Tecnológico de Monterrey
Campus Puebla
Gráficas Computacionales

Documento Técnico Parcial 2

Juan Manuel Rendon Garcia A01731524

Maestro: Dr. Ivan Olmos Pineda

Grupo: TC3022.1

Fecha de entrega: 05/10/2021

Resumen:

Para la entrega de este parcial, se nos pidió entregar un escenario interactivo en el cual se presentarán objetos en 3D a nuestro criterio y se dieran ejemplos de traslación, escala, rotación y movimiento de dichos objetos. Utilizando el uso de pilas de estados para manejar y familiarizarnos con el concepto de estados y como trabajar con ellos. Ejerciendo una jerarquía de eventos y condiciones para todos los objetos.

Introducción:

Se trabajó en la aplicación CODEBLOCKS, la cual ofrece múltiples herramientas y comodidades para programar en GLUT. Teniendo en mente que todos los objetos serian manipulados por medio de una super clase (Llamada Operador en este ejemplo), conectando todas las clases de objetos con esta clase maestra para que únicamente se haga llamado a la manipulación y edición de variables de dichos objetos en la ejecución del escenario. De esta forma reduciendo código entre las múltiples clases creadas y a la vez facilitando el manejo y control de dibujo, actualización y trazos ya que cada clase se maneja por si sola. Siendo la super clase la encargada de mover y/o acomodar los elementos en su lugar.

```
#include <stdio.h>
#include <GL/glut.h>
#include <stdlib.h>
#include "Operador.h"
#include "math.h"

#define PI 3.14159265359

Operador::Operador()
{
    identidad(A);
    identidad(R);
    identidad(T);
    identidad(E);
}

Operador::~Operador()
{
}
void Operador::identidad(){
    identidad(A);
}

void Operador::identidad(float M[][4]){
    for(int i=0; i<4; i++){
        for(int j=0; j<4; j++){
            if(i==j){
                M[i][j] = 1;
            } else {
                M[i][j] = 0;
            }
        }
    }
}
```

Ejemplo de la super clase

```h
#ifndef CUBO_H
#define CUBO_H
#include "Punto.h"
#include "Operador.h"


class Cubo
{
    private:
        Punto misPuntos[8], misPuntosP[8];
        Operador *op;
        float puntosPivote[3], puntosOrigen[3];
    public:
        Cubo(Operador*);
        ~Cubo();
        void setOrigen(float x, float y, float z);
        void draw();
        void update();
};

#endif // CUBO_H
```

Ejemplo de la superclase implementada en otra clase

Codificación:

Siendo la super clase la prioridad en el proyecto, fue la clase mas extensa y con más código debido a las múltiples posibilidades e interacciones que pueda tener con otras clases. Dejando como base las ejecuciones esperadas (movimiento, rotación), y a partir de ellas ir haciendo los procesos de matrices internas. Las llamadas a identidades, acomodar en sus respectivos lugares y multiplicación de matrices. Todos estos procesos se minimizan y facilitan para que el usuario únicamente deba preocuparse de editar valores en términos de X, Y, Z. De esta forma damos acceso rápido y coherente a lo que se necesita editar o manipular en la escena.

```h
void Operador::multM(float M1[][4], float M2[][4], float Res[][4]){
    float tmp[4][4];

    for(int i=0;i<4;i++)
        for(int j=0;j<4;j++)
            tmp[i][j] = 0;

    for(int i=0;i<4;i++){
        for(int j=0;j<4;j++){
                for(int k=0;k<4;k++){
                    tmp[i][j] += M2[i][k]*M1[k][j];
                }
        }
    }

    for(int i=0; i<4; i++){
        for(int j=0; j<4; j++){
            Res[i][j] = tmp[i][j];
        }
    }
}

void Operador::multPuntos(float P[3], float Res[3]){
    float tmp[3];

    for(int i=0; i<3; i++)
        tmp[i] = 0;

    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
                    tmp[i] += P[j]*A[j][i];
        }
        tmp[i] += A[i][3];
    }

    for(int i=0; i<3; i++){
        Res[i] = tmp[i];
    }
}

void Operador::trsPuntos(float x, float y, float z){
    identidad(T);
    T[0][3] = x;
    T[1][3] = y;
```

