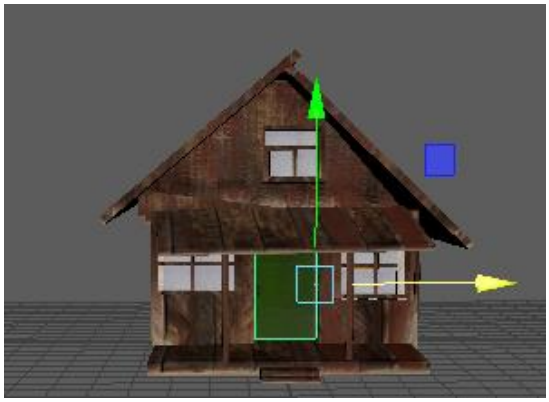
Universidad Nacional Autónoma de México
Facultad de ingeniería

Computación gráfica e interacción humano computadora

Proyecto final

Profesor: Carlos Aldair Román Balbuena

Grupo 4



Pérez Dublán Juan Pablo
Número de cuenta:315086052

Fecha de entrega: 21 de noviembre del 2021

Tabla de contenido

Objetivo

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.

Introducción

Coraje, el perro cobarde, era una caricatura la cual se estrenó el 12 de noviembre de 1999, es una serie original de Cartoon Network, creada por John R Dilwort, esta serie tenía una duración de aproximadamente 22 minutos cada capítulo, fueron creadas cuatro temporadas con un total de 43 episodios. Era una serie con tres personajes principales, Coraje que era el perro, Muriel y Justo, que eran los dueños. Básicamente era una serie la cual trataba sobre un perro que le temía a todo, pero al final siempre lograba enfrentar sus miedos para salvar a los dueños que tanto amaba. Tristemente esta caricatura fue cancelada en noviembre del 2002, algunas fuentes mencionan que fue cancelada porque algunos padres no les gustaba. pero siempre será una serie que a muchos nos recuerde nuestra infancia y que a final de cuentas siempre nos enseñaba algo, esto es que siempre enfrentemos nuestros miedos, sin importar que tan difícil sea para nosotros.

Alcance

Crear la fachada de una casa la cual sea casi una réplica de la original. También se busca crear un espacio o cuarto que este dentro de esta casa. Además, dentro de este cuarto debe haber siete objetos tres de ellos con animaciones sencillas y dos de ellos con animaciones complejas. En este proyecto en especial se busca una réplica exacta, pero con estilo realista.

Manual de usuario

Como podemos observar al compilar el programa nos arroja en la pantalla la casa del perro cobarde para poder observar de una buena manera la casa tenemos que usar las flechas y el mouse

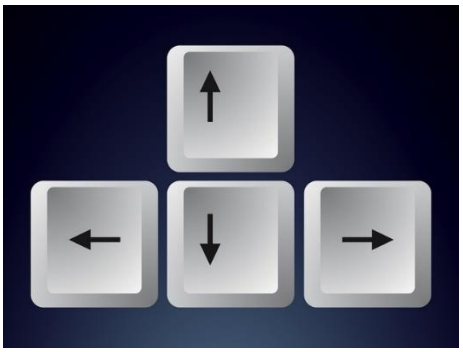


Imagen 1. Flechas de teclado

- Con la flecha de abajo podremos retroceder la pantalla y tener una visión completa de la casa
- Con la flecha de arriba podemos ir hacia delante de esta manera podremos adentrarnos a la casa.
- Con la flecha izquierda y derecha podremos movernos a sus respectivos lados.

Imagen 2 y 3. Mouse de laptop y mouse de computadora de escritorio

- Con el mouse de la pantalla podremos tener un mejor enfoque del de la casa

Ahora bien, se puede usar la combinación del mouse con las teclas para visualizar de una mejor manera la casa

Imagen 4. Fachada de la casa

De primera instancia podemos observar la facha de frente de la casa, tiene un texturizado de madera, cuanta con tres ventanas y una puerta, del costado derecho cuenta igual con par de ventanas. Para poder ingresar al primer piso de la casa podemos oprimir la letra “o” del teclado, de esta forma se abrirá la puerta. En la siguiente imagen podemos observar un ejemplo de esto. Incluso ya se puede ver algunos objetos dentro de la casa.

I*magen 5. Fachada con puerta abierta*

Imagen 6. Inteiror de la casa

Como se mencionó anteriormente, nos adentramos a la casa con la flecha de arriba del teclado, de esta manera ya se logra visualizar el interior de la casa y no solo eso, podemos obserbar nuestros 7 objetos

Objeto 1: El sofa de Justo. Este objeto fue creado por mi en maya

Objeto 2. El tapete. Este objeto fue creado y texturizado por mi.

