

**Final Project. User and technical manual**

**Contents**

**User manual**

1. Objective .....	2
2. Preparation of the environment .....	2
3. Reference images .....	5
4. Animations	
4.1 1 Animation .....	9
4.2 2 Animation .....	9
4.3 3 Animation .....	10
4.4 4 Animation .....	10
5. Keyboard overview .....	11

**Technical manual**

6. Objective .....	13
7. Gantt Diagram .....	13
8. Project scope .....	15
9. Limitations .....	15
10. Documentation .....	15
11. Costs .....	34
12. GitHub link .....	34
13. References .....	35

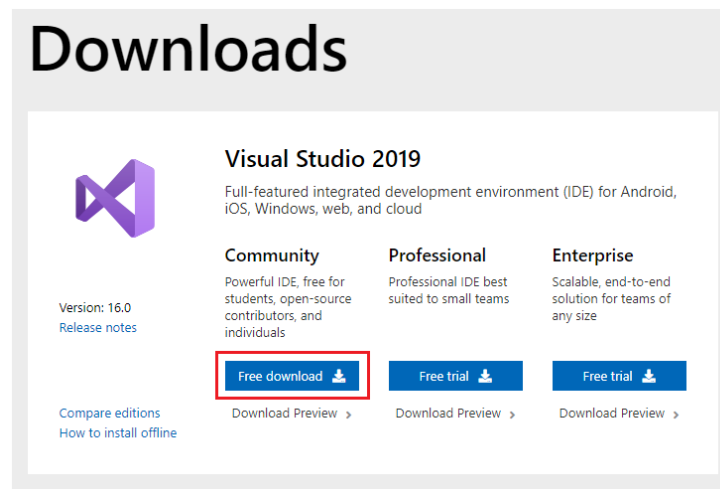
# User Manual

## 1. Objective

Apply and demonstrate the knowledge acquired throughout the course in the Computer Graphics and Human Computer Interaction, using tools such as OpenGL to create a virtual environment.

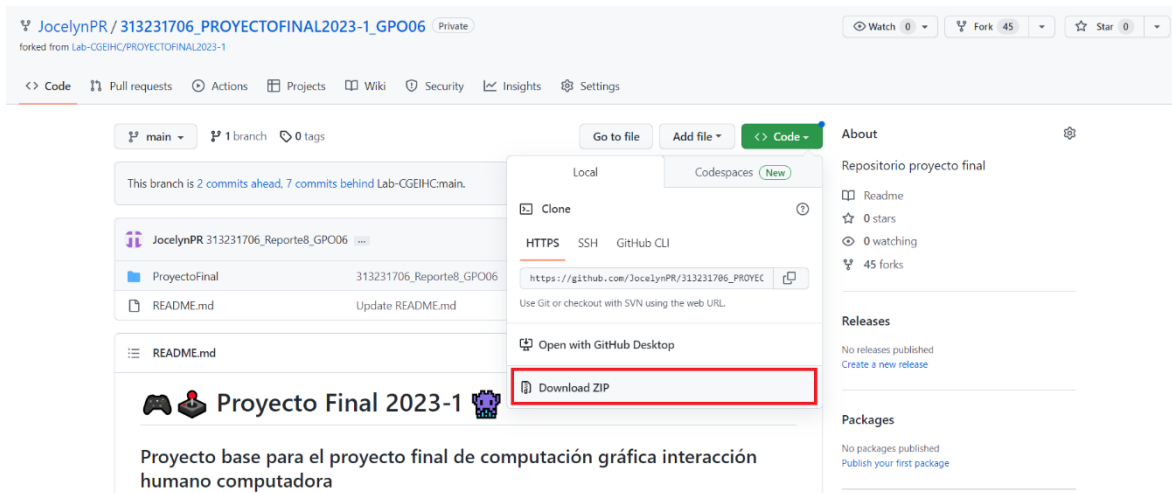
## 2. Preparation of the environment

To start, it is required to have Visual Studio 2019 installed on the respective computer.  
<https://visualstudio.microsoft.com/es/vs/older-downloads/>



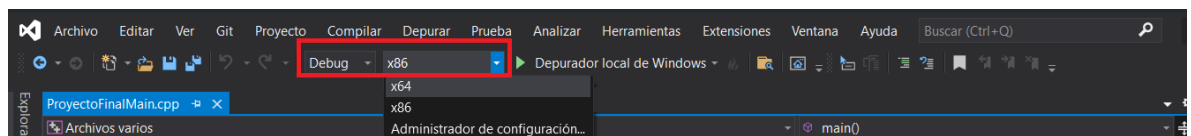
As the next step, you must download the compressed file of the project, it is located on GitHub: [https://github.com/JocelynPR/313231706\\_ProyectoFinal\\_GPO05](https://github.com/JocelynPR/313231706_ProyectoFinal_GPO05)

