

313231706 Grupo:05
Manual de usuario y técnico

Índice

Manual de usuario

1. Objetivo	2
2. Preparación del entorno	2
3. Imágenes de referencia	5
4. Animaciones	
4.1 Animación 1	9
4.2 Animación 2	9
4.3 Animación 3	10
4.4 Animación 4	10
5. Resumen de teclado	11

Manual técnico

6. Objetivo	13
7. Diagrama de Gantt	13
8. Alcance del proyecto	15
9. Limitantes	15
10. Documentación	15
11. Costos	34
12. Enlace GitHub	34
13. Referencias	35

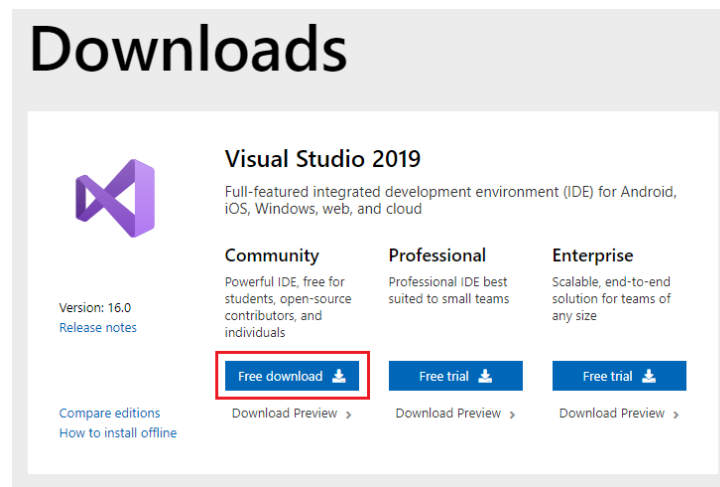
Manual de usuario

1. Objetivo

Aplicar y demostrar los conocimientos adquiridos durante todo el curso de Computación Gráfica e Interacción Humano Computadora, empleando herramientas como OpenGL para la creación de un entorno virtual.

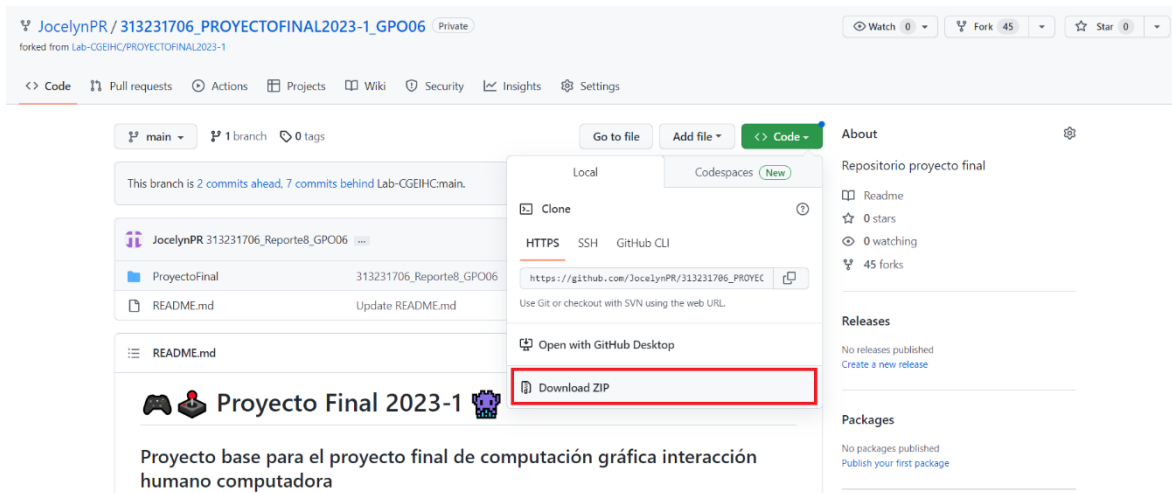
2. Preparación del entorno

Para dar inicio, se requiere tener instalado Visual Studio 2019 en el respectivo equipo de cómputo. <https://visualstudio.microsoft.com/es/vs/older-downloads/>



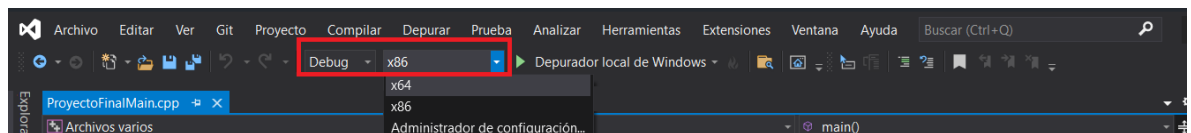
Como siguiente paso se debe descargar el archivo comprimido del proyecto, este se encuentra en GitHub: https://github.com/JocelynPR/313231706_ProyectoFinal_GPO05

Una vez que se tiene el archivo, se debe descomprimir en la dirección deseada

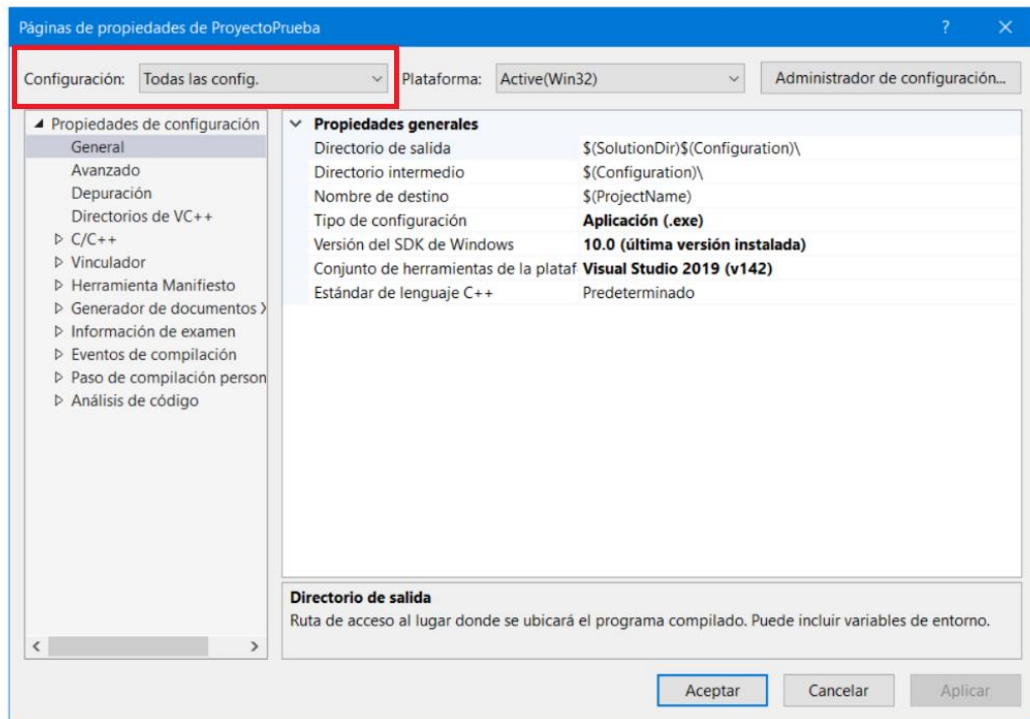


Posteriormente se debe realizar la configuración correspondiente de Visual Studio

1. Abrir el programa
2. Dirigirse a Archivo -> Abrir -> Proyecto o solución y elegir a la ubicación del archivo descomprimido y seleccionar la solución del proyecto.
3. Verificar siempre que en la parte superior se trabaje en una arquitectura x86 sin importar que su equipo sea de 64 bits en caso de que no esté en x86 cambiarlo manualmente.



