UNIVERSIDAD NACIONAL JORGE BASADRE GROHMANN

Facultad de Ingeniería

Escuela Profesional de Ingeniería Informática y Sistemas

Trabajo de Laboratorio

Presentado por:

Gómez Condori. Jean Carlo 2018 - 119027

Caxi Arocutipa, Kevin Dixon 2018 - 119023

Curso: DISEÑO ASISTIDO POR COMPUTADOR

Tacna- Perú

2022

Actividad 01

1) Implemente un menú donde el usuario pueda elegir qué forma dibujar cubos y pirámides. Al elegir una de estas opciones, el objeto debe dibujarse en el centro de la pantalla con un tamaño fijo. Dibuja la pirámide con una base cuadrada como en la figura de abajo.

2) Implemente transformaciones geométricas 3D de traslación, escala y rotación en los puntos de los objetos creados. Recuerde implementar la traslación, la escala y la rotación con operaciones matriciales. Para la rotación, no use una matriz para cada dirección, esto es más complicado y requiere más operaciones de matriz. Utilice el concepto de Cuaternion, aplicando la matriz que se presenta a continuación para rotar los puntos.

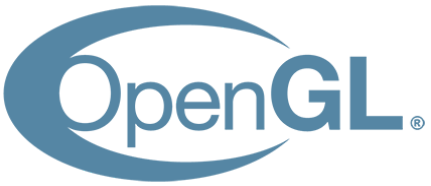
$$M_R = \begin{bmatrix} x^2(1-c)+c & xy(1-c)-zs & xz(1-c)+ys & 0 \\ yx(1-c)+zs & y^2(1-c)+c & yz(1-c)-xs & 0 \\ zx(1-c)-ys & zy(1-c)+xs & z^2(1-c)+c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

donde $c = cos(\theta)$, $s = sin(\theta)$ y $(x; y; z)$ las coordenadas del vector unitario $u \rightarrow$ alrededor del cual debe ocurrir la rotación.
Observación: En lugar de rotar con las teclas direccionales, utilice las letras x, s y z para rotar alrededor de los respectivos ejes. Los sentidos de giro se pueden modificar utilizando, por ejemplo, x para el sentido de las agujas del reloj y s para el sentido contrario a las agujas del reloj. Las transformaciones de escala y traslación deben continuar usando los mismos accesos directos (+ y - para la escala y el clic del mouse para la traslación).

Solución

1. Introducción

Para esta presente actividad se ha investigado sobre las funciones que nos ofrece la librería OpenGL, para el diseño de formas geométricas 3D.

Librería a usar. OpenGL
OpenGL (Open Graphics Library) es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. 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.   

2. Explicación del código

3. Librerías y definiciones

Librería stdio.h
Las funciones declaradas en stdio pueden clasificarse en dos categorías: funciones de manipulación de ficheros y funciones de manipulación de entradas y salidas. Cierra un fichero a través de su puntero. Abre un fichero para lectura, para escritura/reescritura o para adición.

Librería iostream
La librería iostream es un componente de la biblioteca estándar (STL) del lenguaje de programación C++ que es utilizado para operaciones de entrada/salida. Su nombre es un acrónimo de Input/Output Stream. El flujo de entrada y salida de datos en C++ (y su predecesor C) no se encuentra definida dentro de la sintaxis básica y se provee por medio de librerías de funciones especializadas como iostream

Librería GL/glut.h
GLUT Mecanismos (del inglés OpenGL Utility Toolkit) es una biblioteca de utilidades para programas OpenGL que principalmente proporciona diversas funciones de entrada/salida con el sistema operativo.