Once you have the file, you must unzip it in the desired direction

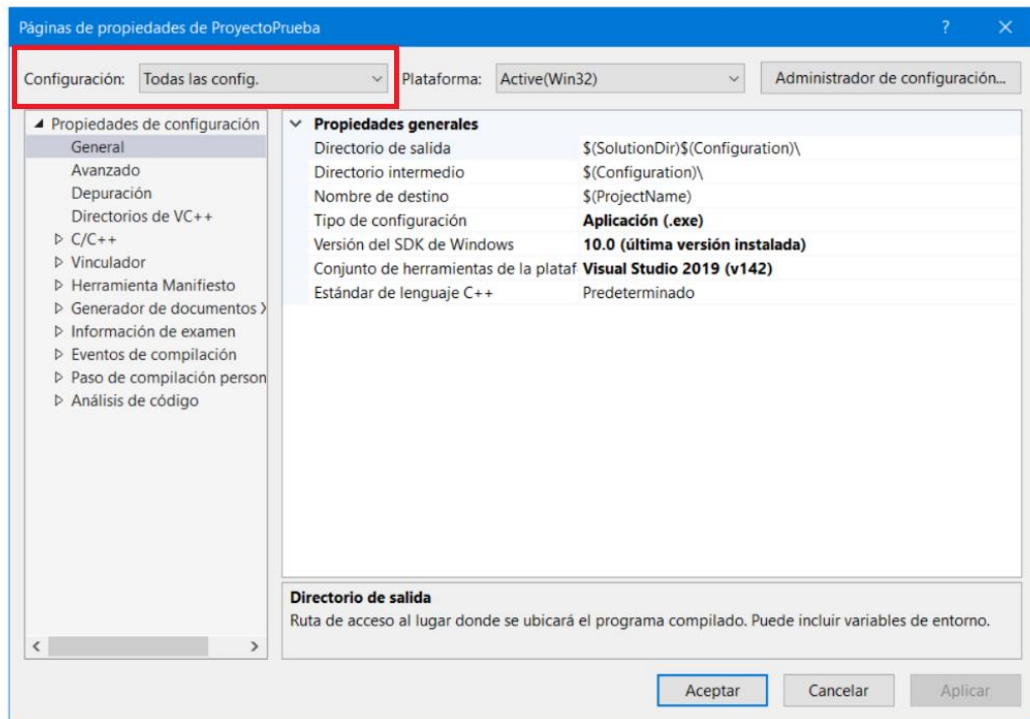


Subsequently, the corresponding configuration of Visual Studio must be carried out

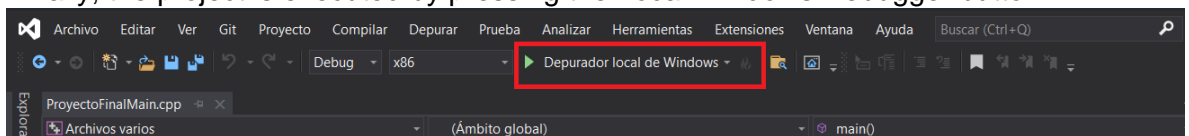
1. Open the program.
2. Go to File -> Open -> Project or solution and choose the location of the file.
3. Always verify that the upper part works on an x86 architecture regardless of whether your computer is 64-bit in case it is not in x86 change it manually.



4. Right click on the project name and go to the *Properties* option
5. A new window should appear. At the top of this window appears *Configuration* and next to it a drop-down list, in which they must select *All Configurations*.



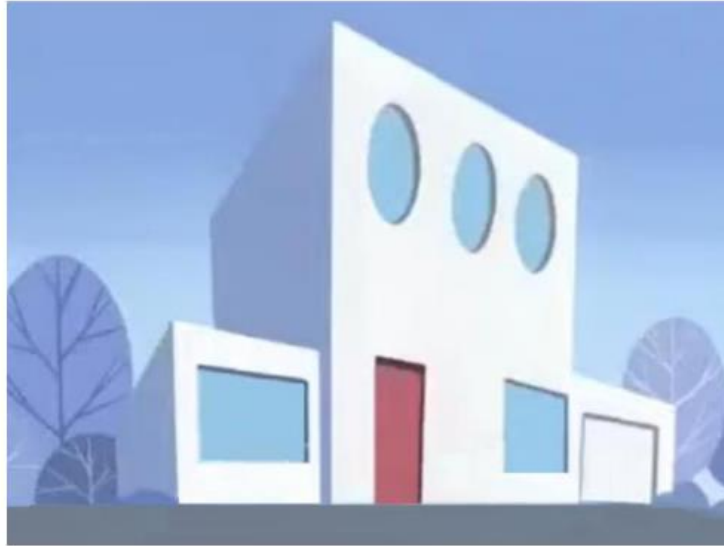
6. Select the *C/C++ -> General*.
7. On the *Additional Include Directories* tab click, select edit and add the following:  
 $\$(SolutionDir)/External Libraries/GLEW/include$   
 $\$(SolutionDir)/External Libraries/GLFW/include$   
 To accept.
8. Select the *Linker -> General -> Additional library directories*, select edit and the following should be added:  
 $\$(SolutionDir)/External Libraries/GLEW/lib/Release/Win32$   
 $\$(SolutionDir)/External Libraries/GLFW/lib-vc2015$   
 $lib;assimp/lib;SOIL2/lib$   
 To accept..
9. On the *Linker* tab -> *Input -> Additional dependencies* and add: `soil2-debug.lib;assimp-vc140-mt.lib;opengl32.lib;glew32.lib;glfw3.lib;`  
 To accept..
10. Finally, the project is executed by pressing the Local Windows Debugger button.