Objeto 3. La mesa de madera. Este objeto fue comprado en Turbosquid por 3 dolares. Fue texturizado y editado por mi

Objeto 4. Lampara. Es un objeto descargado en turbosquid. Editado y texturizado pro mi

Objeto 5. Silla mesedora de Muriel. Esta silla fue creada y texturizada por mi en maya

Objeto 6. Cuadro. Este cuadro fue creado y texturizado por mi.

Objeto 7. Reloj antiguo. Este objeto fue descargado. Fue editado y texturizado por mi.

Objeto 8. Espejo. Este objeto no se logró realizar de una manera adecuada ya que el texturizado se pierde. Pero fue creado y elaborado por mi.

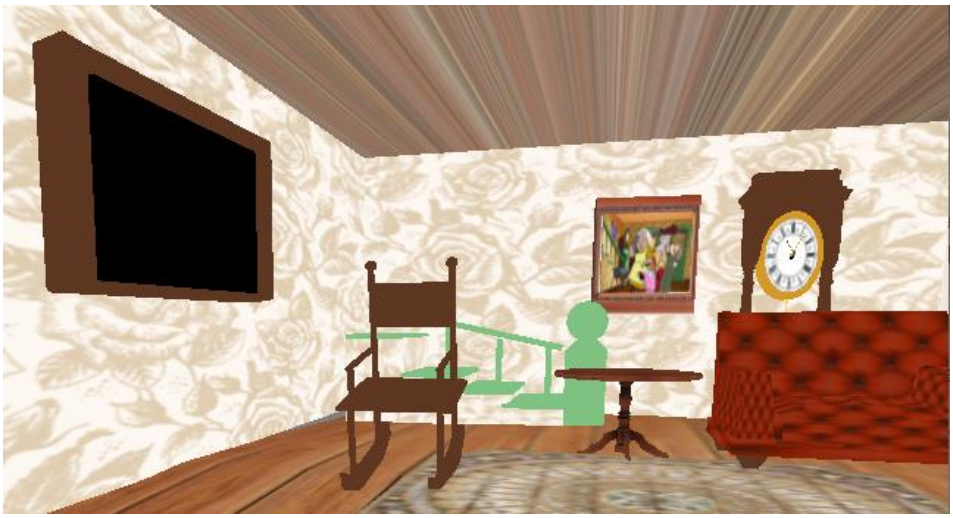


Imagen 7. Silla mesedora

El objeto 5 es otra animación, esta simula su función que es meser. Con la tecla “C” logramos simular esta acción.

El objeto 7 tiene otra animación, la cual tiene un “edgar” el cual se logra animar con la tecla “V”

Diagrama de Gantt

El diagrama este hecho en semanas ya que fue un periodo de mas de un mes. Las ultimas semanas se puede observar que se crearon más objetos y se realizaron mas actividades, esto no quiere decir que fue menos complejo, si no que se le dedico más horas de trabajo las ultimas semanas

Rubro\semana	Semana1	Semana2	Semana 3	Semana 4	Semana 5	Semana 6
Mueble 1						
Mueble 2						
Mueble 3						
Mueble 4						
Mueble 5						
Mueble 6						

Mueble 7						
Fachada						
Animación 1						
Animación 2						
Animación 3						
Adaptación del código						
Manual de usuario						
Reporte del proyecto						
Entrega						

Tabla 1. Diagrama de Gantt

Limitantes.

1.-Tiempo. Para este proyecto aun que se buscó el tiempo suficiente para realizarlo, no se pudo terminar las 5 animaciones.

2.- Conocimientos. Aun que se logró hacer varios ejemplos en Maya para realizar objetos y texturizarlos, no son suficientes en el curso para poder tener un conocimiento amplio de modelado en Maya.

3.- Comprensión. Aun que se tuvo el apoyo del profesor en todo momento, programar o analizar el código en C++ resultó un tanto complejo.

4.- Objetos. Los objetos nos se prestaban como para crear animaciones con sentido.

Análisis financiero del proyecto

En la siguiente tabla se busca hacer un análisis financiero, para tener una dimensión de cuanto costó elaborar el proyecto y cuanto valor tiene.

En actividad se mencionará cada objeto que se realizó y las animaciones que se crearon. En costo viene el precio que tuvo el objeto (si es el caso). En tiempo requerido se pondrán las horas o minutos que se llevó realizar el objeto o la animación. Los objetos que se descargaron se tomaran en cuenta el tiempo de búsqueda, el tiempo de adaptación y el tiempo de texturizado. El precio final, es el preció que se otorga tomando en cuenta las columnas anteriores. Las animaciones de igual manera tienen un precio, debido a que también requirió tiempo y conocimientos realizarlas. Por último cabe destacar que en adaptación es por la

parte de trasladar y acomodar a los objetos dentro de la casa.Los precios se tomarán en cuenta con pesos mexicanos.

