inserta en el pixel de la imagen, lo que genera la el negativo de la imagen.



Codificación

Parte de Segmentación

```
public void segmentacion()
  {
     int ri,rf,gi1,gf1,gi2,gf2,bi1,bf1,bi2,bf2, r,g,b,rgb,black;
     ri=rf=gi1=gf1=gi2=gf2=bi1=bf1=bi2=bf2=r=g=b=rgb=black=0;
     ri=149; rf=253;
     gi1=85; gf1=130; gi2=150; gf2=200;
     bi1=107; bf1=150; bi2=207; bf2=230;

     Color myColor = new Color(0, 0, 0);
     black=myColor.getRGB();

     for(int i=0; i<img.getWidth(); i++)
             {
                for(int j=0; j<img.getHeight(); j++)
                   {
                      rgb=img.getRGB(i, j);
                      b = (rgb)&0xFF;
                      g = (rgb>>8)&0xFF;
                      r = (rgb>>16)&0xFF;
                      if(ri<=r && r<=rf || gi1<=g && g<=gf1 && gi2<=g && g<=gf2 || bi1<=b && b<=bf1 && bi2<=b &&
b<=bf2)
                      {
                         System.out.println("Coordenada sin modificar: ["+i+", "+j+"]");
                      }else{
                            img.setRGB(i, j, black);
                            }
                   }
             }
     f.repaint();
     f.pack();
     f.setVisible(true);
```

```
}
```

Operaciones Punto

Función identidad

public void identidad()

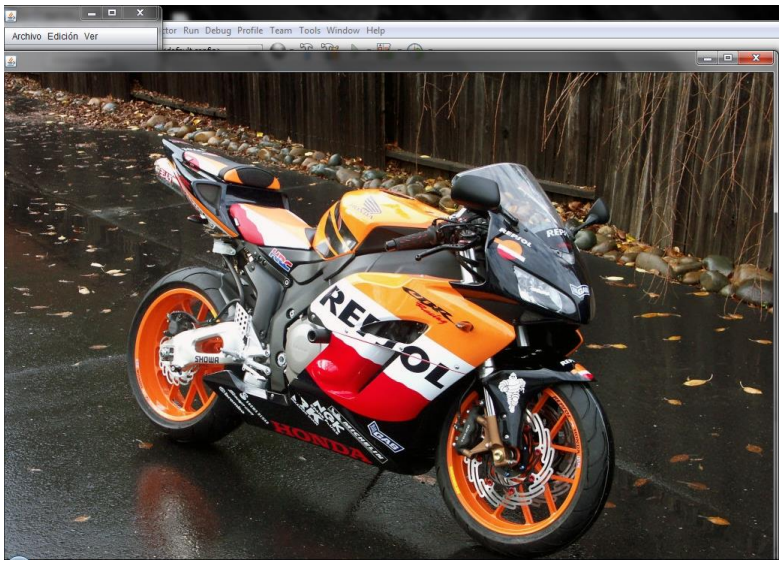
```
{
    int rgb=0;
    int []B = new int [img.getHeight()*img.getWidth()];
    int []G = new int [img.getHeight()*img.getWidth()];
    int []R = new int [img.getHeight()*img.getWidth()];
    int k=0;

    for(int i=0; i<img.getWidth(); i++)
      {
        for(int j=0; j<img.getHeight(); j++)
          {
            rgb = img.getRGB(i, j);
            img.setRGB(i, j, rgb);
            B[k] = (rgb)&0xFF;
            G[k] = (rgb>>8)&0xFF;
            R[k] = (rgb>>16)&0xFF;
            k++;
          }
      }
    f.repaint();
    f.pack();
    f.setVisible(true);
  }
```

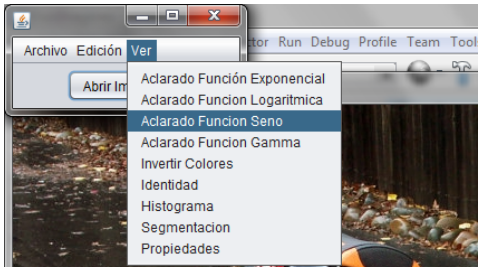
public void negativo()

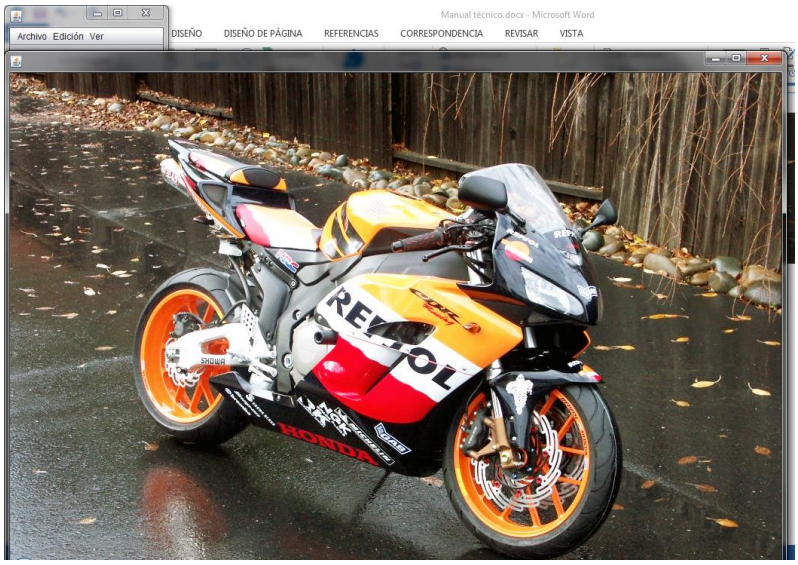
```
{
    int rgb=0;
    Color myColor = new Color(255, 255, 255);
    int b= myColor.getRGB();
    for(int i=0; i<img.getWidth(); i++)
      {
```

```
        for(int j=0; j<img.getHeight(); j++)
          {
             rgb = img.getRGB(i, j);
             img.setRGB(i, j, b-rgb);
          }
      }

   f.repaint();
   f.pack();
   f.setVisible(true);

}
```

- **Funciones punto sobre la imagen (Aclarado con Función Seno, logarítmica y exponencial)**
  - Se realiza una lectura pixel a pixel de la imagen y se coloca el valor arrojado por cada una de las funciones correspondientes en la imagen guardada en el buffer y vuelve a imprimirse en el frame, el resultado es la misma imagen con un aclarado acorde a la función utilizada.







Codificación

```
public void AclaradoFuncionLogaritmica() //Funcion Logaritmica
  {

     int alpha = 1;
     int q = 255;
     int A = (int) (255/log(256));
     int rgb = 0;
     int B = 0;
     int G = 0;
     int R = 0;

     Color col = new Color(255, 255, 255);

     for(int i=0; i<img.getWidth(); i++)
       {
          for(int j=0; j<img.getHeight(); j++)
            {
               rgb = img.getRGB(i, j);
               B = (rgb)&0xFF;
               G = (rgb>>8)&0xFF;
               R = (rgb>>16)&0xFF;
               col = new Color((int)Math.floor(A * log(alpha*(R+1))),(int)Math.floor(A *
log(alpha*(G+1))),(int)Math.floor(A * log(alpha*(B+1))));
               img.setRGB(i, j, col.getRGB());
            }
       }
```

```
    f.repaint();
    f.pack();
    f.setVisible(true);

}

        public void AclaradoFuncionSeno() //Funcion Seno
{

    int alpha = 1;
    int q = 255;
    int u = q;
    double k = Math.PI/(2*q);
    int A = (int) (255/log(256));
    int rgb = 0;
    int B = 0;
    int G = 0;
    int R = 0;

    Color col = new Color(255, 255, 255);

    for(int i=0; i<img.getWidth(); i++)
      {
         for(int j=0; j<img.getHeight(); j++)
           {
              rgb = img.getRGB(i, j);
              B = (rgb)&0xFF;
              G = (rgb>>8)&0xFF;
              R = (rgb>>16)&0xFF;
```

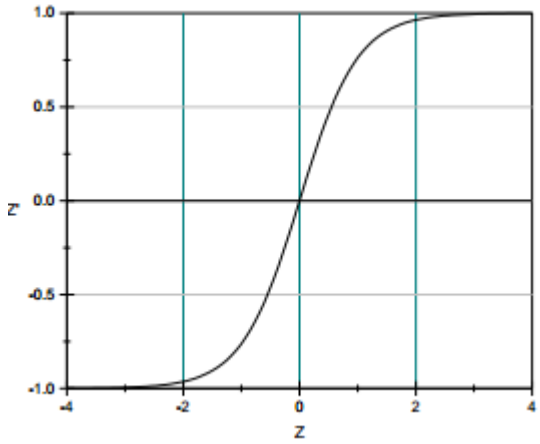
```
                col = new Color((int)Math.floor(q*Math.sin((Math.PI*R)/(2*q))),
(int)Math.floor(q*Math.sin((Math.PI*G)/(2*q))), (int)Math.floor(q*Math.sin((Math.PI*B)/(2*q))));
                img.setRGB(i, j, col.getRGB());
            }
        }
    f.repaint();
    f.pack();
    f.setVisible(true);

  }

  public void AclaradoFuncionExponencial() //Funcion exponencial
  {

    int alpha = 1;
    int q = 255;
    double A = q/(1-(Math.exp(-alpha)));
    int rgb = 0;
    int B = 0;
    int G = 0;
    int R = 0;

    Color col = new Color(255, 255, 255);

    for(int i=0; i<img.getWidth(); i++)
      {
        for(int j=0; j<img.getHeight(); j++)
          {
            rgb = img.getRGB(i, j);
            B = (rgb)&0xFF;
```

```
                G = (rgb>>8)&0xFF;
                R = (rgb>>16)&0xFF;
                col = new Color((int)Math.floor(A*(1-(Math.exp(-((alpha*R)/q))))), (int)Math.floor(A*(1-(Math.exp(-((alpha*G)/q))))), (int)Math.floor(A*(1-(Math.exp(-((alpha*B)/q)))))));
                img.setRGB(i, j, col.getRGB());
            }
        }
    f.repaint();
    f.pack();
    f.setVisible(true);

  }
```

- **Funciones punto sobre la imagen (Filtro de contraste con Función Sigmoide)**

En la transformación sigmoide los tonos oscuros se oscurecerán más y los claros se aclararán más, éste comportamiento lógicamente aumentará el contraste, es decir la separación (distinción) entre claros y oscuros crecerá en la imagen transformada por la función sigmoide.

Un tipo de filtro sigmoide se basa en la función tangente hiperbólica, su forma gráfica se muestra a continuación.

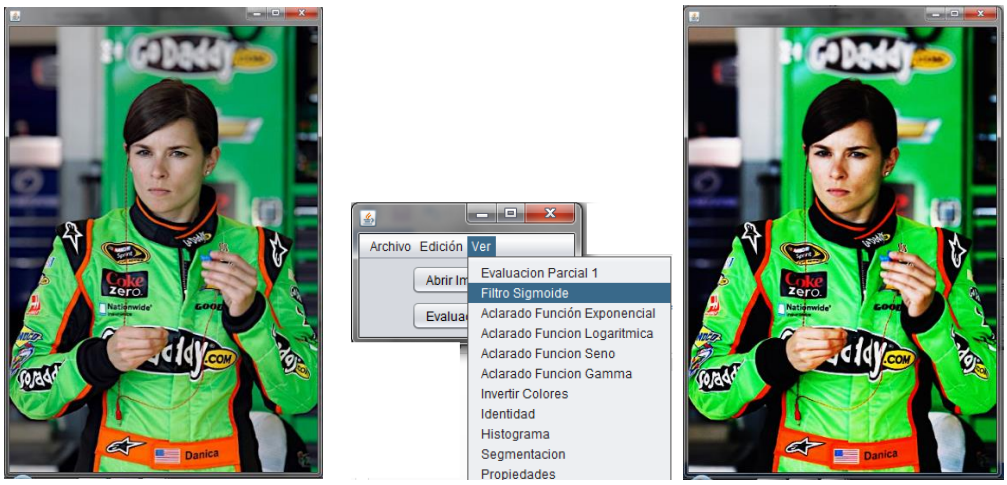


El cálculo de la función Sigmoide en Java se realizó de la siguiente manera:

(int) Math.floor((255/2)* (1 + Math.tanh(Math.toRadians(alpha*(X-(255/2))))))

X representa el valor de rojo, verde o azul del RGB del pixel leído, en este caso la función no fue implementada para trabajar con un rango especificado por el usuario así que se toma el rango total del canal [0, 255]

Ejemplo de ejecución



Codificación (Filtro Sigmoide)

```
public void ContrasteFuncionSigmoide()
  {
```

```
    // Función (q/2)*(1+tan(alpha(x-(q/2))))
    float alpha = 1;
     int q = 255;
     int rgb = 0;
     int B = 0;
     int G = 0;
     int R = 0;
   Color col = new Color(255, 255, 255);


     for(int i=0; i<img.getWidth(); i++)
       {
          for(int j=0; j<img.getHeight(); j++)
            {
               rgb = img.getRGB(i, j);
               B = (rgb)&0xFF;
               G = (rgb>>8)&0xFF;
               R = (rgb>>16)&0xFF;
               col = new Color((int) Math.floor((255/2)* (1 + Math.tanh(Math.toRadians(alpha*(R-(255/2)))))),(int)
Math.floor((255/2)* (1 + Math.tanh(Math.toRadians(alpha*(G-(255/2)))))),(int) Math.floor((255/2)* (1 +
Math.tanh(Math.toRadians(alpha*(B-(255/2))))));
               img.setRGB(i, j, col.getRGB());
            }
       }
     f.repaint();
     f.pack();
     f.setVisible(true);


  }
```

Desarrollo de Aplicación para Parcial I

Para la evolución del primer parcial se generó una nueva interfaz muy sencilla donde poder aplicar varios filtros de corrección, ya sea de aclarado oscurecimiento etc. Con parámetros de modificación especificando rangos de modificación en los valores de los canales RGB para que la función seleccionada no sea aplicada a todos los valores sino solo los que se encuentren en el rango [x0, x1] especificado por el usuario, así como valores para alpha según sea el caso.

Modificaciones realizadas a las funciones de corrección para aplicación solo a rangos de colores RGB especificados por el usuario

Para todas las funciones se les realizo una modificación para que no exista un salto tan marcado al solo presentar una fracción de la función general (para que se vea una variación de color mucho más suave entre dos pixeles que se encuentren en la frontera de los rangos especificados). Esta modificación es igual a trasladar la gráfica de la función hasta el origen, para este fin en cada una de las funciones se realizó una resta del valor inicial X0 a cada uno de los valores de x (valor del canal) y al final de la función una suma del mismo X0 (representa el despeje del valor de y en el plano cartesiano y que también debe ser modificado para poder trasladar la función al origen); y que la aplicación de la función se detenga alcanzando el punto máximo del rango en este caso representado por 'q'; además la ejecución de las funciones solo se realiza como tal en los valores del canal que se encuentren dentro del rango especificado y los demás valores se reasignan con el valor de la función identidad antes vista.