4. Dar click derecho sobre el nombre del proyecto y dirigirse a la opción de *Propiedades*.
5. Debe de aparecer una nueva ventana. En la parte de arriba de esta ventana aparece *Configuración* y a su lado una lista desplegable, en la cual deben de seleccionar *Todas las Configuraciones*.



6. Seleccionar la pestaña *C/C++ -> General*.
7. En la pestaña de *Directorios de inclusión adicionales* dar click, seleccionar editar y agregar lo siguiente: `$(SolutionDir)/External Libraries/GLEW/include`
`$(SolutionDir)/External Libraries/GLFW/include`

Aceptar.

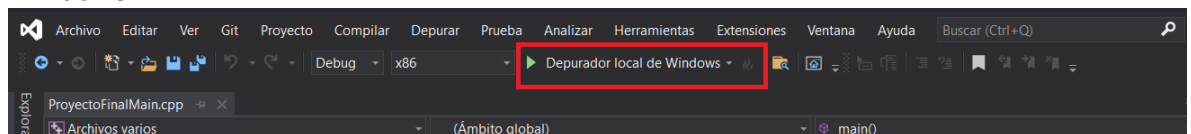
8. Seleccionar la pestaña *Vinculador -> General -> Directorios de bibliotecas adicionales*, seleccionar editar y se debe agregar lo siguiente:
`$(SolutionDir)/External Libraries/GLEW/lib/Release/Win32`
`$(SolutionDir)/External Libraries/GLFW/lib-vc2015`
`lib;assimp/lib;SOIL2/lib`

Y aceptar.

9. En la pestaña *Vinculador -> Entrada -> Dependencias adicionales* y añadir: `soil2-debug.lib;assimp-vc140-mt.lib;opengl32.lib;glew32.lib;glfw3.lib;`

Aceptar.

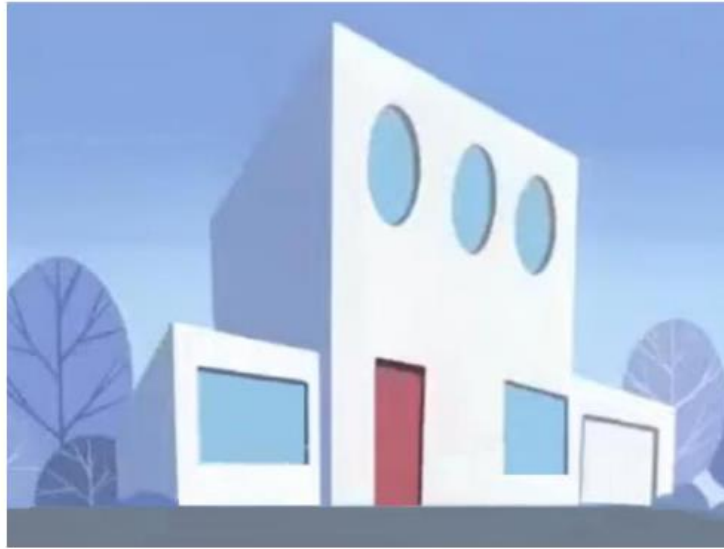
10. Finalmente, se ejecuta el proyecto presionando el botón *Depurador local de Windows* de Windows.



3. Imagen de referencia

Para la creación del entorno virtual, se muestran las siguientes imágenes de referencia para diseñar los respectivos escenarios seleccionados (fachada y dos cuartos) y se listan los 10 elementos que se van a recrear.