Ejemplo de múltiples funciones en la super clase

```
class Operador{
    private:

        stack <Matriz> pila;
        void identidad(float M[][4]);
        void multM(float M1[][4], float M2[][4], float Res[][4]);

    public:
        float T[4][4], R[4][4], E[4][4], A[4][4];
        Operador();
        ~Operador();
        void identidad();
        void trs(float x, float y, float z);
        void esc(float x, float y, float z);
        void trsPuntos(float x, float y, float z);
        void escPuntos(float x, float y, float z);
        void trsX(float b, float c, float d);
        void trsX(float deg);
        void trsY(float a, float d);
        void trsY(float deg);
        void trsZ(float deg);
        void multPuntos(float Puntos[3], float M[3]);
        void rotar(float deg, float p1[3], float p2[3]);
        void rotX(float deg);
        void rotY(float deg);
        void rotZ(float deg);

        void push();
        void pop();
};
```

Ejemplo de todas las funciones y privilegios en la super clase

Una vez probada y compilada la super clase de forma satisfactoria, se procedió a la creación de 3 clases de objetos para que aparezcan y manipulen en el escenario. Tomando en cuenta que los objetos serian formas geométricas sencillas y con dimensiones de 1, se creo un cubo, pirámide y diamante.

```
Esfera::Esfera(Operador *opera)
{
    for(int i = 0; i < 26; i++)
        misPuntos[i].setOperador(opera);

    misPuntos[0].setValues(0,0,0); //Base
    int lvly = 1;
    int lvl = 1;
    for(int i = 1; i < 25; i = i + 8)
    {
        misPuntos[i].setValues((0.25 * lvl), (0.25 * lvly), 0);
        misPuntos[i+1].setValues((0.25 * lvl), (0.25 * lvly), (0.25 * lvl));
        misPuntos[i+2].setValues(0, (0.25 * lvly), (0.25 * lvl));
        misPuntos[i+3].setValues((0.25 * lvl), (0.25 * lvly), (-0.25 * lvl));
        misPuntos[i+7].setValues((-0.25 * lvl), (0.25 * lvly), (0.25 * lvl));
        misPuntos[i+6].setValues(0, (0.25 * lvly), (-0.25 * lvl));
        misPuntos[i+5].setValues((-0.25 * lvl), (0.25 * lvly), (-0.25 * lvl));
        misPuntos[i+4].setValues((-0.25 * lvl), (0.25 * lvly), 0);
        lvl += 1;
        lvly += 1;
        if (lvl > 2)
            lvl = 1;
    }
    misPuntos[25].setValues(0,1,0); //Cabeza

    for (int i = 0; i < 5; i++)
        misPuntosP[i] = misPuntos[i];

    for (int i = 0; i < 3; i++)
    {
        puntosOrigen[i] = 0;
        puntosPivote[i] = 0;
    }
    op = opera;
}
```

```
Triangulo::Triangulo(Operador *opera)
{
    for(int i = 0; i < 8; i++)
        misPuntos[i].setOperador(opera);

    misPuntos[0].setValues(0,0,0); //Origen
    misPuntos[1].setValues(1,0,0); //Punto x
    misPuntos[2].setValues(0,0,1); //Punto z
    misPuntos[3].setValues(1,0,1); //Punto esquina
    misPuntos[4].setValues(0.5,1,0.5); //Pico

    for (int i = 0; i < 5; i++)
        misPuntosP[i] = misPuntos[i];

    for (int i = 0; i < 3; i++)
    {
        puntosOrigen[i] = 0;
        puntosPivote[i] = 0;
    }
    op = opera;
}
```

Ejemplo de los códigos de las clases previamente mencionadas

Finalmente una vez creadas todas las clases que formaran parte del escenario es que se crea la clase Main que funcionará como nuestro escenario. Dentro de esta clase es que se hace llamado a todas las clases y además se inicializan valores como tamaños, dimensiones, ejes, etc… Esta clase tiene la libertad de ser editada y manipulada a gusto propio y da libertad de lo que se quiere exponer.