Aclarado Función Gamma

```
R = (int)(Math.floor((q-x)*(Math.pow((double)(R/q), alpha)))+x);
```

Nota: el valor de x, q y alpha son especificados por el usuario y pasados por parámetros.

Aclarado Función Logarítmica

```
double A = (double) ((q-x)/(Math.log(255-x)));

r = (double)Math.floor(A * log(alpha*((R-x)+1))+x);
```

Nota: el valor de x, q y alpha son especificados por el usuario y pasados por parámetros.

Aclarado Función Seno

```
r = (int)Math.floor((q-x)*Math.sin((Math.PI*(R-x))/(2*(q-x)))+x);
```

Nota: el valor de x, q y alpha son especificados por el usuario y pasados por parámetros.

Aclarado Función Exponencial

```
double A = ((q-x)/(1-(Math.pow(Math.E, -alpha))));

r=(int)Math.round((float)(A*(1-(Math.pow(Math.E, -((alpha*(R-x))/(q-x))))))+x);
```

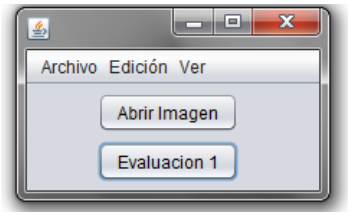
Nota: el valor de x, q y alpha son especificados por el usuario y pasados por parámetros.

Al Usuario final

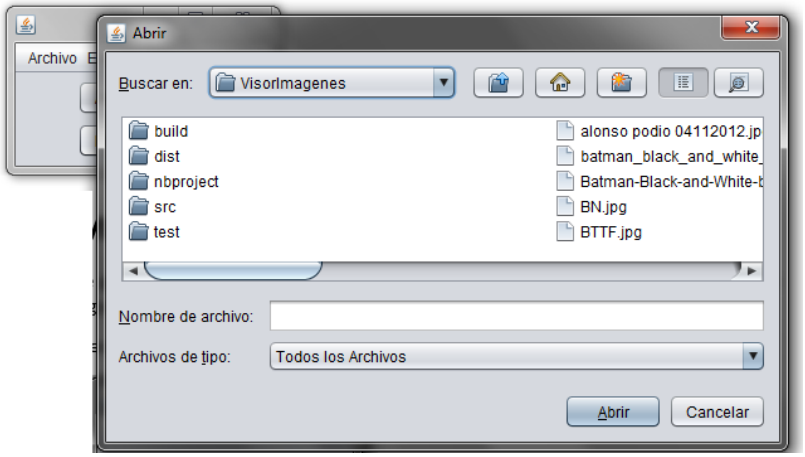
Modo de ejecución de Parcial I

Se generó una nueva y muy sencilla interfaz para este fin y la forma de ejecución es la siguiente:

Ejecutar nuestra aplicación **VisorImagenes.jar**; se mostrara la ventana principal de nuestra aplicación.

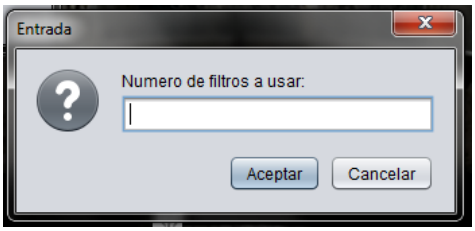


Utilizar el nuevo botón creado para la Evaluación del Parcial 1.

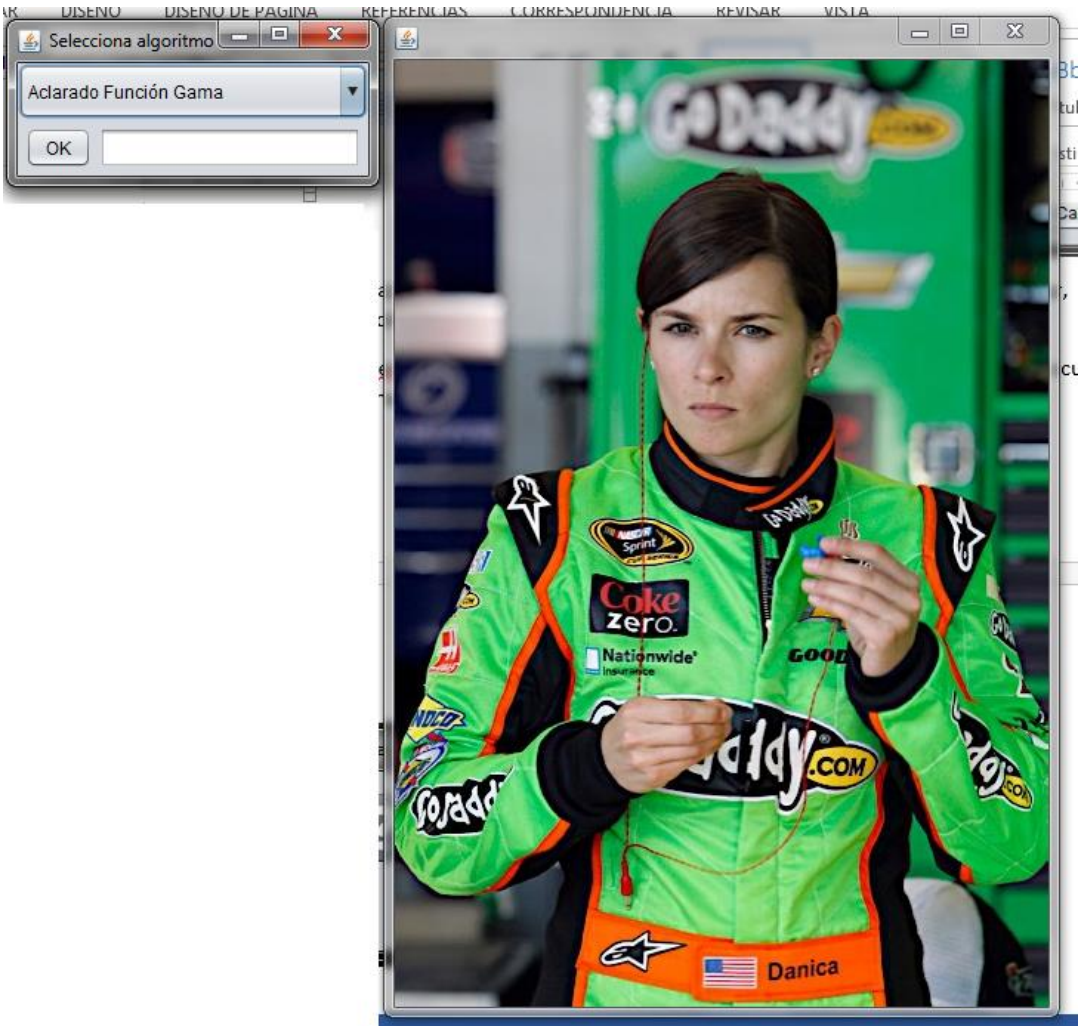


Inmediatamente se abre una ventana para la selección de la imagen a trabajar, Seleccionamos nuestra imagen y clic en Abrir.

Aparecerá una nueva ventana donde especificamos el número de filtros que ocuparemos en la imagen abierta y damos clic en aceptar.



Ahora se abrirá una nueva ventana que es la que nos permite la ejecución de cada una de las funciones de filtros implementadas. Se selecciona el filtro deseado y se da clic en el botón OK



Nota 1: Cada uno de los filtros seleccionados nos pedirá ingresemos el rango del canal [x0, x1] en el que deseamos aplicar el filtro así como el valor de alpha si es necesario.

Nota 2: Las funciones implementadas anteriormente (función identidad, negativo, mostrar histograma, etc.) pueden ser ejecutadas desde la ventana principal de nuestra aplicación. En el menú “Ver” esta cada una de las opciones.

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Graficación

Dr. Iván Olmos Pineda

Verano 2015

Alumno: Mario Alejandro Cortez Salinas 201302537

PROYECTO FINAL

ESCENARIO

Introducción

Los gráficos por computador se han convertido en una potente herramienta para la producción rápida y económica de imágenes. Prácticamente no existe ninguna tarea en la que la representación gráfica de la información no pueda aportar ventaja y, por tanto, no sorprende encontrar gráficos por computadora en muchos sectores.

El proyecto final consiste en realizar un escenario que contenga los elementos que se aprendieron en el curso para crear escenas que tengan movimiento, con texturas, y tipos de vistas.

Se decidió por realizar una vista 3D de una casa (la mía) en la que te puedes mover por ella, también se le ponen texturas para simular el pasto, las paredes y otros elementos. Se le ponen elementos que tengan movimientos.

Conceptos

Texturas

Una textura, desde el punto de vista de su almacenamiento en memoria es un array de datos. Este array de datos representa una imagen, que utilizaremos para mapearla sobre un polígono.

Podemos rellenar dicho array bien cargando una imagen desde un fichero.

La información de cada uno de los componentes puede ser:

- Las componentes RGB del color.
- Índices de color.
- Niveles de luminancia (grises).
- Para poder ver las texturas deberemos activarlas con la función:
- **glEnable(GL_TEXTURE_2D);**
- Especificar la textura.
- Normalmente partiremos de una imagen almacenada en algún formato standard (bmp, jpg, gif, rgb, etc). Para poder cargar esta imagen necesitaremos algún programa o biblioteca que lea este tipo de ficheros. Por la red se pueden encontrar

todo de tipo de cargadores de imagen. Debemos asegurarnos, utilizando algún editor de imágenes, que el ancho y el alto de la imagen es potencia de dos.

- Si la imagen la tenemos en formato **bmp** podemos utilizar también una función de la biblioteca glaux para cargarla:
- **AUX_RGBImageRec *auxDIBImageLoad(Filename);**
- Esta función nos devolverá una estructura de la cual podremos leer la información de tamaño y datos:
- **typedef struct _AUX_RGBImageRec {**
  GLint sizeX, sizeY;

  unsigned char *data;

  } AUX_RGBImageRec;
- Una vez hemos cargada de fichero la textura deseada, debemos definir la textura en OpenGl. Para ello utilizamos la función **glTextImage2D**

Transformaciones

Una transformación geométrica es un procedimiento que permite convertir un punto en otro: $P \rightarrow P'$. O dicho de otro modo, al aplicar la transformación geométrica al punto

Las transformaciones que más comúnmente se usan son las traslaciones, escalados y rotaciones. Es muy importante hacer notar que las transformaciones geométricas se aplican antes de que el objeto sea visualizado, y que son precisamente los vértices transformados los que se usan para crear las primitivas.

Esto es así ya que se sabe que OpenGL aplicará las transformaciones a los vértices antes de usarlos para la visualización. Las transformaciones se definen OpenGL de la siguiente manera:

Traslación: glTranslate[fd](Δx,Δy,Δz)

Escalado: glScale[fd](Sx,Sy,Sz)

Rotacion: glRotate[fd](α,x,y,z)

Produce una rotación de α grados alrededor del eje definido por el vector V(x, y,z), en sentido contrario a las agujas del reloj.

La idea principal es que OpenGL utiliza una pila para combinar y almacenar las transformaciones. OpenGL tiene varias pilas de transformaciones: para las transformaciones geométricas y de la cámara, la pila GL_MODELVIEW, y para las proyecciones, la pila GL_PROJECTION (existen otras pilas). Para activar alguna de estas pilas se hace uso de la orden glMatrixMode(pila). Activada la pila GL_MODELVIEW, cada vez que OpenGL encuentra una transformación la combina con la que haya en el top de la pila (es importante recordar que las transformaciones geométricas se pueden combinar.

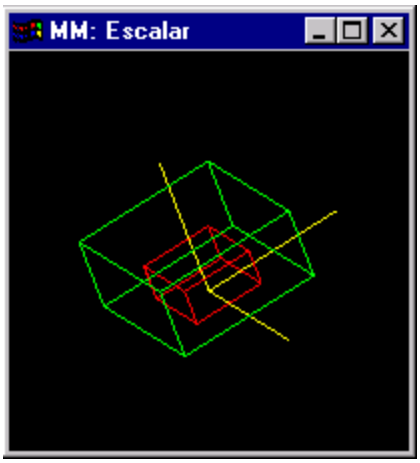
Si tengo dos transformaciones T0 y T1, la transformación T obtenida como el producto de $T0 * T1$ produce el mismo resultado que aplicarlas individualmente).

Escalar

Gracias a la función de escalado podemos aumentar/disminuir un objeto en cuanto a tamaño.

Ejemplo:

```
glScalef(2.0f, 4.0f, 1.0f);
```

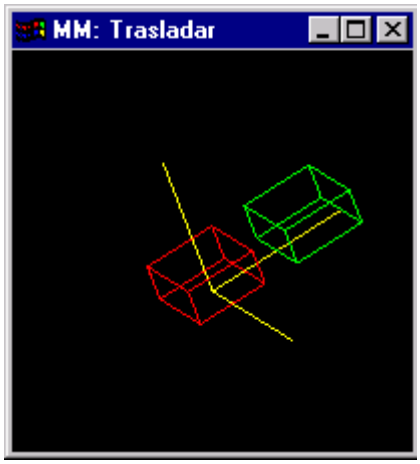


Trasladar

Esta es precisamente una transformación afín imposible de realizar en cartesianas si no se incluye una suma de matrices. Pero nosotros no queremos sumar, tan sólo multiplicar.

Ejemplo:

glTranslatef (x,y,z);

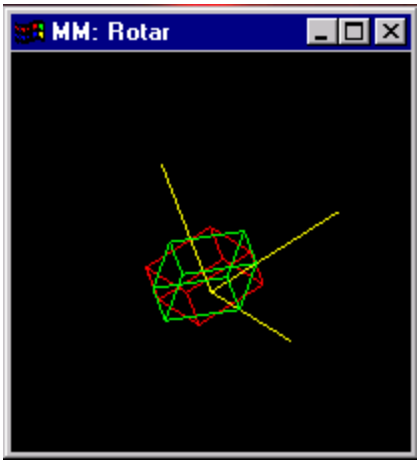


Rotar

La rotación debe realizarse siempre alrededor de un determinado eje de referencia. Podemos rotar alrededor del eje X, del eje Y o del eje Z.

Ejemplo:

```
glRotatef(angulo, 0.0f, 1.0f, 0.0f);
```



Las matrices y OpenGL

Anteriormente hemos visto que OpenGL guarda la transformación de los objetos en una matriz. A esta matriz se le denomina matriz de visualización/modelado, porque se emplea para estas dos funciones.

Además de esta transformación, OpenGL posee otra matriz muy importante, que es la matriz de proyección, en la que se guarda la información relativa a la "cámara" a través de la cual vamos a visualizar el mundo.

Al realizar operaciones que modifiquen alguna de estas dos matrices, tendremos que cambiar el "modo de matriz", para que las operaciones afecten a la matriz que nos interesa.