Fachada

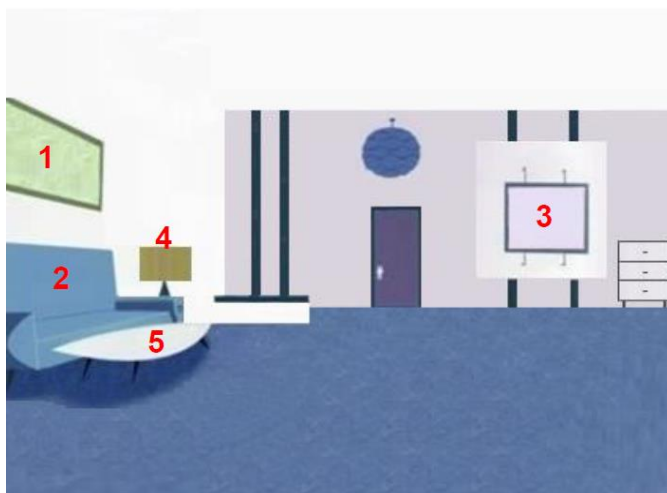


Cuarto 1

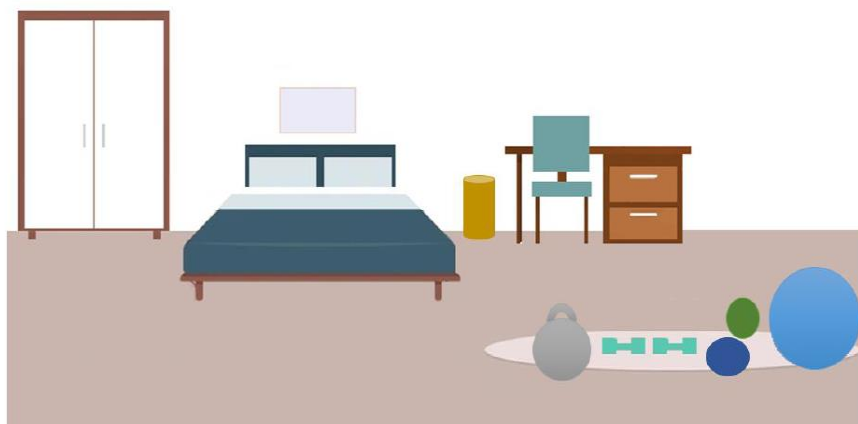


Elementos

1. Cuadro
2. Sillón
3. TV
4. Lampara
5. Mesa

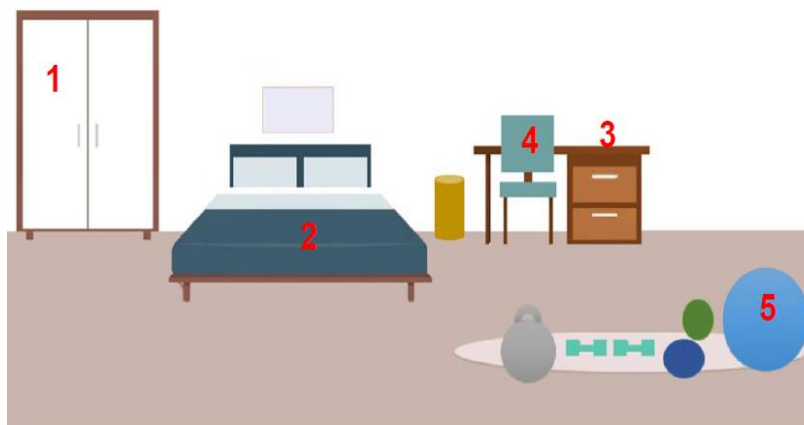


Cuarto 2



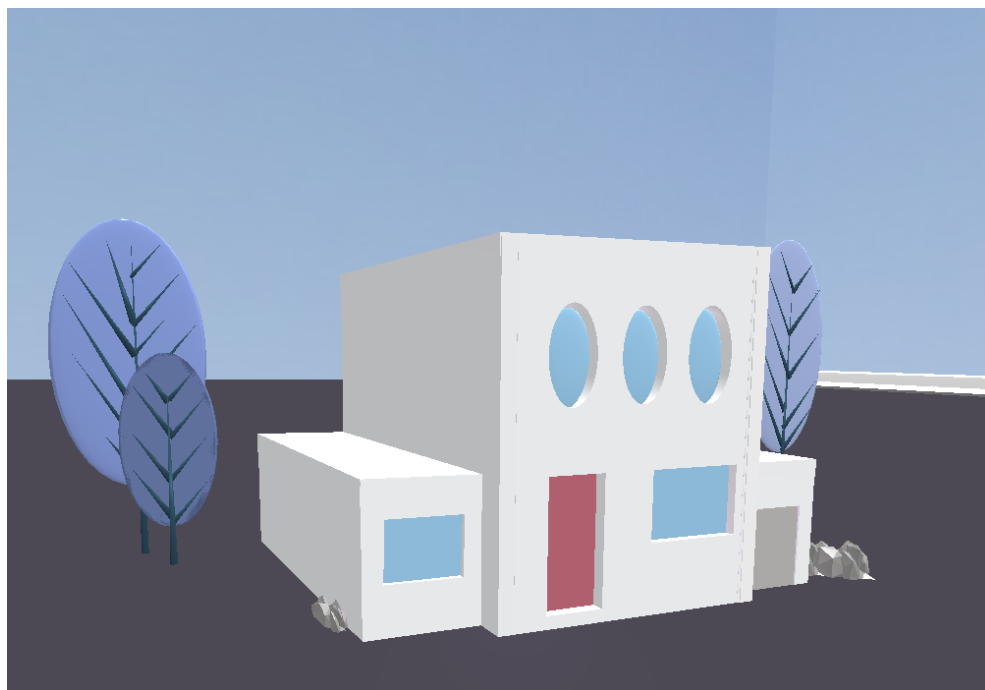
Elementos

1. Armario
2. Cama
3. Escritorio
4. Silla
5. Pelota

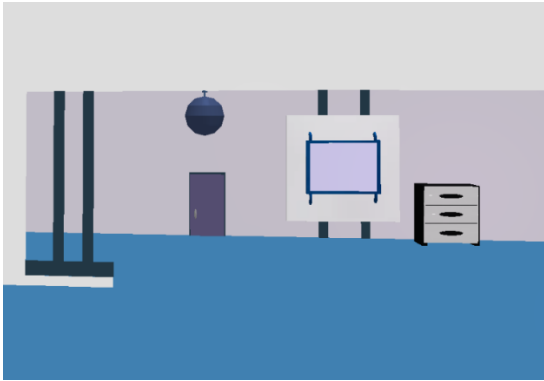


Resultados

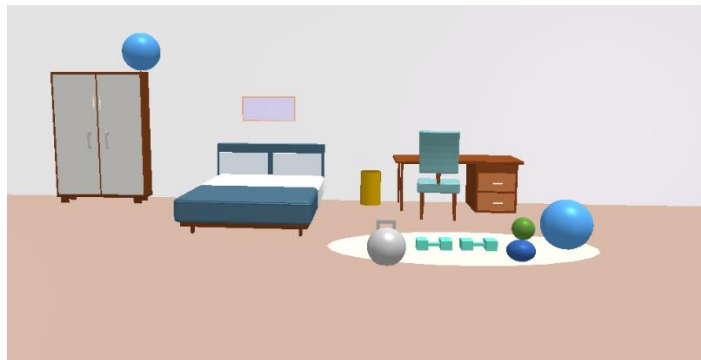
Fachada



Cuarto 1



Cuarto 2

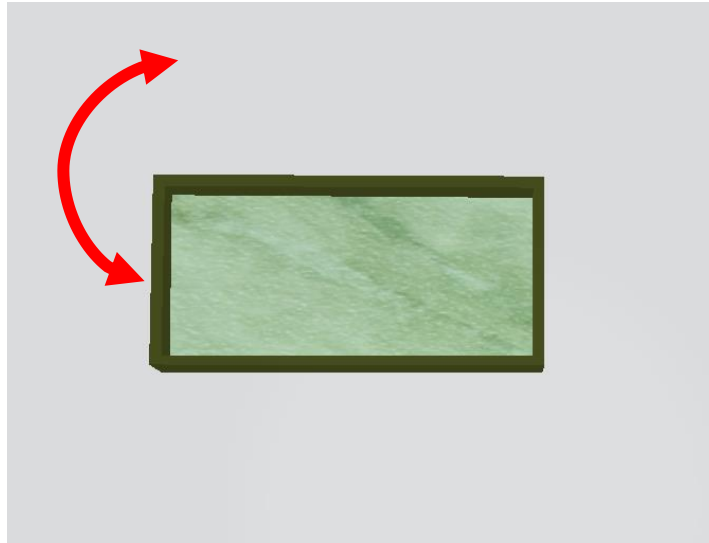


4. Animaciones

A continuación, se muestran 4 animaciones consideradas en este proyecto, cada una está basada en los movimientos que realizamos en nuestras casas al momento de llevar a cabo algún cambio de ubicación de los muebles u objetos.

4.1 Animación 1

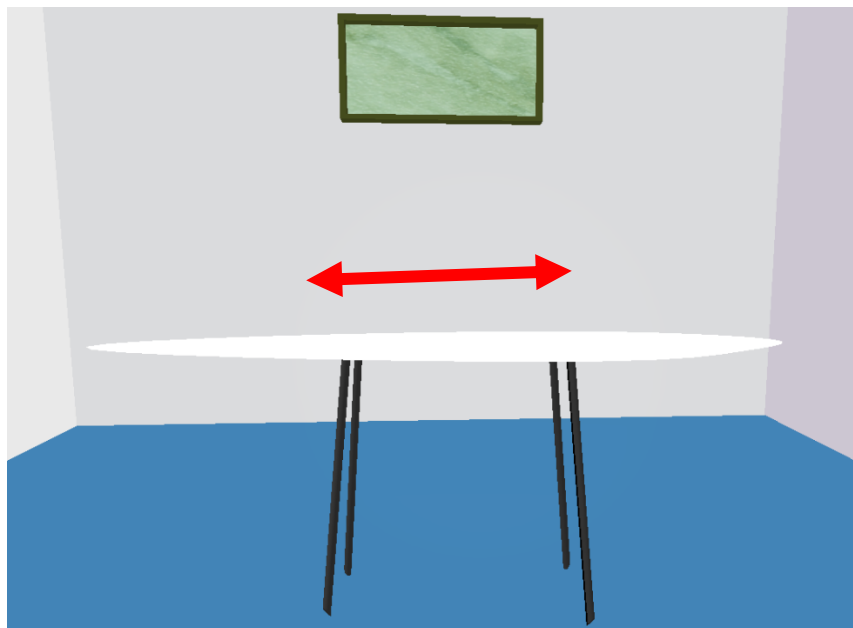
Esta animación consiste en un cuadro que realiza un movimiento vertical y se encuentra a un costado del muro divisorio.



Para iniciar esta animación, se necesita presionar la tecla “I”, para detenerla basta con volver a presionar “I”.

4.2 Animación 2

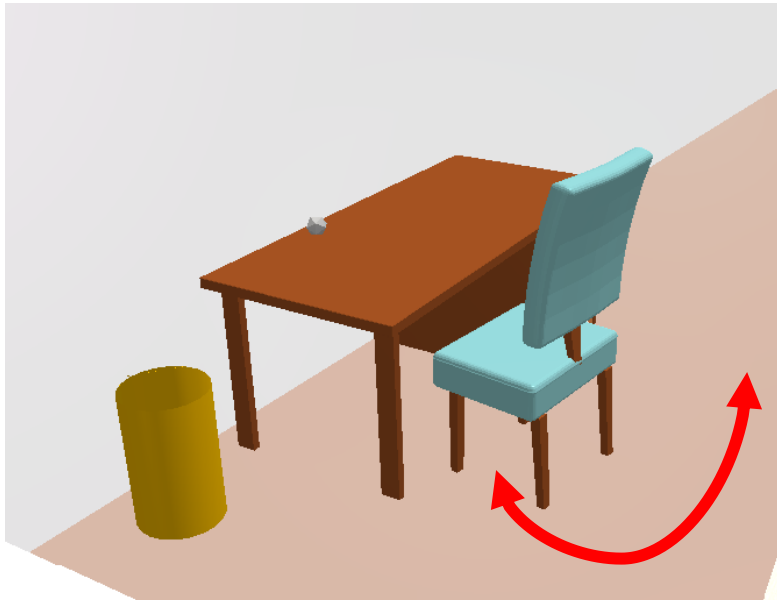
Se trata de una mesa que efectúa un movimiento horizontal.



