UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

LABORATORIO DE COMPUTACION GRAFICA E INTERACCION HUMANO COMPUTADORA

ROMAN BALBUENA CARLOS ALDAIR

316026703

MANUAL TECNICO PROYECTO FINAL

SEMESTRE 2022-2

11/05/2022

INDICE

OBJETIVOS

Aplicar y demostrar los conocimientos adquiridos durante todo el curso para recrear un escenario grafico animado en el software de Visual Studio.

INTRODUCCION

Para comprender como se realizó este proyecto se presenta a continuación un manual técnico donde primeramente se explican los requerimiento solicitados al inicio del proyecto para mostrar que no hubo confusión dentro de lo solicitado, y tener una base para planearnos objetivos claros que cumpliremos a lo largo de la aplicación del proyecto, de igual manera se explicara cada uno de los elementos utilizados para su realización, así como las herramientas y softwares de desarrollo donde a partir de un diccionario de variables y funciones podremos plasmar nuestras ideas planteadas.

REQUERIMIENTOS

Se deberá seleccionar una fachada y un espacio que pueden ser reales o ficticios y presentar imágenes de referencia de dichos espacios para su recreación 3D en OpenGL.

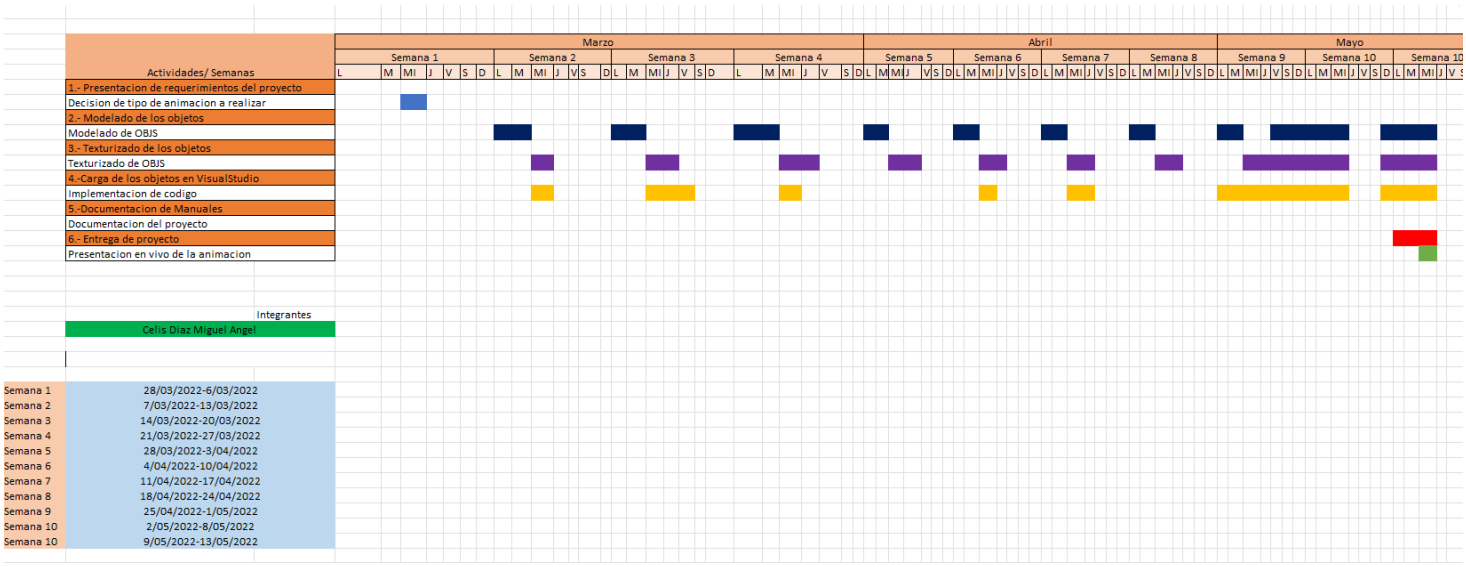
En la imagen de referencia se debe visualizar 7 objetos que se van a recrear virtualmente donde dichos objetos deben ser lo mas parecido a su imagen de referencia, así como su ambientación.