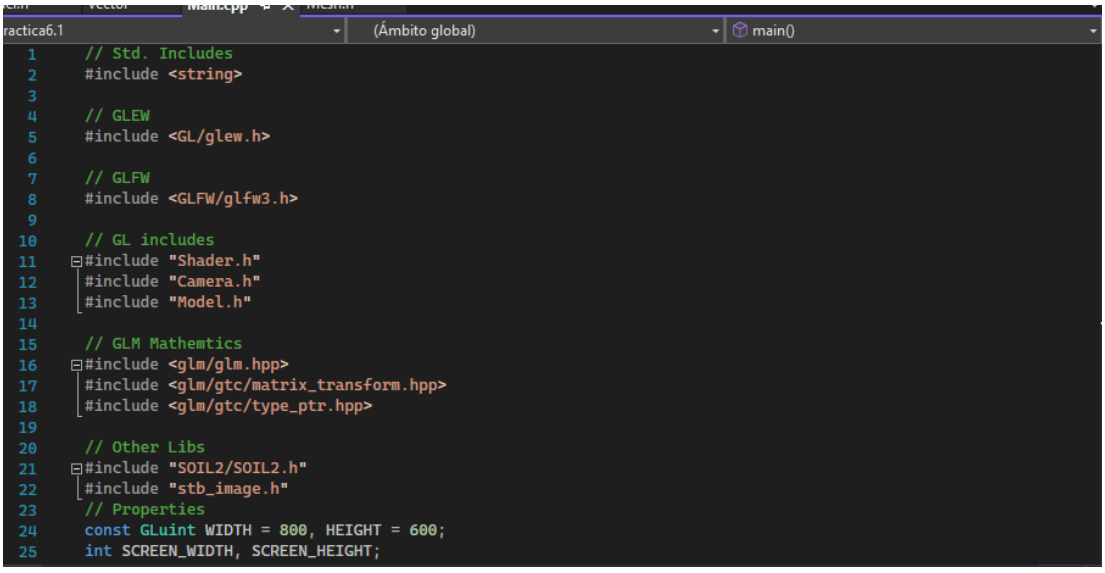
Análisis financiero			
Actividad	Costo	Tiempo requerido	Precio final
sofá	$0.00	5 hrs	$300.00
Silla	$0.00	4hrs	$200.00
Mesa	$60.00	2 hrs	$300.00
Lampara	$0.00	2 hrs	$200.00
Reloj	$0.00	4hrs	$200.00
Cuadro	$0.00	60 min	$150.00
Tapete	$0.00	60 min	$120.00
Fachada	$0.00	8hrs	$2500.00
Animación 1	$0.00	5 hrs	$200.00
Animación 2	$0.00	3 hrs	$200.00
Animación 3	$0.00	60 min	$200.00
Adaptación	$0.00	2 hrs	$500.00
Total	-	-	$5070.00

Tabla 2. Análisis de costos

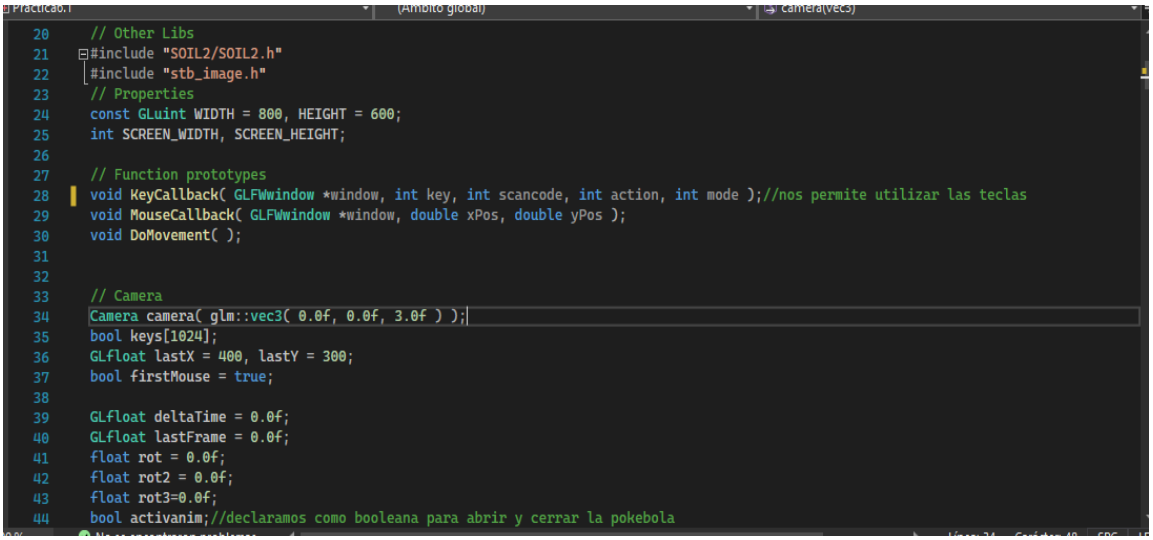
A parte de las horas y actividades plasmadas en el análisis del financiero se requirieron de más horas para algunas variantes del proyecto. Por otro lado, el precio de cada objeto son basados en páginas como turbosquid.