Es necesario presionar la tecla “O” para activar la animación, para realizar una pausa se debe presionar la misma tecla.

4.3 Animación 3

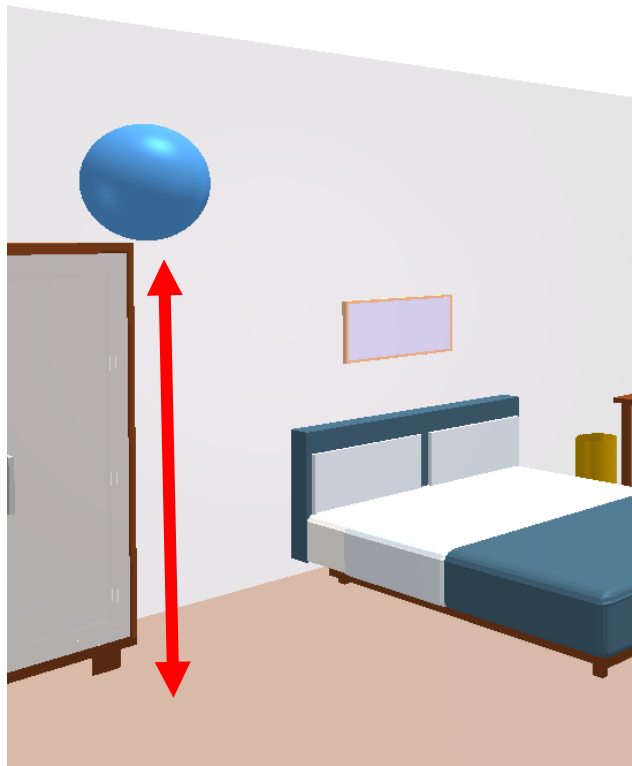
Al oprimir la letra “P” en el teclado, se produce un movimiento con la silla que se encuentra frente a el escritorio.



Para detener o reiniciar la animación se debe pulsar la misma tecla “P”


















4.4 Animación 4

Cuando se presiona la tecla “U” da comienzo esta animación que efectúa la caída libre de la pelota y rebota en el piso hasta que pierde altura. Al oprimir nuevamente “U” se reinicia la animación.



5. Resumen del teclado

Tecla	Acción	Descripción
I	Animación 1	Movimiento Cuadro
O	Animación 2	Movimiento Mesa
P	Animación 3	Movimiento Silla
U	Animación 4	Caída libre de pelota
W	Adelante	Movimiento hacia adelante en el entorno

	Atrás	Movimiento hacia atrás en el entorno
	Derecha	Movimiento hacia la derecha en el entorno
	Izquierda	Movimiento hacia la izquierda en el entorno
	Termina ejecución	Termina ejecución de la aplicación
	Adelante	Movimiento hacia adelante en el entorno
	Atrás	Movimiento hacia atrás en el entorno
	Derecha	Movimiento hacia la derecha en el entorno
	Izquierda	Movimiento hacia la izquierda en el entorno
	PointLight enciende	Enciende las 4 PointLight
	Movimiento PointLight	Movimiento en eje x + de PointLight lado izquierdo parte trasera
	Movimiento PointLight	Movimiento en eje x - de PointLight lado izquierdo parte trasera
	Movimiento PointLight	Movimiento en eje x + de PointLight lado izquierdo parte delantera
	Movimiento PointLight	Movimiento en eje x - de PointLight lado izquierdo parte delantera
	Movimiento PointLight	Movimiento en eje x + de PointLight lado derecho parte trasera
	Movimiento PointLight	Movimiento en eje x - de PointLight lado derecho parte trasera
	Movimiento PointLight	Movimiento en eje x + de PointLight lado derecho parte delantera
	Movimiento PointLight	Movimiento en eje x - de PointLight lado derecho parte delantera

Manual técnico

6. Objetivo

El alumno deberá aplicar y demostrar los conocimientos adquiridos durante todo el curso de Computación Gráfica e Interacción Humano Computadora, empleando herramientas como OpenGL, Maya, Visual Studio y GIMP para la recreación 3D de un entorno virtual, que consiste en una fachada y un cuarto donde deben visualizarse como mínimo 10 objetos lo más parecidos a la imagen de referencia.

7. Diagrama de Gantt

Se empleó la gestión y seguimiento de un diagrama de Gantt para agilizar y cumplir con objetivos de entrega, contemplar las tareas, el tiempo de duración de cada actividad para que se cumplan adecuadamente, así como las fechas de entrega programadas.

RESPONSABLE DEL PROYECTO

INICIO DEL PROYECTO

Jocelyn Karina Peña Reyes

16 de Noviembre de 2022

	NOVIEMBRE			DICIEMBRE												DICIEMBRE										ENERO								
ACTIVIDAD	16	18	24	1	2	5	7	9	11	12	13	14	15	16	17	18	20	21	26	27	28	29	30	3	4	5	6	7	8	9				
INICIO																																		
Asignacion de Proyecto																																		
Asignación de entregable de imágenes de referencia																																		
Funcionamiento de software																																		
DISEÑO, MODELADO Y TEXTURIZADO																																		
Muros y piso																																		
Sillon y mesa																																		
Pantalla y mueble																																		
Esfera y lampara																																		
Fachada																																		
Cuadro y Jardin																																		
Armario																																		
Pelotas, pesas y bote de basura																																		
Cama y papel																																		
Escritorio y silla																																		
Tapete y cuadro																																		
CORRECCIONES																																		
Texturizado de lampara, mesa y mueble																																		
Escalado de sillón y esfera																																		
Color de muros y piso																																		
Traslación de armario y cama																																		
Texturizado de silla																																		
Escalado de escritorio y tapete																																		
Color de pesas y pelotas																																		
Traslación de cuadros																																		
Escalado de fachada																																		
IMPLEMENTACION																																		
Animaciones																																		
Skybox																																		
Iluminación																																		
Archivo ejecutable																																		
Elaboración de manual de usuario																																		
Elaboración de manual técnico																																		

8. Alcance del proyecto

Este documento describe el contenido en el proyecto final de Computación Gráfica e Interacción Humano Computadora, que contempla realizar la recreación de una fachada, dos cuartos y 10 objetos en un entorno virtual, considerando todas las fases necesarias para que el desarrollo del proyecto se realice en tiempo y forma, definiendo cada tarea y llevando a cabo un monitoreo constante del trabajo realizado de acuerdo con el diagrama de Gantt. El proyecto tendrá una duración aproximada de 7 semanas y deberá entregarse con una fecha límite de 10 de enero de 2023.

Además, se realiza una descripción de cada acción o actividades a realizar y se da seguimiento a las interacciones que se desarrollarán. Asimismo, elaborar un manual de usuario y técnico en español e inglés, que permite a las personas un buen entendimiento, así como el uso de las funcionalidades que este posee.

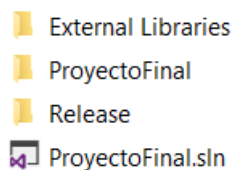
9. Limitantes

- Afectación en tiempos de trabajo en el proyecto ya sea por mal tiempo, asuntos de trabajo o escolares.
- Aplicar muchas acciones correctivas o modificaciones.
- Pérdida de archivos o documentos.