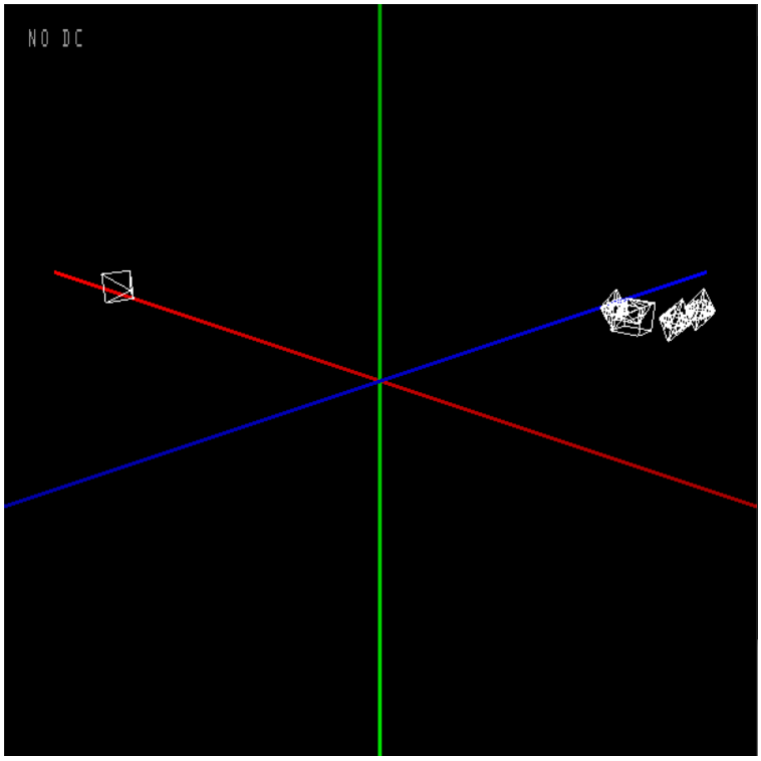
```
#include <stdlib.h>
#include "Punto.h"
#include "Cubo.h"
#include "Linea.h"
#include "Esfera.h"
#include "Triangulo.h"
#include "Operador.h"
#include <GL/glut.h>

//Variables dimensiones de la pantalla
int WIDTH=600;
int HEIGTH=600;
//Variables para establecer los valores de gluPerspective
float FOVY=60.0;
float ZNEAR=0.01;
float ZFAR=100.0;
//Variables para definir la posicion del observador
//gluLookAt(EYE_X,EYE_Y,EYE_Z,CENTER_X,CENTER_Y,CENTER_Z,UP_X,UP_Y,UP_Z)
float EYE_X=10.0;
float EYE_Y=5.0;
float EYE_Z=10.0;
float CENTER_X=0;
float CENTER_Y=0;
float CENTER_Z=0;
float UP_X=0;
float UP_Y=1;
float UP_Z=0;
//Variables para dibuiar los ejes del sistema
float X_MIN=-20;
float X_MAX=20;
float Y_MIN=-20;
float Y_MAX=20;
float Z_MIN=-20;
float Z_MAX=20;
```

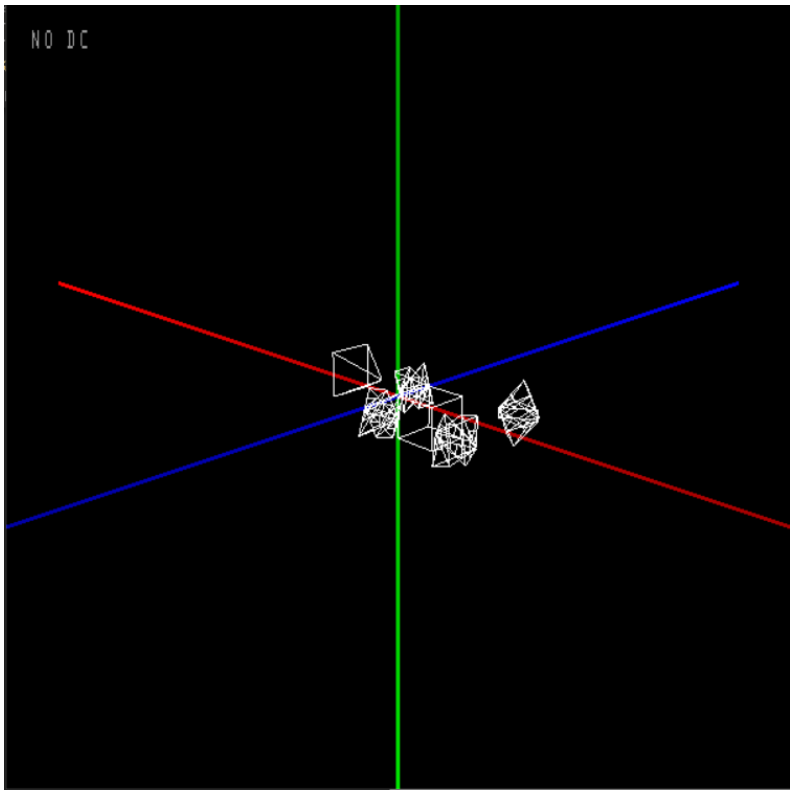
Ejemplo de inicialización de variables y objetos