Para ello utilizaremos las funciones glMatrixMode(GL_MODELVIEW) o © 2003 - Jorge García (Bardok) 17 Curso de introducción a OpenGL (v1.0) Las funciones básicas de OpenGL glMatrixMode(GL_PROJECTION).

Además, existen dos funciones que permiten guardar y restaurar los valores de la matriz activa en una pila.

La función glPushMatrix() guarda una matriz en la cima de la pila, y glPopMatrix() la saca, y la restaura. Esto lo podemos utilizar para dibujar un objeto y, antes de dibujar el siguiente, restauramos la transformación inicial.

Finalmente, comentar la operación glLoadIdentity(), que carga la matriz identidad como matriz activa.

El dibujado en OpenGL

Para dibujar en OpenGL, tenemos que habilitar el modo de dibujado, establecer las opciones de dibujado de cada vértice, y dibujar cada uno de ellos. Al terminar de dibujar una figura, finalizamos el modo de dibujado.

Para comenzar a dibujar, utilizaremos el comando glBegin(), dónde el modo de dibujado vendrá dado por una constante:

Parámetro	***Descripción***
GL_POINTS	Se dibujan vertices separados
GL_LINES	Cada par de vértices se interpreta como una línea
GL_POLYGON	Los vértices describen el contorno de un polígono
GL_TRIANGLES	Cada triplete de vértices de interpreta como un triángulo
GL_QUADS	Cada cuarteto de vértices se interpreta como un cuadrilátero
GL_LINE_STRIP	Líneas conectadas
GL_LINE_LOOP	Líneas conectadas, con unión entre el primer y último vértice
GL_TRIANGLE_STRIP	Se dibuja un triángulo, y cada nuevo vértice se interpreta con un triángulo entre los dos anteriores vértices y el nuevo
GL_TRIANGLE_FAN	Se dibujan triángulos con un vértice común
GL_QUAD_STRIP	Igual que el TRIANGLE_STRIP, con cuadriláteros

Entre las funciones que permiten establecer los atributos de cada vértice, están aquellas que nos permiten seleccionar su color (glColor*), normal (glNormal*), coordenadas de textura (glTexCoord*), etc.

Finalmente, las funciones de dibujado de vértices tienen la forma “glVertex*”.

El color en OpenGL

OpenGL puede utilizar dos modos de color: color RGBA y color indexado. Nosotros vamos a centrarnos en el color RGBA. Este color recibe este nombre porque se compone de cuatro componentes: Rojo (Red), Verde (Green), Azul (Blue) y canal Alfa, o transparencia.

La orientación de las caras en OpenGL

Un polígono tiene dos caras, delantera y trasera. La manera de saber qué cara es la delantera, y cual la trasera, es que, si miramos la delantera, los vértices se habrán dibujado en orden antihorario.

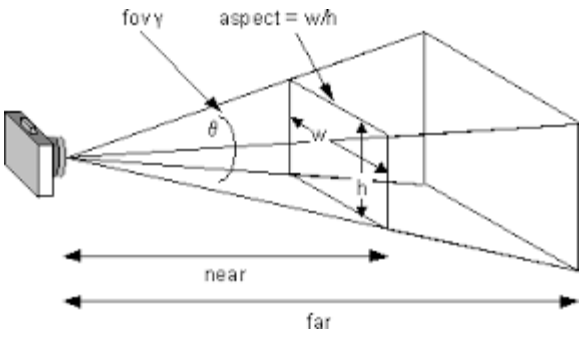
La proyección en OpenGL

En el modo de proyección podemos especificar cómo va a afectar la posición de un objeto a su visualización. Tenemos dos maneras de visualizar el espacio: con una proyección ortográfica, y con una proyección perspectiva:

✂ La proyección ortográfica:

La proyección ortográfica nos permite visualizar todo aquello que se encuentre dentro de un cubo, delimitado por los parámetros de la función glOrto. A la hora de visualizar, la distancia al observador sólo se tiene en cuenta para determinar si el objeto está dentro o fuera del cubo...

✂ La proyección perspectiva: La proyección perspectiva delimita un volúmen de visualización dado por un ángulo de cámara, y una relación alto/ancho. La distancia al observador determinará el tamaño con el que un objeto se visualiza.



Dibujando

Para poder tener varias texturas se cargan en un arreglo y asi se tienen disponibles todo el tiempo para ser usadas

Para poder cargar imágenes usamos “FreeImage.h” con sus respectivas sentencias para agregarlas a nuestro arreglo.

```
//variables para manejo de texturas
char *texturefiles[] = {
    "texturaPasto2ymedio.jpg","paredgris.jpg","ladrillo.jpg","paredRosa.jpg","pisoMad.jpg","cielo.jpg","television.jpg","cuadro1.jpg","cuadro2.jpg","pared-ladril
};

bool readImage()
{
    //image format
    FREE_IMAGE_FORMAT fif = FIF_UNKNOWN;
    //pointer to the image, once loaded
    FIBITMAP *dib(0), *dib_copy(0);
    //pointer to the image data
    BYTE* bits(0);
    //image width and height
    unsigned int width(0), height(0);
    //check the file signature and deduce its format
    for(int i=0;i<=10;i++)
    {
    fif = FreeImage_GetFileType(texturefiles[i], 0);
```

Se tienen funciones en las que se le agrega textura, esto se hace especificando las coordenadas del polígono a rellenar, asignando la textura que se usara y dando las posiciones que se tomaran de la textura.

Primero trasladamos el objeto a la posición que tendrá en el escenario, se le asigna una textura y se especifican las coordenadas.

Con “BindTexture” especificaremos que textura usaremos, dentro de los parámetros ingresamos el número del arreglo en donde este la textura que queremos.

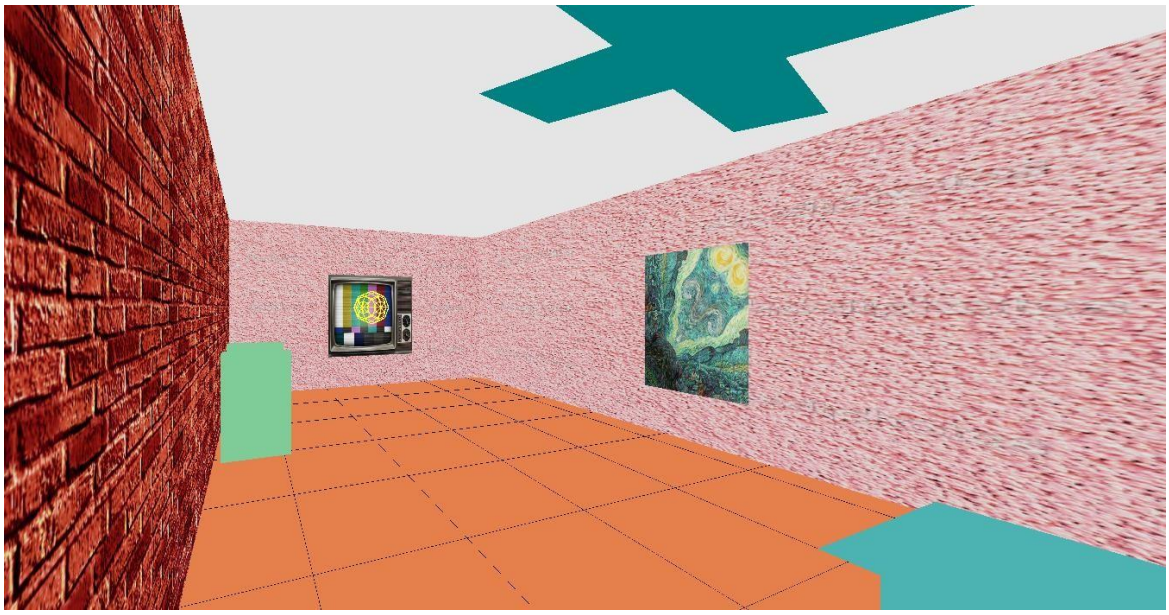
“glTextCord2f” con esta funcion opengl nos permite decir que parte de nuestra imagen se asignara a ese vértice del polígono.

```
void pintarPasto(){
    glPushMatrix();
    glTranslatef (0.0f,3.0f,0.0f);
    glColor3f(0.9, 0.9, 0.9);
    glEnable(GL_TEXTURE_2D);    //Dibuja pasto texturizado
    glBindTexture(GL_TEXTURE_2D, texture[0]);
    glBegin(GL_POLYGON);
        glTexCoord2f(0.0f,0.0f);
        glVertex3i(0,0,0);
        glTexCoord2f(0.0f,1.0f);
        glVertex3i(85,0,0);
        glTexCoord2f(1.0f,1.0f);
        glVertex3i(85,0,-52);
        glTexCoord2f(1.0f,0.0f);
        glVertex3i(0,0,-52);
    glEnd();
    glBindTexture(GL_TEXTURE_2D, texture[0]);
    glBegin(GL_POLYGON);
        glTexCoord2f(0.0f,0.0f);
        glVertex3i(0,0,-52);
```

Para poder tener un mejor control de los elementos del escenario se tiene una función que dibuja los elementos de cada cuarto, así se divide todo el escenario por zonas. Se usan las funciones “PushMatrix” y “PopMatrix” para tener control de lo que se esta dibujando.

“glTraslatef” se especifican las coordenadas X Y y Z del plano a donde nos deseamos mover.

```
void pintarCMama(){
    glPushMatrix();
    glTranslatef (0.0f,3.0f,0.0f);
    glColor3f(0, 0, 0);
    glBegin(GL_LINES);
    for(int i=87;i<100;i=i+3)
    {
        glVertex3i(i,0,0);
        glVertex3i(i,0,-70);
    }
    for(int i=0;i>-100;i=i-5)
    {
        glVertex3i(85,0,i);
        glVertex3i(100,0,i);
    }
    glEnd();
    glColor3f(1, .3, .3);
    glBegin(GL_POLYGON);
        glVertex3i(85,0,0);
        glVertex3i(100,0,0);
        glVertex3i(100,0,-16);
        glVertex3i(85,0,-16);
```

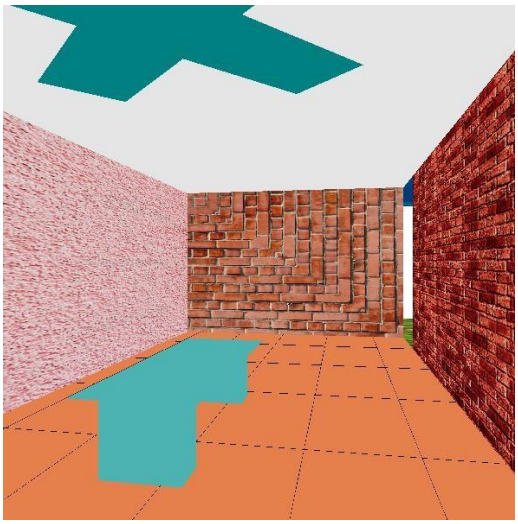


En muchas ocasiones se necesita de un polígono rectangular, como para la base de la mesa por ejemplo pero también se usa para crear nuestro sillón, variando sus posiciones y colores, por eso se tiene como parámetros de estrada ciertos datos.

“glutSolidCube” cuando necesitamos un un cubo esta funcion nos ayuda ya que solo le pasamos un parámetro y esa será la medida de cada uno de los lados de cubo que creará.

“glColor3f” ahí se especifica de qué color queremos que sea nuestro objeto a dibujar, los colores se definen en una escala del 0-1 RGB

```
void pintarTronco(float x, float y, float z){
    glPushMatrix();
    glTranslatef (x,y,z);
    glScalef(2.0f, 4.0f, 1.0f);
    glutSolidCube(1);
    glPopMatrix();

}
```



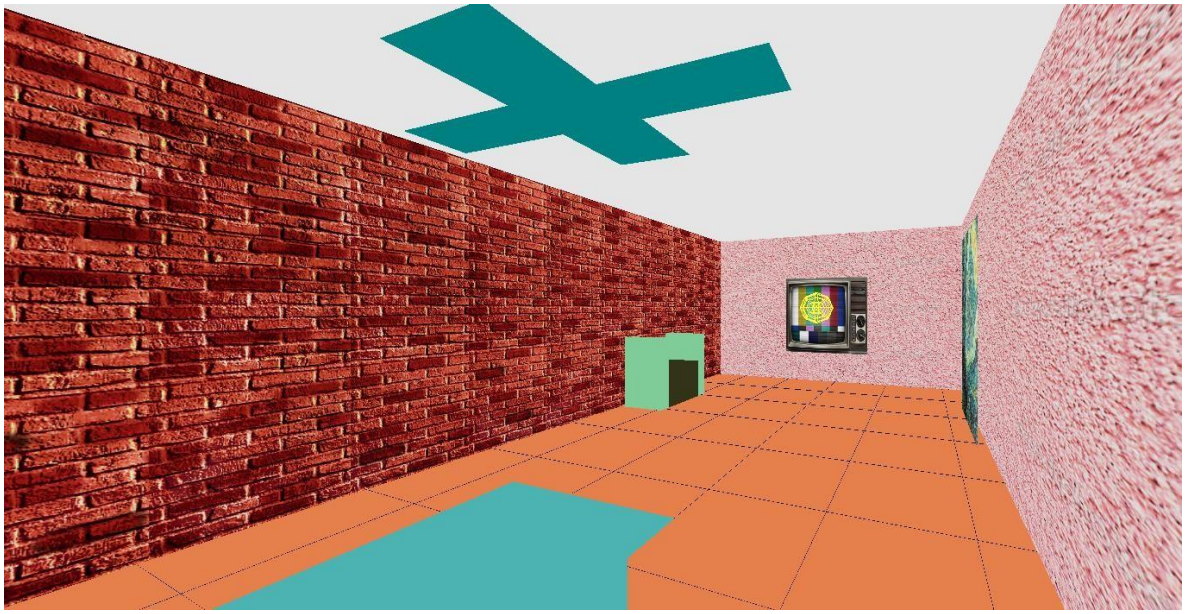
Anim_tele controla la animación que se presenta en la televisión de nuestro escenario, para que nuestro objeto no salga del área donde debe estar se reinicia a su posición inicial.

“glutWireSphere” es para crear nuestra esfera y le pasamos como parámetro su tamaño.

```
void Anim_tele(){
    glTranslatef(tras, 8.0f, -68.0f);
    glRotatef(angulo, 0.0f, 1.0f, 0.0f);
    glColor3f(8.0f, 8.0f, 0.2f);
    glutWireSphere(1.0f, 8, 8);
    if (tras>92.7)
    tras=91.3;
    tras=tras+0.001;
}
```

Para poder tener un ventilador rotando, los cuales se dibujan con polígonos se tiene su propia funcion que hace que giren en torno al centro de las figuras.

```
void pintarVentilador (float x, float y, float z)
{
    glTranslatef(x, y, z);
    glRotatef(angulo, 0.0f, 1.0f, 0.0f);
    glColor3f(0.0f, 0.5f, 0.5f);
    glBegin(GL_POLYGON);
        glVertex3i(-4,0,1);
        glVertex3i(4,0,1);
        glVertex3i(4,0,-1);
        glVertex3i(-4,0,-1);
    glEnd();
    glBegin(GL_POLYGON);
        glVertex3i(1,0,-4);
        glVertex3i(1,0,4);
        glVertex3i(-1,0,4);
        glVertex3i(-1,0,-4);
    glEnd();
}
```



Algunos de los objetos que tenemos en el escenario tienen movimiento por lo que se tienen que actualizar los datos que ayudan a controlar su rotación o traslación. Tambien aquí se manda a llamar a todas las funciones que dibujan nuestras texturas u objetos. Se hacen muchas funciones para tener mejor control.