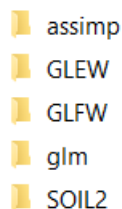
10 Documentación

Estructura del proyecto

El proyecto tiene la siguiente estructura



Durante el desarrollo se utilizaron las siguientes bibliotecas

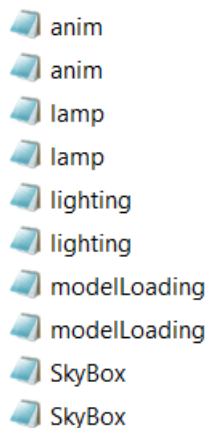


- **Assimp:** es una librería que nos servirá para cargar modelos o escenas 3D almacenados en gran variedad de formatos, en este caso se emplea Wavefront

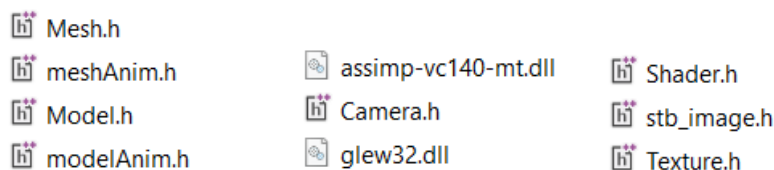
Object (*.obj), Assimp puede cargar información de vértices, coordenadas de textura, normales, materiales, animación, y otros.

- **GLEW:** es una librería multiplataforma escrita en C/C++, destinada a ayudar en la carga y consulta de extensiones de OpenGL
- **GLFW:** es una biblioteca de utilidad ligera para uso con OpenGL. Proporciona la capacidad de crear y dirigir ventanas y aplicaciones OpenGL, así como recibir la entrada de teclado y ratón.
- **glm:** define sobre C/C++ los tipos de datos utilizados, así como numerosas funciones y operadores asociados a estos tipos de datos.
- **SOIL2:** es una biblioteca utilizada principalmente para cargar texturas en OpenGL.

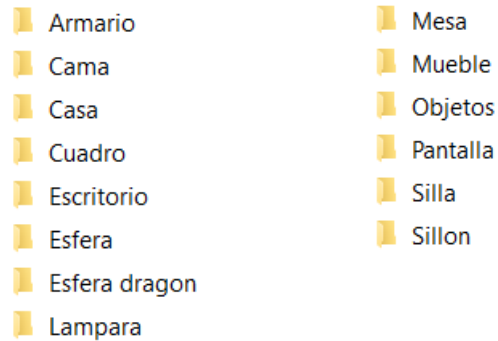
Asimismo, se emplearon los respectivos shaders que fueron utilizados para iluminación, animaciones y el skybox.



Además, se cuenta con archivos dll y de encabezado que hacen que la iluminación, texturas, bibliotecas, modelos y cámara funcionen correctamente.



Para llevar a cabo las actividades programadas, se hicieron carpetas para agregar cada uno de los modelos implementados con OpenGL



Código

Dentro del código se encuentran declaradas todas las bibliotecas y shaders que se emplean durante el desarrollo del proyecto.

```
#include <iostream> //Procesamiento de entradas y salidas en forma de una secuencia de bytes.
#include <cmath> //Conjunto de funciones para realizar operaciones matemáticas

// GLEW
#include <GL/glew.h> //Determina qué extensiones de OpenGL son compatibles con la plataforma

// GLFW
#include <GLFW/glfw3.h> //Permite crear y administrar ventanas

// Other Libs
#include "stb_image.h" //Procesamiento de imágenes

// GLM Mathematics
#include <glm/glm.hpp> //Uso de diferentes librerías dadas por C.
#include <glm/gtc/matrix_transform.hpp> //Define funciones que generan matrices de transformación
#include <glm/gtc/type_ptr.hpp> //Maneja la interacción entre punteros y vectores, tipos de matrices.

//Load Models
#include "SOIL2/SOIL2.h" //Carga de texturas dentro de OpenGL

// Other includes
#include "Shader.h" //Se ejecuta en una etapa del proceso
#include "Camera.h" //Carga de la cámara dentro de la escena
#include "Model.h" //Carga de modelos en OpenGL
#include "Texture.h" //Carga de texturas en OpenGL
#include "modelAnim.h" //Carga de animaciones en OpenGL
```

Posteriormente se declaran los prototipos de funciones

```
void KeyCallback(GLFWwindow *window, int key, int scancode, int action, int mode);  
//Lectura de teclado  
  
void MouseCallback(GLFWwindow *window, double xPos, double yPos); //Lectura de mouse  
  
void DoMovement(); //Movimiento de las posiciones de la cámara según la entrada del  
usuario  
  
void animacion(); //Uso para animación
```

Como siguiente paso, se definen las dimensiones de la ventana donde se visualizará el resultado. En donde se emplean 1200 x 1000 pixeles en total.

```
const GLuint WIDTH = 1200, HEIGHT = 1000;  
  
int SCREEN_WIDTH, SCREEN_HEIGHT;
```

Asimismo, se especifican las variables de la cámara y su posición inicial dentro de la escena.

```
Camera camera(glm::vec3(-10.0f, 10.0f, -50.0f));  
  
GLfloat lastX = WIDTH / 2.0;  
GLfloat lastY = HEIGHT / 2.0;  
bool keys[1024];  
bool firstMouse = true;  
  
float movCamera = 0.0f;
```

Por otro lado, se tiene detalle sobre los shaders que se emplearon para la iluminación, animación y skybox.

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

Modelos

Como siguiente paso se especifica la llamada de cada uno de los modelos con formato .obj y que son importados desde el software Maya.

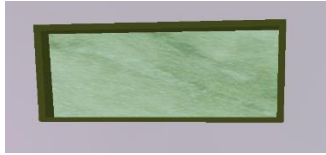
```
Model Casa((char*)"Models/Casa/casa.obj");
Model Cuadro((char*)"Models/Cuadro/cuadro.obj");
Model Esfera((char*)"Models/Esfera/esfera.obj");
Model Sillon((char*)"Models/Sillon/sillon.obj");
Model Piso((char*)"Models/Esfera dragon/Piso.obj");
Model Lampara((char*)"Models/Lampara/lampara.obj");
Model Mesa((char*)"Models/Mesa/mesa.obj");
Model Mueble((char*)"Models/Mueble/mueblelisto.obj");
Model Pantalla((char*)"Models/Pantalla/pantalla.obj");
Model Armario((char*)"Models/Armario/armario.obj");
Model Cama((char*)"Models/Cama/cama.obj");
Model Escritorio((char*)"Models/Escritorio/escritorio.obj");
Model Silla((char*)"Models/Silla/silla.obj");
Model Objeto1((char*)"Models/Objetos/tapete.obj");
Model Objeto2((char*)"Models/Objetos/cuadroCuarto.obj");
Model Objeto3((char*)"Models/Objetos/pelotaAzul.obj");
Model Objeto4((char*)"Models/Objetos/pelotaVerde.obj");
Model Objeto5((char*)"Models/Objetos/pelotaGrande.obj");
Model Objeto6((char*)"Models/Objetos/bote.obj");
Model Objeto7((char*)"Models/Objetos/pesa.obj");
Model Objeto8((char*)"Models/Objetos/pesita.obj");
```

Se realiza una carga de todos los modelos declarados anteriormente, se emplean transformaciones básicas (rotación, traslación o escala) según sea el caso.

```
// Sillón
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f)); //Se modifica la escala
model = glm::translate(model, glm::vec3(-3.0f, 0.3f, -6.0f)); //Se realiza una
traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Sillon.Draw(lightningShader); //Se dibuja el modelo
```