Se debe subir un documento pdf donde se muestre claramente su fachada y el cuarto a recrear, así como también un listado de los 7 objetos que se van a desarrollar dentro de dicho ambiente.

ALCANCES

Tomar como referencia la fachada de la caricatura “Coraje: El perro cobarde” asi como su entorno, e implementar las herramientas necesarias para poder recrearlo y tratar de hacerlo lo mas realista posible, de igual manera se hará una animación de algunos objetos seleccionados para justificar que el seguimiento de los temas presentados en el curso.

CRONOGRAMA



Actividades/ Semanas
1.- Presentacion de requerimientos del proyecto
Decision de tipo de animacion a realizar
2.- Modelado de los objetos
Modelado de OBJS
3.- Texturizado de los objetos
Texturizado de OBJS
4.-Carga de los objetos en VisualStudio
Implementacion de codigo
5.-Documentacion de Manuales
Documentacion del proyecto
6.- Entrega de proyecto
Presentacion en vivo de la animacion

Integrantes
Celis Diaz Miguel Angel

Semana 1	28/03/2022-6/03/2022
Semana 2	7/03/2022-13/03/2022
Semana 3	14/03/2022-20/03/2022
Semana 4	21/03/2022-27/03/2022
Semana 5	28/03/2022-3/04/2022
Semana 6	4/04/2022-10/04/2022
Semana 7	11/04/2022-17/04/2022
Semana 8	18/04/2022-24/04/2022
Semana 9	25/04/2022-1/05/2022
Semana 10	2/05/2022-8/05/2022
Semana 10	9/05/2022-13/05/2022

IMÁGENES DE REFERENCIA

Imagen de la fachada

Imagen del cuarto

Objetos recreados:

- Sofá
- Televisión
- Mecedora
- Lampara
- Reloj
- Alfombra
- Banco

HERRAMIENTAS DE DESARROLLO

- Maya AutoDesk 2022
- Visual Studio 2019
- Gimp 2.10.12
- Quixel Mixer

DICCIONARIO DE VARIABLES Y FUNCIONES

- Variables

Nombre	Tipo	Valor
movelightPosx	float	0.0f
movelightPosy	float	0.0f
movelightPosz	float	0.0f
rot	float	0.0f
activaanim	bool	False
rotPuerta	float	0.0f

rotacionPuerta	int	0
PuertaAbierta	bool	False
PuertaCerrada	bool	False
rotMolino	float	0.0f
rotacionMolino	int	0
GiroMolino	bool	False
rotMecedora	float	0.0f
rotacionMecedora	int	0.0f
Activo	bool	False
Movimiento	bool	False
movKitXNave	float	0.0
movKitYNave	float	0.0
movKitZNave	float	90.0
circuito	bool	False
circuito2	bool	False
recorrido1	bool	True
Recorrido2	bool	False
Recorrido3	bool	False
Recorrido4	bool	False
Recorrido5	bool	False
keys	bool	[1024]
firstMouse	bool	True
active	bool	-

- Funciones

```
// 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();
void animacion2();
```

Nombre	Descripción
KeyCallback	Es la función que hace interactuar al usuario con el ambiente, toma como entrada el teclado y a partir de la tecla presionada realiza una acción especifica
MouseCallback	Toma como entrada el movimiento del mouse para poder mover la cámara de nuestro espacio ambientado
DoMovement	Función donde se declaran las acciones que se realizaran después de llamar al KeyCallback
animacion	Función donde se explica el recorrido de nuestra primera animación para el platillo
animacion2	Función donde se explica el segundo recorrido de nuestra animación para el platillo

- Shaders

```
// Setup and compile our shaders
Shader shader("Shaders/modelLoading.vs", "Shaders/modelLoading.frag");
Shader lampshader("Shaders/lamp.vs", "Shaders/lamp.frag");
Shader lightingShader("Shaders/lighting.vs", "Shaders/lighting.frag");
```

Nombre	Descripción
lampshader	Dentro de este shader se encuentran dos archivos, el .vs y el .frag donde el vs se encarga de la información de los vértices y el frag de la información de los colores