Documentación del código

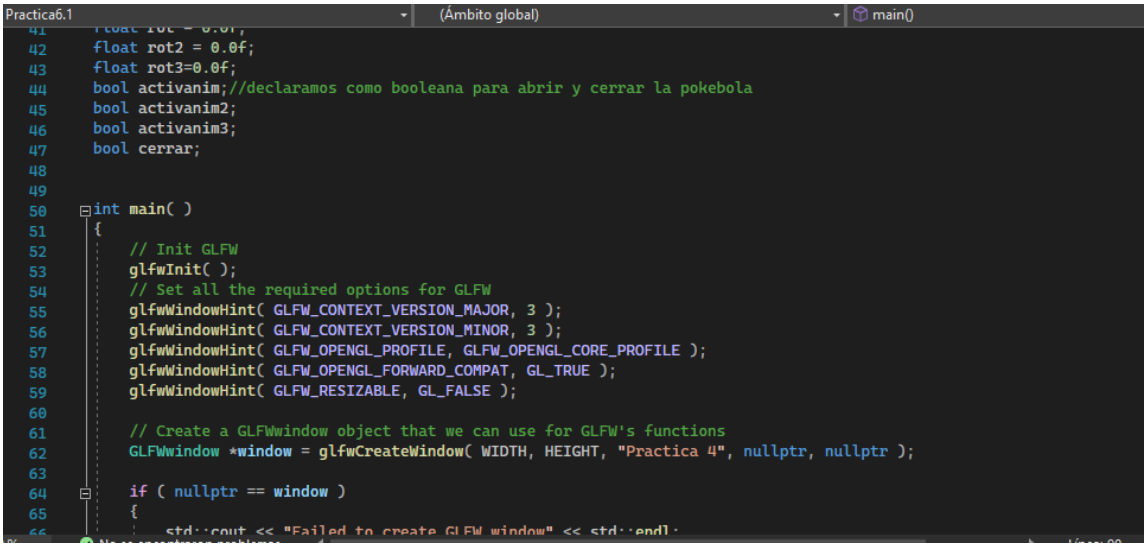
Para este proyecto se utilizó el código base de la práctica número 6 del presente semestre (2022-1).



En esta parte del código tenemos las bibliotecas de Glopen que utilizamos durante el semestre.

