Transformaciones geométricas 3D

Integrantes:

- Huaycani Gomez, Katherine Liliana 2018 - 119037
- Cohaila Aquino, Yaneli Suguey 2018 - 119034

Actividad 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.
- Implemente transformaciones geométricas 3D de traslación, escala y rotación en los puntos de los objetos creados.
- 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).

Intento 1

Con matrices de Transformaciones geométricas 3D de openGl

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

using namespace std;

int ancho = 800, alto =600;
float tx = 1, ty = 0, tz = 0;
float tam = 0;
float Sx = 0.9999, Sy = 0.9999, Sz = 0.9999;
int opct = 0,opcta = 0,opcts = 0,opctd = 0,opctw = 0;
int opcr = 0, opcr1 = 0, opcr2 = 0;
int opcs = 0, opcs1 = 0, opcs2 = 0;
float angulo = -0.02;
int contadorcaras = 0;
```

```
//Figura poligonal cubo
float MatrixP[24][3] = {
        {0.2,-0.2,-0.2}, {0.2,0.2,-0.2}, {-0.2,0.2,-0.2}, {-0.2,-0.2,-0.2},
        {0.2,-0.2,0.2}, {0.2,0.2,0.2}, {-0.2,0.2,0.2}, {-0.2,-0.2,0.2},
        {0.2,-0.2,-0.2}, {0.2,0.2,-0.2}, {0.2,0.2,0.2}, {0.2,-0.2,0.2},
        {-0.2,-0.2,0.2}, {-0.2,0.2,0.2}, {-0.2,0.2,-0.2}, {-0.2,-0.2,-0.2},
        {0.2,0.2,0.2}, {0.2,0.2,-0.2}, {-0.2,0.2,-0.2}, {-0.2,0.2,0.2},
        {0.2,-0.2,-0.2}, {0.2,-0.2,0.2}, {-0.2,-0.2,0.2}, {-0.2,-0.2,-0.2}

};

void pintaPixel(double Cx1, double Cy1, double Cz1, double Cx2,
double Cy2, double Cz2,int R, int G, int B) {
        glBegin(GL_LINES); //dibuja la linea en cada vertice que
enviamos
        glPointSize(1);
        glColor3f(1.0f, 0.5f, 0.0f);
        glVertex3f(Cx1, Cy1, Cz1);
        glVertex3f(Cx2, Cy2, Cz2);
        glEnd();
        glFlush();
}
```

```
void dibujaPlano(float Mimatriz[24][3]) {

        for (int i = 0; i < 18-1; i++)
  {
    if(i%3!=0){
      pintaPixel(Mimatriz[i][0], Mimatriz[i][1],Mimatriz[i][2],  Mimatriz[i +
1][0], Mimatriz[i + 1][1],Mimatriz[i + 1][2],
      0, 2, 8);
    }
  }
}

void dibujaPlano(float Mimatriz[24][3]) {

        for (int i = 0; i < 18-1; i++)
  {
    if(i%3!=0){
      pintaPixel(Mimatriz[i][0], Mimatriz[i][1],Mimatriz[i][2],  Mimatriz[i +
1][0], Mimatriz[i + 1][1],Mimatriz[i + 1][2],
      0, 2, 8);
    }
  }
}
```

```
void Rotacion(float Mimatriz[24][3],int a){

float Xc = Mimatriz[20][0];
float Yc = Mimatriz[20][1];
float Zc = Mimatriz[20][2];
if(a==1) angulo=-0.05;
if(a==2) angulo=0.05;

float MatRot[4][4] = {

{Xc*Xc*(1-cos(angulo))+cos(angulo) ,   Xc*Yc*(1-cos(angulo))-Xc*sin(angulo) ,
Xc*Zc*(1-cos(angulo))+Yc*sin(angulo),0},
{Yc*Xc*(1-cos(angulo))+Zc*sin(angulo) , Yc*Yc*(1-cos(angulo))+ cos(angulo) ,
Yc*Zc*(1-cos(angulo))-Xc*sin(angulo),0},
{Zc*Xc*(1-cos(angulo))-Yc*sin(angulo),  Zc*Yc*(1-cos(angulo))+Xc*sin(angulo) ,
Zc*Zc*(1-cos(angulo))+sin(angulo) ,0},
            {0, 0, 0 , 1}};
for (int i = 0; i < 24; i++)
{
Mimatriz[i][0] = ((Mimatriz[i][0] * MatRot[0][0]) + (Mimatriz[i][1] * MatRot[0][1]) +
(Mimatriz[i][2] * MatRot[0][2]) + MatRot[0][3]);
Mimatriz[i][1] = ((Mimatriz[i][0] * MatRot[1][0]) + (Mimatriz[i][1] * MatRot[1][1]) +
(Mimatriz[i][2] * MatRot[1][2]) + MatRot[1][3]);
Mimatriz[i][2] = ((Mimatriz[i][0] * MatRot[2][0]) + (Mimatriz[i][1] * MatRot[2][1]) +
(Mimatriz[i][2] * MatRot[2][2]) + MatRot[2][3]);
}
dibujaPlano(Mimatriz);}
```