- Modelos

```
// Load models

Model armchair((char*)"Models/Armchair/armchair.obj");
Model Lampara((char*)"Models/Lampara/Lampara.obj");
Model SueloExt((char*)"Models/SueloExterior/SueloTierra.obj");
Model TV((char*)"Models/TV/TV/TV.obj");
Model Fachada((char*)"Models/Fachada/Fachada.obj");
Model PuertaD((char*)"Models/Puerta/Puerta.obj");
Model MarcoPuertaD((char*)"Models/Puerta/MarcoPuerta.obj");
Model PuertaT((char*)"Models/Fachada/PuertaTrasera.obj");
Model Tubo((char*)"Models/Fachada/TuboFachada.obj");
Model Molino((char*)"Models/Molino/Molino.obj");
Model MolinoAnim((char*)"Models/Molino/MolinoCirculo.obj");
Model Escalera((char*)"Models/Escalera/Escalera.obj");
Model Alfombra((char*)"Models/Alfombra/Alfombra.obj");
Model Banco((char*)"Models/Banco/BancoMadera.obj");
Model Mecedora((char*)"Models/Mecedora/Mecedora.obj");
Model Reloj((char*)"Models/Reloj/Reloj.obj");
Model Telefono((char*)"Models/Telefono/Telefono.obj");
Model UFO((char*)"Models/UFO/UFO.obj");
Model Ventanas((char*)"Models/Ventanas/Ventanas.obj");
glm::mat4 projection = glm::perspective(camera.GetZoom(), (float)SCREEN_WIDTH / (float)SCREEN_HEIGHT, 0.1f, 400.0f);
```

Nombre	Direccion
armchair	Models/Armchair/armchair.obj
Lampara	Models/Lampara/Lampara.obj
SueloExt	Models/SueloExterior/SueloTierra.obj
TV	Models/TV/TV/TV.obj
Fachada	Models/Fachada/Fachada.obj
PuertaD	Models/Puerta/Puerta.obj
MarcoPuertaD	Models/Puerta/MarcoPuerta.obj
PuertaT	Models/Fachada/PuertaTrasera.obj
Tubo	Models/Fachada/TuboFachada.obj
Molino	Models/Molino/Molino.obj
MolinoAnim	Models/Molino/MolinoCirculo.obj
Escalera	Models/Escalera/Escalera.obj
Alfombra	Models/Alfombra/Alfombra.obj
Banco	Models/Banco/BancoMadera.obj
Mecedora	Models/Mecedora/Mecedora.obj
Reloj	Models/Reloj/Reloj.obj
Telefono	Models/Telefono/Telefono.ob
UFO	Models/UFO/UFO.obj
Ventanas	Models/Ventanas/Ventanas.obj"

CODIGO

Incluimos todas las librerías a utilizar para hacer todos los cálculos necesarios para la recreación de nuestro ambiente

```
// Std. Includes
#include <string>

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

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

#include "stb_image.h"

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

// Other Libs
#include "SOIL2/SOIL2.h"


// GL includes
#include "Shader.h"
#include "Camera.h"
#include "Model.h"
```

Declaramos nuestras funciones a utilizar junto con sus parámetros que necesitara, de igual manera ponemos el tamaño de la ventana a utilizar así como los valores de la cámara que nos ayudara a ver por completo nuestro ambiente.

```
// 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();
void animacion2();

// Properties
const GLuint WIDTH = 800, HEIGHT = 600;
int SCREEN_WIDTH, SCREEN_HEIGHT;



// Camera
Camera camera(glm::vec3(0.0f, 0.0f, 0.0f));
bool keys[1024];
GLfloat lastX = 400, lastY = 300;
bool firstMouse = true;
// Light attributes
glm::vec3 PosIni(0, 0, 0);
bool active;
```