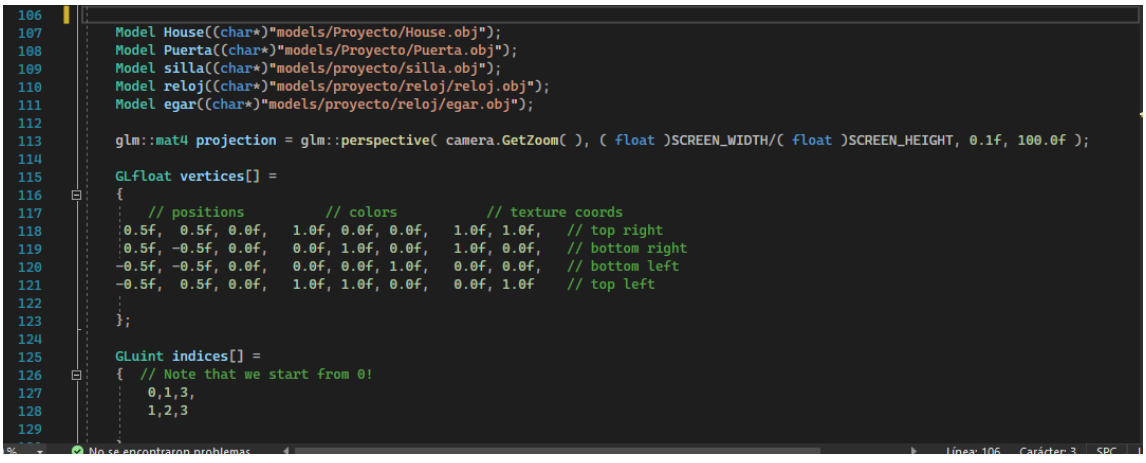


Aquí cabe destacar las funciones que declaramos, que son vitales para el funcionamiento del programa, que son keyCallback, MouseCallback y Domovement, estas funciones nos permiten usar las teclas y el mouse para el espacio virtual. Por otro lado, se encuentran las variables flotantes de “rot” y de “activanim”, estas variables funcionan para crear las animaciones.




```
float rot2 = 0.0f;
float rot3=0.0f;
bool activanim;//declaramos como booleana para abrir y cerrar la pokebola
bool activanim2;
bool activanim3;
bool cerrar;


int main( )
{
    // Init GLFW
    glfwInit( );
    // Set all the required options for GLFW
    glfwWindowHint( GLFW_CONTEXT_VERSION_MAJOR, 3 );
    glfwWindowHint( GLFW_CONTEXT_VERSION_MINOR, 3 );
    glfwWindowHint( GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE );
    glfwWindowHint( GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE );
    glfwWindowHint( GLFW_RESIZABLE, GL_FALSE );

    // Create a GLFWwindow object that we can use for GLFW's functions
    GLFWwindow *window = glfwCreateWindow( WIDTH, HEIGHT, "Practica 4", nullptr, nullptr );

    if ( nullptr == window )
    {
        std::cout << "Failed to create GLFW window" << std::endl;
```

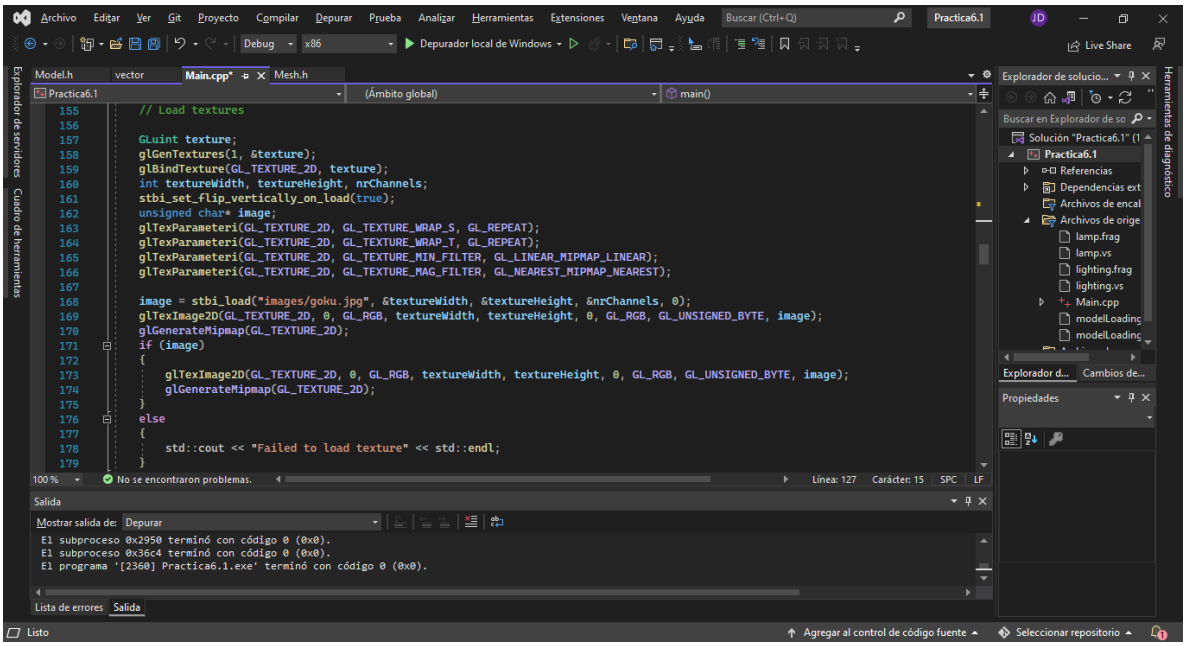
En esta parte del código tenemos el set de opciones para GLFW



```
Model House((char*)"models/Proyecto/House.obj");
Model Puerta((char*)"models/Proyecto/Puerta.obj");
Model silla((char*)"models/proyecto/silla.obj");
Model reloj((char*)"models/proyecto/reloj/reloj.obj");
Model egar((char*)"models/proyecto/reloj/egar.obj");

glm::mat4 projection = glm::perspective( camera.GetZoom( ), ( float )SCREEN_WIDTH/( float )SCREEN_HEIGHT, 0.1f, 100.0f );

GLfloat vertices[] =
{
    // positions          // colors           // texture coords
 0.5f,  0.5f, 0.0f,   1.0f, 0.0f, 0.0f,   1.0f, 1.0f,   // top right
 0.5f, -0.5f, 0.0f,   0.0f, 1.0f, 0.0f,   1.0f, 0.0f,   // bottom right
-0.5f, -0.5f, 0.0f,   0.0f, 0.0f, 1.0f,   0.0f, 0.0f,   // bottom left
-0.5f,  0.5f, 0.0f,   1.0f, 1.0f, 0.0f,   0.0f, 1.0f    // top left

};

GLuint indices[] =
{  // Note that we start from 0!
    0,1,3,
    1,2,3
```