1. **Variables**

   Coordenada de rotación: coordRotacion
   Coordenada de traslación: coordTraslacion
   Coordenada de escala: coordEscalacion
   Tecla por el usuario : tecla

2. **Funciones**

Función escala Estamos inicializando los valores de la coordenada de escala en un **for**
Función drawcubo En esta función estamos trazando todos los vértices y colores de la figura de un cubo.
Función drawpiramide En esta función estamos trazando todos los vértices y colores de la figura de una pirámide.
Función display En esta función donde llamamos a la función de trazar figura según acorde a la elección del usuario
Función transformación En esta función estamos inicializando los parámetros de los procesos de transformación (escala, rotación, escala).
Función teclado transformacion En esta función estamos definiendo los teclados para poder rotar o escalar en sistema.
Función keyboard En esta función estamos definiendo los teclados para poder desplazar,la figura en el sistema.
Función Update En esta función estamos utilizando para llamar la función de teclado para hacer las operaciones de transformación en el sistema.
Función principal (int main) En esta función estamos llamando a todas las funciones y un menú de selección de figura.

3. Código final

```
#include <stdio.h>
#include <iostream>
#include <GL/glut.h>

#define VK_2 0x32
#define VK_4 0x34
#define VK_6 0x36
#define VK_8 0x38
```

```
#define VK_9 0x39

using namespace std;

int tecla;

float coordrotacion[2];
float coordtraslacion[2];
float coordescalacion[2];
float interval = 1000 / 60;

void inicoordenadas(){
	for(int i=0; i<=2; i++){
		coordrotacion[i]=0;
		coordtraslacion[i]=0;
		coordescalacion[i]=0;
	}
}

void escala(){
	for(int i=0; i<=2; i++){
	coordescalacion[i]=0.2;
	}
}

void drawcubo(){

	 glBegin(GL_POLYGON);
		glColor3f(1,1,0);
		glVertex3f(  0.5, -0.5, -0.5 );
		glVertex3f(  0.5,  0.5, -0.5 );
		glVertex3f( -0.5,  0.5, -0.5 );
		glVertex3f( -0.5, -0.5, -0.5 );
		glEnd();

	 glBegin(GL_POLYGON);
	 glColor3f(   1.0,  1.0, 1.0 );//blanco
	 glVertex3f(  0.5, -0.5, 0.5 );
	 glVertex3f(  0.5,  0.5, 0.5 );
	 glVertex3f( -0.5,  0.5, 0.5 );
	 glVertex3f( -0.5, -0.5, 0.5 );
	 glEnd();

	 glBegin(GL_POLYGON);
	 glColor3f(  1.0,  0.0,  1.0 );//morado
	 glVertex3f( 0.5, -0.5, -0.5 );
	 glVertex3f( 0.5,  0.5, -0.5 );
	 glVertex3f( 0.5,  0.5,  0.5 );
	 glVertex3f( 0.5, -0.5,  0.5 );
	 glEnd();

	 glBegin(GL_POLYGON);
	 glColor3f(   0.0,  1.0,  0.0 );
	 glVertex3f( -0.5, -0.5,  0.5 );
	 glVertex3f( -0.5,  0.5,  0.5 );
	 glVertex3f( -0.5,  0.5, -0.5 );
	 glVertex3f( -0.5, -0.5, -0.5 );
	 glEnd();
```