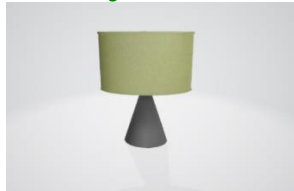
```
// Cuadro
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(3.0f, 2.0f, 3.0f)); //Se modifica la escala
model = glm::translate(model, glm::vec3(-4.0f, 1.1f, -5.65f)); //Se realiza una
traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Cuadro.Draw(lightningShader); //Se dibuja el modelo
```



```
// Mesa
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 4.0f, 1.0f)); //Se modifica la escala
model = glm::translate(model, glm::vec3(-24.0f, 0.1f, -15.2f)); //Se realiza una
traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Mesa.Draw(lightningShader); //Se dibuja el modelo
```



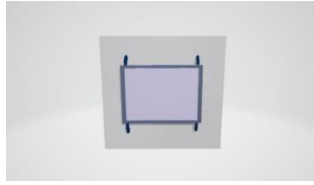
```
// Lampara
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.5f, 1.5f, 1.5f)); //Se modifica la escala
model = glm::translate(glm::mat4(1.0f), glm::vec3(-23.0f, 4.8f, -15.0f)); //Se
realiza una traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Lampara.Draw(lightningShader); //Se dibuja el modelo
```



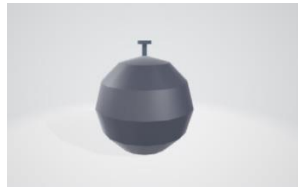
```
// Mueble
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.1f, 1.1f, 1.3f)); //Se modifica la escala
model = glm::translate(model, glm::vec3(11.0f, 0.2f, -8.5f)); //Se realiza una
traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Mueble.Draw(lightningShader); //Se dibuja el modelo
```



```
// Pantalla
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.5f, 1.5f, 1.5f)); //Se modifica la escala
model = glm::translate(model, glm::vec3(0.6f, 2.0f, -7.0f)); //Se realiza una
traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Pantalla.Draw(lightningShader); //Se dibuja el modelo
```



```
// Esfera
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(0.9f, 0.9f, 0.9f)); //Se modifica la escala
model = glm::translate(model, glm::vec3(-4.5f, 8.8f, -11.9f)); //Se realiza una
traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Esfera.Draw(lightningShader); //Se dibuja el modelo
```



```
// Armario
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(2.5f, 2.5f, 2.5f)); //Se modifica la escala
model = glm::rotate(model, glm::radians(-180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //Se
realiza una rotación
model = glm::translate(model, glm::vec3(17.0f, 0.9f, 45.7f)); //Se realiza una
traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Armario.Draw(lightningShader); //Se dibuja el modelo
```



```
// Cama
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Se modifica la escala
model = glm::rotate(model, glm::radians(178.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //Se
realiza una rotación
model = glm::translate(model, glm::vec3(15.0f, 2.5f, 82.7f)); //Se realiza una
traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Cama.Draw(lightningShader); //Se dibuja el modelo
```



```
// Escritorio
model = glm::mat4(1);
```

```

model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Se modifica la escala
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //Se
realiza una rotación
model = glm::translate(model, glm::vec3(-14.0f, 2.7f, 77.8f)); //Se realiza una
traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Escritorio.Draw(lightningShader); //Se dibuja el modelo

```



```

// Silla
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Se modifica la escala
model = glm::rotate(model, glm::radians(240.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //Se
realiza una rotación
model = glm::translate(model, glm::vec3(-85.5f, 1.5f, 32.5f)); //Se realiza una
traslación al modelo
model = glm::rotate(model, rotSILLA * glm::radians(-12.0f), glm::vec3(0.0f, 1.0f,
0.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Silla.Draw(lightningShader); //Se dibuja el modelo

```



```

// Bote
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Se modifica la escala
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //Se
realiza una rotación
model = glm::translate(model, glm::vec3(-10.0f, 1.5f, 97.1f)); //Se realiza una
traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Objeto6.Draw(lightningShader); //Se dibuja el modelo

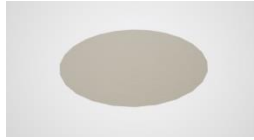
```



```

// Tapete
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.5f, 1.7f, 1.2f)); //Se modifica la escala
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //Se
realiza una rotación
model = glm::translate(model, glm::vec3(-5.7f, 1.4f, 71.0f)); //Se realiza una
traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Objeto1.Draw(lightningShader); //Se dibuja el modelo

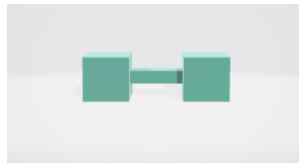
```



```
// Pesa
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Se modifica la escala
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //Se
realiza una rotación
model = glm::translate(model, glm::vec3(1.0f, 2.5f, 72.0f)); //Se realiza una
traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Objeto7.Draw(lightningShader); //Se dibuja el modelo
```



```
// Pesita
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Se modifica la escala
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //Se
realiza una rotación
model = glm::translate(model, glm::vec3(3.0f, 2.5f, 74.0f)); //Se realiza una
traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Objeto8.Draw(lightningShader); //Se dibuja el modelo
```



```
// Pelota azul
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Se modifica la escala
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //Se
realiza una rotación
model = glm::translate(model, glm::vec3(1.7f, 2.6f, 73.6f)); //Se realiza una
traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Objeto3.Draw(lightningShader); //Se dibuja el modelo
```

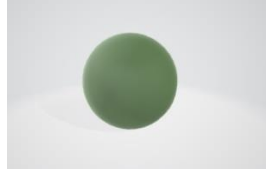


```
// Pelota verde
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Se modifica la escala
```

```

model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //Se
realiza una rotación
model = glm::translate(model, glm::vec3(1.5f, 2.6f, 75.4f)); //Se realiza una
traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Objeto4.Draw(lightningShader); //Se dibuja el modelo

```



```

// Pelota azul grande
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Se modifica la escala
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //Se
realiza una rotación
model = glm::translate(model, glm::vec3(1.5f, 2.7f, 75.4f)); //Se realiza una
traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Objeto5.Draw(lightningShader); //Se dibuja el modelo

```



```

// Cuadro cuarto
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.2f, 1.0f, 1.0f)); //Se modifica la escala
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //Se
realiza una rotación
model = glm::translate(model, glm::vec3(-1.2f, 8.0f, 77.0f)); //Se realiza una
traslación al modelo
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Objeto2.Draw(lightningShader); //Se dibuja el modelo

```



Animaciones

Se declaran las variables que se utilizan en cada una de las animaciones y se especifica su función.

```
//Animación CUADRO
```

<code>bool</code> animacionCUADRO = <code>false</code> ;	Animación de cuadro inicializada en falso
--	---

<code>bool regresoCUADRO = false;</code>	Animación regreso de cuadro inicializada en falso
--	---

//Animación MESA

<code>bool animacionMESA = false;</code>	Animación de la mesa inicializada en falso
<code>float pos = -5.0f;</code>	Definición de posición mesa
<code>bool regresoMESA = false;</code>	Animación regreso de la mesa inicializada en falso

// Animación SILLA

<code>bool animacionSILLA = false;</code>	Animación de silla inicializada en falso
<code>float rotSILLA = 10.7f;</code>	Definición de rotación de silla
<code>bool regresoSILLA = false;</code>	Animación regreso de silla inicializada en falso

//Animación PELOTA

<code>float auxiliar;</code>	Empleada para terminar el rebote de la pelota según sea el caso
<code>float PELOTA_x = -11.0f;</code>	Movimiento de la pelota en x
<code>float PELOTA_z = 0.0f;</code>	Movimiento de la pelota en z
<code>float PELOTA_y = 8.8f;</code>	Movimiento de la pelota en y
<code>int est = 0;</code>	Estados de la pelota
<code>bool animacionPELOTA = false;</code>	Animación de la pelota inicializada en falso
<code>bool regresoPELOTA = false;</code>	Animación regreso de la pelota inicializada en falso
<code>float altura1 = 8.8f;</code>	Definición de la altura 1 de la pelota
<code>float Tiempo = 0.0f;</code>	Definición de tiempo de la pelota
<code>float altura2 = altura1;</code>	Definición de la altura 2 de la pelota
<code>float velocidadPELOTA = 1.0f;</code>	Definición de la velocidad de la pelota

Animación 1. Consiste en un cuadro que realiza un movimiento vertical.