Para esta parte del código, cargamos los modelos, en este caso dentro del modelo con nombre “house” se cargó la fachada con varios objetos, ya que variós de los modelos no requerían de una animación. La puerta, la silla se cargarón a parte porque son los primeros modelos a animar, después se cargo el reloj y luego “edgar”

egar es el modelo o el objeto el cual tendrá movimiento con respecto al reloj. Ádemas en esta parte del código tenemos la projeccion en perspectiva, que es la forma en la que visualizaremos los modelos.



```
// Load textures

GLuint texture;
glGenTextures(1, &texture);
glBindTexture(GL_TEXTURE_2D, texture);
int textureWidth, textureHeight, nrChannels;
stbi_set_flip_vertically_on_load(true);
unsigned char* image;
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST_MIPMAP_NEAREST);

image = stbi_load("images/goku.jpg", &textureWidth, &textureHeight, &nrChannels, 0);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, textureWidth, textureHeight, 0, GL_RGB, GL_UNSIGNED_BYTE, image);
glGenerateMipmap(GL_TEXTURE_2D);
if (image)
{
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, textureWidth, textureHeight, 0, GL_RGB, GL_UNSIGNED_BYTE, image);
    glGenerateMipmap(GL_TEXTURE_2D);
}
else
{
    std::cout << "Failed to load texture" << std::endl;
}
```

Esta parte es muy importante ya que nos va a permitir cargar las texturas de nuestros modelos.

```
glm::mat4 view = camera.GetViewMatrix();
glUniformMatrix4fv(glGetUniformLocation(shader.Program, "projection"), 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(glGetUniformLocation(shader.Program, "view"), 1, GL_FALSE, glm::value_ptr(view));

// Transladar y rotar puerta
glm::mat4 model(1);
model = glm::translate(model, glm::vec3(1.0f, 2.0f, 4.0f));//translate para puerta
model = glm::rotate(model, glm::radians(-rot), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(glGetUniformLocation(shader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
Puerta.Draw(shader);

// translate y rotate silla
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(-1.5f, 0.5f, -1.5f));//translate para silla
model = glm::rotate(model, glm::radians(rot2), glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(glGetUniformLocation(shader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
silla.Draw(shader);

//translate  de reloj
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(0.5f, 0.5f, -1.7f));//translate para silla
glUniformMatrix4fv(glGetUniformLocation(shader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));
reloj .Draw(shader);
//translate y rotate de egar
```

Aquí es donde se empieza a manipular el modelo, ya que por ser varios objetos tridimensionales se tuvieron que cargar por separado, a de más de que de esta forma se puedierón realizar las animaciones. En casi todos los objetos primero se

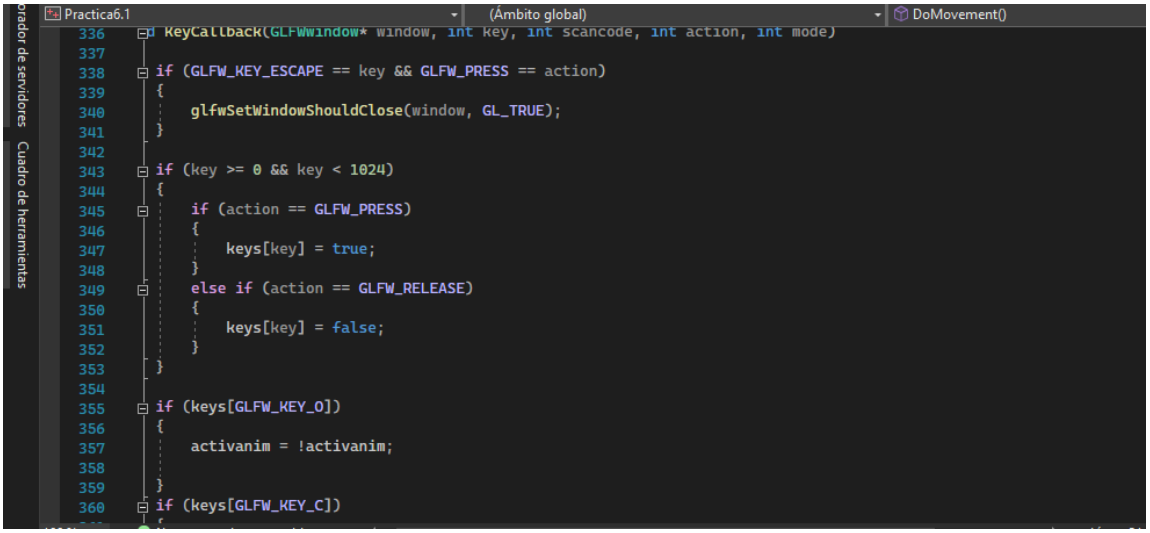
setea la matriz, se translada el objeto y se rota, en este caso la rotación es la animación. Estas animaciones se llevarón acabo moviendo el centro del modelo en maya y adaptandolo a su eje corriespodiente aquí en el código.

```
// Moves/alters the camera positions based on user input
// 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);
    }
```

