MANUAL TÉCNICO

Índice

1.- Objetivos....1
2.- Información Técnica....1
2.1.- Hardware y Software utilizado para la realización del proyecto....1
2.2.- Referencias para la obtención de modelos....1
2.3.- Requerimientos necesarios para la ejecución del proyecto....2
2.4.- Licenciamiento....2
2.5.- Instalación....2
3.- Planeación del Proyecto y Diagrama de GANT....4
3.1.- Actividad 1....4
3.2.- Actividad 2....4
3.3.- Actividad 3....4
3.4.- Actividad 4....5
3.5.- Actividad 5....5
3.6.- Actividad 6....5
3.7.- Actividad 7....5
3.8.- Actividad 8....5
3.9.- Actividad 9....5
3.10.- Actividad 10....5
3.11.- Actividad 11....5
3.12.- Actividad 12....6
3.13.- Actividad 13....6
3.14.- Actividad 14....6
3.15.- Diagrama de Gantt....6
3.16.- Costo Final del Proyecto....6
4.- ALCANCE DEL PROYECTO....7
5.- Limitantes....7
6.- Documentación del código....8
6.3.- Dimensiones de la ventana....9
6.4.- Cámara....9
6.5.- Definición de variables a utilizar en animaciones simples y complejas....10
6.6.- Definición de variables de iluminación....10
6.7.- Carga de Modelos....11
6.8.- Carga de Shaders....11
6.9.- SkyBox....11

6.10.- Definición de Luces direccionales, puntos de luz y spotlights. 12
6.11.- Dibujado de modelos 12
6.12.- Función de animación 13
6.13.- Función de KeyCallback 14
6.14.- Función MouseCallback 14
6.15.- Función DoMovement 14
7.- Conclusiones generales 16

1.- Objetivos

El objetivo del presente trabajo es realizar la documentación del proyecto final.

El proyecto final tiene como objetivo principal aplicar y demostrar los conocimientos adquiridos durante el curso. Para esto se debe recrear un ambiente real dentro de un entorno virtual, modelando objetos para un parque temático de dinosaurios. El proyecto debe tener animaciones simples y animaciones complejas.

2.- Información Técnica

2.1.- Hardware y Software utilizado para la realización del proyecto

El proyecto final fue realizado bajo una serie de especificaciones tanto de software como de hardware con el objetivo de ayudar a que el desarrollo de este fuera lo más eficiente posible, por ello, se optó por utilizar un equipo cuyo hardware permitiera que el procesamiento del proyecto fuese lo más rápido posible.

Hardware utilizado:

- Procesador Intel Core i7 7700HQ 4 núcleos 8 hilos
- Tarjeta de video Nvidia GeForce 1050 4GB
- Memoria RAM de 16 GB

Software Utilizado

- Visual Studio 2022
- GIMP
- MAYA

2.2.- Referencias para la obtención de modelos

Para la realización del proyecto fue necesario recrear objetos del entorno real a un entorno virtual. Para esto fue necesario recurrir a la herramienta de Modelado MAYA, sin embargo, no todos los modelos que contiene el programa fueron creados desde 0. Fue necesario apoyarse de diversas páginas de internet las cuales brindan diferentes modelos con formato OBJ y textura para facilitar la realización del proyecto.

Algunas de las páginas utilizadas para la obtención de modelos con extensión OBJ fueron:

- https://www.turbosquid.com/
- https://free3d.com/es/modelos-3d/obj
- https://www.cgtrader.com/es/gratis-3d-modelos
- https://www.artec3d.com/es/3d-models/art-and-design

2.3.- Requerimientos necesarios para la ejecución del proyecto

Debido a que el proyecto fue creado bajo ciertas condiciones de Hardware, es posible que para que se pueda ejecutar de manera adecuada o similar al entorno de desarrollo se cumplan con los siguientes requerimientos mínimos de Hardware y Software

Hardware Mínimo

- Procesador Intel Core I3, I5, I7 de séptima generación en Adelante
- Procesador Ryzen 3,5,7 de segunda generación en adelante
- 8 GB de RAM
- 5 GB de almacenamiento

Software Requerido

- Sistema Operativo Windows 7 o superior

2.4.- Licenciamiento

Al ser un Proyecto Final con el objetivo de obtener una calificación, el proyecto no necesita licencia alguna, sin embargo, es necesario poner las referencias de las páginas de las cuales fueron obtenidos algunos modelos con el fin de dar crédito a quienes realizaron algunos de los modelos que fueron utilizados para el proyecto

2.5.- Instalación

Al ser un proyecto realizado en un lenguaje de programación C++, si el usuario lo requiere puede utilizar el ambiente de programación de su agrado que le permita manejar las librerías de C++. Sin embargo, con la finalidad de que no tenga problemas al momento de modificar el código o compilarlo, se recomienda utilizar alguna versión del Software Visual Studio en sus versiones 2019 en adelante.