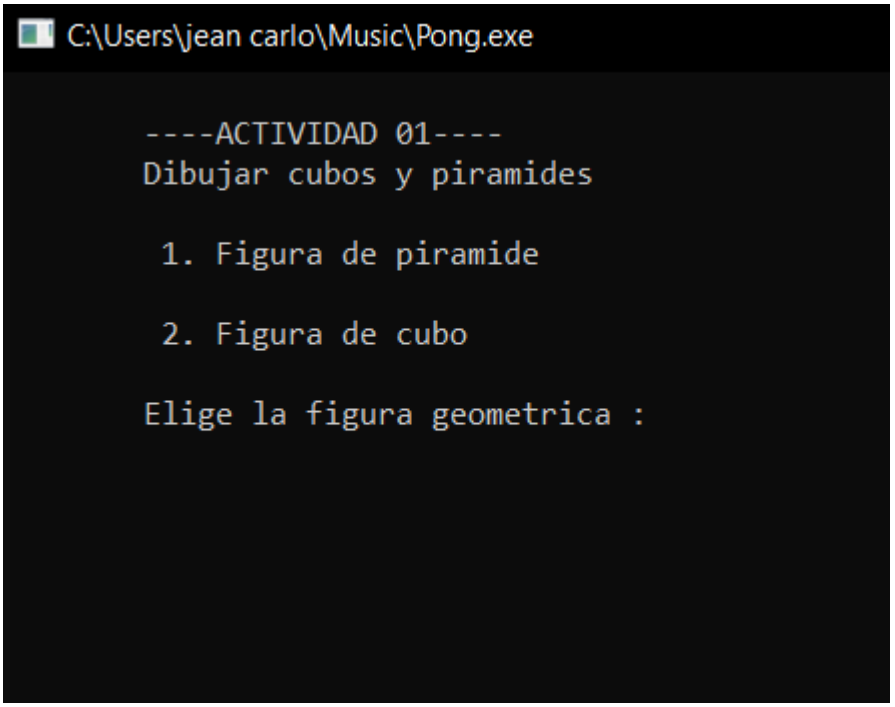
```
            glBegin(GL_POLYGON);
            glColor3f(   0.0,  0.0,  1.0 );//azul
            glVertex3f(  0.5,  0.5,  0.5 );
            glVertex3f(  0.5,  0.5, -0.5 );
            glVertex3f( -0.5,  0.5, -0.5 );
            glVertex3f( -0.5,  0.5,  0.5 );
            glEnd();

            glBegin(GL_POLYGON);
            glColor3f(   1.0,  0.0,  0.0 );// rojo
            glVertex3f(  0.5, -0.5, -0.5 );
            glVertex3f(  0.5, -0.5,  0.5 );
            glVertex3f( -0.5, -0.5,  0.5 );
            glVertex3f( -0.5, -0.5, -0.5 );
            glEnd();
}

void drawpiramide(){

          glBegin(GL_QUADS);
                    glColor3f(1,0,0);//rojo
                    glVertex3d(-2,0,-2);

                    glColor3f(1,1,0);//amarillo
                    glVertex3d(-2,0,2);

                    glColor3f(0,0,1);//azul
                    glVertex3d(2,0,2);

                    glColor3f(0,1,0);//verde
                    glVertex3d(2,0,-2);
                    glEnd();

                    glBegin(GL_TRIANGLES);
                    glColor3f(1,1,1);//blanco
                    glVertex3d(0,2,0);
                    glVertex3d(-2,0,-2);
                    glVertex3d(2,0,-2);
                    glEnd();

                    glBegin(GL_TRIANGLES);
                    glColor3f(1,1,1);//blanco
                    glVertex3d(0,2,0);
                    glColor3f(1,0,0);//rojo
                    glVertex3d(2,0,-2);
                    glVertex3d(2,0,2);
                    glEnd();

                    glBegin(GL_TRIANGLES);
                    glColor3f(0,1,0);//verde
                    glVertex3d(0,2,0);
                    glVertex3d(-2,0,2);
                    glVertex3d(-2,0,-2);
                    glEnd();

                    glBegin(GL_TRIANGLES);
                    glColor3f(1,1,1);//blanco
                    glVertex3d(0,2,0);
                    glColor3f(0,0,1);
```

```
                glVertex3d(2,0,2);
                glVertex3d(-2,0,2);
                glEnd();
}

void transformacion(){

                //ESCALAMIENTO
                glScalef( coordescalacion[0], coordescalacion[1], coordescalacion[2] );

                //TRASLACION
                glTranslatef( coordtraslacion[0], coordtraslacion[1], 0 );

                //ROTACION
                glRotatef(coordrotacion[0], 1.0, 0.0, 0.0);
                glRotatef(coordrotacion[1], 0.0, 1.0, 0.0);
                glRotatef(coordrotacion[2], 0.0, 0.0, 1.0);

}
void display(){

  glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
  glLoadIdentity();
        transformacion();
        if(tecla == 2){
                drawcubo();
        }
        if(tecla == 1){
                drawpiramide();
        }

  glFlush();
  glutSwapBuffers();
}

void tecladotransformacion( unsigned char tecla, int x, int y)
{
        switch(tecla)
        {
                case'x':coordrotacion[0] += 1;//derecha
                                break;
                case's':coordrotacion[1] += 1;//izquierda
                                break;
                case'z':coordrotacion[2] += 0.01;//arriba
                                break;

                case'-': coordescalacion[0] -= 0.02;
                                 coordescalacion[1] -= 0.02;
                                 coordescalacion[2] -= 0.02;
                                break;
                case'+': coordescalacion[0] += 0.02;
                                coordescalacion[1] += 0.02;
                                 coordescalacion[2] += 0.02;
                                break;
        }
  glutPostRedisplay();
}

void keyboard(){
```