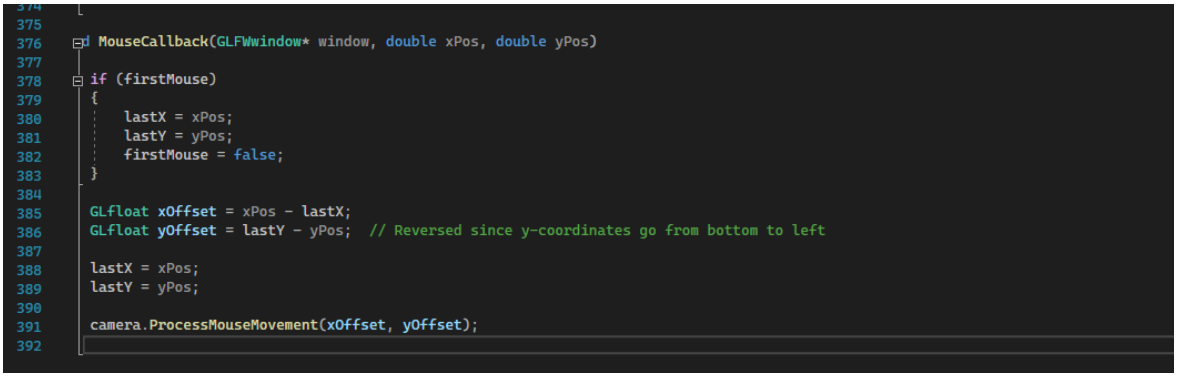
El DoMovement como ya lo había mencionado nos va permitir movernos en nuestro ambiente, no se menciona en el manual de usuario ya que su función es similar a las flechas.

```
Practicab1                (Ámbito global)                DoMovement()
}
else if (!activanim)
{
    if (rot > 0)
    {
        rot -= 0.1f;
    }

}
if (activanim2)//animación para la silla
{
    if (rot2 < 10)// se rota solo 10 grados ya que no requiere de mucho movimiento
    {
        rot2+= 0.1f;
    }
}
else if (!activanim2)
{
    if (rot2 > 0)
    {
        rot2-= 0.1f;
    }

}
if (activanim3)// animación para el "edgar" del reloj, esta se consiera una rotación muy pequeña, debido a la dimensión del reloj
No se encontraron problemas.                Línea: 315   Carácter: 10   SPC
```

Estas condicionales son las que permiten crean las animaciones, se utilizara la rotación en grados y cada rotación varía dependiendo el objeto.



Esta es la penúltima parte del código es donde se asignan las teclas en la función de KeyCallbak, las teclas que se utilizadas se mencionan en el manual de usuario.



Y esta última parte es la función que permite el uso del mouse para movernos mejor.

Diccionario de funciones y variables

Variable o función	Descripción
`Model House((char*)"models/Proyecto/House.obj");`	Carga el modelo de la casa. Aquí se carga la fachada pero de igual manera se cargan varios objetos dentro de ella.
`Model Puerta((char*)"models/Proyecto/Puerta.obj");`	Cargamos la puerta a parte ya que es la que va a tener la primera animación
`Model silla((char*)"models/proyecto/silla.obj");`	De igual manera cargamos la silla a parte porque es la que va a tener la segunda animación
`Model reloj((char*)"models/proyecto/reloj/reloj.obj");`	Carga del reloj

`Model egar((char*)"models/proyecto/reloj/egar.obj");`	Este modelo es parte del reloj, pero es la parte que va a tener la tercer animación
`glm::mat4 projection = glm::perspective( camera.GetZoom( ), ( float )SCREEN_WIDTH/( float )SCREEN_HEIGHT, 0.1f, 100.0f );`	Cargamos la proyección en pespectiva
`float rot = 0.0f;`	Esta variable recibe parámetros flotantes, ya que va a ser la variable de rotación para la animación
`float rot2 = 0.0f;`	La variable rot2 recibe parámetros flotantes, va a ser la segunda variable de rotación
`float rot3=0.0f;`	Tercera variable de rotación de igual manera recibe parámetros flotantes
`bool activanim;`	La variable activeanim res la variable que se utiliza para activar o desactivar la animación de la puerta
`bool activanim2;`	Misma función de activanim2 pero esta es para la animación de la silla
`bool activanim3;`	Variable que recibe parámetros booleanos, sirve para activar

`glm::mat4 model(1);`
`model = glm::translate(model, glm::vec3(1.0f, 2.0f, 4.0f));//translate para puerta`
`model = glm::rotate(model, glm::radians(-rot), glm::vec3(0.0f, 1.0f, 0.0f));`
 	