Declaramos e inicializamos todas nuestras variables para las animaciones que implementaremos

```
// Light attributes
glm::vec3 lightPos(7.0f, 0.5f, 1.5f);
float movelightPosx = 0.0f;
float movelightPosy = 0.0f;
float movelightPosz = 0.0f;
GLfloat deltaTime = 0.0f;
GLfloat lastFrame = 0.0f;
float rot = 0.0f;
bool activanim = false;

//Variables para la animacion de la puerta
float rotPuerta = 0.0f;
int rotacionPuerta = 0;
bool PuertaAbierta = false;
bool PuertaCerrada = false;

//Variables para la animacion del molino
float rotMolino = 0.0f;
int rotacionMolino = 0;
bool GiroMolino = false;

//Variables para la animacion de la mecedora
float rotMecedora = 0.0f;
int rotacionMecedora = 0;
bool Activo = false;
bool Movimiento = false;
```

```
//Animación de la nave
float movKitXNave = 0.0;
float movKitZNave = 0.0;
float rotKitNave = 90.0;

//variables para controlar los estados de animacion.
bool circuito = false;
bool circuito2 = false;
bool recorrido1 = true;
bool recorrido2 = false;
bool recorrido3 = false;
bool recorrido4 = false;
bool recorrido5 = false;
```

Declaramos nuestros shaders para utilizarlos en la carga de modelos

```
// Setup and compile our shaders
Shader shader("Shaders/modelLoading.vs", "Shaders/modelLoading.frag");
Shader lampshader("Shaders/lamp.vs", "Shaders/lamp.frag");
Shader lightingShader("Shaders/lighting.vs", "Shaders/lighting.frag");
```

Para cargar todos los modelos le debemos de asignar un nombre y poner la dirección exacta de la carpeta donde se encuentran todos los archivos necesarios para la carga de los objetos.

```
// Load models

Model armchair((char*)"Models/Armchair/armchair.obj");
Model Lampara((char*)"Models/Lampara/Lampara.obj");
Model SueloExt((char*)"Models/SueloExterior/SueloTierra.obj");
Model TV((char*)"Models/TV/TV/TV.obj");
Model Fachada((char*)"Models/Fachada/Fachada.obj");
Model PuertaD((char*)"Models/Puerta/Puerta.obj");
Model MarcoPuertaD((char*)"Models/Puerta/MarcoPuerta.obj");
Model PuertaT((char*)"Models/Fachada/PuertaTrasera.obj");
Model Tubo((char*)"Models/Fachada/TuboFachada.obj");
Model Molino((char*)"Models/Molino/Molino.obj");
Model MolinoAnim((char*)"Models/Molino/MolinoCirculo.obj");
Model Escalera((char*)"Models/Escalera/Escalera.obj");
Model Alfombra((char*)"Models/Alfombra/Alfombra.obj");
Model Banco((char*)"Models/Banco/BancoMadera.obj");
Model Mecedora((char*)"Models/Mecedora/Mecedora.obj");
Model Reloj((char*)"Models/Reloj/Reloj.obj");
Model Telefono((char*)"Models/Telefono/Telefono.obj");
Model UFO((char*)"Models/UFO/UFO.obj");
Model Ventanas((char*)"Models/Ventanas/Ventanas.obj");
glm::mat4 projection = glm::perspective(camera.GetZoom(), (float)SCREEN_WIDTH / (float)SCREEN_HEIGHT, 0.1f, 400.0f);
```

Empezamos a utilizar el lampshader para la carga de los modelos y en todo caso si necesitamos hacer una transformación básica al objeto la realizamos en esta parte con sus respectivas funciones.