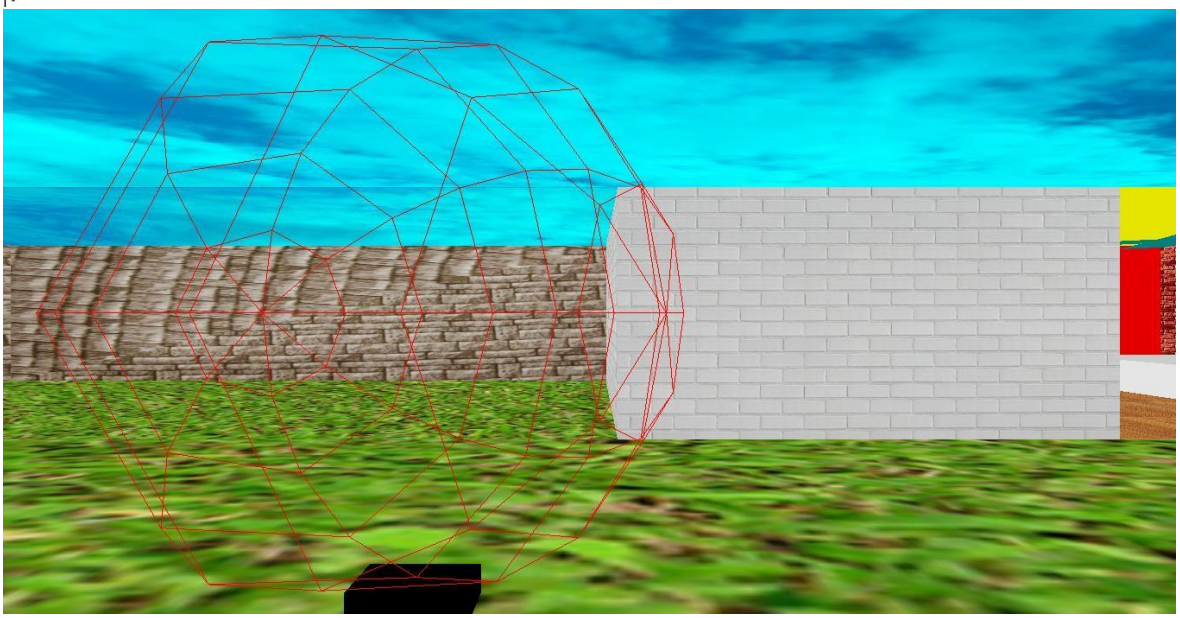
```
void display()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();


    if (!lookAt){
        glRotatef(anguloCamaraY, 0.0f, 1.0f, 0.0f);
        glRotatef(anguloCamaraX,1.0f,0.0f,0.0f);

        glTranslatef(0.0f, -8.0f, posObjeto); // zoom
        glTranslatef(pos2Objeto, 0.0f, 0.0f); // traslacion en eje x

    } else {
        gluLookAt(-5,10,5,0,0,0,1,12,4);
    }

    glPushMatrix();
    pintarCoordenadas();
    glPopMatrix();

    glPushMatrix();
    pintarContorno();
    glPopMatrix();

    glPushMatrix();
    pintarVentilador(90,12,-5);
    glPopMatrix();
```

Keyboard es una funcion que con un “switch” en caso de que se presione una tecla detectara cual fue y en base a eso realizara una acción para después llamar a la funcion “display” con lo que se actualizaran los datos con los nuevos que se hayan ingresado por el teclado.

```
void keyboard(unsigned char key, int x, int y){
    switch(key) {
    case '1':
        anguloCamaraY--;
        display();
        break;
    case '2':
        anguloCamaraY++;
        display();
        break;
    case '3':
        if(pos2Objeto<-2)
        pos2Objeto++;
        printf("en x vale: %f \n", pos2Objeto);
        display();
        break;
    case '4':
        if(pos2Objeto>-98)
        pos2Objeto--;
        printf("en x vale: %f \n", pos2Objeto);
        display();
        break;
    case '5':
        if(posObjeto<68)
        posObjeto++;
        printf("en zoom vale: %f \n", posObjeto);
        display();
        break;
```

En el escenario se tiene una esfera que simula una esfera de juegos, para dibujarla primero se traslada al punto donde estará situada, después se rota con base al ángulo que se encuentre en ese momento (se actualizara para que se dé la ilusión de movimiento), asignamos el color de nuestra esfera y se dibuja.

```
void Esfera(){
    glTranslatef(40.0f, 8.0f, -40.0f);
    glRotatef(angulo, 0.0f, 1.0f, 0.0f);
    glColor3f(1.0f, 0.0f, 0.0f);
    glutWireSphere(4.5f, 8, 8);
}
```



Bibliografía

Graficos por computadora con OpenGL: Donald Baker, Tercera Edicion, Prentice Hall 2006.

BENÉMERITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

Proyectofinal: Escenario en 30 con elementos

Rotativos

Profesor

Dr. Iván Olmos Pineda

Ob jetivo: El proposito del proyecto final es la construccion de un escenario en 3D con las herramientas de OpenGl, el escenario debe contener elementos de traslacón y de rotacion así como el uso de texturas en los objetos, además se pueden usar materiales y aprovechar la fuente de iliminación para darle mas realse a los objetos.

Descripción del escenario: El escenario consiste de un casa situada en el campo, la cual est construida usando coordenadas en 3D, consiste de dos pisos, en un total de 4 cuartos, las ventanas y las puertas se abren y en su interior, para crear este efecto se usó la instruccion glrotatef. Se colocó una textura en las paredes exteriores de la casa usando la biblioteca glaux. En el interior de la casa se colocaron muebles los cuales se crearon a través de diferentes funciones, las cuales rotan cubos en 3d de manera que los muebles tienen volumen y se pueden apreciar desde diferentes ángulos. Los muebles y la casa tienen diferentes colores y diferentes materiales, se usaron las propiedades de la luz parapoder observarlos desde diferentes perspectivas y que no parecieran opacos, se uso la instruccion glNormal.

En el exterior de la casa se colocaron una serie de árboles los cuales se construyeron usando piramides en 3d para la parte superior y un cubo para la parte inferior,se texturizaron con imagenes de hojas. Para la parte del piso, se us un cubo el cual se texturiz para darle el efecto de pasto. Se coloc un sol en 3d rotando sobre el escenario y se colocaron dos automoviles los cuales siguen una trayectoria recta, al rededor del escenario.

Prara describir las trayectorias de los objetos se usaron varias funciones, la funcion que describe el movimiento en linea recta para la traslacion del sol y la de un automovil.

Funciones implementadas:

Se crearon 25 funciones las cuales permiten el trazo de los objetos. En la funciones que dibujan los muebles, se realiza el trazo del de cada mueble usando cubos definidos previamente en otra funcion, además se definieron los materiales, ambiente y color de los objetos.

En la funcion display1 se crea la representacion de una cabaña, la cual fue construida usando las funciones glCuads y glTriangles, para el efecto de perpectiva se rotaron algunas figuras para dar el efecto de profundidad, se usaron tres texturas, las cuales permiten crear el efecto piedra blanca en las paredes de la casa. Para el techo de la casa se usó textura tipo piso. Se

usaron las instrucciones glTraslate y glRotate y con el fin de no cambiar el punto de origen se usaron las instrucciones glPushMatrix y glPopMatrix.

Para el trazado del sol y con el fin de lograr darle volumen y una textura brillante se realizaron cambios en la funcion de inicializacion, las instruciones qusadas permiten definir un color para los tipos de luz del objeto y la posicion de la luz para darle el efecto mas brillante en la parte superior del sol y asì creal el efecto mas real de volumen.

Para la costrucción del autómovil se crearon varias funciones para crear las diferentes partes del coche las cuales usan distintas texturas. Se implement una función para el cuerpo del coche y se mapeo esta figura con una imagen de textura azul platinada, para dar el efecto final. Para las ventanas se us una textura tipo vidrio y para el dibujo de las llantas se usó la instruccion glSolidSphere para la figura de la llanta y glCUADS para los rayos de las llantas. La lantas del coche van rotando a una cierta velocidad cada vez que avanza el automivil, por lo cual hay una combinacion de dos movimientos, una traslacion a medida que avanza el coche y una rotación respecto al automivil en cada momento.

En la función display principal se hace el llamado de estas 25 funciones y con el fin de

crear efectos de movimiento diversos se modifican los parametros que describen los diversos estados del objeto, a traves de los parametros de angulos y de posicion donde se realizan incrementos constantes.

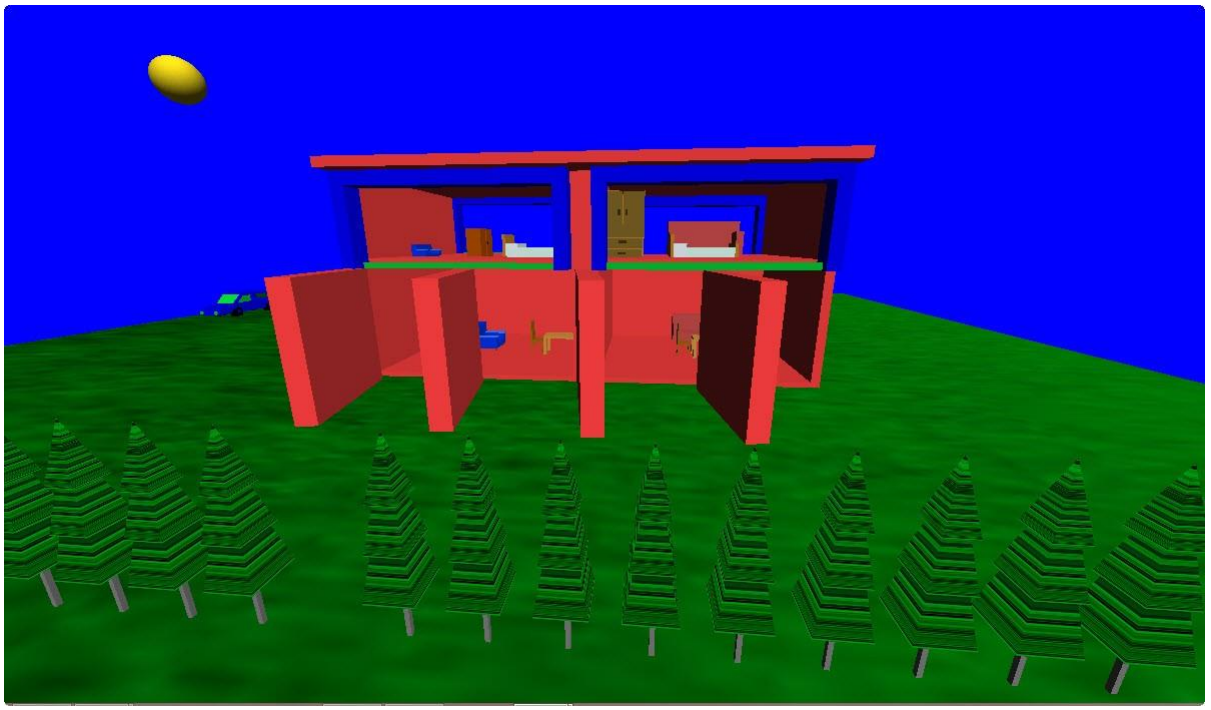
En la funcion Visualizacion se activa la matriz de proyeccion y se define el tipo de proyeccion a usar.Se carga en la matriz de proyeccion la matriz identidad, para resetearla y poder trabajar sobre ella, se redimenciona la ventana, los parámetros con los que se trabajan son ancho y alto. Se agregaron funciónes adicionales a través de la funcion que indica que al pulsar numeros y letras.

Para controlar otras teclas adicionales se implementa la función donde de indica que si se pulsa los numeros 1 y 5 abren las ventanas de la casa, las teclas de direcion nos sirven para rotar el escenario y visualizarlo en una vista completa de 360 grados. Ademas podemos ver el escenario desde la parte de arriba o una vista desde la superficie del pasto.

Finalmente se emple una textura en el pasto para dar el efecto de estar en el campo. Sobre

esta textura se superponen todos los objetos creados mencionados anteriormente para lograr el efecto final que se muestra acontinuacion.

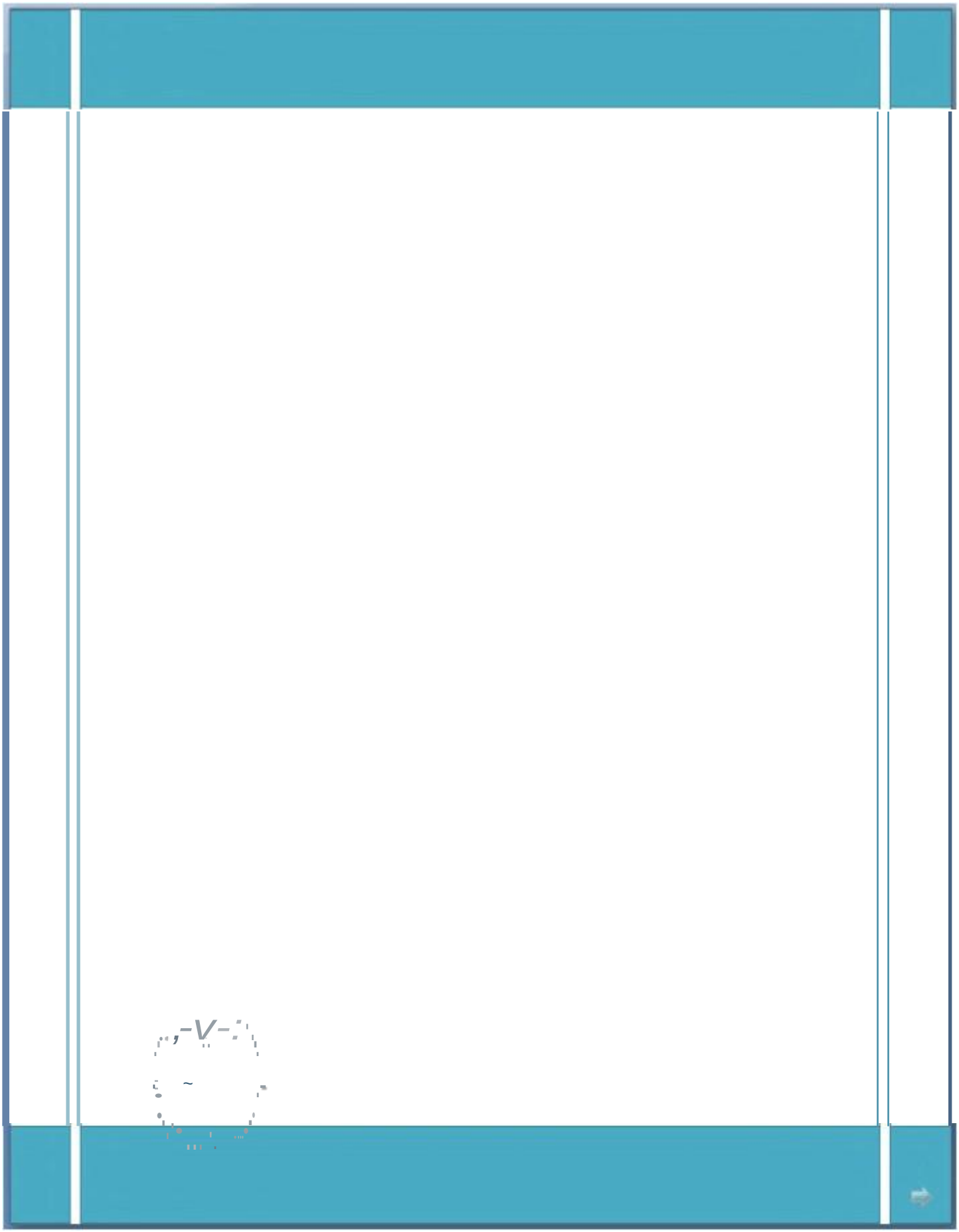
Resultado:



References

[1] Graficos por computadoras con OpenGL, Heam Baker, Prentice Hall, 2005.