`glUniformMatrix4fv(glGetUniformLocation(shader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));`
`Puerta.Draw(shader);`
	En esta función es donde se setea la matriz, aquí es donde se nos permite cargar el modelo de puerta, además de que ya se le asignan la variable “rot”, y se traslada el objeto para que este donde debe estar. La variable “rot” en esta ocasión se esta rotando en Y

`model = glm::mat4(1);`
`model = glm::translate(model, glm::vec3(-1.5f, 0.5f, -1.5f));//translate para silla`
`model = glm::rotate(model, glm::radians(rot2), glm::vec3(1.0f, 0.0f, 0.0f));`
 	
`glUniformMatrix4fv(glGetUniformLocation(shader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));`
`silla.Draw(shader);`
	Se setea la matriz y se dibuja la silla, para esta función ocupamos la segunda variable de rotación “rot2” y se utiliza el eje x para la rotación

`model = glm::mat4(1);`
`model = glm::translate(model, glm::vec3(0.5f, 0.5f, -1.7f));//translate para silla`
 	
`glUniformMatrix4fv(glGetUniformLocation(shader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));`
`reloj .Draw(shader);`
	Se setea la matriz y se dibuja el reloj esta función no tiene variable de rotación, pero si se traslada, esto es debido a que se cargó a parte

`model = glm::mat4(1);`
`model = glm::translate(model, glm::vec3(0.5f, 0.5f, -1.7f));//translate para silla`
`model = glm::rotate(model, glm::radians(rot3), glm::vec3(0.0f, 1.0f, 0.0f));`
 	
`glUniformMatrix4fv(glGetUniformLocation(shader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));`
`egar.Draw(shader);`
	Se setea la matriz y se dibuja ”egar” esta función si tiene la variable rot3 ya que es la tercer animación y va a rotar en el eje Y

`model = glm:: mat4(1);`
 	
`glUniformMatrix4fv(glGetUniformLocation(shader.Program, "model"), 1, GL_FALSE, glm::value_ptr(model));`
`House.Draw(shader);`
	Esta función es la ultima matriz que se setea, pero es la más importante ya que es donde se dibuja la casa y 5 objetos dentro de ella. Los cuales son El sofá, la lampara, el tapete, el cuadro, el espejo, la mesa.

`if (activanim)//animación para puerta`
`{`
`if (rot < 90)// se consiedera la apertura regular o promedio de una puerta como 90 °`
`{`
`rot += 0.1f;`
`}`
`}`
`else if (!activanim)`
`{`
`if (rot > 0)`
`{`
`rot -= 0.1f;`
`}`
	Estas condicionales reciben los parámetros booleanos de activeanime1, esta función nos va a permitir realizar la animación

`if (activanim2)//animación para la silla`
`{`
`if (rot2 < 10)// se rota solo 10 grados ya que no requiere de mucho movimiento`
`{`
`rot2+= 0.1f;`
`}`
`}`
`else if (!activanim2)`
`{`
`if (rot2 > 0)`
`{`
`rot2-= 0.1f;`
`}`
	Esta función nos permite el correcto funcionamiento de activanim2

`if (activanim3)// animación para el "edgar" del reloj, esta se consiera una rotación muy pequeña, debido a la dimensión del reloj`
`{`
`if (rot3 < 1)`
`{`
`rot3 += 0.1f;`
`}`
`}`
`else if (!activanim3)`
	Esta función permite el correcto funcionamiento de activanim3

{
if (rot3 > -1)
{
rot3 -= 0.1f;
}
 	
}
}	

void KeyCallback(GLFWwindow* window, int key, int scancode, int action, int mode)
{
if (GLFW_KEY_ESCAPE == key && GLFW_PRESS == action)
{
glfwSetWindowShouldClose(window, GL_TRUE);
}
	Esta función nos va a permitir hacer uso de las teclas

if (keys[GLFW_KEY_O])
{
activanim = !activanim;
 	
}
if (keys[GLFW_KEY_C])
{
activanim2 = !activanim2;
 	
}
if (keys[GLFW_KEY_V])
{
activanim3 = !activanim3;
 	
}
	Estas condicionales son las que nos permiten marcar el uso de las teclas o,c y v, cada una para su respectiva animación

void MouseCallback(GLFWwindow* window, double xPos, double yPos)
{
if (firstMouse)
{
lastX = xPos;
lastY = yPos;
firstMouse = false;
}
	Esta función nos va a permitir hacer uno del mouse de nuestro dispositivo para poder movernos en nuestro espacio de modelado al momento de compilar el código

Conclusión

Es un proyecto muy interesante, pero un tanto laborioso y complejo, pero fue un proyecto muy completo, ya que con este se logré reafirmar muchos de los conocimientos y conceptos que adquirí desde la primera práctica. Por otro lado, fue muy satisfactorio poder realizar todo esto. A pesar de que no logré terminar mis animaciones por falta tiempo, estoy muy satisfecho por todo lo que logré en la elaboración tanto del modelado como del código.