Cargamos todos los modelos a utilizar en conjunto de sus parámetros.

```
//Dibujamos todos los elementos de la fachada
view = camera.GetViewMatrix();

//Empezamos a usar el lampshader para la carga de los objetos
lampshader.Use();
modelLoc = glGetUniformLocation(lampshader.Program, "model");
viewLoc = glGetUniformLocation(lampshader.Program, "view");
projLoc = glGetUniformLocation(lampshader.Program, "projection");
glBindVertexArray(0);
glUniformMatrix4fv(viewLoc, 1, GL_FALSE, glm::value_ptr(view));
glUniformMatrix4fv(projLoc, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));

    //Para el Suelo
model = glm::mat4(1.0f);
//model = glm::translate(model, glm::vec3(lightPos.x + movelightPosx, lightPos.y + movelightPosy, lightPos.z + movelightPosz));
model = glm::scale(model, glm::vec3(20.0,0.0f,20.0f));
glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
glBindVertexArray(VAO);
SueloExt.Draw(lampshader);
glBindVertexArray(0);
```

```
        //Para la fachada
         model = glm::mat4(1.0f);
        //model = glm::translate(model, glm::vec3(4.0f,0.0f,0.0f));
        //model = glm::scale(model, glm::vec3(0.02f));
        glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
        glBindVertexArray(VAO);
        Fachada.Draw(lampshader);
        glBindVertexArray(0);

        //Para la puerta delantera

        //Para el marco de la puerta
            model = glm::mat4(1.0f);
            model = glm::translate(model, glm::vec3(-4.373f, 9.25f, 30.083));
             //model = glm::scale(model, glm::vec3(0.02f));
            glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
            glBindVertexArray(VAO);
            MarcoPuertaD.Draw(lampshader);
            glBindVertexArray(0);

        //Para la animacion de la puerta9

        model = glm::mat4(1.0f);
        model = glm::translate(model, glm::vec3(-4.32f,9.415f, 30.083));
        model = glm::rotate(model, glm::radians(rotPuerta), glm::vec3(0.0f, 1.0f, 0.0f));
        //model = glm::scale(model, glm::vec3(0.02f));
        glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
        glBindVertexArray(VAO);
        PuertaD.Draw(lampshader);
        glBindVertexArray(0);
```

```
    //Para la puerta trasera
    model = glm::mat4(1.0f);
    //model = glm::translate(model, glm::vec3(4.0f, 0.0f, 0.0f));
    //model = glm::scale(model, glm::vec3(0.02f));
    glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
    glBindVertexArray(VAO);
    PuertaT.Draw(lampshader);
    glBindVertexArray(0);

    //Para el tubo de la fachada

     model = glm::mat4(1.0f);
    //model = glm::translate(model, glm::vec3(4.0f, 0.0f, 0.0f));
    //model = glm::scale(model, glm::vec3(0.02f));
    glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
    glBindVertexArray(VAO);
    Tubo.Draw(lampshader);
    glBindVertexArray(0);

    //Para el molino
        //Para la base del molino
        model = glm::mat4(1.0f);
        //model = glm::translate(model, glm::vec3(4.0f, 0.0f, 0.0f));
        //model = glm::scale(model, glm::vec3(0.02f));
        glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
        glBindVertexArray(VAO);
        Molino.Draw(lampshader);
        glBindVertexArray(0);
```

```
            //Para la animacion del molino
            model = glm::mat4(1.0f);
            model = glm::translate(model, glm::vec3(74.645f, 85.051f, -82.005f));
            model = glm::rotate(model, glm::radians(rotMolino), glm::vec3(0.0f, 0.0f, 1.0f));
            //model = glm::scale(model, glm::vec3(0.02f));
            glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
            glBindVertexArray(VAO);
            MolinoAnim.Draw(lampshader);
            glBindVertexArray(0);
        //Dibujamos los elementos de la habitacion

            //Para el sillon
            model = glm::mat4(1.0f);
            //model = glm::translate(model, glm::vec3(4.0f, 0.0f, 0.0f));
            //model = glm::scale(model, glm::vec3(0.02f));
            glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
            glBindVertexArray(VAO);
            armchair.Draw(lampshader);
            glBindVertexArray(0);

            //Para la lampara
            model = glm::mat4(1.0f);
            //model = glm::translate(model, glm::vec3(4.0f, 0.0f, 0.0f));
            //model = glm::scale(model, glm::vec3(0.02f));
            glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
            glBindVertexArray(VAO);
            Lampara.Draw(lampshader);
            glBindVertexArray(0);
```

```
        //Para la Television

        model = glm::mat4(1.0f);
        //model = glm::translate(model, glm::vec3(4.0f, 0.0f, 0.0f));
        //model = glm::scale(model, glm::vec3(0.02f));
        glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
        glBindVertexArray(VAO);
        TV.Draw(lampshader);
        glBindVertexArray(0);

        //Para la escalera
        model = glm::mat4(1.0f);
        //model = glm::translate(model, glm::vec3(4.0f, 0.0f, 0.0f));
        //model = glm::scale(model, glm::vec3(0.02f));
        glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
        glBindVertexArray(VAO);
        Escalera.Draw(lampshader);


        //Para la alfombra

        model = glm::mat4(1.0f);
        //model = glm::translate(model, glm::vec3(4.0f, 0.0f, 0.0f));
        //model = glm::scale(model, glm::vec3(0.02f));
        glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
        glBindVertexArray(VAO);
        Alfombra.Draw(lampshader);
```

```
    //Para la animacion de la mecedora
    model = glm::mat4(1.0f);
    model = glm::translate(model, glm::vec3(0.0f, 2.0f, 18.0f));
    model = glm::rotate(model, glm::radians(-5.0f), glm::vec3(0.0f, 0.0f, 1.0f));
    model = glm::rotate(model, glm::radians(rotMecedora), glm::vec3(0.0f, 0.0f, 1.0f));
    //model = glm::scale(model, glm::vec3(0.02f));
    glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
    glBindVertexArray(VAO);
    Mecedora.Draw(lampshader);


    //Para el reloj
    model = glm::mat4(1.0f);
    //model = glm::translate(model, glm::vec3(4.0f, 0.0f, 0.0f));
    //model = glm::scale(model, glm::vec3(0.02f));
    glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
    glBindVertexArray(VAO);
    Reloj.Draw(lampshader);


    //Para la Nave
    model = glm::mat4(1);
    glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
    model = glm::translate(model, glm::vec3(movKitXNave, 0, movKitZNave));
    //model = glm::rotate(model, glm::radians(rotKitNave), glm::vec3(0.0f, 1.0f, 0.0));
    glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
    UFO.Draw(lampshader);
```

```
//Para el telefono

model = glm::mat4(1);
//model = glm::translate(model, glm::vec3(4.0f, 0.0f, 0.0f));
//model = glm::scale(model, glm::vec3(0.02f));
glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
glBindVertexArray(VAO);
Telefono.Draw(lampshader);
glBindVertexArray(0);

model = glm::mat4(1);
//model = glm::translate(model, glm::vec3(4.0f, 0.0f, 0.0f));
//model = glm::scale(model, glm::vec3(0.02f));
glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
glBindVertexArray(VAO);
Ventanas.Draw(lampshader);
glBindVertexArray(0);
```