3



BENÉMERITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

Examen 1: Programar una biblioteca de figuras geométricas en C++ y OpenGL.

Profesor:

Dr. Iván Olmos Pineda

Materia:
Graficación

Alumna:

Torres Juan Amalia (201319421)

Fecha: 20-02-2015

Objetivo: El proposito de este proyecto es crear una biblioteca de funciones para graficar las figuras geométricas basicas: círculo, triangulo, cuadrilatero y posteriormente crear un escenario usando estas funciones. Las clases de esta biblioteca deben ser programadas usando el lenguaje C++ y haciendo uso únicamente de la herramienta GL-POINTS de la biblioteca gráfica OpenGL. Todas estas clases deben aplicar las operaciones de traslacion, rotacion, escalamiento y deformación a sus objetos.

Conceptos matemáticos utilizados:

En este trabajo se programaron varios tipos de métodos que nos permiten realizar transformaciones de las figuras en el plano a continuacion se da una breve descripción de estas funciones.

Sabemos que una transformacion que crea una imagen que es congruente con la figura original se llama una transformacion rígida, o isometría. En este trabajo se programaron métodos para los tres tipos de transformaciones rígidas: traslacion, rotacion, y reflexión.

Una traslacion a través de un vector $\vec{p} =< x_0, y_0 >$ mueve todos los puntos de una figura sumandoles a través del vector (x_0, y_0), matematicamente esta transformación es una funcion lineal $\vec{p} : R^2 \rightarrow R^2$ que realiza una suma vectorial en cada punto $p = (x, y) \rightarrow R^2$ de la figura original y lo convierte en un punto $p^0 = (x, y) + (x_0, y_0)$. Usando las propiedades de las bases en el espacio vectorial R^3 se puede demostrar que una Transformacion de traslacion en el plano R^2 a través del vector (x_0, y_0) tiene una representacion matricial por medio de la matriz de orden 3×3, MT, definida por:

$$\begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Una transformacion de rotacion permite mover una figura un número determinado de grados respecto al origen. Formalmente una rotacion (de θ grados) es una funcion lineal $R(\theta) : R^2 \rightarrow R^2$ tal que a cada punto (x, y) lo transforma en el punto $(x^0, y^0) = (x\cos(\theta) - y\text{sen}(\theta),\ x\text{sen}(\theta) + y\cos(\theta))$. Esta transformación tiene la siguiente representacion matricial con matrices de orden 3×3:

$$\begin{bmatrix} \cos(\theta) & \text{sen}(\theta) & 0 \\ \text{sen}\theta & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La transformacion de reflexion T en el plano permite mover una figura de un punto a otro usando un eje de simétria, existen 4 tipos de reflexiones dependiendo de el eje que tomemos como referencia, por ejemplo si queremos reflejar una figura respecto al eje x, la reflexion respecto al este eje, T_x, envía el punto (x, y) en el punto $(x, -y)$. Similarmete las reflexiones en el eje y y en el eje XY T_y y T_{xy} se definen por $T_y(x, y) = (-x, y)$. y $T_{xy}(x, y) = (-x, -y)$.

Estas transformaciones del plano tiene una representación matricial a través de las matrices siguientes:

$$M_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad M_y = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad M_{xy} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Dado que en estas representaciones a través de matrices se toma como referencia el origen de coordenadas, si queremos realizar algunas de estas operaciones debemos trasladar la figura al origen y posteriormente realizar las operaciones matriciales. Cuando se realizan varias operaciones a la vez basta con calcular las matrices de estas operaciones y realizar su producto de matrices, para posteriormente operar esta matriz resultante con cada uno en los puntos de las figuras. No es necesario aplicar las transformaciones en cada uno de los puntos de una figura dada, pues dependiendo de su forma podemos considerar varios casos. Si la figura es un triangulo necesitamos aplicarle las transformaciones a cada uno de sus vértices, si es un cuadrado, solo es necesario aplicarle las operaciones a los puntos de su diagonal, y si es un recta basta con conocer a plicarselo a un punto.

Otra operacion considerada en este trabajo es la operación de escalamiento, la cual reduce o aumenta el tamaño de la figura, esta deformacion puede ser igual o diferente en ambas dimensiones, y no cambia sustancialmente la deforma de la figura. Esta transformación tiene la siguiete representación matricial ME:

$$\begin{bmatrix} r1 & 1 & 0 \\ 1 & r2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

donde r1 es la magnitud de escalamiento en la coordenada x y r2 es la magnitud de escalamiento de la coordenada y de un punto p = (x, y).

Por último, consideramos las operaciones de deformación que nos permiten cambiar la forma de una figura dependiendo del tipo factor de deformacion, existen tres tipos de estas transformaciones, deformación en x, en y y respecto a una linea de referencia. Estas transformaciones lineales se pueden representar usando matrices cuadradas de dimension 3 de la siguiete forma:

$$\begin{bmatrix} 1 & shx & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

El estudio de estas transformaciones a través de sus representaciones matriciales nos facilita el trabajo de programacion de dichas funciones, pues supone calcular estas matrices, y operarlas a través de la multiplicacion usual de matrices y finalmente aplicarlas a cada uno de los puntos que queremos tomar como base para la transformacion.

Descripción del proyecto:

Para realizar la implementacion de los métodos que realizan las transformaciones de rotación, escala, traslacion, deformacion y reflexion se usó la representación matricial de estas funciones lineales mencionadas en el apartado anterior.

Se programaron 6 clases, la clase punto, la clase línea, la clase triangulo, la clase matrices, la clase cuadrilátero y la clase círculo. Cada una de las 6 clases programadas consta de dos archivos, un archivo .cpp y un archivo .h.

El archivo de extension .h contiene las declaraciones de constantes, variables y funciones de las que consta la clase, y el archivo .cpp implementa el codigo para las funciones declaradas en el archivo de encabezado. En estas clases solo contienen los constructores y métodos basicos para establecer y obtener valores y un método para mandar a dibujar la figura en pantalla.

Cada una de estas clases se program aprovechando las ventajas de la programación orientada a objetos,

primero se crea una clase Punto, de ella heredan sus propiedades y métodos, la clase linea y la clase circulo;

las clases triangulo y cuadrilatero heredan de la clase línea.

En clase matrices se definen las operaciones y matrices que fueron usadas en cada una de las clases. Con el fin de realizar una programacion mas eficiente,todas las clases heredan los métodos de la clase matrices.

En el caso de la clase línea, se program un método, llamado dibujarlinea, el cual contiene la implementacion

en lenguaje C del método por incrementos para trazado de líneas, este método es parte de la tarea2 del curso, en esta funcion solo usamos la instruccion GL-POINTS de la biblioteca OpenGL.

Las clases triangulo y cuadrilatero usan la funcion dibuja linea para para construir los lados de sus objetos. Asimismo, usan los métodos de la clase Matrices para crear las matrices de rotación, traslación, deformación, etc. y operar estas matrices, finamlmente teniendo calculada la matriz de transformación final se las aplican

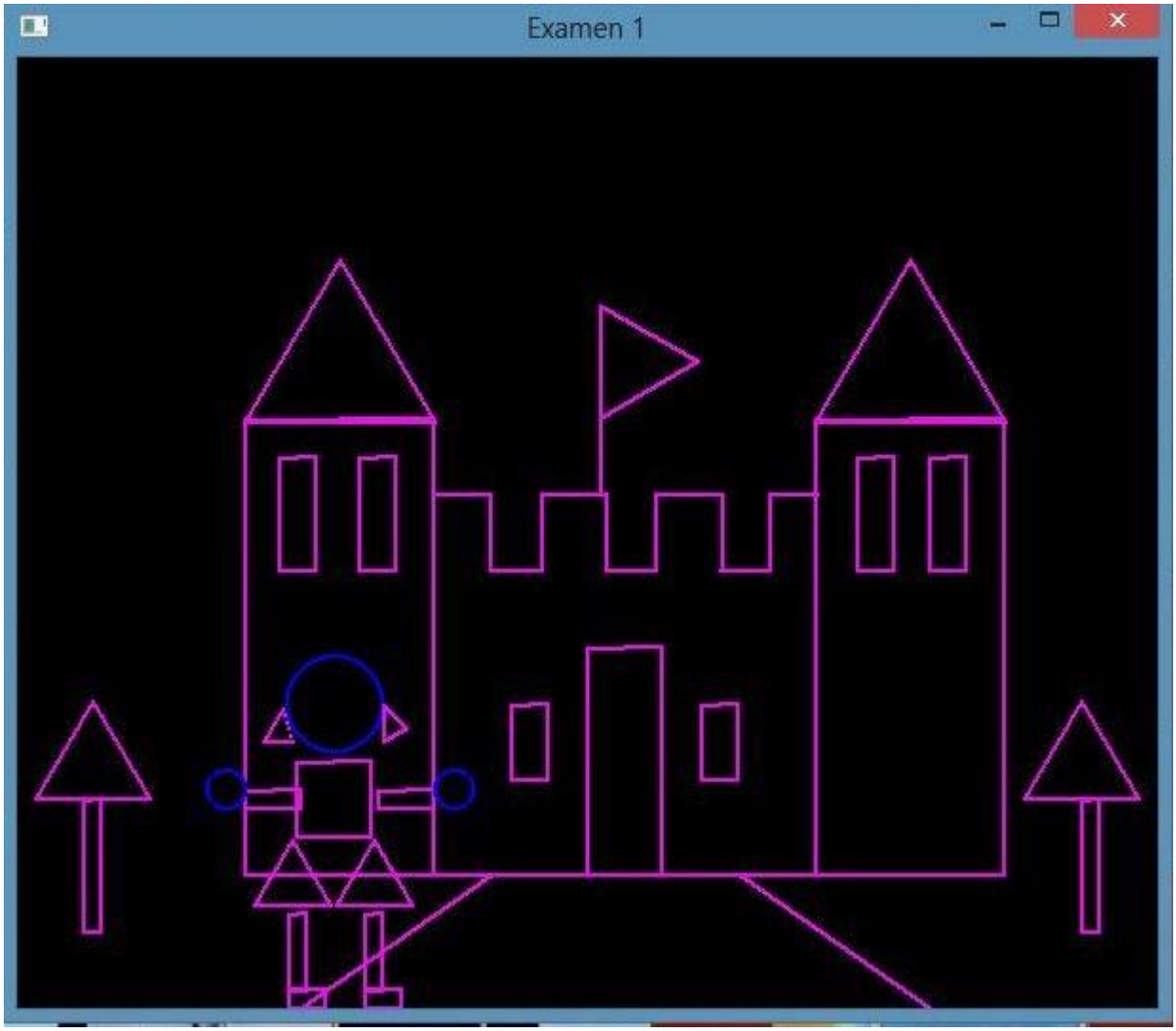
a los puntos de figuras unitarias, dibujadas en el origen de la pantalla, y finalmente la funcion dibuja de estas clases realiza una traslacion en el punto de la pantalla donde debe ser colocada y la dibuja en esa posición.

La clase círculo usa elementos de tipo punto para definir las coordenadas de su centro, el método dibujaCirculo de esta clase realiza la grafica de un círculo usando el médodo por incrementos, el cual aprovecha la simetría de la figura para calcular solo los puntos de la circunferencia del primer cuadrante y luego usando reflexiones, calcula los puntos de los cuadrantes restantes.

Para la realizacion del escenario final, se cre una serie de figuras usando la biblioteca de figuras geométricas

cradas en este proyecto, para crear una figura dada, en una cierta posición de la pantalla, se instancia el tipo de objeto a dibujar y se llaman a cada uno de sus métodos que se desean operar, ejemplo rotación, traslacion,escalamiento,las cuales preparan las matrices correspondientes y finalmente se invoca a la funcion draw(x, y) la cual opera estas matrices y la aplica a los puntos de la figura y manda la figura en la pantalla en la posicion (x, y).

Resultados:



Conclusiones: Al realizar esta proyecto aprendí a codificar en lenguaje C++ pues aunque ya tenía experien- cia programando en el paradigma orientado a objetos, sólo había programado en java. En cuanto a los concep- tos matematicos empleados para la programación de los métodos de transformaciones de rotación,escalamiento, traslacion, etc., recordé conceptos como representación matricial de una transformación lineal respecto a una base.

Me resulta interesante aprender a programar usando la herramienta openGL pues con ella se pueden realizar graficos con relativa facilidad los cuales con otros lenguajes de programación resultarían más complicados de hacer.

References

[1] Serge Lang, Introduccion al algebra Lineal, 3da. Ed. Addison-Wesley Iberoamericana. 1971.

Proyecto Final

Objetivo

En este proyecto final se busca implementar las funciones de translación, rotación y escalamiento esto con el propósito de darle animación a un escenario 3D el cual con el uso de las funciones que trae opengl por defecto, las figuras sean capaces de moverse de un punto a otro, rotarse respecto a un eje de rotación y crecer respecto a algún, escalar además de implementar texturas en este mismo.

En este proyecto se hará un escenario el cual debe de llevar las 3 operaciones básicas antes mencionadas, este escenario se trabajara en el plano 3D (X, Y, Z).