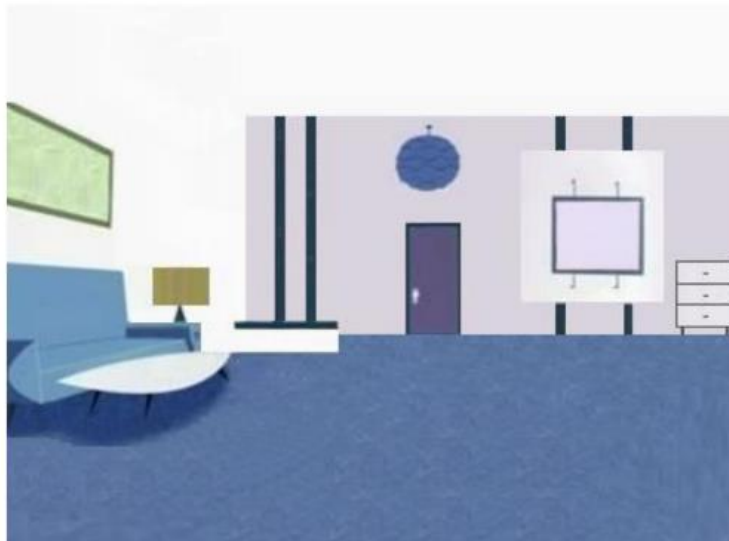
### 3. Reference images

For the creation of the virtual environment, the following reference images are shown to design the respective selected scenarios (facade and two rooms) and the 10 elements to be recreated are listed.