El software se puede descargar en el siguiente enlace:

- https://visualstudio.microsoft.com/es/vs/community/

Si lo que se quiere únicamente es ejecutar el entorno virtual se debe de descargar carpeta que contiene todo el proyecto mediante el siguiente enlace:

- https://github.com/LFernandoGO/ProyectoFCG

El repositorio contiene todo el proyecto cargado, por lo que para descargar es necesario dar click en el botón verde llamado “Code”, se desplegará un menú de opciones el cual incluye la opción “Download ZIP” seleccionar esa opción y esperar a que se descargue el proyecto.

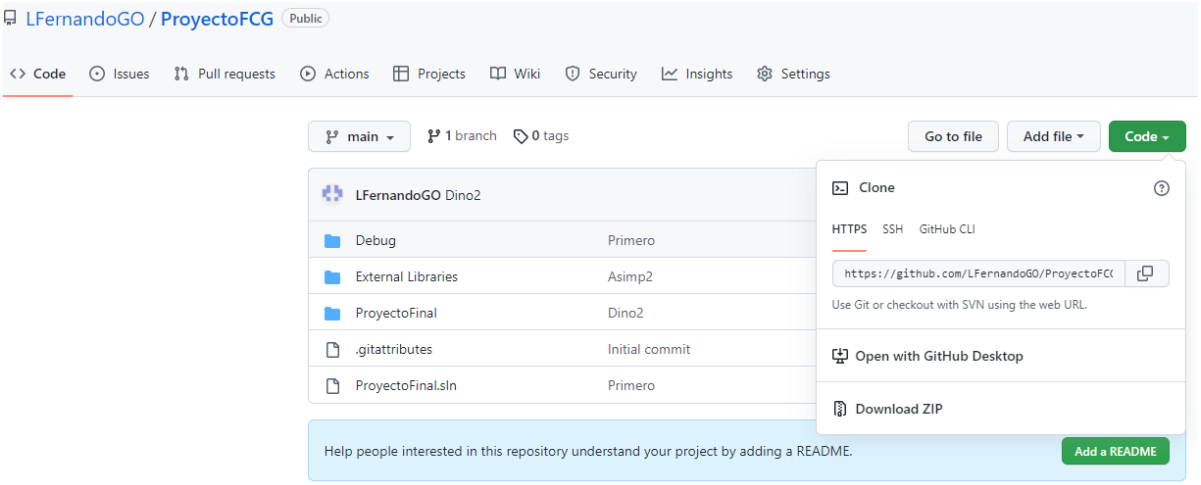

Imagen1. Github

Una vez que el proyecto fue descargado hay que extraer los archivos. Para esto se recomienda crear una carpeta en el Escritorio y extraer los archivos dentro de esa carpeta. Una vez extraídos los archivos se crearán varias carpetas, entrar a la que dice “Release”

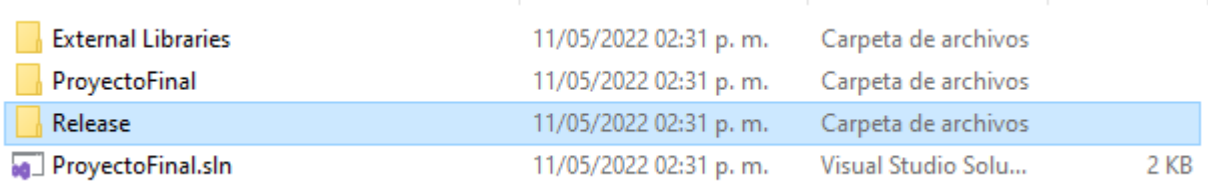

Imagen2. Explorador2

Dentro de la carpeta reléase se encuentran varios archivos, entre ellos el ejecutable. Para poder usar el proyecto es necesario abrir el archivo ejecutable con extensión ".exe" llamado "Proyecto Final". Una vez hecho lo anterior, se podrá navegar dentro del entorno virtual.