// Animación 1 Cuadro

```

    if (animacionCUADRO)
    {
        if (!regresoCUADRO)
        {
            if (rot < 15.0f)
            {
                rot += 0.05f;
            }
            else
            {
                regresoCUADRO = true;
            }
        }
        else
        {

```

```

        if (rot > 5.0f)
        {
            rot -= 0.05f;
        }
        else
        {
            regresoCUADRO = false;
        }
    }
}

```

Para activar o detener la animación basta con presionar la tecla “I”.

```

//Animación Cuadro
if (glfwGetKey(window, GLFW_KEY_I) == GLFW_PRESS)
    animacionCUADRO ^= true;
    regresoCUADRO = false;

```

Animación 2. Se trata de una mesa que efectúa un movimiento horizontal

```

//Animación 2 Mesa
if (animacionMESA)
{
    if (!regresoMESA)
    {
        if (pos < 2.0f)
        {
            pos += 0.09f;
        }
        else
        {
            regresoMESA = true;
        }
    }
    else
    {
        if (pos > -6.0f)
        {
            pos -= 0.09f;
        }
        else
        {
            regresoMESA = false;
        }
    }
}

```

Para iniciar o pausar la animación se debe presionar la tecla “O”.

```

//Animación mesa

```

```

if (glfwGetKey(window, GLFW_KEY_O) == GLFW_PRESS)
    animacionMESA ^= true;
regresoMESA = false;

```

Animación 3. Al oprimir la letra “P” en el teclado, se produce un movimiento silla que se encuentre frente al escritorio.

```

//Animación SILLA
if (glfwGetKey(window, GLFW_KEY_P) == GLFW_PRESS)
    animacionSILLA ^= true;
regresoSILLA = false;

```

```

// Animación 3 SILLA
if (animacionSILLA)
{
    if (!regresoSILLA)
    {
        if (rotSILLA < 5.5f)
        {
            rotSILLA += 0.01f;
        }
        else
        {
            regresoSILLA = true;
        }
    }
    else
    {
        if (rotSILLA > 4.1f)
        {
            rotSILLA -= 0.01f;
        }
        else
        {
            regresoSILLA = false;
        }
    }
}

```

Animación 4. Cuando se presiona la tecla “U” da comienzo esta animación que efectúa la caída libre de la pelota y rebota en el piso hasta que pierde altura. Al oprimir nuevamente “U” se reinicia la animación.

```

//Animación PELOTA
if (glfwGetKey(window, GLFW_KEY_U) == GLFW_PRESS)
{
    PELOTA_x = -11.0f;
    PELOTA_z = 0.0f;
    PELOTA_y = 8.5f;
    est = 0;
    animacionPELOTA ^= true;
    regresoPELOTA = false;
    altura1 = 8.5f;
    Tiempo = 0.0f;
    altura2 = altura1;
    velocidadPELOTA = 1.0f;
}

```

```

//Animación 4 PELOTA

if (animacionPELOTA)
{
    switch (est)
    {
        case 0: //Hacia abajo
            if (PELOTA_y > 0.0f)
            {
                Tiempo += 0.05;
                auxiliar = (altura2 - (0.5 * 9.81 * pow(Tiempo,
2)))));

                if (auxiliar < 0.0)
                    PELOTA_y = 0.0;
                else
                    PELOTA_y = auxiliar;
            }
            else
            {
                velocidadPELOTA = (9.81 * Tiempo) / 0.5; //
                altura2 = altura2 * 0.6;
                Tiempo = 0.0f;
                if (altura2 < 0.2)
                    est = 2;
                else
                    est = 1;
            }
            break;
        case 1: //Hacia arriba
            if (PELOTA_y < altura2)
            {
                Tiempo += 0.05;
                auxiliar = 9.81 * pow(Tiempo, 2);
                if (auxiliar > altura2)
                    PELOTA_y = altura2;
            }

```

```

        else
            PELOTA_y += 0.05;
    }
    else
    {
        Tiempo = 0.0f;
        est = 0;
    }

case 2:
    PELOTA_y = PELOTA_y;
    break;
default:
    break;
}
}

```

Skybox

Se especifican las respectivas caras para el entorno, se trabaja con formato png y tga.

```

vector<const GLchar*> faces;
    faces.push_back("SkyBox/right.png");
    faces.push_back("SkyBox/left.png");
    faces.push_back("SkyBox/top.png");
    faces.push_back("SkyBox/bottom.tga");
    faces.push_back("SkyBox/back.png");
    faces.push_back("SkyBox/detras.png");

```

Iluminación

Se define la posición de las 4 PointLight que se encuentran en cada esquina de la fachada

```

glm::vec3 pointLightPositions[] = {
    glm::vec3(44.0f,0.0f,23.0f), //lado derecho, frente
    glm::vec3(-46.0f,0.0f,-95.0f), //lado izquierdo, atrás
    glm::vec3(-45.0f,0.0f,24.0f), //lado izquierdo, frente
    glm::vec3(48.0f,0.0f,-95.0f) //lado derecho, atrás
};

```

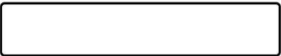
Se define la siguiente variable para encender las PointLight al presionar la tecla "space"

<code>bool</code> active;	Se activa/ enciende PointLight
---------------------------	--------------------------------

```

if (keys[GLFW_KEY_SPACE])
{
    active = !active;
    if (active)
    {
        Light1 = glm::vec3(1.0f, 0.0f, 1.0f);
        Light2 = glm::vec3(0.04f, 0.16f, 0.45f);
        Light3 = glm::vec3(0.27f, 1.0f, 0.4f);
        Light4 = glm::vec3(0.2f, 0.0f, 1.0f);
    }
    else
    {
        Light1 = glm::vec3(0);
        Light2 = glm::vec3(0);
        Light3 = glm::vec3(0);
        Light4 = glm::vec3(0);
    }
}
}

```

Tecla	Acción	Descripción
	PointLight activa	Se activan las 4 PointLight y comienzan a parpadear



Además, se produce movimiento de las 4 PointLight en el eje x positivo como negativo. Esto se puede hacer presionando las siguientes teclas:

```

/*pointLight*/
//Lado izquierdo - detras
if (keys[GLFW_KEY_1])
{
    pointLightPositions[1].x += 0.01f;
}



if (keys[GLFW_KEY_2])
{
    pointLightPositions[1].x -= 0.01f;
}

```

Tecla	Acción	Descripción
	Movimiento PointLight	Movimiento en eje x + de PointLight lado izquierdo parte trasera
	Movimiento PointLight	Movimiento en eje x - de PointLight lado izquierdo parte trasera



```
//Lado izquierdo - frente
if (keys[GLFW_KEY_3])
{
    pointLightPositions[2].x += 0.01f;
}

if (keys[GLFW_KEY_4])
{
    pointLightPositions[2].x -= 0.01f;
}
```

Tecla	Acción	Descripción
	Movimiento PointLight	Movimiento en eje x + de PointLight lado izquierdo parte delantera
	Movimiento PointLight	Movimiento en eje x - de PointLight lado izquierdo parte delantera

```
//Lado derecho - detras
if (keys[GLFW_KEY_5])
{
    pointLightPositions[3].x += 0.01f;
}

if (keys[GLFW_KEY_6])
{
    pointLightPositions[0].x -= 0.01f;
}
```

Tecla	Acción	Descripción
	Movimiento PointLight	Movimiento en eje x + de PointLight lado derecho parte trasera
	Movimiento PointLight	Movimiento en eje x - de PointLight lado derecho parte trasera

```
//Lado derecho - enfrente
if (keys[GLFW_KEY_7])
{
    pointLightPositions[0].x += 0.01f;
}
if (keys[GLFW_KEY_8])
{
    pointLightPositions[0].x -= 0.01f;
}
```

}

Tecla	Acción	Descripción
	Movimiento PointLight	Movimiento en eje x + de PointLight lado derecho parte delantera
	Movimiento PointLight	Movimiento en eje x - de PointLight lado derecho parte delantera

Funciones

Se detallan cada una de las funciones empleadas y se agrega una breve descripción de su uso.