Facade



Room 1

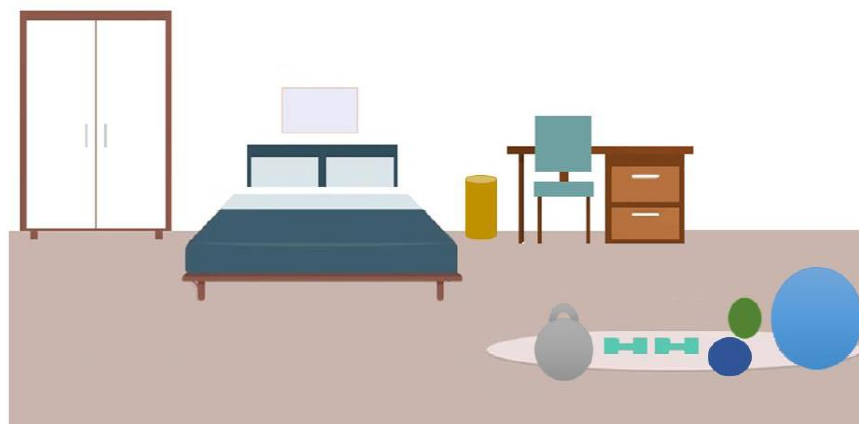


## Elements

1. Picture
2. Armchair
3. TV
4. Lamp
5. Table

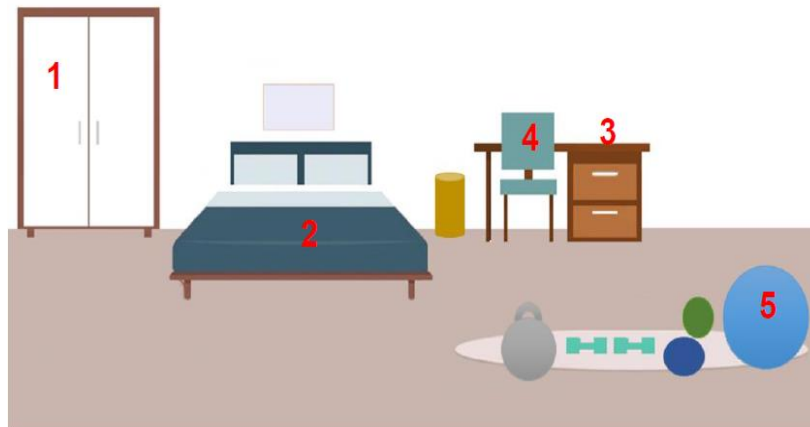


## Room 2



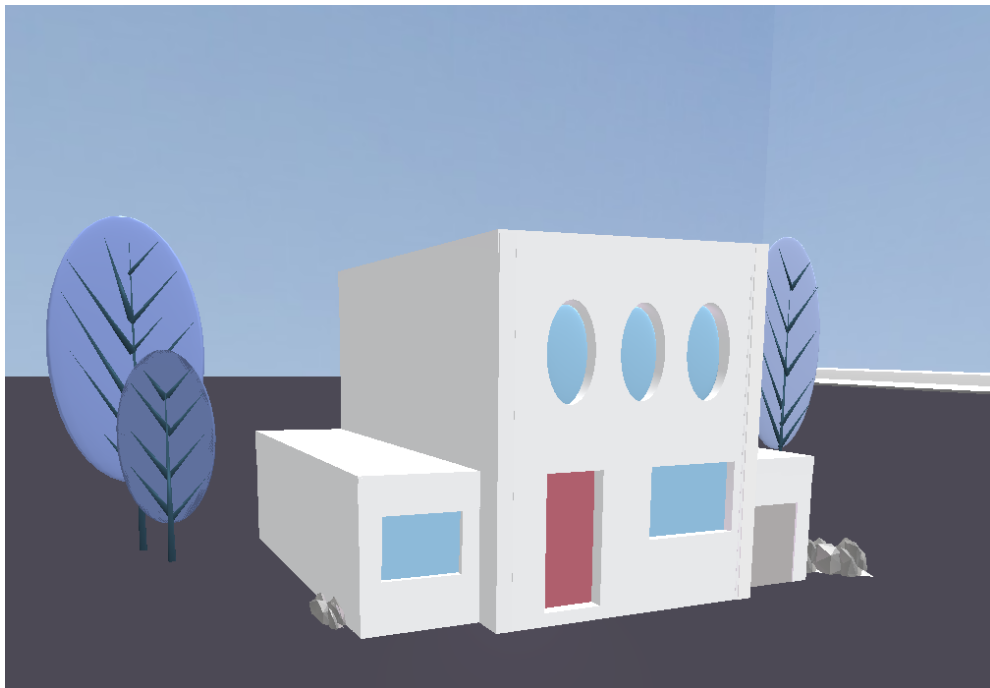
## Elements

1. Wardrobe
2. Bed
3. Desk
4. Chair
5. Ball

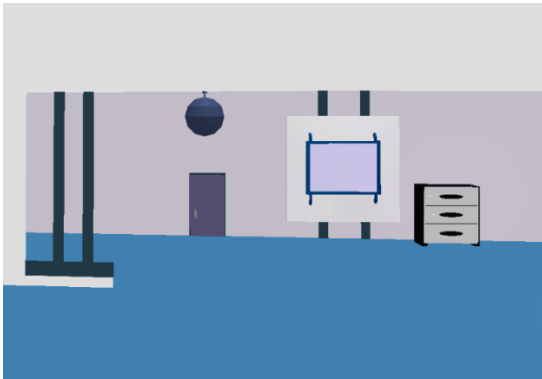


## Results

## Facade



Room 1



Room 2



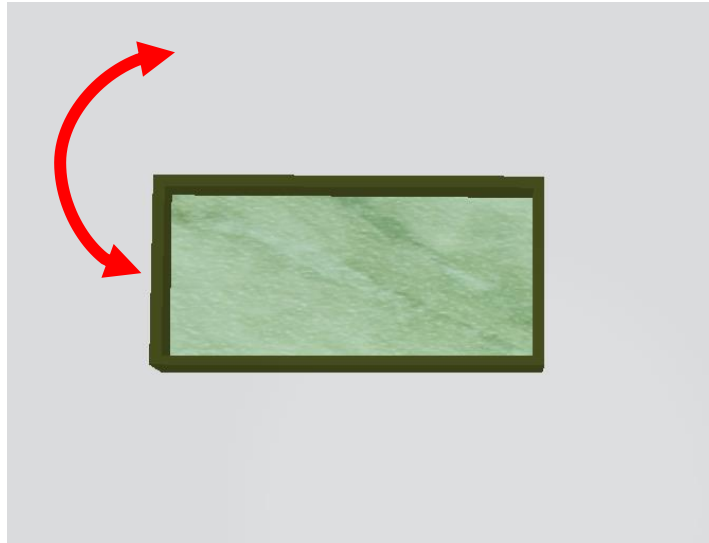
#### 4. Animations

Below are 4 animations considered in this project, each one is based on the movements we make in our homes when carrying out a change in the location of furniture or objects.



### 4.1 Animation 1

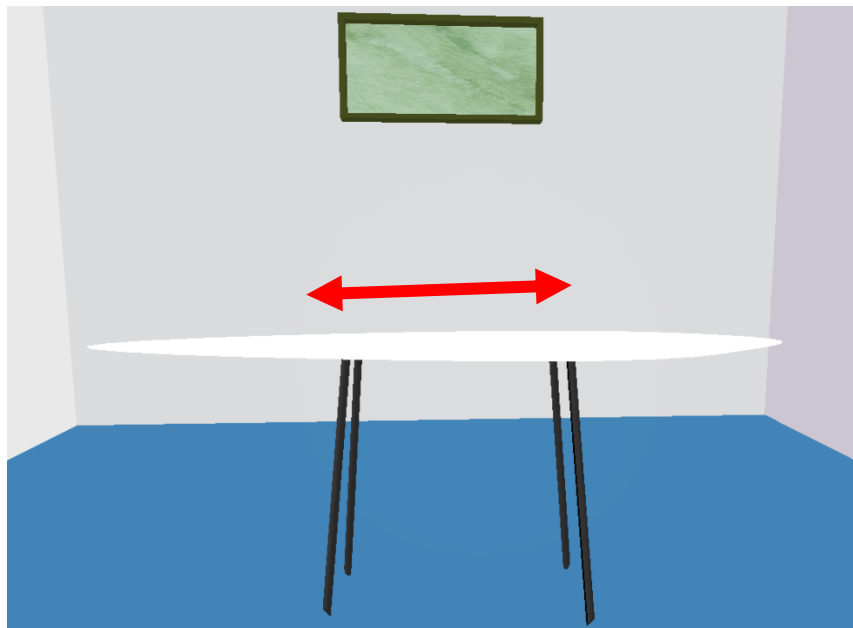
This animation consists of a box that makes a vertical movement and is located on one side of the dividing wall.



To start this animation, you need to press the “I” key, to stop it, just press “I” again.