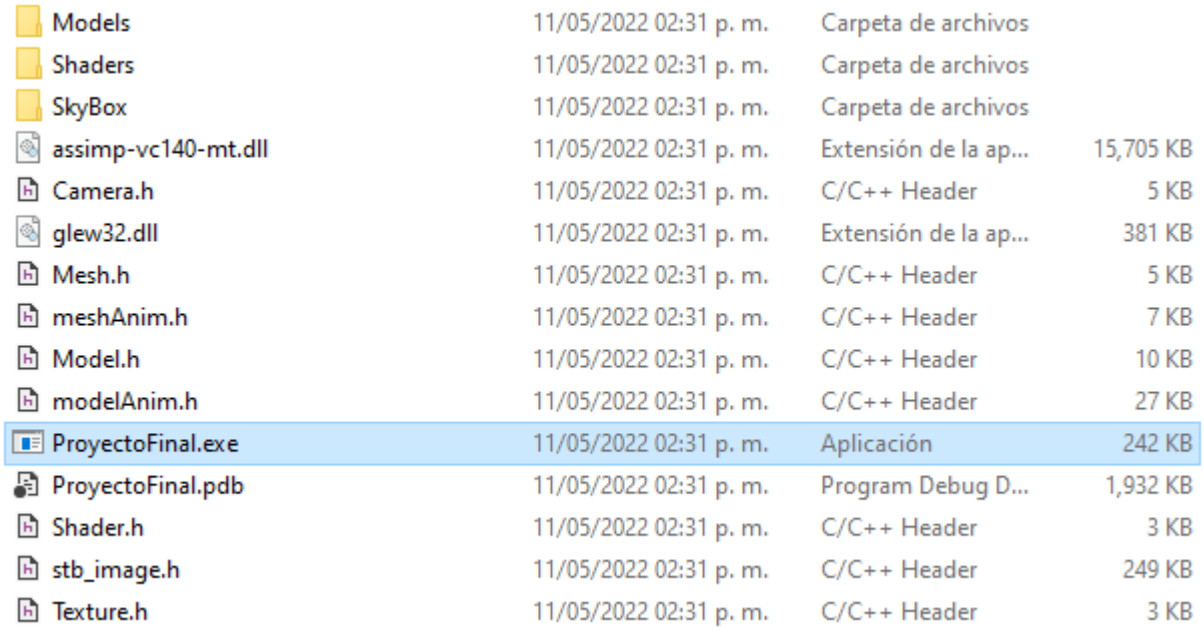

Imagen3. Explorador 3.

3.- Planeación del Proyecto y Diagrama de GANT

En este apartado se detallan las actividades que se hicieron para la realización del proyecto, en qué consisten, cuanto tiempo tomaron y cuál fue el resultado final.

Costo presupuestado: $ 30,000.00 MXN

3.1.- Actividad 1

Descripción: Propuesta de proyecto
Tiempo: Del 9 al 20 de marzo de 2022
Costo Final: $ 0.00 MXN

3.2.- Actividad 2

Descripción: Revisión y aprobación del proyecto
Tiempo: Del 21 al 27 de marzo de 2022
Costo Final: $ 0.00 MXN

3.3.- Actividad 3

Descripción: Búsqueda de modelos
Tiempo: Del 28 de marzo al 3 de abril de 2022
Costo Final: $ 9,000.00 MXN

3.4.- Actividad 4

Descripción: Realización de la geometría de los modelos
Tiempo: Del 4 al 17 de abril de 2022
Costo Final: $ 18,000.00 MXN

3.5.- Actividad 5

Descripción: Texturizado de los modelos
Tiempo: Del 18 al 24 de abril de 2020
Costo Final: $ 9,000.00 MXN

3.6.- Actividad 6

Descripción: Juntar modelos y darles proporción
Tiempo: Del 25 al 27 de abril de 2022
Costo Final: $ 500.00 MXN

3.7.- Actividad 7

Descripción: Animación 1
Tiempo: Del 28 de abril al 1 de mayo de 2022
Costo Final: $ 2,000.00 MXN

3.8.- Actividad 8

Descripción: Animación 2
Tiempo: Del 2 al 12 de mayo de 2022.
Costo Final: $ 2,000.00 MXN