Se estará trabajando como escenario interactivo este consiste en una vista de un cuarto con una ventana y dos puertas, la ventana tiene una vista de un paisaje, este cuarto contendrá: una cama, una mesa, un buro, diversos cuadros con imágenes, una bocina, puertas, retratos.

Se podrá salir del cuarto al patio en el cual se podrá observar el paisaje y el cielo.

En el patio habrá texturas referentes al patio de cualquier casa también habrá objetos los cuales se moverán al darle una orden con el teclado.

Antecedentes

Se necesita saber cómo aplicar las operaciones translación, rotación y escalamiento estas las hemos visto en clase.

También se necesita conocer las primitivas de opengl para dibujar nuestro escenario y saber cómo texturizar cada figura.

Como se implemento

Se necesita saber y no solo eso sino comprender como es la estructura de nuestro ciclo infinito para poder limpiar pantalla sin afectar a las demás figuras

Primero dibujo mi escenario para esto utilizo estas funciones las cuales son absolutas:

paredes();

ventana();

piso();

mesa();

cuadros();

buro();

retratos();

cama();

suelo();

puertas();

Estos dibujos no se mueven están dados los puntos en 3D sin que se les realice ningún tipo de operación.

En las funciones:

pelota(z);

esta función dibuja una pelota con ayuda de la función para dibujar esferas se hace una translación de esta figura (pelota) al ordenarle con la tecla numérica “3” con la que se mueve simulando que se ha pateado la pelota estas operaciones se aplican con las funciones predeterminadas de opengl.

juego(z);

Esta función dibuja un juego de niños y lo texturiza, el cual está rotando el dibujo es absoluto y en la función de dibujo se le aplican las transformaciones para que gire.

bocina(k);

Esta función dibuja una bocina (un cubo) y lo texturiza, este está escalándose

(creciendo y decreciendo)

Cajones ();

Esta función dibuja los cajones del buro y los texturiza, con ayuda del teclado se le puede ordenar que cajón se quiere abrir.

Puertas();

Esta función dibuja las puertas y las texturiza, en la función de dibujo se verifica la posición del lookAt() para que así en alguna posición determinada las puertas se abran o se cierre.

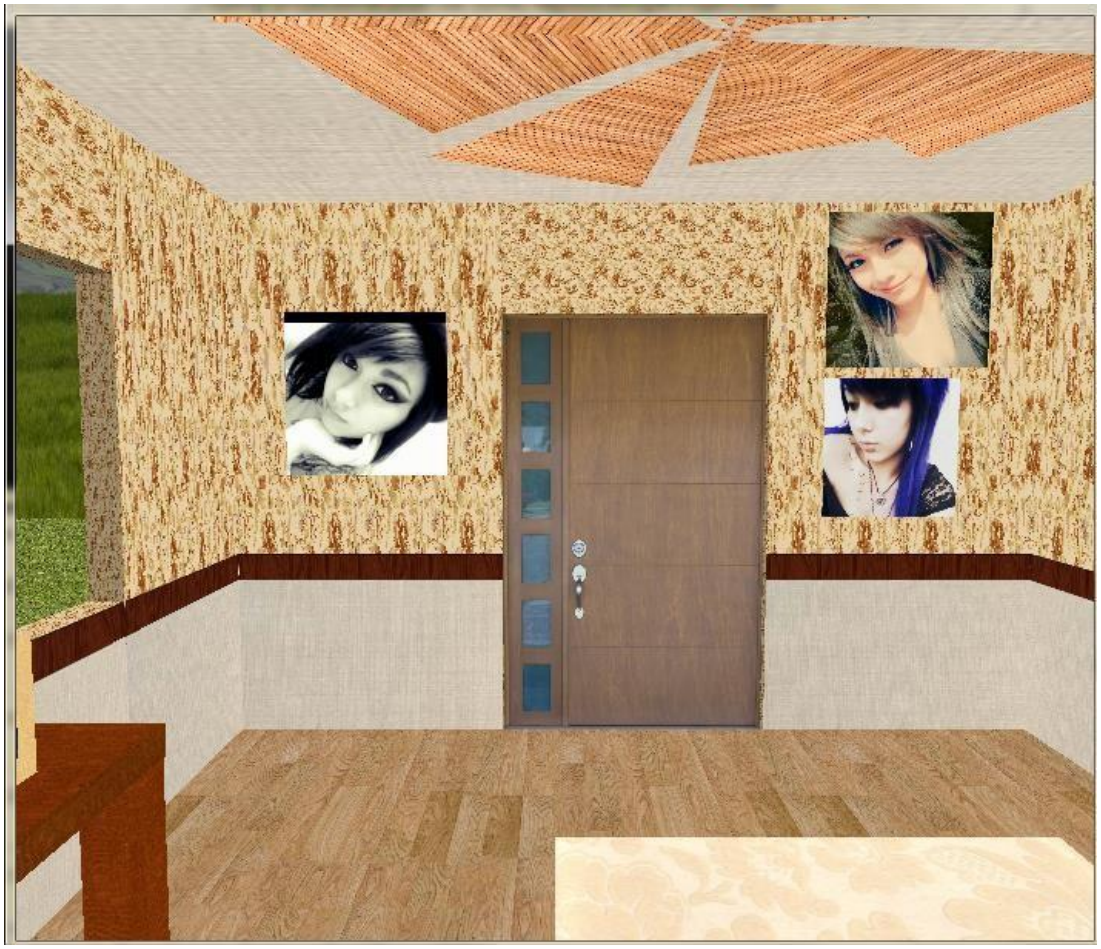
Keyboard()

He implementado esta función que controla la posición del observador mediante el uso del teclado.

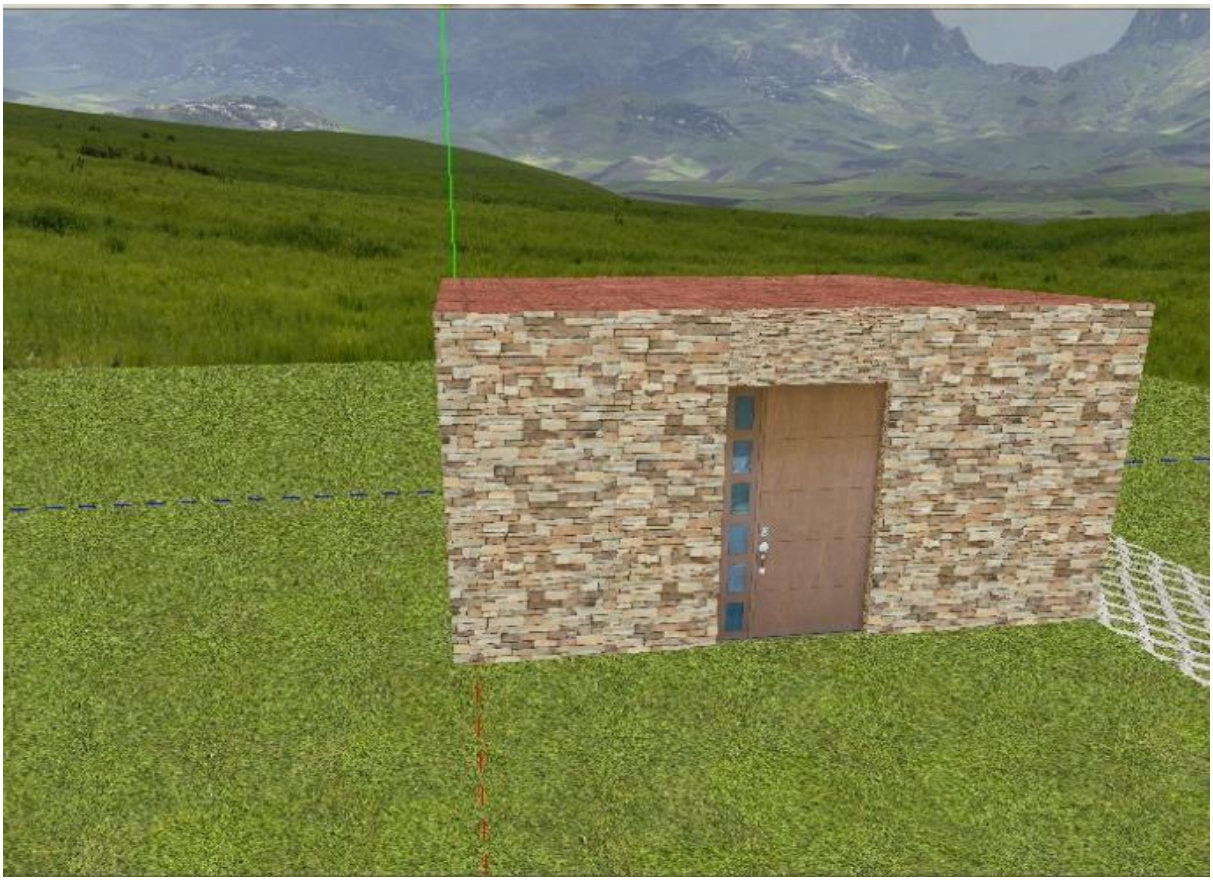
readImage()

en eta función se hace la lectura de cada imagen o textura, estas se encuentran nombradas en un arreglo para que asi la funcion pueda ir llamando a cada una de las texturas.

Resultados




te soñé.



BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de computación

Documento técnico:

“Escenario con texturas”

Materia: Graficación

Nombre del alumno: Ruiz Flores Jorge Mauricio

Nombre del profesor: Olmos Pineda Iván

Introducción

El objetivo del proyecto fue retomar todos los temas abarcados hasta el momento en el curso, aplicando el uso de texturas para dar mayor realismo al escenario.

Conceptos:

Traslación

Para realizar la translación de los personajes del escenario se uso el vector de coordenadas (variables) donde se generan las figuras, que a través de la lectura de un carácter en el teclado se modifica su posición donde se genera el polígono.

Texturización

Es la operación que consisten en colocar imágenes en un polígono tratando de dar un mayor realismo a la figura, para cargar texturas en OpenGL es necesario hacer uso de bibliotecas gráficas, además cabe destacar que en la creación del escenario no se uso un mapa de 24 bits, sino que se uso un canal extra que es el canal alfa para poder dar transparencia a la textura, por ello se usan 32 bits.

Se uso la biblioteca gráfica Corona para poder cargar las texturas en memoria.

Consideraciones sobre corona:

- Corona carga imágenes a 32 bits directamente, por lo que no es necesario agregar funciones extra
- Las texturas se cargan al revés por lo que las imágenes van a salir invertidas con respecto al eje Y, pero no es un gran problema ya que se puede invertir nuevamente la textura para que tome su forma original.
- Hay que recordar que para que no tome color distinto la textura se debe de aplicar un tinte blanco con glColor.

- Es necesario recordar que la forma en que se crean los polígonos afecta en cómo se muestra la textura.

Pseudocódigo (Quad's)

```
dibujar ()

{

    ActivarTexturas(Textura2D);

    SeleccionarTextura (Textura2D,textura);      //textura es el identificador

                                                 //de la misma
```

```
        Tintado(1,1,1);                    //Colocamos un tintado blanco
        dibujaCuadrado();
            CoordenadaTextura (0.0f, 0.0f);   //(0,0) con respecto a la textura
            vertice(X,Y);                     //creamos el vértice superior izquierdo
            CoordenadaTextura(1.0f, 0.0f);    //(1,0) con respecto a la textura
            vertice(X',Y);                    //creamos el vértice superior derecho
            CoordenadaTextura(1.0f,1.0f);     //(1,1) con respecto a la textura
            vertice(X',Y');                   //creamos el vértice inferior derecho
            CoordenadaTextura(0.0f,1.0f);     //(0,1) con respecto a la textura
            vertice(X,Y');                    //creamos el vértice inferior izquierdo
        FindibujaCuadrado();


    DesactivarTexturas(Textura2D);

}
```

Nota: Se puede observar que los polígonos se crean con respecto al movimiento de las manecillas del reloj, pero también es fácil de ver que la textura no se aplica en el mismo sentido, sino que se está aplicando a la inversa puesto que las texturas se cargan al revés.

Escenario de ejemplo

Dibujado del escenario

Se puede observar en la imagen anterior que la mayoría de los polígonos que son necesarios son cuadriláteros, por ello la primitiva empleada en totalidad fue GL_QUADS, para deformar los polígonos se hizo uso de los vértices que los conforman.

La mayoría de las texturas que se usaron fueron imágenes en formato .png puesto que permiten guardar el canal alfa para transparencia, aun que algunas de las imágenes que se usaron tienen formato .bmp o de mapa de bits.

Recordando que las texturas se cargan al revés en memoria, se invirtió la coordenada Y de los vértices de la textura para que al aplicarla tomara su forma original, también se hizo uso de la característica de OpenGL para duplicar la textura y reflejarla con respecto a los ejes.

Para poder hacer que los personajes se trasladaran sólo se modifico el vector de coordenadas del Quad, ya que al presionar algunas teclas se modifican las coordenadas y la textura que se aplica al polígono, con ello se buscó crear una animación por transición de texturas.

Finalmente para mostrar los cambios entre texturas se hizo uso de la función Sleep, con ello se buscó alentar un poco el cambio de texturas y la forma en que se muestra en pantalla.

Conclusión

El proyecto me permitió implementar ideas que tenía sobre el uso de texturas en polígonos para crear animaciones, aun que inicialmente me fue muy complicado entender cómo se manejaban las texturas con las bibliotecas graficas, luego de buscar y leer algunas documentaciones y textos en la red sobre OpenGL, FreeImage, Soil y Devil logre encontrar una librería gráfica muy fácil de usar (Corona) aun que un gran problema que posee la librería es que se dejo de actualizar desde el año 2003.

Además, el proyecto me permitió reafirmar conocimientos sobre el manejo de primitivas de OpenGL y de vectores, aun que no llegue a utilizar la función glRotate ya que no pude texturizar una esfera para poder crear un sol.

Finalmente, pienso que el proyecto fue bastante útil pues permitió interactuar con las funciones que nos provee OpenGl, además de plantearnos la manera de crear un entorno visualmente agradable e interactivo.

Introducción

La práctica para esta ocasión consistía en hacer un pequeño escenario usando transformaciones y solo líneas, todo esto en 2D. La complicación estaba en que, con objetivo de reafirmar los conocimientos teóricos, teníamos que realizar las transformaciones y los dibujos "a mano", es decir, realizar las transformaciones con multiplicaciones de matrices y el dibujado solo con líneas.

Librería Figuras.h

Para hacer la programación un poco más transparente incluí una librería "**Figuras.h**" que, emulando las funciones de OpenGL, contiene lo necesario para llevar a cabo las transformaciones y tratar las figuras como polígonos mientras se mantenía la regla de "*solo líneas*".

Transformaciones

La librería cuenta con una **Matriz de *double's*** llamada **Matrix** que sirve para multiplicar a todos los puntos por dibujarse.