### 4.2 Animation 2

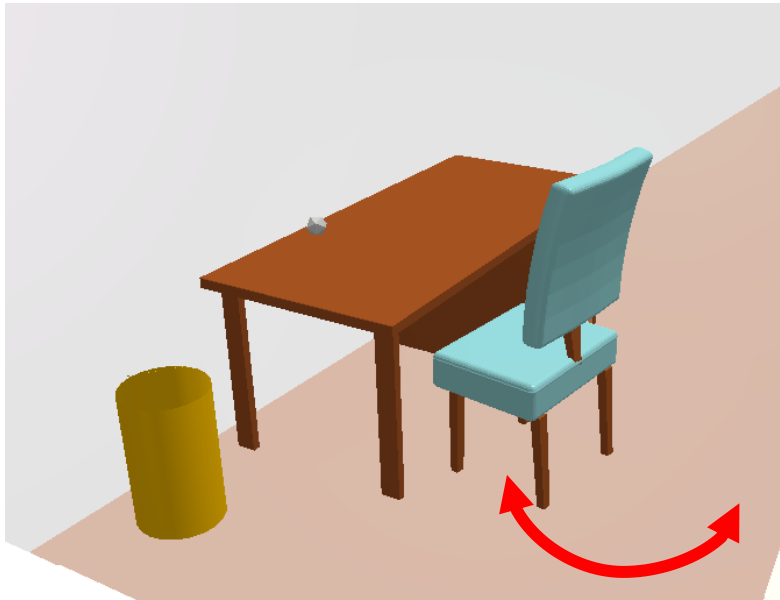
It is a table that makes a horizontal movement.



It is necessary to press the "O" key to activate the animation, to pause it you must press the same key.

### **4.3 Animation 3**

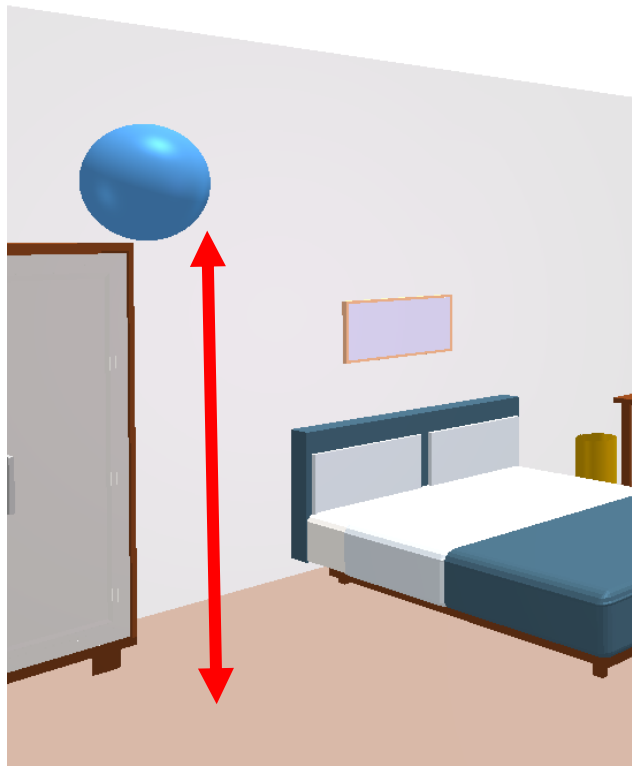
Pressing the letter "P" on the keyboard causes a movement with the chest of drawers that is located near the main entrance.



To stop or restart the animation, you must press the same "P" key.









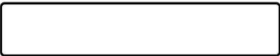








### **4.4 Animation 4**

When the "U" key is pressed, this animation begins, causing the ball to free fall and bounce off the floor until it loses height. Pressing "U" again restarts the animation.



## 5 Keyboard overview

Key	Action	Description
I	Animation 1	Movement picture
O	Animation 2	Movement table
P	Animation 3	Movement Furniture
U	Animation 4	Ball free fall
W	Forward	Forward movement in the environment

	Back	Movement backwards in the environment
	Right	Movement to the right in the environment
	Left	Movement to the left in the environment
	End of execution	End of application execution
	Forward	Forward movement in the environment
	Back	Movement backwards in the environment
	Right	Movement to the right in the environment
	Left	Movement to the left in the environment
	PointLight turns on	Turn on the 4 PointLight
	PointLight Movement	Movement in x axis + of PointLight left side rear
	PointLight Movement	Movement in x axis - of PointLight left side rear
	PointLight Movement	Movement in x axis + of PointLight left side front
	PointLight Movement	Movement in x axis - of PointLight left side front
	PointLight Movement	Movement in x axis + of PointLight right side rear
	PointLight Movement	Movement in x axis - of PointLight right side rear
	PointLight Movement	Movement in x axis + of PointLight right side front
	PointLight Movement	Movement in x axis - of PointLight right side front

## **Technical manual**

### **6 Objective**

The student must apply and demonstrate the knowledge acquired throughout the Computer Graphics and Human Computer Interaction course, using tools such as OpenGL, Maya, Visual Studio and GIMP for the 3D recreation of a virtual environment, consisting of a facade and a room where At least 10 objects that are most similar to the reference image must be displayed.

### **7 Diagrama de Gantt**

The management and monitoring of a Gantt chart was used to expedite and meet delivery objectives, contemplate the tasks, the duration of each activity so that they are adequately fulfilled, as well as the scheduled delivery dates.

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 Scope of the project

This document describes the content in the final project of the Computer Graphics and Human-Computer Interaction laboratory, which includes recreating a facade, two rooms and 10 objects in a virtual environment, considering all the necessary phases for the development of the project to be performed in a timely manner, defining each task and carrying out constant monitoring of the work carried out in accordance with the Gantt diagram. The project will have a maximum duration of approximately 7 weeks and must be delivered by a deadline of January 10, 2023.