3.9.- Actividad 9

Descripción: Animación 3
Tiempo: Del 12 al 15 de mayo de 2022
Costo Final: $ 2,000.00 MXN

3.10.- Actividad 10

Descripción: Animación 4
Tiempo: Del 16 al 18 de mayo de 2022
Costo Final: $ 2,000.00 MXN

3.11.- Actividad 11

Descripción: Animación 5
Tiempo: Del 19 al 20 de mayo de 2022
Costo Final: $ 2,000.00 MXN

3.12.- Actividad 12

Descripción: Animaciones extra
Tiempo: Del 4 al 20 de mayo de 2022
Costo Final: $ 2,000.00 MXN

3.13.- Actividad 13

Descripción: Pruebas y correcciones
Tiempo: 21 de mayo de 2022
Costo Final: $ 1,000.00 MXN

3.14.- Actividad 14

Descripción: Documentación
Tiempo: 22 de mayo de 2022
Costo Final: $ 2,000.00 MXN

3.15.- Diagrama de Gantt

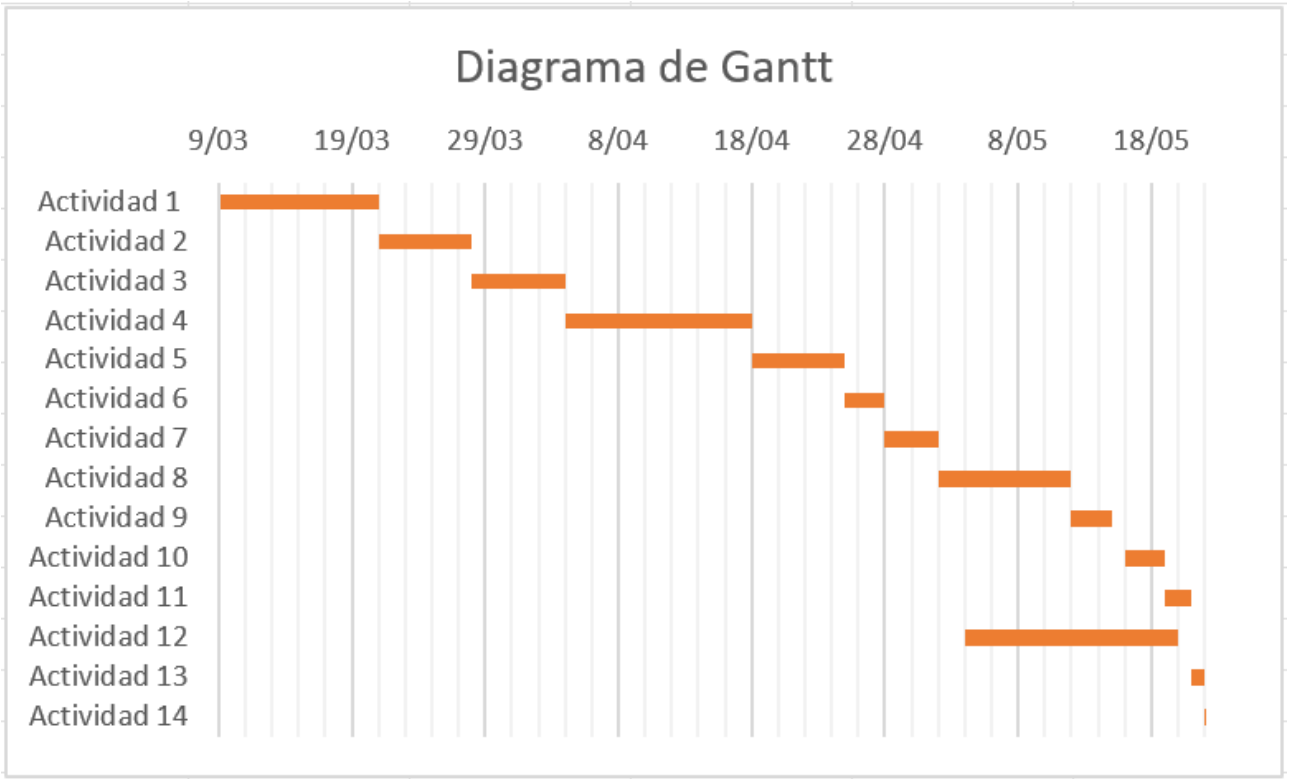

Imagen4. Diagrama de Gantt

3.16.- Costo Final del Proyecto

El proyecto tuvo un costo final de 51,500.00 MXN por lo que podemos concluir que se requirieron mas recursos debido a que se ocuparon mas modelos y animaciones extras de las que se tenían planeadas, sin embargo, cada una de las adiciones al presupuesto inicial fueron con el fin de mejorar el resultado final del proyecto.