Además de esto están las cuatro funciones que manejan a esta matriz:

- **resetMatrix();** Sin retorno. Esta función escribe en el arreglo **Matrix** una matriz identidad de 3x3.

- **Trasladar(double tx, double ty);** Sin retorno. Esta función hace las veces de *glTranslatef.* Recibe dos números reales y los asignas en sus respectivas posiciones en una matriz identidad de nombre *tmp* multiplica la matriz *tmp* por la matriz **Matrix** y guarda el resultado en *Result*. Finalmente se vacía el contenido de *Result* en **Matrix.**

- **Rotar(double theta);** Sin retorno. Esta función hace las veces de *glRotatef.* Recibe un número real y calcula números necesarios usando senos y cosenos y los asignas en sus respectivas posiciones en una matriz identidad de nombre *tmp* multiplica la matriz *tmp* por la matriz **Matrix** y guarda el resultado en *Result*. Finalmente se vacía el contenido de *Result* en **Matrix.**

- **Escalar(double sx, double sy);** Sin retorno. Esta función hace las veces de *glScalef.*

  Recibe dos números reales y los asignas en sus respectivas posiciones en una matriz identidad de nombre *tmp* multiplica la matriz *tmp* por la matriz **Matrix** y guarda el resultado en *Result*. Finalmente se vacía el contenido de *Result* en **Matrix.**

Manejo de dibujos

Para el manejo de dibujos decidí tratar a cada uno de los mismos como listas ligadas (FiFo, cola) de puntos para manejar de manera más dinámica no solo el dibujado de los mismos sino también la delimitación de puntos.

Para este aspecto de la biblioteca necesite hacer uso de una estructura llamada Nodo que cuenta con dos números flotantes (x, y) y un apuntador de tipo Nodo. En una línea posterior redefino esta estructura como ***Tlista**

Para este aspecto solo incluí una función puesto que era la única que necesitaba:

- **insertar(Tlista &lista, float x, float y);** Sin retorno. Esta función es el típico *push* de cualquier lista ligada. Inserta un nuevo nodo al final de la lista creando un nodo *q* con los datos recibidos y ocupa otro nodo *t* (que empieza en la misma dirección que *lista*) para recorrer el arreglo hasta el último elemento, ahí, en la variable apuntador de ese nodo inserta el nodo *q*.

Dibujado

Una vez declaradas las figuras como listas ligadas el dibujado se hizo más fácil al solo requerir que se recorriera el arreglo ejecutando un glVertex por cada nodo mientras estos valores se multiplicaban por el arreglo Matrix.

Finalmente, en la biblioteca, para el dibujado ocupe la siguiente función:

- **dibujar(Tlista lista, int dibujado);** Sin retorno. La primera variable que recibe es la dirección del primer nodo de la lista mientras que la segunda define qué tipo de dibujado va a hacer: *Polígono* o *Línea* (la única diferencia es que con los polígonos se dibuja una línea que va desde el último nodo hasta el primero y con las líneas no). Con un *while* se crea el primer ciclo que termina hasta que la lista se vacía, se abre un *glBegin*, se inicia un nuevo ciclo while que termina cuando la lista se termina o cuando el nodo contiene (0, 0) en (x, y) respectivamente (esto me permite dibujar varias líneas o polígonos dentro de un mismo dibujo), dentro del ciclo se toman *x*, *y* del nodo y se multiplican por los valores de *Matrix* para entonces ser dibujado con un *glVertex*, se ejecuta el *glEnd* al terminar el segundo ciclo y por último se cierra el primer ciclo.

Main

Es parte es la simple. El archivo main contiene las siguientes funciones:

- **reshape(int width, int height);** Sin retorno. La función reshape estándar.

- **init();** Sin retorno. La función crea un fondo negro con glClearColor para contrastar (sin usar un glColor en el programa las líneas siempre salen blancas, no sé por qué). Y se 'dibujan' (se insertan los puntos) todos las figuras que se van a ocupar.

- **display();** Sin retorno. Vale hacer énfasis en la variable i que a diferencia de tiempo lleva un tempo delimitado que va de -10 a 10 para la animación del brazo que saluda. A partir de ahí se usan la funciones de la librería figura para crear varias instancias de los dibujos creados en init() y animarlos.

- **idle();** Sin retorno. Se incrementa la variable tiempo, se da un Sleep de 33 y se re invoca la función display().

- **main(int argc, char **argv);** Retorna un entero, 0 en caso de éxito. La función main estándar.

Bibliografía

Gil T., M. A. (8 de Diciembre de 2004). *Apuntadores en C y C++.* Obtenido de Las TIC en la educación: https://utch2009.files.wordpress.com/2010/04/apuntadores.pdf

lami20j. (Junio de 2015). *La lista enlazada simple*. Obtenido de Kioskea.net:

http://es.kioskea.net/faq/2842-la-lista-enlazada-simple

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de computación

Documento técnico:

“Escenario 3D”

Materia: Graficación

Nombre del alumno: Ruiz Flores Jorge Mauricio

Nombre del profesor: Olmos Pineda Iván

Introducción

El objetivo del proyecto fue aplicar todos los conocimientos adquiridos en el curso, plasmándolos en un entorno tridimensional e intentando generar un escenario creativo donde se hiciera uso de la iluminación, manejo de cámara y texturas.

Conceptos:

Traslación

Operación que consta en mover de lugar un polígono modificando las coordenadas de los puntos que lo forman; se hizo uso de la translación para colocar los modelos 3D en el escenario.

Rotación

Operación que consiste en girar un polígono respecto a un eje de coordenadas.

Escalamiento

Operación que consiste en hacer más grande un polígono u objeto del escenario a través de la multiplicación de cada uno de los puntos que lo conforman por un escalar.

Texturización

Es la operación que consisten en colocar imágenes en un polígono tratando de dar un mayor realismo a la figura o al escenario; se hizo uso de texturas para generar el cielo del escenario así como para darle mayor realismo a los personajes y a las construcciones.

Modelo 3D

Figura creada a través de polígonos (primitivas) generalmente triángulos y cuadriláteros; se hizo uso de una librería externa para cargar modelos tridimensionales.

Pseudocódigo (escenario)

//Nota: solo se colocara un ejemplo de cómo se dibujan los modelos en el escenario y como está estructurado el proyecto.

```
dibujar ()

{

    PosLuz [] = {X, Y, Z, Tipo de luz};
    LuzAmbiental[]  =  {R,G,B,A};
    LuzEspecular[]  = {R,G,B,A};
    LuzDifusa[] = {R,G,B,A};
    DibujarCielo();

    ConfigurarLuz(PosLuz,LuzAmbiental,LuzEspecular,LuzDifusa);
    ActivarIluminación(Luz a activar);

    EmpujarMatriz();                //Colocación de modelo precargado
        Traslacion(X, Y, Z);        //Se coloca el modelo en su posicion
        Escalamiento(X, Y, Z);      //Se escala el modelo
        ActivarTextura();           //Se activan las texturas
        AplicarTextura(Textura);

            DibujarModelo(Modelo);

        DesactivarTextura();

    SacarMatriz();
```

```
        DesactivarLuz(Luz a desactivar);
        Regresar;
    }
```

Dibujado del escenario

El contexto del escenario es una ciudad futurista, para hacer el cielo se hizo uso de la primitiva GL_QUADS para generar un cubo y se cargo la textura del mismo (skybox), para dar un efecto de movimiento del cielo y del agua se uso la rotación.

Los modelos utilizados para el escenario fueron una ciudad y cuatro robots, dos de los robots giran alrededor de la ciudad debido a que se les aplico una traslación y rotación.

Para que el usuario pudiera hacer uso de la cámara se agregaron teclas que permiten modificar la posición de la cámara en el plano así como para modificar el punto hacia donde observa la cámara.

Finalmente para dar mayor realismo al escenario se colocó una luz posicional (puntual), además se combinaron varios tipos de luz y se modificó su intensidad con respecto al canal RGBA.

Conclusión

El proyecto me permitió implementar los conocimientos adquiridos durante todo el curso y plantearme como generar un escenario interesante y atractivo para el usuario.

Me permitió reflexionar sobre cómo resolver el problema de generar figuras (mallas) complejas y como texturizar los polígonos que las conforman.

Pienso que es un poco complicado configurar los controles para manejar la cámara debido a que no solo se debe de modificar la posición de la cámara sino también se debe de modificar la dirección hacia donde ve la cámara.

Finalmente creo que el manejo de luces es bastante interesante ya que le da algo de realismo al escenario pero puede ser un poco difícil de utilizar puesto que existen distintos tipos de iluminación.

PROYECTO 3D GRAFICACIÓN

Méndez Barrios José Antonio

BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA

FACULTAD CIENCIAS DE LA COMPUTACIÓN

PROFESOR – IVAN OLMOS


BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA
1578

INTRODUCCIÓN

Hoy en día la animación 3D, es una de las técnicas más empleadas y con mayor auge en la industria cinematográfica y televisiva. En la animación 3D, los elementos, personajes y escenarios se construyen o modelan en 3 dimensiones o ejes. El ordenador y las diferentes herramientas (software)los más conocidos son "Autocad", Autodesk

3Ds Max y "Autodesk Maya que se utilizan permiten generar toda clase de formas aplicar todo tipo de características superficiales, efectos especiales. Permitiendo expresar ideas y conceptos de manera gráfica por medio de imágenes en movimiento.

Hoy en día se encuentra la animación 3D en la mayoría sino es que en todos de los comerciales, películas, programas de televisión, Pag Web etc.

Todos los que hacen animación 3D, deben de seguir las siguientes etapas las cuales pueden llegar a variar dependiendo del proyecto

A partir del momento en el cual se tiene un concepto o de una idea

- Desarrollo del guion
- Realización del storyborad
- Audio (locuciones y voces de los personajes)
- Modelado 3D (creación del personajes en 3D)
- Set up (es esqueleto que le permitirá que el personaje se mueva)
- Texturización
- Animación

- Efectos especiales
- Render
- Compost del vídeo. (Corrección de color y Edición)
- Musicalización y efectos de audio.
- Transfer al formato que se requiera.

Dentro de los productos que se pueden elaborar por medio de la animación 3D están:

- Producción de comerciales digitales.

- Animación 3D
  - Diseño de personaje
- Modelado y animación de personajes
  - Animación 3D con look 2D.
  - Back digitales
  - Cierres y Cortinillas
  - Integración de 3D a realidad
- Diseño gráfico 3D
  - Ilustración 3D
  - Modelado de producto
  - Modelado de objeto 3D
  - Modelado de personaje 3D
  - Ilustración 2D
- Videos animados 3D
  - Corporativos
  - Educativos
  - Institucionales
  - Cortometrajes
  - Largometrajes
  - Efectos Especiales 3D
- Animación 2D
  - Creación de Story boards

OBJETIVO DEL PROYECTO

El Objetivo del proyecto es modelar un escenario en 3D utilizando las técnicas y herramientas que aprendimos en clase, tales como la rotación, la traslación, iluminación, texturas entre otras.

Para crear las figuras debemos emplear un volumen definido para cada una de las figuras, manipular de igual forma su iluminación (brillo).

Agregar movimiento a la figura que hemos creado donde posteriormente la hemos ambientado con texturas.

ANALISIS DE LA ESTRUCTURA DE NUESTRO CÓDIGO

Declaramos librerias

```
#include <GL/glut.h>
#include <iostream>
```

Ajustamos el ángulo de proyección, es decir , como la distancia de la cámara

```
float Distancia=300.0;
```

Declaramos cada una de sus partes del robot para dibujarlas después llamarlas conforme a su nombre

*Primer cilindro de arriba

*Cilindro más grande del robot

*Ambos brazos del robot

*Ambas piernas del robot

```
GLuint cabeza;
GLuint cuerpoprincipal;
GLuint brazosdelrobot;
GLuint piernasdelrobot;
```

Creación o declaración de los brazos, para trabajar cada uno por separado

```
enum {brazoderecho, brazoisquierdo};
```

creación de ambas piernas pero con diferentes rotaciones y traslaciones en una misma operación

```
enum {musloderecho, piernaderecha, musloiquierdo, piernaizquierda};
```

Creación de las texturas

```
GLfloat mat_diffuse   [] = {1, 0, 1, 1};
```

Creación de la lámpara

```
GLfloat mat_specular  [] = {0.5, 0.5, 0.5, 1};
```

Valor del brillo que le pondremos a las figuras

```
GLfloat mat_shininess [] = {20};
```

Valores predeterminados a las lámparas

```
GLfloat light_position[] = {-9.9, -1.0, 3.0, 0.0};
```

creación y rotación del robot para que este de sus caminadas con la condición if de cada ángulo

```
GLfloat r=0,cx=0,cy=0,cz=0,pbi=0,pbd=0;
GLfloat varp=0,eyex=0,eyey=-5,eyez=25;
```

En esta se crea la escena de los colores de las figuras y le vamos cambiando los valores para que cambien de color

```
GLfloat cilindro     [4]={0,0,0,1};                    //color del cilindro
GLfloat triangulo    [4]={1,0,1,1};                   //color del triangulo
GLfloat cuadro       [4]={0,0,1,1};                     //color del cuadro
GLfloat cuadrofrente [4]={1.0,1.0,0.0,1};             //color del dodecahedro
```

Creación de color rojo todo el robot y sus partes de los brazos cuerpo principal cabeza y piernas

```
GLfloat colorrobott [4]={1,0,0,1};
GLfloat colorrobot  [4]={1,0,0,1};
```

En esta llevamos como un apuntador *robot que nos dirá e indicara como se debe trabajar las partes del robot

```
GLfloat rot=0;
GLUquadricObj *robot;
```

Posicionamiento o generar la pantalla conforme a los pixeles o es la pantalla de ejecución o la posición de pantalla en ejecución

```
GLfloat WinWidth=1000, WinHeight=690;
```

Creación de las texturas para después dar una aclaración y dar los colores que se declararon un poco antes de estas líneas.

```
void luces()
{
glLightfv    (GL_LIGHT0, GL_POSITION, light_position);
glMaterialfv (GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv (GL_FRONT, GL_SHININESS, mat_shininess);


    glEnable(GL_CULL_FACE);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_STENCIL);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_NORMALIZE);
}


GLfloat WinWidth=1000, WinHeight=690;
```