In addition, a description of each action or activities to be carried out is made and the interactions that will be developed are followed up. Likewise, to prepare a user and technical manual in Spanish and English, which allows people a good understanding, as well as the use of the functionalities that it has.

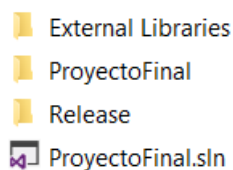
## 9 Limitations

- Affectation in times of work in the project, be it due to bad weather, work or school matters.
- Apply many corrective actions or modifications.
- Loss of files or documents.

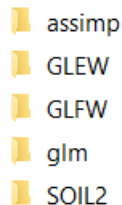
## 10 Documentation

### Project structure

The project has the following structure



During the development the respective libraries were used

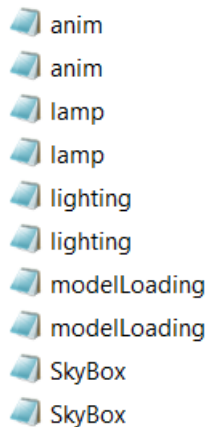


- **Assimp:** is a library that will help us load 3D models or scenes stored in a variety of formats, in this case Wavefront Object (\*.obj) is used.

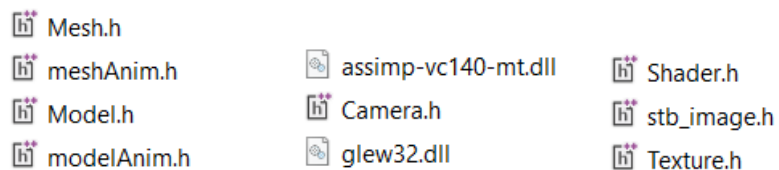
Assimp can load information on vertices, texture coordinates, normals, materials, animation, and others.

- **GLEW:** is a cross-platform library written in C/C++, intended to help load and query OpenGL extensions.
- **GLFW:** is a lightweight utility library for use with OpenGL. It provides the ability to create and direct OpenGL windows and applications, as well as receive keyboard and mouse input.
- **glm:** It defines on C/C++ the data types used, as well as numerous functions and operators associated with these data types.
- **SOIL2:** is a library mainly used to load textures in OpenGL.

Likewise, the respective shaders that were used for lighting, animations and the skybox were used.



In addition, there are dll and header files that make the lighting, textures, libraries, models and camera work correctly.





To carry out the programmed activities, folders were made to add each of the models implemented with OpenGL



## Code

All the libraries and shaders that are used during the development of the project are declared in the code.

```
#include <iostream> // Processing of inputs and outputs in the form of a sequence of
bytes.
#include <cmath> // Set of functions to perform mathematical operations

// GLEW
#include <GL/glew.h> // Determines which OpenGL extensions are supported by the
platform

// GLFW
#include <GLFW/glfw3.h> // Allows you to create and manage windows

// Other Libs
#include "stb_image.h" // Image processing

// GLM Mathematics
#include <glm/glm.hpp> //Use of different libraries given by C
#include <glm/gtc/matrix_transform.hpp> //Define functions that generate
transformation matrices
#include <glm/gtc/type_ptr.hpp> // Handles the interaction between pointers and
vectors, types of arrays

//Load Models
#include "SOIL2/SOIL2.h" // Loading textures within OpenGL

// Other includes
#include "Shader.h" // It is executed in one stage of the process
#include "Camera.h" // Camera loading within the scene
#include "Model.h" //Loading models in OpenGL
#include "Texture.h" // Loading textures in OpenGL
#include "modelAnim.h" // Loading animations in OpenGL
```

Subsequently, the function prototypes are declared.

```
void KeyCallback(GLFWwindow *window, int key, int scancode, int action, int mode);  
//Keyboard reading  
  
void MouseCallback(GLFWwindow *window, double xPos, double yPos); //Mouse reading  
  
void DoMovement(); //Movement of camera positions based on user input  
  
void animacion(); //Use for animation
```

As a next step, the dimensions of the window where the result will be displayed are defined. Where 1200 x 1000 pixels are used in total.

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

Likewise, the variables of the camera and its initial position within the scene are specified.

```
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;
```

On the other hand, there is detail about the shaders that were used for lighting, animation and 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");
```

## Models

The next step is to specify the call of each of the models with .obj format and that are imported from the Maya software.

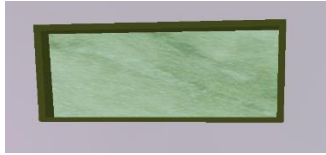
```
Model Casa((char*)"Models/Casa/casa.obj");
Model Cuadro((char*)"Models/Cuadro/cuadro.obj");
Model Esfera((char*)"Models/Esfera/esfera.obj");
Model Sillon((char*)"Models/Sillon/sillon.obj");
Model Piso((char*)"Models/Esfera dragon/Piso.obj");
Model Lampara((char*)"Models/Lampara/lampara.obj");
Model 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");
```

A load of all the previously declared models is carried out, basic transformations (rotation, translation or scale) are used as the case may be.