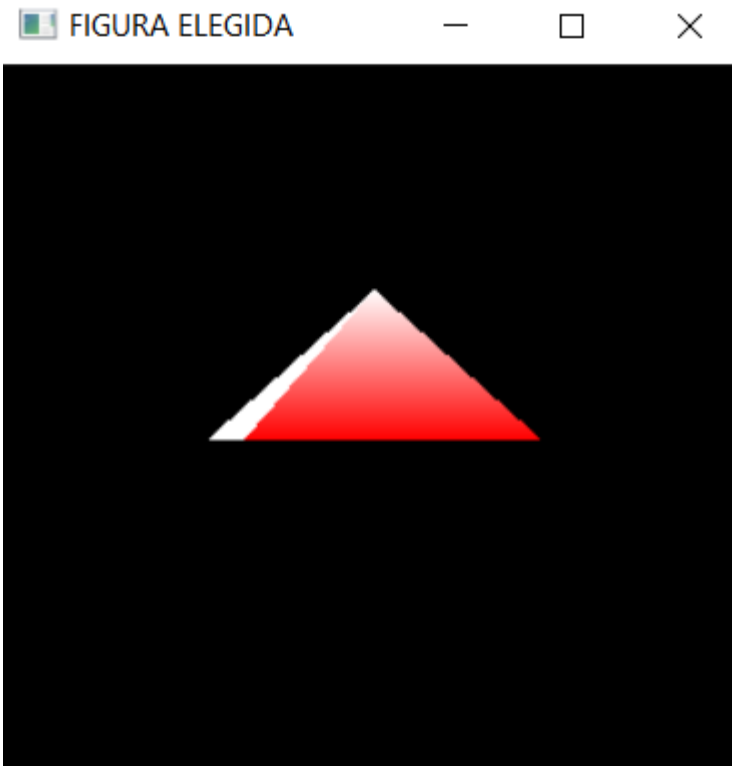
```
	if (GetAsyncKeyState(VK_2)){
	coordtraslacion[1] -= 0.1;
		}
	else if (GetAsyncKeyState(VK_8)){
	coordtraslacion[1] += 0.1;
		}
	if (GetAsyncKeyState(VK_4)){
	coordtraslacion[0] -= 0.1;
		}
	else if (GetAsyncKeyState(VK_6)){
	coordtraslacion[0] += 0.1;
		}
	if (GetAsyncKeyState(VK_9)){
			exit(0);
		}
}


void Update(int value)
{
	keyboard();
	glutTimerFunc(interval, Update, 0);
	glutPostRedisplay();
}

int main(int argc, char* argv[]){

		glutInit(&argc,argv);
		glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
		escala();

		cout<<"\n\t----ACTIVIDAD 01----\n";
		cout<<"\tDibujar cubos y piramides\n";
		cout<<"\n\t 1. Figura de piramide \n";
		cout<<"\n\t 2. Figura de cubo \n";

		cout<<"\n\tElige la figura geometrica : ";
		cin>>tecla;

		glutCreateWindow("FIGURA ELEGIDA");
		glEnable(GL_DEPTH_TEST);
		glutDisplayFunc(display);
		glutTimerFunc(interval, Update, 0);
		glutKeyboardFunc(tecladotransformacion);

		glutMainLoop();
		return 0;
}
```