Pruebas de ejecución:

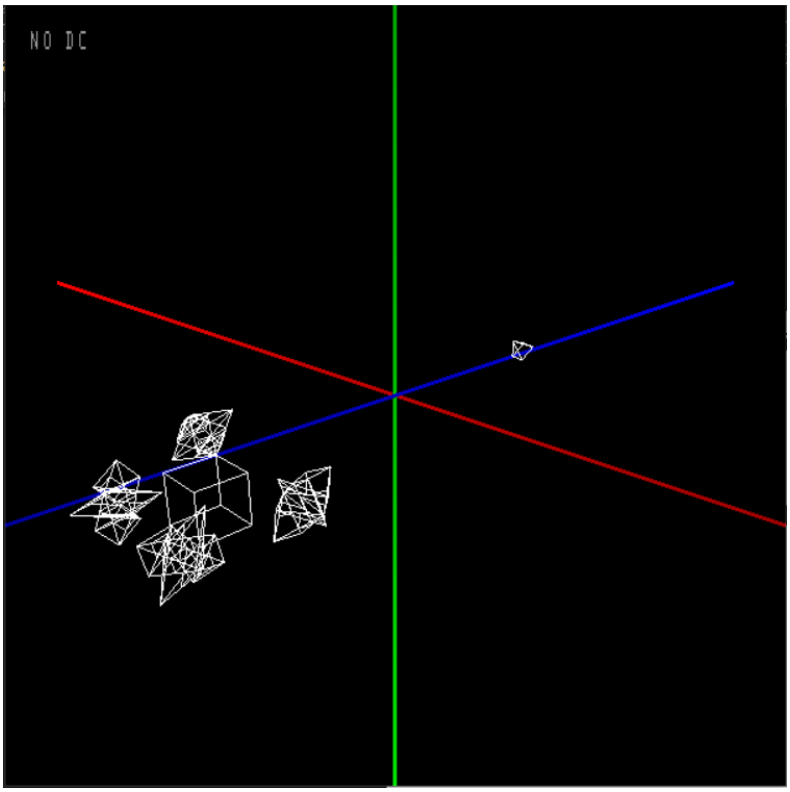
Dada mi selección de figuras geométricas, decidí crear una escena corta de unas naves espaciales. En la cual cuatro diamantes giran alrededor de un cubo mientras una pirámide intercepta el convoy y es derribada. Haciendo uso de traslaciones por objetos en conjunto y además rotaciones de cada objeto independiente junto con otra jerarquía de eventos para la nave piramidal, es que se utilizó la gran mayoría de comandos y recursos para demostrar una escena en 3D con movimiento, cambios y de interés. Tomando en cuenta que el uso de tuplas fue necesario para todo cambio y/o reposicionamiento de los elementos junto con ser el enfoque principal de dicha entrega.



Ejemplo 1: Inicializacion de escena



Ejemplo 2: Objetos en movimiento



Ejemplo 3: Manipulación de ángulos de objetos

Conclusiones:

Si bien fue mas complicado que el proyecto pasado, fue mas divertido poder trabajar con figuras tridimensionales e incluso conceptualizar una pequeña “aventura” que demostrar como parte de un proyecto parcial. Claramente requiere mucho mas detalle y algunas funciones no fueron correctas o ejecutadas en relación a la matriz madre en la cual ocurren todos los cambios. Mas se satisfacen varios puntos de la entrega y ejecutan de forma correcta los procesos previamente vistos. Aplicando lo visto en este parcial y mejorando el diseño y representación grafica de elementos y objetos.