```
void Traslacion(float Mimatriz[24][3]){
            float MatTras[4][4] = {
            { 1,0,0,tx },
             { 0,1,0,ty },
             {0,0,1,tz} ,
             {0,0,0,1} };

if(opcta ==1 && opctw == 1){
            tx = -0.75;ty = -0.75;
} else { if(opcta == 1 && opcts==1){
            tx = -0.75;ty = 0.75;
} else { if(opctd ==1 && opctw == 1){
            tx = 0.75;ty = -0.75;
} else { if(opctd == 1 && opcts==1){
            tx = 0.75;ty = 0.75;
} else {
            if (opcta == 1){tx = -1;ty = 0;}
            if (opctd == 1){tx = 1;ty = 0;}
            if (opctw == 1){tx = 0;ty = -1;}
            if (opcts == 1){tx = 0;ty = 1;}
}
}}}

float punto1 = Mimatriz[12][0];
float punto2 = Mimatriz[12][1];
```

```
for (int i = 0; i < 107; i++)
		{
		Mimatriz[i][0] = ((Mimatriz[i][0] * MatTras[0][0]) +
(Mimatriz[i][1] * MatTras[0][1]) + (Mimatriz[i][2] * MatTras[0][2]) +
MatTras[0][3]);
		Mimatriz[i][1] = ((Mimatriz[i][0] * MatTras[1][0]) +
(Mimatriz[i][1] * MatTras[1][1]) + (Mimatriz[i][2] * MatTras[1][2]) +
MatTras[1][3]);
		Mimatriz[i][2] = ((Mimatriz[i][0] * MatTras[2][0]) +
(Mimatriz[i][1] * MatTras[2][1]) + (Mimatriz[i][2] * MatTras[2][2]) +
MatTras[2][3]);
}
dibujaPlano(Mimatriz);
}
```

```
void Escalacion(float Mimatriz[24][3], int a){
float Xc = Mimatriz[12][0];
float Yc = Mimatriz[12][1];
float Zc = Mimatriz[12][2];
float MatEsc[4][4] = {
		{Sx,0, 0, Xc * (1 - Sx)},
		{0,Sy,0,Yc*(1-Sy)},
		{0,0,Sz,Zc*(1-Sz)},
		{0,0,0,1} };
if(a == 1){
		Sx = 1.0013; Sy = 1.0013; Sz = 1.0013;
}if(a == 0){
		Sx = 0.995; Sy = 0.995;nSz = 0.995;
} for (int i = 0; i < 107; i++)
		{
Mimatriz[i][0] = ((Mimatriz[i][0] * MatEsc[0][0]) + (Mimatriz[i][1] *
MatEsc[0][1]) + (Mimatriz[i][2] * MatEsc[0][2]) + MatEsc[0][3]);
Mimatriz[i][1] = ((Mimatriz[i][0] * MatEsc[1][0]) + (Mimatriz[i][1] *
MatEsc[1][1]) + (Mimatriz[i][2] * MatEsc[1][2]) +
MatEsc[1][3]);Mimatriz[i][1] = ((Mimatriz[i][0] * MatEsc[2][0]) +
(Mimatriz[i][1] * MatEsc[2][1]) + (Mimatriz[i][2] * MatEsc[2][2]) +
MatEsc[2][3]);}
		dibujaPlano(Mimatriz);
}
```

```
void teclado(unsigned char tecla, int x, int y) { //enviamos como parametro la letra que deseemos
switch (tecla){
	case 'a':
		if (opcta == 0){
			opct = 1; opcta = 1; opctd = 0;
		}else{
			opcta = opct = 0;}
		glutPostRedisplay();
		break;
	case 'd':
		if (opctd == 0){
			opct = 1;nopctd = 1; opcta = 0;
		}else{
			opctd = opct = 0;}
		glutPostRedisplay();
		break;
	case 'w':
		if (opctw == 0){
			opct = 1; opctw = 1; opcts = 0;
		}else{
			opctw = opct = 0;}
			glutPostRedisplay();
			break;}
```