En nuestra función DoMovement colocamos todas las acciones que va a realizar nuestro espacio siempre y cuando reciba una interacción con la función KeyCallback

Para este caso asignamos las teclas con las cuales nos moveremos dentro del espacio animado que serán: [W, A, S, D].

```
// Moves/alters the camera positions based on user input
void DoMovement()
{
    // 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);
    }
```

Describiremos el movimiento que realizaran cada una de las animaciones de nuestros objetos.

```
if (PuertaAbierta)
{
    if (rotPuerta <= 90.0f)
    {
        rotPuerta += 1.0f;
    }

    if (rotPuerta == 90.0f)
    {
        rotacionPuerta = 1;

    }
}
if (PuertaCerrada)
{
    if (rotPuerta >= 0.0f)
    {
        rotPuerta -= 1.0f;
    }

    if (rotPuerta == 0)
    {
        rotacionPuerta = 0;
    }
}
```

```
if (GiroMolino)
{

    if (rotMolino <= 360)
    {
        rotMolino += 1;
    }

    if (rotMolino == 360)
    {
        rotacionMolino = 1;
        rotMolino = 0;
    }

}

if (Movimiento) {
    rotMecedora += (Activo) ? -0.1f : 0.1f;
    if (rotMecedora > 21.0f) //Movimiento hacia atras
        Activo = true;
    if (rotMecedora < 0.0f) //Movimiento hacia adelante
        Activo = false;
}
if (keys[GLFW_KEY_B]) {
    Movimiento = true;
}

if (keys[GLFW_KEY_V]) {
    Movimiento = false;
}

if (keys[GLFW_KEY_I])
{
    circuito = true;
}

if (keys[GLFW_KEY_O])
{
    circuito = false;
    circuito2 = false;
}
if (keys[GLFW_KEY_X])
{
    circuito2 = true;
}
```