4.- ALCANCE DEL PROYECTO

El proyecto tiene como objetivo final que se logre la implementación de un ambiente real en un ambiente virtual siguiendo las siguientes especificaciones.

- Se debe de entregar en tiempo y forma a más tardar el lunes 23 de mayo de 2022 a las 23:59 horas.
- Los objetos recreados repetidamente contaran como un objeto dentro de la evaluación
- Las animaciones realizadas deben tener contexto
- Para que una animación sea compleja no tiene que ser lineal, es decir, no solo basta con que se haga la transformación básica del proyecto
- Las actividades se deben realizar y entregar de acuerdo con lo estipulado en el apartado de planeación y diagrama de Gantt.

5.- Limitantes

A lo largo del desarrollo del proyecto se atravesaron algunas limitantes, las cuales se listan a continuación:

- Se presentaron problemas al momento de la programación del código, por lo que cuando estaba a punto de llegarse a una solución o correcta implementación de animación, se tuvo que comenzar con el trabajo prácticamente desde 0.
- El hardware de la computadora con el que fue desarrollado el proyecto si bien no es deficiente, tampoco es el de última generación, por lo que en ocasiones los tiempos de carga y respuesta de los modelos en Visual Studio llegaban a ser altos
- Algunos modelos obtenidos mediante las páginas indicadas al principio de este documento resultaron ser muy pesados, es decir, se componían de muchos polígonos, lo que hacía que el tiempo de carga del proyecto fuera alto. Ante esto, algunos modelos como el escritorio y la televisión tuvieron que ser remplazados por uno nuevo realizado desde 0 en el software de modelado MAYA.
- El tiempo con el que se contó para la terminación del proyecto final no permitió implementar algunas mejoras de interacción como lo pueden ser:
    - Limitar al usuario a que no pueda atravesar paredes o ir mas allá del entorno modelado

6.- Documentación del código

Uno de los requerimientos para la entrega del proyecto es que se use el código proporcionado a lo largo del semestre por el profesor, lo cual ayudó y facilitó mucho el desarrollo del proyecto. El presente apartado tiene como objetivo explicar en qué consiste cada parte del código del proyecto final.

6.1.- Carga de encabezados

La carga de archivos de encabezado es de suma importancia, en ella definimos las bibliotecas en lenguaje C++ que se van a utilizar, además de archivos de encabezado que nos sirven para poder cargar los shaders del proyecto

```
#include <iostream>
#include <cmath>

// GLEW
#include <GL/glew.h>

// GLFW
#include <GLFW/glfw3.h>

// Other Libs
#include "stb_image.h"
```

```
// GLM Mathematics
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtc/type_ptr.hpp>

//Load Models
#include "SOIL2/SOIL2.h"


// Other includes
#include "Shader.h"
#include "Camera.h"
#include "Model.h"
#include "Texture.h"
#include "modelAnim.h"
```

Imagen5. Codigo1

6.2.- Declaración de prototipos de funciones

En este apartado se cargan los prototipos de las funciones que ayudan a la interacción del usuario con el programa, es decir, aquellas en las que se puede interactuar con la cámara y los objetos del programa.

La función “KeyCallback” se encarga de recoger
una interacción en teclado, es decir, cada que el usuario presione una tecla, el programa tendrá la interacción que le sea indicada. Lo mismo sucede con la función “MouseCallback” cada que se mueva el ratón se interactuará con la cámara.