```
	case 's':
		if (opcts == 0){
			opct = 1; opcts = 1; opctw = 0;
		}else{
			opcts = opct = 0;}
		glutPostRedisplay();
		break;
	case 'q':
		opcs2 = 0;
		if (opcs1 == 0){
			opcs1 = 1;opcs = 0;
		}else{
			opcs1 = opcs = 0;}
		glutPostRedisplay();
		break;
	case 'e':
		opcs1 = 0;
		if (opcs2 == 0){
			opcs2 = 1;opcs = 1;
		}else{
			opcs2 = opcs = 0;}
		glutPostRedisplay();
		break;
```

```
case 'r':
		opcr2 = 0;
		if (opcr1 == 0){ opcr1 = 1;opcr = 1;
		}else{ opcr1 =opcr  = 0;}
		glutPostRedisplay();
		break;
case 'f':
		opcr1 = 0;
		if (opcr2 == 0){ opcr2 = 1;opcr = 2;
		}else{ opcr2 =opcr = 0;}
		glutPostRedisplay();
		break; } }

void display() {

		glClear(GL_COLOR_BUFFER_BIT); //limpia la pantalla
		if (opcr + opcs + opct == 0) {
			dibujaPlano(MatrixP);}
		if (opct  == 1) {
			Traslacion(MatrixP);}
		if (opcs  == 1 || opcs1 == 1) {
		 Escalacion(MatrixP, opcs);}
		if (opcr1 == 1 || opcr2 == 1 ) {
		Rotacion(MatrixP,opcr);}
		glFlush();
		glutSwapBuffers();
		glutPostRedisplay(); }
```

```
int main(int arg, char* argv[]) {
//INICIALIZAR LA LIBRERIA OPENGL CON SUS PARAMETROS
		glutInit(&arg, argv);
//CONFIGURAR EL MODO DE PANTALLA, ENVIANDO 2 PARAMETROS
LOS COLORES Y LOS BUFFER A UTILIZAR
		glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
//CONFIGURAMOS EL ANCHO Y ALTO DE LA VENTANA
		glutInitWindowSize(ancho, alto);
//CONFIGURAMOS LA POSICION DE LA VENTANA
		glutInitWindowPosition(75, 50);
//CREAR VENTANA Y TITULO
		glutCreateWindow("TRANSFORMADAS GEOMETRICAS 2D");
//llamamos la funcion dps
		glutDisplayFunc(display);
//funcion para hacer uso del teclado
		glutKeyboardFunc(teclado);
//FUNCION QUE REPETIR EL CICLO
		glutMainLoop();
//return 0;
}
```

Intento 2

Con funciones reservadas de openGl

```
#include <stdio.h>
#include <GL/glut.h>
#include <stdlib.h>
#include <iostream>
using namespace std;
double rY = 0;
double rX = 0;
double rZ = 0;
GLfloat X = 0.0f;
GLfloat Y = 0.0f;
GLfloat Z = 0.0f;
GLfloat escala = 1.0f;
int opcion;
```

```
void PIRAMIDE ()
{
	glClearColor(0,0,0,0);
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();

	glRotatef(rX, 1.0, 0.0, 0.0);
	glRotatef(rY, 0.0, 1.0, 0.0);
	glRotatef(rZ, 0.0, 0.0, 1.0);

	glTranslatef(X, Y, Z);
	glScalef(escala, escala, escala);

	glBegin(GL_QUADS);
	glColor3f(1.0,1.0,1.0);
	glVertex3d(-0.2,0,-0.2);
	glVertex3d(-0.2,0,0.2);
	glVertex3d(0.2,0,0.2);
	glVertex3d(0.2,0,-0.2);
	glEnd();

	glBegin(GL_TRIANGLES);
	glColor3f(0,1,0.6);
	glVertex3d(0,0.2,0);
	glVertex3d(-0.2,0,-0.2);
	glVertex3d(0.2,0,-0.2);
	glEnd();

	glBegin(GL_TRIANGLES);
	glColor3f(1,0,0);
	glVertex3d(0,0.2,0);
	glVertex3d(0.2,0,-0.2);
	glVertex3d(0.2,0,0.2);
	glEnd();

	glBegin(GL_TRIANGLES);
	glColor3f(0,1,0);
	glVertex3d(0,0.2,0);
	glVertex3d(-0.2,0,0.2);
	glVertex3d(-0.2,0,-0.2);
	glEnd();

	glBegin(GL_TRIANGLES);
	glColor3f(1.0, 0.1, 0.9);
	glVertex3d(0,0.2,0);
	glVertex3d(0.2,0,0.2);
	glVertex3d(-0.2,0,0.2);
	glEnd();