Nuestra función animación se encarga de realizar el primer circuito de la nave espacial alrededor de la casa, y la función animacion2 realizara el segundo circuito descrito de la nave espacial.

```
void animacion()
{
if (circuito)
{
    if (recorrido1)
    {
        movKitXNave += 1.0f;
        if (movKitXNave > 438)
        {
            recorrido1 = false;
            recorrido2 = true;
        }
    }
    if (recorrido2)
    {
        rotKitNave = -45;
        movKitZNave += 1.0f;
        movKitXNave -= 1.0f;
        if (movKitZNave > 90 && movKitXNave < 0)
        {
            recorrido2 = false;
            recorrido3 = true;

        }
    }
```

```
    if (recorrido3)
    {
        rotKitNave = -180;
        movKitZNave -= 1.0f;
        if (movKitZNave < 0)
        {
            recorrido3 = false;
            recorrido4 = true;
        }
    }
    if (recorrido4)
    {
        rotKitNave = 90;
        movKitXNave += 1.0f;
        if (movKitXNave > 0)
        {
            recorrido4 = false;
            recorrido1 = true;
        }
    }
```

```
void animacion2()
{

    //Movimiento del coche
    if (circuito2)
    {
        if (recorrido1)
        {
            movKitXNave += 1.0f;
            if (movKitXNave > 90)
            {
                recorrido1 = false;
                recorrido2 = true;
            }
        }
        if (recorrido2)
        {
            rotKitNave = 0;
            movKitZNave += 1.0f;
            if (movKitZNave > 90)
            {
                recorrido2 = false;
                recorrido3 = true;

            }
        }
```

```
        if (recorrido3)
        {
            rotKitNave = -90;
            movKitXNave -= 1.0f;
            if (movKitXNave < 0)
            {
                recorrido3 = false;
                recorrido4 = true;
            }
        }

        if (recorrido4)
        {
            rotKitNave = -180;
            movKitZNave -= 1.0f;
            if (movKitZNave < 0)
            {
                recorrido4 = false;
                recorrido5 = true;
            }
        }
        if (recorrido5)
        {
            rotKitNave = 90;
            movKitXNave += 1.0f;
            if (movKitXNave > 0)
            {
```

```
    if (recorrido5)
    {
        rotKitNave = 90;
        movKitXNave += 1.0f;
        if (movKitXNave > 0)
        {
            recorrido5 = false;
            recorrido1 = true;
        }
    }


}
```

Por último, en nuestra función KeyCallback daremos paso a la animación a realizar oprimiendo la tecla que se configuro.

```
    if (keys[GLFW_KEY_P])
    {
        if (rotacionPuerta == 0)
        {
            PuertaAbierta = true;
            PuertaCerrada = false;


        }

        if (rotacionPuerta == 1)
        {
            PuertaAbierta = false;
            PuertaCerrada = true;



        }

    }
```

```
    if (keys[GLFW_KEY_M])
    {
        if (rotacionPuerta == 0)
        {
            GiroMolino = true;


        }

        if (rotacionPuerta == 1)
        {
            GiroMolino = true;



        }

    }

    if (keys[GLFW_KEY_L])
    {
        GiroMolino = false;
    }
```

```
    if (keys[GLFW_KEY_B])
    {

            Movimiento = true;
            //MecedoraAtras = true;
            //MecedoraAdelante = false;


    }

        if (keys[GLFW_KEY_V])
        {
            Movimiento = false;
            //MecedoraAtras = false;
            //MecedoraAdelante = true;



        }

}
```

CONCLUSIONES

Al termino de este proyecto logre aplicar todos los conocimientos obtenidos a lo largo del semestre, de igual manera fue muy interesante conocer las bases que se necesitan para las animaciones que observamos en las películas y como es el proceso que conlleva hacerlas, por ultimo se puede concluir que cumplimos satisfactoriamente con el objetivo planteado al inicio de este proyecto.