Creación de cono, cilindro y los dos cuadros que aparecen en la ejecución del programa

```
static void Esena()
{
    glPushMatrix();
        glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,cuadro);        //tiene a llamar las funciones de color para que haga texturas
        glTranslated(-30,-4, -60);                                   //posicioamiento x,y,z
        glScalef(5,5, 5);                                            //engrandece el cubo o la figura que estamos haiendo
        glutSolidCube(10);                                           //terminamos  por definido el cubo que estamos dibujando
    glPopMatrix();                                                   //fin de la creacion del cubo
```

```
glPushMatrix();
    glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,triangulo);
    glTranslated(-80,-28,20);                                   //posicioamiento x,y,z
    glRotatef(-90,1, 0, 0);
    glScalef(8, 5, 8);
    glutSolidCone(6, 15, 20,20);                                //fin de la creacion del cono
glPopMatrix();


glPushMatrix();
    glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,cuadrofrente);
     glTranslated(30,-17,90);                                   //posicioamiento x,y,z
    glScalef(5,5, 5);
    glutSolidCube(10);                                          //fin de la creacion del cubo de amarillo
glPopMatrix();

      glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,cilindro);
      glTranslated(90,28,-115);                                 //posicioamiento x,y,z
      glRotatef(90, 1, 0, 0);
      gluCylinder(robot, 30, 30, 60, 20, 2);                    //fin de la creacion del cilindro de color negro que esta asta el fo
      gluDisk(robot,0,20,20,2);
  glPopMatrix();
  glFlush();
  glutSwapBuffers();
  glPushMatrix();

glFlush();
glutSwapBuffers();
}
```

Aquí empezamos a construir el robot a partir de las variables que ya habíamos declarado anteriormente

```
void init(void)
{
    glClearColor(1.85,1.85,2,1);
    luces();
    robot= gluNewQuadric();
    gluQuadricDrawStyle(robot,GLU_FILL);
    gluQuadricNormals(robot,GLU_FLAT);

    cabeza=glGenLists(1);
        glNewList(cabeza,GL_COMPILE);
        glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,colorrobott);
        glPushMatrix();
            glRotatef(-90,1,0,0);
            gluCylinder(robot,5.5,5.5,10,50,5);
        glPopMatrix();
    glEndList();
    glFlush();



    cuerpoprincipal=glGenLists(1);
        glNewList(cuerpoprincipal,GL_COMPILE);
        glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,colorrobot);
        glPushMatrix();
            glRotatef(-90,1,0,0);
            glTranslatef(0,0,-12);
            gluCylinder(robot,10,10,32,20,2);
            glTranslatef(0,0,30);
            gluDisk(robot,0,10,20,2);
            glTranslatef(0,0,-12);
            glRotatef(180,1,0,0);
            gluDisk(robot,0,10,20,2);
        glPopMatrix();
    glEndList();
```

```
brazosdelrobot=glGenLists(1);
    glNewList(brazosdelrobot,GL_COMPILE);
    glPushMatrix();
        glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,colorrobott);
        glutSolidCube(5);
        glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,colorrobot);
        gluCylinder(robot,3.1,3.1,19,20,2);
    glPopMatrix();
glEndList();
```

```
    piernasdelrobot=glGenLists(1);
        glNewList(piernasdelrobot,GL_COMPILE);
        glPushMatrix();
            glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,colorrobott);
            glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,colorrobot);
            gluCylinder(robot,4.2,4.2,19,20,2);
        glPopMatrix();
    glEndList();

        glFlush();
    glutSwapBuffers();
}
```

FUNCIÓN PARA CREAR Y DAR MOVIMIENTO DEL BRAZO DERECHO

Esto será mediante la técnica de rotación

```
void creacionbrazoderecho()
{
     glPushMatrix();
            glLoadName(brazoderecho);
            glTranslatef(-11.3,10,0);
            glRotatef(90,1,0,0);
            glRotatef(-10,0,1,0);
            glRotatef(90,0,1,0);
            glRotatef(-90,0,1,0);
            glRotatef(-pbd,1,0,0);
            glCallList(brazosdelrobot);
            glTranslatef(0,0,12);
            glRotatef(90,0,0,1);
        glPopMatrix();
}
```

FUNCIÓN PARACREAR Y DAR EL MOVIMIENTO DEL BRAZO IZQUIERDO

Para la creación del brazo vamos a utilizar variables que nos van a permitir utilizar las técnicas de rotación y traslación.

```
void creacionbrazoizquierdo()
{

        glPushMatrix();
            glLoadName(brazoisquierdo);
            glTranslatef(11.3,10,0);
            glRotatef(90,1,0,0);
            glRotatef(10,0,1,0);
            glRotatef(-90,0,1,0);
            glRotatef(90,0,1,0);
            glRotatef(-pbi,1,0,0);
            glCallList(brazosdelrobot);
            glTranslatef(0,0,12);
            glTranslatef(0,0,11);
            glRotatef(-90,0,0,1);
        glPopMatrix();
}
```

FUNCIÓN PARA CREAR AMBAS PIERNAS Y MANIPULAR SU MOVIMIENTO

Para la creación de las piernas vamos a utilizar variables que nos van a permitir utilizar las técnicas de rotación y traslación .

```
void creaciondeambaspiernas()
{
        glPushMatrix();
            glLoadName(musloderecho);

            glRotatef(90,1,0,0);
            glTranslatef(-5,-5,11);
            glRotatef(pbd,1,0,0);
            glCallList(piernasdelrobot);
            glTranslatef(0,-1,1);
            glLoadName(piernaderecha);
        glPopMatrix();
        glFlush();
```

```
        glLoadName(musloiquierdo);
        glRotatef(90,1,0,0);
        glTranslatef(5,0,11);
        glRotatef(pbi,1,0,0);
        glCallList(piernasdelrobot);
        glTranslatef(0,0,-18);
        glLoadName(piernaizquierda);
        glTranslatef(0,0,-15);
        glPopMatrix();
        glFlush();
    glutSwapBuffers();
```

FUNCIÓN PARA CONSTRUIR LA PIERNA BAJA DEL ROBOT

```
void construcciondebajodepierna()
{
        glPushMatrix();
            glLoadName(musloiquierdo);
            glRotatef(90,1,0,0);
            glTranslatef(5,0,11);
            glRotatef(pbi,1,0,0);//15 a-15
            glCallList(piernasdelrobot);
            glTranslatef(0,0,-18);
            glLoadName(piernaizquierda);
            glTranslatef(0,0,-15);
        glPopMatrix();
        glFlush();
    glutSwapBuffers();
}
```

Con la siguiente función vamos a llamar al cuerpo principal del robot y posteriormente terminar de dibujarlo

```
void Robot(void)
{
    glClear( GL_COLOR_BUFFER_BIT  | GL_DEPTH_BUFFER_BIT );
   Esena();
    glPushMatrix();
    glTranslatef(cx,cy,cz);
    glRotatef(r,0,1,0);
        glPushMatrix();
            glCallList(cuerpoprincipal);
            glTranslatef(0,20,0);
            glCallList(cabeza);
            glTranslatef(0,4,0);
            glRotatef(0,1,0,0);
        glPopMatrix();
```

```
        creacionbrazoderecho();                 //la primera declaracion de arriba para hacer lo ultimo
        creacionbrazoizquierdo();               //se declaro arriba para su ejecucion
        creaciondeambaspiernas();               //se crearon en un mismo solo para no hacer otro
        construccciondebajodepierna();          //terminacion asta abajo para completar el robot

glPopMatrix();
glFlush();
glutSwapBuffers();
```

Recalculamos la transformación de visualización

```
void setViewTrans()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(-1,1,-1,1,2,1000);

    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();

    gluLookAt(eyex,eyey,eyez,0,0,0,0,0,1);
    glTranslatef( 0, 0,-Distancia);
    glRotatef(rot,0,1,0);
}
```

```
void WinReshapeFcn(GLsizei ancho,GLsizei alto)
{
    glViewport(0,0,ancho,alto);
    WinWidth= ancho;
    WinHeight = alto;
    setViewTrans();
}
```

FUNCIONES DE TECLADO PARA DAR MOVIMIENTO AL ROBOT (DESPLAZAMIENTO).

Para el movimiento de las extremidades del robot vamos a utilizar un switch donde nos permita decidir hacia donde será la dirección del robot.

Movimiento hacia atrás, mediante un caso de switch asignamos una dirección a nuestro robot, esto será por medio de una tecla.

```
void teclas(GLint opc, GLint x, GLint y)
{
    switch (opc)
    {
        case GLUT_KEY_UP:
                r=180;
            if((pbi==0&&pbd==0)||(pbi==-25&&pbd==25))
            {
                    pbi=25;
                    pbd=-25;
            }
            else
            {
                pbi=-25;
                pbd=25;
            }
                cz=cz-5;
                if(cz==-15&&cx<1&&cx>-70)


 {
 cz=cz+5;
 }
 if(cz==55&&cx<-45&&cx>-115)
 {
 cz=cz+5;
 }
 if(cz==-75&&cx>50&&cx<130)
 {
 cz=cz+5;
```

```
Robot();
if(cz==-290)
{
    for(int i=0;i<26;i++)
    {
        cy=cy-5;
        Robot();
        Sleep(10);
    }
 }
break;
```

Movimiento hacia adelante, mediante un caso de switch asignamos a una tecla una dirección para nuestro robot.

```
case GLUT_KEY_DOWN:

        r=0;

        if((pbi==0&&pbd==0)||(pbi==-25&&pbd==25)){
            pbi=25;
            pbd=-25;
        }else{
            pbi=-25;
            pbd=25;
        }
    cz=cz+5;

    if(cz==-105&&cx<1&&cx>-70)
    {
    cz=cz-5;
    }
    if(cz==-10&&cx<-40&&cx>-110)
    {
```

```
Robot();
if(cz==190)
{
    for(int i=0;i<26;i++)
     {
        cy=cy-5;
        Robot();
        Sleep(10);

    }
}
 if(cy==-130)
 {
    cx=0;cy=180;cz=0;
        for(int j=0;j<36;j++)
    {
        cy=cy-5;
        Robot();
        Sleep(10);
    }
```

Movimiento hacia la derecha mediante un caso de switch, manipulamos una tecla para dar una dirección de nuestro robot

```
case GLUT_KEY_RIGHT:
        r=90;
        if((pbi==0&&pbd==0)||(pbi==-25&&pbd==25)){
            pbi=25;
            pbd=-25;
        }else{
            pbi=-25;
            pbd=25;
        }
     cx=cx+5;
    if(cx==-65&&cz<-25&&cz>-105)
    {
    cx=cx-5;
    }
    if(cx==-115&&cz<50&&cz>-15)
    {
    cx=cx-5;
    }
    if(cx==5&&cz>30&&cz<110)


  Robot();
  if(cx==160)
  {
     for(int i=0;i<26;i++)
     {
         cy=cy-5;
         Robot();
         Sleep(10);

     }
  }
  break;
```

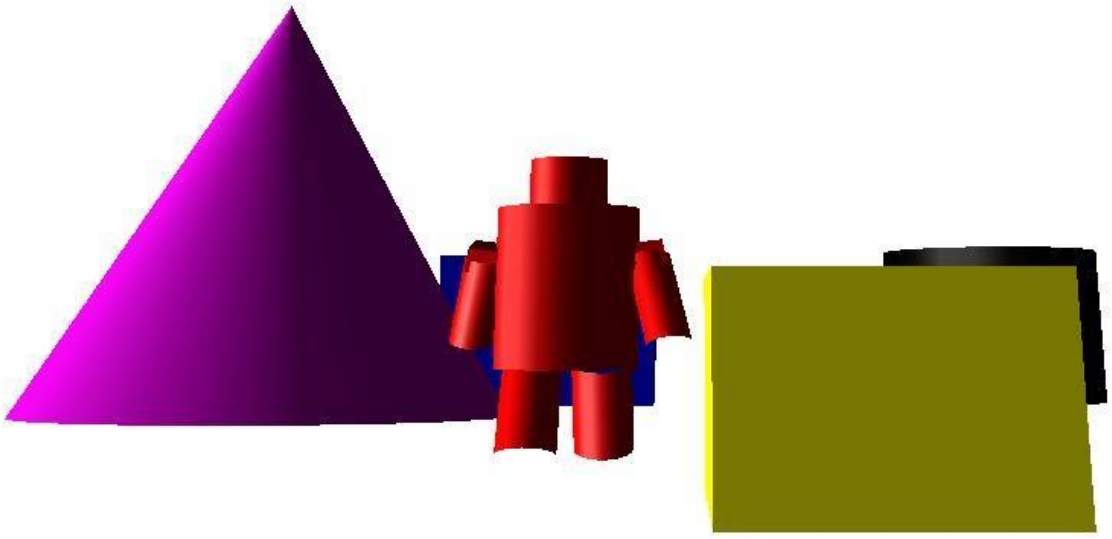
Movimiento hacia la izquierda mediante un caso del switch, manipulamos una tecla de modo que pueda tener un desplazamiento.

```
case GLUT_KEY_LEFT:
        r=-90;
        if((pbi==0&&pbd==0)||(pbi==-15&&pbd==15))
        {
            pbi=15;
            pbd=-15;
        }
         else
        {
            pbi=-15;
            pbd=15;
        }
     cx=cx-5;
     if(cx==0&&cz<-25&&cz>-105)
     {
    cx=cx+5;
    }
    if(cx==-40&&cz<50&&cz>-10)
    {


   Robot();
   if(cx==-165)
   {
      for(int i=0;i<26;i++)
       {
          cy=cy-5;
          Robot();
          //std::cout<<cy<<"\n";
          Sleep(10);

       }
   }
   break;
```

Para finalizar únicamente manipulamos nuestra ventana de salida.

```
int main( int argc, char *argv[])
{
    glutInit( &argc, argv );

    glutInitWindowPosition( 100,0 );
    glutInitWindowSize(WinWidth,WinHeight);
    glutInitDisplayMode( GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH );
    glutCreateWindow("ROBOT");
    glutDisplayFunc(Robot);
    glutReshapeFunc(WinReshapeFcn);
    glutSpecialFunc(teclas);

    init();
    glutMainLoop();
}
```

CAPTURA DE LA SALIDA



FUNCIONES PARA MANIPULAR LA VENTANA

Para finalizar dentro de colocamos algunas funciones que nos van a servir para manipular la ventana de ejecución de nuestro programa y algunos gráficos:

se inicializa pantalla grafica	glutInit(&argc, argv);
establece modo de visualización	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
se establece la posición de la pantalla grafica	glutInitWindowPosition(100,100);
se establece ancho y altura de la ventana	glutInitWindowSize(400,400);
se crea la venta de visualización	glutCreateWindow("Mi primer dibujo");
se ejecuta la función de inicialización de parámetros	init();
se envían gráficos a pantalla	glutDisplayFunc(dibujaPuntos);
optimiza el refrescado de gráficos para hacerlo más fluido	glutMainLoop();

CONCLUSIÓN/ OPINIÓN PERSONAL

El desarrollar un escenario con ambiente en 3D nos permitió divertirnos un poco con las herramientas que conocimos en clase, incluso logramos implementar algunas técnicas que no habíamos podido agregar a prácticas anteriores. Se comprendió mejor el uso de la iluminación para las figuras además logramos darle movimiento a nuestra figura principal.

Al principio no teníamos muy claro que escenario íbamos a crear porque la idea era utilizar la mayoría de las técnicas aprendidas, al final optamos por crear un robot con movimiento entre figuras en 3D.

REFERENCIAS

http://blackcrystal.wordpress.com/2009/07/21/opengl-y-c-como-usar- las-teclas-de-flecha