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

```
void CUBO ()
{
	glClearColor(0,0,0,0);
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();

	glRotatef(rX, 1.0, 0.0, 0.0);
	glRotatef(rY, 0.0, 1.0, 0.0);
	glRotatef(rZ, 0.0, 0.0, 1.0);

	glTranslatef(X, Y, Z);
	glScalef(escala, escala, escala);

	glBegin(GL_POLYGON);
	glColor3f(1.0, 1.0, 1.0);
	glVertex3f(0.2,-0.2,-0.2);
	glVertex3f(0.2,0.2,-0.2);
	glVertex3f(-0.2,0.2,-0.2);
	glVertex3f(-0.2,-0.2,-0.2);
	glEnd();

	glBegin(GL_POLYGON);
	glColor3f(0.0, 1.0, 0.60);
	glVertex3f(0.2,-0.2,0.2);
	glVertex3f(0.2,0.2,0.2);
	glVertex3f(-0.2,0.2,0.2);
	glVertex3f(-0.2,-0.2,0.2);
	glEnd();

	glBegin(GL_POLYGON);
	glColor3f(1.0, 0.5, 0.0);
	glVertex3f(0.2,-0.2,-0.2);
	glVertex3f(0.2,0.2,-0.2);
	glVertex3f(0.2,0.2,0.2);
	glVertex3f(0.2,-0.2,0.2);
	glEnd();

	glBegin(GL_POLYGON);
	glColor3f(0.6, 0.80, 0.51);
	glVertex3f(-0.2,-0.2,0.2);
	glVertex3f(-0.2,0.2,0.2);
	glVertex3f(-0.2,0.2,-0.2);
	glVertex3f(-0.2,-0.2,-0.2);
	glEnd();

	glBegin(GL_POLYGON);
	glColor3f(1.0, 0.1, 0.9);
	glVertex3f(0.2,0.2,0.2);
	glVertex3f(0.2,0.2,-0.2);
	glVertex3f(-0.2,0.2,-0.2);
	glVertex3f(-0.2,0.2,0.2);
	glEnd();

	glBegin(GL_POLYGON);
	glColor3f(0.0, 1.0, 0.0);
	glVertex3f(0.2,-0.2,-0.2);
	glVertex3f(0.2,-0.2,0.2);
	glVertex3f(-0.2,-0.2,0.2);
	glVertex3f(-0.2,-0.2,-0.2);
	glEnd();

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

```
void raton(int boton, int estado, int x , int y)
{
	if(boton == GLUT_LEFT_BUTTON && estado == GLUT_DOWN)
	{
		X+= 0.1;
		Y+= 0.1;
	}if(boton == GLUT_RIGHT_BUTTON && estado == GLUT_DOWN)
	{
		X-= 0.1;
		Y-= 0.1;
	}
	glutPostRedisplay();
}

void teclas (unsigned char key, int x , int y)
{
	switch(key){
		case '+':
			escala += 0.1;
			break;

		case '-':
			escala -= 0.1;
			break;

		//rotación
		case 'w':
			rX += 5;
			break;

		case 's':
			rX -= 5;
			break;

		case 'e':
			rY += 5;
			break;

		case 'r':
			rY -= 5;
			break;

		case 'a':
			rZ += 5;
			break;

		case 'd':
			rZ -= 5;
			break;
	}
	glutPostRedisplay();
}
```

```
int main (int argc, char* argv[])
{
	cout<<"-------------------TRANSFORMACIONES GEOMÉTRICA 3D--------------------"<<endl;
	cout<<"  Escoja la figura a graficar: "<<endl;
	cout<<"     1. Cubo"<<endl;
	cout<<"     2. Piramide"<<endl;
	cin>>opcion;

	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGB);
	glutInitWindowSize(800,800);
	glutInitWindowPosition(0,0);

	switch(opcion){
		case 1:

			glutCreateWindow("Transformaciones 3D");
			glEnable(GL_DEPTH_TEST);
			glutDisplayFunc(CUBO);
			glutKeyboardFunc(teclas);
			glutMouseFunc(raton);
			glutMainLoop();
			break;
		case 2:

			glutCreateWindow("Transformaciones 3D");
			glEnable(GL_DEPTH_TEST);
			glutDisplayFunc(PIRAMIDE);
			glutKeyboardFunc(teclas);
			glutMouseFunc(raton);
			glutMainLoop();
			break;
	}

	//glutSpecialFunc(flechasteclado);
	//glutMainLoop();
	return 0;
}
```