Capturas del programa

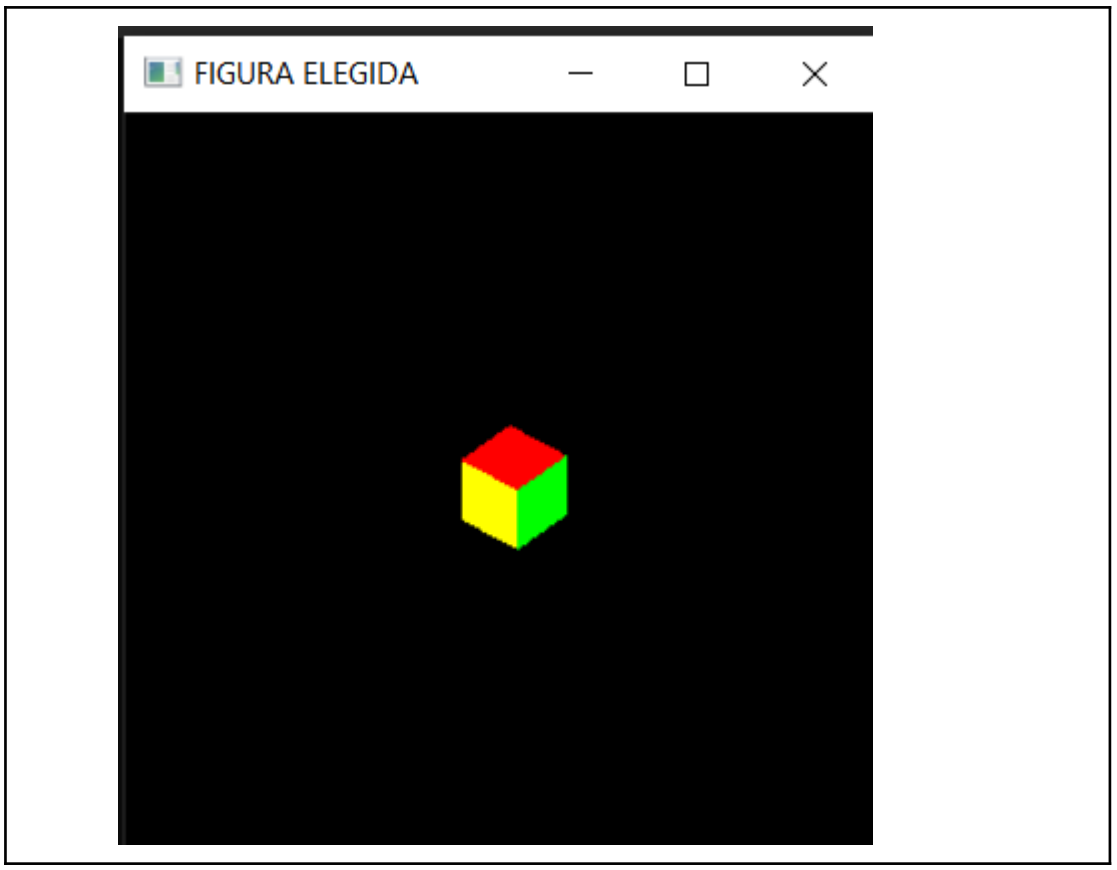
1. Menú principal



2. Dibujo de una pirámide



3. Dibujo de un cubo



Herramientas

Para este trabajo grupal, se ha utilizado plataformas de comunicación virtual para el desarrollo de este trabajo. Discord, para la comunicación efectiva entre los integrantes del grupo, y también el repositorio de Git-Hub, para el compartimiento de código mutuo.