Las funciones DoMovement y animación ayudan a programar el código que realizará las animaciones, es decir, en ellas se dirán los circuitos de las animaciones complejas o se pondrán limites en las rotaciones o translaciones en las animaciones simples.

```
// Function prototypes
void KeyCallback(GLFWwindow *window, int key, int scancode, int action, int mode);
void MouseCallback(GLFWwindow *window, double xPos, double yPos);
void DoMovement();
void animacion();
```

Imagen6. Codigo2

6.3.- Dimensiones de la ventana

Este apartado de código únicamente define las dimensiones que tendrá la ventana ejecutable del programa. En este caso se optó por una resolución HD.

```
// Window dimensions
const GLuint WIDTH = 1080, HEIGHT = 720;
int SCREEN_WIDTH, SCREEN_HEIGHT;
```

Imagen7. Codigo3

6.4.- Cámara

Este es el apartado en donde se declara la posición inicial de la cámara.

```
// Camera
Camera  camera(glm::vec3(-100.0f, 4.5f, -25.0f));
GLfloat lastX = WIDTH / 2.0;
GLfloat lastY = HEIGHT / 2.0;
bool keys[1024];
bool firstMouse = true;
float range = 0.0f;
float rot = 0.0f;
```

Imagen8. Codigo4

6.5.- Definición de variables a utilizar en animaciones simples y complejas

Para la realización de las animaciones simples y complejas es necesario definir variables. Algunas son variables de activación, es decir, que cuando se presione una tecla cambie su estado de “False” a “True”, algunas otras son variables de incremento, las cuales nos ayudan para poder dar la animación de traslación o rotación de un objeto.

```
//CUELLO LARGO
bool animaCuello1 = true;
bool animaCuello2 = true;
bool animaCuello3 = false;
bool animaCuello4 = true;
bool animaCuello5 = false;
float rotLargoCuello = 0.0f;
float rotLargoCola = 0.0f;
```

Imagen9. Código 5

```
//MANOS PERSONAS GRADAS
bool animaMano1 = true;
bool animaMano2 = true;
bool animaMano3 = false;
float rotMano = 0.0f;
```

Imagen10. Codigo6

6.6.- Definición de variables de iluminación

Definen la posición de la luz, la posición inicial de los modelos y la dirección de la luz. Además, la posición de los puntos de Luz.

```
// Light attributes
glm::vec3 lightPos(0.0f, 0.0f, 0.0f);
glm::vec3 PosIni(-95.0f, 1.0f, -45.0f);
glm::vec3 lightDirection(0.0f, -1.0f, -1.0f);

// Positions of the point lights
glm::vec3 pointLightPositions[] = {
    glm::vec3(posX,posY,posZ),
    glm::vec3(0,0,0),
    glm::vec3(0,0,0),
    glm::vec3(0,0,0)
};

glm::vec3 LightP1;
```

Imagen11. Código7

6.7.- Carga de Modelos

Se cargan los modelos, la estructura es la misma para todos los modelos, pero cambia dependiendo de la carpeta en la que estén ubicados

```
// Carga de modelos CONSTRUCCIONES
Model Domo((char*)"Models/Construcciones/Domo.obj");
Model Lab((char*)"Models/Construcciones/Lab.obj");
Model Grada((char*)"Models/Construcciones/Grada.obj");
Model Arbol((char*)"Models/Construcciones/Arbol.obj");
Model Villas((char*)"Models/Villas/Villas.obj");
```

Imagen12. Codigo8.

6.8.- Carga de Shaders

Similar a la carga de modelos, pero con los archivos de Shaders.

```
Shader lightingShader("Shaders/lighting.vs", "Shaders/lighting.frag");
Shader lampShader("Shaders/lamp.vs", "Shaders/lamp.frag");
Shader lampShader2("Shaders/lamp2.vs", "Shaders/lamp2.frag");
Shader SkyBoxshader("Shaders/SkyBox.vs", "Shaders/SkyBox.frag");
Shader animShader("Shaders/anim.vs", "Shaders/anim.frag");
Shader Anim2("Shaders/anim2.vs", "Shaders/anim2.frag");
```

Imagen13. Codigo8.

6.9.- SkyBox

Se definen los vértices del Skybox, es decir, que tan grande será el cubo en el que estarán cargadas las texturas y se cargan las texturas del Skybox.

```
GLfloat skyboxVertices[] = {
    // Positions
    -1.0f,  1.0f, -1.0f,
    -1.0f, -1.0f, -1.0f,
    1.0f, -1.0f, -1.0f,
    1.0f, -1.0f, -1.0f,
    1.0f,  1.0f, -1.0f,
    -1.0f,  1.0f, -1.0f,
```

```
//SkyBox
GLuint skyboxVBO, skyboxVAO;
glGenVertexArrays(1, &skyboxVAO);
glGenBuffers(1,&skyboxVBO);
glBindVertexArray(skyboxVAO);
glBindBuffer(GL_ARRAY_BUFFER, skyboxVBO);
glBufferData(GL_ARRAY_BUFFER, sizeof(skyboxVertices),&skyboxVertices,GL_STATIC_DRAW);
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 3, GL_FLOAT,GL_FALSE, 3 * sizeof(GLfloat), (GLvoid *)0);

// Load textures
vector<const GLchar*> faces;
faces.push_back("SkyBox/right.tga");
faces.push_back("SkyBox/left.tga");
faces.push_back("SkyBox/top.tga");
faces.push_back("SkyBox/bottom.tga");
faces.push_back("SkyBox/back.tga");
faces.push_back("SkyBox/front.tga");
```