```
// Armchair
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f)); //Scale is changed
model = glm::translate(model, glm::vec3(-3.0f, 0.3f, -6.0f)); // A translation is
made to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Sillon.Draw(lightningShader); // The model is drawn
```



```
// Picture
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(3.0f, 2.0f, 3.0f)); //Scale is changed
model = glm::translate(model, glm::vec3(-4.0f, 1.1f, -5.65f)); // A translation is
made to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Cuadro.Draw(lightningShader); // The model is drawn
```



```
// Table
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 4.0f, 1.0f)); //Scale is changed
model = glm::translate(model, glm::vec3(-24.0f, 0.1f, -15.2f)); // A translation is
made to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Mesa.Draw(lightningShader); // The model is drawn
```



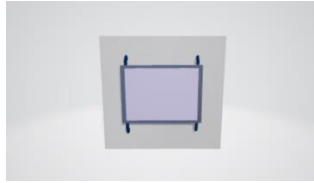
```
// Lamp
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.5f, 1.5f, 1.5f)); //Scale is changed
model = glm::translate(glm::mat4(1.0f), glm::vec3(-23.0f, 4.8f, -15.0f)); // A
translation is made to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Lampara.Draw(lightningShader); // The model is drawn
```



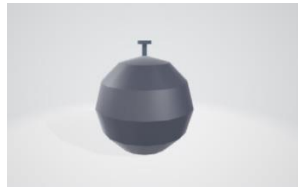
```
// Furniture
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.1f, 1.1f, 1.3f)); //Scale is changed
model = glm::translate(model, glm::vec3(11.0f, 0.2f, -8.5f)); // A translation is
made to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Mueble.Draw(lightningShader); // The model is drawn
```



```
// TV
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.5f, 1.5f, 1.5f)); //Scale is changed
model = glm::translate(model, glm::vec3(0.6f, 2.0f, -7.0f)); // A translation is made
to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Pantalla.Draw(lightningShader); // The model is drawn
```



```
// Sphere
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(0.9f, 0.9f, 0.9f)); //Scale is changed
model = glm::translate(model, glm::vec3(-4.5f, 8.8f, -11.9f)); // A translation is
made to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Esfera.Draw(lightningShader); // The model is drawn
```



```
// Wardrobe
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(2.5f, 2.5f, 2.5f)); //Scale is changed
model = glm::rotate(model, glm::radians(-180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //A
rotation is performed
model = glm::translate(model, glm::vec3(17.0f, 0.9f, 45.7f)); // A translation is
made to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Armario.Draw(lightningShader); // The model is drawn
```



```
// Bed
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Scale is changed
model = glm::rotate(model, glm::radians(178.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //A
rotation is performed
model = glm::translate(model, glm::vec3(15.0f, 2.5f, 82.7f)); // A translation is
made to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Cama.Draw(lightningShader); // The model is drawn
```



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

```

model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Scale is changed
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //A
rotation is performed
model = glm::translate(model, glm::vec3(-14.0f, 2.7f, 77.8f)); // A translation is
made to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Escritorio.Draw(lightningShader); // The model is drawn

```



```

// Chair
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Scale is changed
model = glm::rotate(model, glm::radians(240.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //A
rotation is performed
model = glm::translate(model, glm::vec3(-85.5f, 1.5f, 32.5f)); // A translation is
made to the model
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); // The model is drawn

```



```

// Trash can
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Scale is changed
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //A
rotation is performed
model = glm::translate(model, glm::vec3(-10.0f, 1.5f, 97.1f)); // A translation is
made to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Objeto6.Draw(lightningShader); // The model is drawn

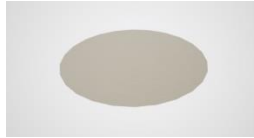
```



```

// Rug
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.5f, 1.7f, 1.2f)); //Scale is changed
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //A
rotation is performed
model = glm::translate(model, glm::vec3(-5.7f, 1.4f, 71.0f)); // A translation is
made to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Objeto1.Draw(lightningShader); // The model is drawn

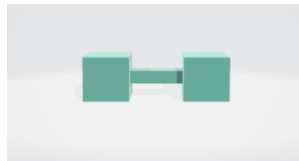
```



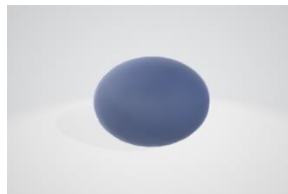
```
// Dumbbell
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Scale is changed
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //A
rotation is performed
model = glm::translate(model, glm::vec3(1.0f, 2.5f, 72.0f)); //A translation is made
to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Objeto7.Draw(lightningShader); // The model is drawn
```



```
// Small dumbbell
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Scale is changed
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //A
rotation is performed
model = glm::translate(model, glm::vec3(3.0f, 2.5f, 74.0f)); //A translation is made
to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Objeto8.Draw(lightningShader); // The model is drawn
```



```
// Blue ball
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Scale is changed
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //A
rotation is performed
model = glm::translate(model, glm::vec3(1.7f, 2.6f, 73.6f)); //A translation is made
to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Objeto3.Draw(lightningShader); // The model is drawn
```

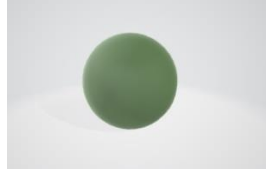


```
// Green ball
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Scale is changed
```

```

model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //A
rotation is performed
model = glm::translate(model, glm::vec3(1.5f, 2.6f, 75.4f)); // A translation is made
to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Objeto4.Draw(lightningShader); // The model is drawn