Nombre	Descripción
<code>glfwMakeContextCurrent (GLFWwindow *window)</code>	Hace que el contexto de la ventana especificada sea actual para el subproceso de llamada.
<code>glfwGetFramebufferSize (window, &SCREEN_WIDTH, &SCREEN_HEIGHT);</code>	Cambia el tamaño de una ventana.
<code>glfwSetKeyCallback (window, KeyCallback)</code>	Establece la devolución de llamada de tecla de la ventana especificada, que se llama cuando se presiona, repite o suelta una tecla.
<code>glViewport(0, 0, SCREEN_WIDTH, SCREEN_HEIGHT);</code>	Establece la ventana gráfica.
<code>glfwWindowShouldClose(window)</code>	Devuelve el valor del indicador de cierre de la ventana especificada.
<code>glAreTexturesResident (GLsizei n, const GLuint *textures, GLboolean *residences);</code>	Determinar si los objetos de textura especificados residen en la memoria de textura.
<code>glArrayElement (GLint i);</code>	Especifica los elementos de matriz utilizados para representar un vértice.
<code>glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);</code>	Especifica aritmética de píxeles.
<code>glAlphaFunc (GLenum func, GLclampf ref);</code>	Permite a la aplicación establecer la función de prueba alfa.
<code>glBegin (GLenum mode);</code>	Delimitan los vértices de un primitivo o un grupo de primitivos similares.
<code>glBindTexture (GLenum target, GLuint texture);</code>	Permite crear una textura con nombre enlazada a un destino de textura.
<code>glBitmap (GLsizei width, GLsizei height, GLfloat xorig, GLfloat yorig, GLfloat xmove, GLfloat ymove, const GLubyte *bitmap);</code>	Dibuja un mapa de bits.
<code>glfwSwapBuffers(window);</code>	Intercambia los búfer frontal y posterior de la ventana

	especificada cuando se renderiza con OpenGL
<code>glDrawArrays(GL_TRIANGLES, 0, 36);</code>	Especifica varios primitivos que se van a representar.
<code>glfwTerminate();</code>	Destruye todas las ventanas y cursores restantes, restaura las rampas gamma modificadas y libera cualquier otro recurso asignado.
<code>glTexCoord1d (GLdouble s);</code>	Establece las coordenadas de textura actuales.
<code>glfwInit();</code>	Inicializa la biblioteca GLFW.
<code>glfwGetTime();</code>	Devuelve el valor del temporizador GLFW.
<code>glfwCreateWindow(WIDTH, HEIGHT, "Iluminacion 2", nullptr, nullptr);</code>	Crea una ventana y su contexto OpenGL
<code>glfwPollEvents();</code>	Procesa solo aquellos eventos que ya están en la cola de eventos y luego regresa inmediatamente.
<code>glTranslated (GLdouble x, GLdouble y, GLdouble z);</code>	Multiplica la matriz actual por una matriz de traducción.
<code>glDisable(GL_BLEND);</code>	Deshabilitan las funcionalidades de OpenGL.
<code>glClear (GLbitfield mask);</code>	Borra los búferes en valores preestablecidos.
<code>glDrawElements (GLenum mode, GLsizei count, GLenum type, const void *indices);</code>	Representa primitivos a partir de datos de matriz.
<code>glFlush (void);</code>	Fuerza la ejecución de funciones OpenGL en tiempo finito.
<code>glLightf (GLenum light, GLenum pname, GLfloat param);</code>	Devuelve valores de parámetros de origen de luz.
<code>glRotated (GLdouble angle, GLdouble x, GLdouble y, GLdouble z);</code>	Multiplica la matriz actual por una matriz de rotación.
<code>glClear (GLbitfield mask);</code>	Especifica un plano con el que se recorta toda la geometría.
<code>glColorMaterial (GLenum face, GLenum mode);</code>	Hace que un color de material realice un seguimiento del color actual.
<code>glEndList (void);</code>	Crean o reemplazan una lista para mostrar.
<code>glNormal3b (GLbyte nx, GLbyte ny, GLbyte nz);</code>	Establece el vector normal actual.
<code>glEnable(GL_BLEND);</code>	Habilitan las funcionalidades de OpenGL.
<code>glScaled (GLdouble x, GLdouble y, GLdouble z);</code>	Multiplican la matriz actual por una matriz de escalado general.
<code>glVertex2d (GLdouble x, GLdouble y);</code>	Especifica un vértice.
<code>glfwGetKey(window, GLFW_KEY_I)</code>	Devuelve el último estado informado para la tecla especificada a la ventana especificada.
<code>glfwSetWindowShouldClose(window, GL_TRUE);</code>	Establece el valor del indicador de cierre de la ventana especificada.
<code>glIndexf (GLfloat c);</code>	Establece el índice de color actual.
<code>glAccum (GLenum op, GLfloat value);</code>	Funciona en el búfer de acumulación.

11 Costos

Para determinar el costo final del proyecto se consideran los costos de mano de obra, servicios y materiales empleados durante el desarrollo de todas las actividades necesarias para el correcto funcionamiento del proyecto.

CONCEPTO	IMPORTE
Costos fijos	
Internet	\$ 700.00
Electricidad	\$ 290.00
Sueldos	\$ 35,000.00
Mantenimiento de equipo	\$ 420.00
Software y PC	\$ 700.00
Subtotal	\$37,110.00
Costos variables	
Materiales	\$ 500.00
Insumos	\$ 150.00
Subtotal	\$ 650.00
Subtotal	\$ 37,760.00
IVA 16%	\$6,042.00
Utilidad	\$6,570.00
TOTAL	\$50,370.00

12 Enlace

Todos los archivos del proyecto se encuentran en GitHub. Además, se puede descargar el comprimido, en el siguiente enlace:

https://github.com/JocelynPR/313231706_ProyectoFinal_GPO05

13 Bibliografía

Microsoft. (s.f.). Visual Studio. Recuperado el 24 de noviembre de 2022 de <https://visualstudio.microsoft.com/es/vs/older-downloads/>

Alex. G. (Enero 2004). Programación con OpenGL. Recuperado el 20 de diciembre de 2022 de https://lc.fie.umich.mx/~rochoa/Manuales/OPEN_GL/TUTORIAL.pdf

Microsoft. (s.f.). Introducción a OpenGL. Recuperado el 27 de diciembre de 2022 de <https://learn.microsoft.com/es-es/windows/win32/opengl/introduction-to-opengl>

GLFW. (s.f.). Funciones OpenGL. Recuperado el 30 de diciembre de 2022 de https://www.glfw.org/docs/3.1/group__window.html#ga37bd57223967b4211d60ca1a0bf3c832