Imagen14. Codigo9

6.10.- Definición de Luces direccionales, puntos de luz y spotlights.

Se definen los tipos de luz que se pueden utilizar del lightingShader, los cuales son: Directional Light, Point Light y SpotLight.

```
// Directional light
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.direction"), -0.2f, -1.0f, -0.3f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.ambient"), 0.5f, 0.5f, 0.5f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.diffuse"), 0.4f, 0.4f, 0.4f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.specular"), 0.5f, 0.5f, 0.5f);

// Point light 1
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].position"), pointLightPositions[0].x, pointLightPositions[0].y,
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].ambient"), 0.05f, 0.05f, 0.05f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].diffuse"), LightP1.x, LightP1.y, LightP1.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].specular"), LightP1.x, LightP1.y, LightP1.z);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].constant"), 1.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].linear"), 0.09f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].quadratic"), 0.032f);
// SpotLight
glUniform3f(glGetUniformLocation(lightingShader.Program, "spotLight.position"), camera.GetPosition().x, camera.GetPosition().y, camera.GetPosition()
glUniform3f(glGetUniformLocation(lightingShader.Program, "spotLight.direction"), camera.GetFront().x, camera.GetFront().y, camera.GetFront().z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "spotLight.ambient"), 0.0f, 0.0f, 0.0f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "spotLight.diffuse"), 0.0f, 0.0f, 0.0f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "spotLight.specular"), 0.0f, 0.0f, 0.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "spotLight.constant"), 1.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "spotLight.linear"), 0.09f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "spotLight.quadratic"), 0.032f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "spotLight.cutOff"), glm::cos(glm::radians(12.5f)));
glUniform1f(glGetUniformLocation(lightingShader.Program, "spotLight.outerCutOff"), glm::cos(glm::radians(15.0f)));
```

Imagen15. Codigo10

6.11.- Dibujado de modelos

Hay varios tipos de dibujado de modelo, en sí la estructura es la misma, sin embargo, puede cambiar dependiendo de si el modelo tiene o no animación y en qué consista, si una traslación o rotación.

Dibujado sin animación:

```
view = camera.GetViewMatrix();
model = glm::mat4(1);
model = glm::translate(tmp, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::translate(model, glm::vec3(posX, posY, posZ));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(lightingShader.Program, "transparencia"), 0.0);
Grada.Draw(lightingShader);
```

Imagen16. Codigo11

Dibujado con animación:

```
view = camera.GetViewMatrix();
model = glm::mat4(1);
model = glm::translate(model, PosIni + glm::vec3(movKitX+38.594, 2.376, movKitZ+116.496));
model = glm::rotate(model, glm::radians(rotKitCoche1), glm::vec3(0.0f, 1.0f, 0.0));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(lightingShader.Program, "transparencia"), 0.0);
Coche.Draw(lightingShader);
```

Imagen17. Codigo12

6.12.- Función de animación

La estructura de esta función es simple, únicamente se crea un circuito con diferentes recorridos, en los cuales el objeto que se vea afectado podrá rotar y trasladarse dependiendo de los límites que se declaren en sus respectivas variables de traslación, rotación y limitación. A continuación, se muestra la función de animación con el circuito de animación compleja del coche.

```
void animacion()
{
    //Movimiento del coche
    if (circuito)
    {
        if (recorrido1)
        {
            movKitZ += 0.1f;
            if (movKitZ > 40)
            {
                recorrido1 = false;
                recorrido2 = true;
            }
        }
        if (recorrido2)
        {
            rotKit = 90;
            movKitX += 0.1f;
            if (movKitX > 40)
            {
                recorrido2 = false;
                recorrido3 = true;

            }
        }

        if (recorrido3)
        {
            rotKit = 180;
            movKitZ -= 0.1f;
            if (movKitZ < 10)
            {
                recorrido3 = false;
                recorrido4 = true;

            }
        }
        if (recorrido4)
        {
            rotKit = 270;
            movKitX -= 0.1f;
            if (movKitX < 0)
            {
                recorrido4 = false;
                recorrido5 = true;
            }
        }

        if (recorrido5)
        {
            rotKit = 180;
            movKitZ -= 0.1f;
            if (movKitZ < 0)
            {
                recorrido5 = false;
                recorrido6 = true;
            }
        }

        if (recorrido6)
        {
            rotKit = 0;
            movKitZ += 0.1f;
            if (movKitZ > 45)
            {
                recorrido6 = false;
                recorrido1 = true;
            }
        }

    }
```

Imagen18. Codigo13

6.13.- Función de KeyCallback

La estructura de esta función es la misma para tecla que se asigne. Dependiendo de la tecla que se presione se hará una acción distinta, a continuación, se muestran las teclas que se asignaron para activar y desactivar la animación compleja del coche

```
// Teclas animacion COMPLEJA COCHE
if (keys[GLFW_KEY_1]){ ... }

if (keys[GLFW_KEY_2]){ ... }

// Teclas animacion compleja PTERODACTILO
if (keys[GLFW_KEY_3]){ ... }

if (keys[GLFW_KEY_4]){ ... }
```

Imagen19. Codigo14.

6.14.- Función MouseCallback

La estructura de esta función es decirle al programa que con el ratón de la computadora se puede mover la cámara del programa.

```
void MouseCallback(GLFWwindow *window, double xPos, double yPos)
{

    if (firstMouse)
    {
        lastX = xPos;
        lastY = yPos;
        firstMouse = false;
    }

    GLfloat xOffset = xPos - lastX;
    GLfloat yOffset = lastY - yPos;  // Reversed since y-coordinates go from bottom to left

    lastX = xPos;
    lastY = yPos;

    camera.ProcessMouseMovement(xOffset, yOffset);
}
```

Imagen20. Codigo15.

6.15.- Función DoMovement

En esta función se pueden declarar teclas, es decir, se puede hacer lo mismo que en la función KeyCallback y además, se pueden hacer bloques de código que funcionan a partir de variables de activación que se mandar a llamar en la función KeyCallback

Ejemplo de definición de teclas nuevas y su acción (Cámara en este caso)

```
// Camera controls
if (keys[GLFW_KEY_W] || keys[GLFW_KEY_UP])
{
    camera.ProcessKeyboard(FORWARD, deltaTime);

}

if (keys[GLFW_KEY_S] || keys[GLFW_KEY_DOWN])
{
    camera.ProcessKeyboard(BACKWARD, deltaTime);


}

if (keys[GLFW_KEY_A] || keys[GLFW_KEY_LEFT])
{
    camera.ProcessKeyboard(LEFT, deltaTime);


}

if (keys[GLFW_KEY_D] || keys[GLFW_KEY_RIGHT])
{
    camera.ProcessKeyboard(RIGHT, deltaTime);
}
```

Imagen21. Codigo16

Ejemplo de la animación de un objeto (Puerta) mediante su variable de activación.

```
// ANIMACION PUERTA
if (animaPuerta1)
{
    if (rotPuerta <= 90)
    {
        rotPuerta += 0.5;
    }
}

if (animaPuerta2)
{
    if (rotPuerta >= 0)
    {
        rotPuerta -= 0.5;
    }
}
```

Imagen22. Codigo17

7.- Conclusiones generales

Se puede concluir que el proyecto final se realizó de manera adecuada. Se lograron implementar y poner en práctica los conocimientos teóricos adquiridos durante el curso. Se ocuparon temas como texturizado, iluminación, modelado jerárquico, transformaciones básicas y animación.

De la misma forma, el proyecto fue de ayuda para darnos cuenta de la planeación y los recursos que se necesitan en un proyecto de esta magnitud. Al inicio los costos de la planeación fueron desarrollados de una forma, contemplando ciertas condiciones, sin embargo, con forme se evolucionó en el proyecto nos dimos cuenta de que no se podía seguir avanzando de acuerdo con lo planeado, por lo que fue necesario el uso de más recursos para completar el proyecto de manera adecuada.