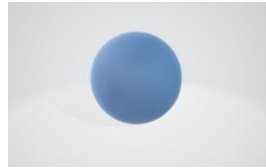
```



```

// Big blue ball
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f)); //Scale is changed
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //A
rotation is performed
model = glm::translate(model, glm::vec3(1.5f, 2.7f, 75.4f)); // A translation is made
to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Objeto5.Draw(lightningShader); // The model is drawn

```



```

// Picture 2
model = glm::mat4(1);
model = glm::scale(model, glm::vec3(1.2f, 1.0f, 1.0f)); //Scale is changed
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f)); //A
rotation is performed
model = glm::translate(model, glm::vec3(-1.2f, 8.0f, 77.0f)); // A translation is
made to the model
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Objeto2.Draw(lightningShader); // The model is drawn

```



## Animations

The variables used in each of the animations are declared and their function is specified.

```
//Animation PICTURE
```

<code>bool animacionCUADRO = false;</code>	Picture animation initialized to false
<code>bool regresoCUADRO = false;</code>	Return picture animation initialized to false



//Animation TABLE

<code>bool animacionMESA = false;</code>	Table animation initialized to false
<code>float pos = -5.0f;</code>	Definition of table position
<code>bool regresoMESA = false;</code>	Animation return of table initialized to false

// Animation CHAIR

<code>bool animacionSILLA = false;</code>	Chair animation initialized to false
<code>float rotSILLA = 10.7f;</code>	Chair Rotation Definition
<code>bool regresoSILLA = false;</code>	Animation return of chair initialized to false

//Animation BALL

<code>float auxiliar;</code>	It is used to finish the rebound of the sphere as the case maybe
<code>float PELOTA_x = -11.0f;</code>	Movement of the ball in x
<code>float PELOTA_z = 0.0f;</code>	Movement of the ball in z
<code>float PELOTA_y = 8.8f;</code>	Movement of the ball in y
<code>int est = 0;</code>	States of the ball
<code>bool animacionPELOTA = false;</code>	Ball animation initialized to false
<code>bool regresoPELOTA = false;</code>	Animation return of ball initialized to false
<code>float altura1 = 8.8f;</code>	Definition of the height 1 of the ball
<code>float Tiempo = 0.0f;</code>	Ball Time Definition
<code>float altura2 = altura1;</code>	Definition of the height 2 of the ball
<code>float velocidadPELOTA = 1.0f;</code>	Definition of the velocity of the ball

**Animation 1.** It consists of a picture that performs a vertically movement.

// Animation 1 PICTURE

```

    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;
        }
    }
}

```

To activate or stop the animation, simply press the “I” key.

```

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

```

**Animation 2.** It is a table that makes a horizontally movement.

```

//Animation 2 TABLE
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;
        }
    }
}
}

```

To start or pause the animation, you must press the “O” key.

```

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

```

**Animation 3.** By pressing the letter "P" on the keyboard, a movement is produced with the chair.

```
//Animation CHAIR
if (glfwGetKey(window, GLFW_KEY_P) == GLFW_PRESS)
    animacionSILLA ^= true;
    regresoSILLA = false;

// Animation 3 CHAIR
    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;
            }
        }
    }
}
```

**Animation 4.** When the "U" key is pressed, this animation begins, making the ball free fall and bounce off the floor until it loses height. Pressing "U" again restarts the animation.

```
//Animation BALL
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;
    }

```

```

//Animation 4 BALL

    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

The respective faces for the environment are specified, working with png and tga format.

```

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");

```

## Illumination

The position of the 4 PointLight that are in each of the corners of the facade is defined.

```

glm::vec3 pointLightPositions[] = {
    glm::vec3(44.0f,0.0f,23.0f), //left, front
    glm::vec3(-46.0f,0.0f,-95.0f), //left, back
    glm::vec3(-45.0f,0.0f,24.0f), //right, front
    glm::vec3(48.0f,0.0f,-95.0f) //right, back
};

```

The following variable is defined to turn on the PointLight when pressing the "space" key

<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);
    }
}
}

```

Key	Action	Description
<input type="text"/>	PointLight turns on	All 4 PointLight are activated and begin to flash

In addition, movement of the 4 PointLight occurs on the positive and negative x-axis. This can be done by pressing the following keys:

```

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

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

```

Key	Action	Description
<input type="text" value="1"/>	PointLight Movement	Movement in x axis + of PointLight left side rear
<input type="text" value="2"/>	PointLight Movement	Movement in x axis - of PointLight left side rear

```

//Left - front
if (keys[GLFW_KEY_3])
{
    pointLightPositions[2].x += 0.01f;
}

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

```

Key	Action	Description
3	PointLight Movement	Movement in x axis + of PointLight left side front
4	PointLight Movement	Movement in x axis - of PointLight left side front

```

//Right - back
if (keys[GLFW_KEY_5])
{
    pointLightPositions[3].x += 0.01f;
}

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

```

Key	Action	Description
5	PointLight Movement	Movement in x axis + of PointLight right side rear
6	PointLight Movement	Movement in x axis - of PointLight right side rear

```

//Right - front
if (keys[GLFW_KEY_7])
{
    pointLightPositions[0].x += 0.01f;
}
if (keys[GLFW_KEY_8])
{
    pointLightPositions[0].x -= 0.01f;
}

```

}

Key	Action	Description
7	PointLight Movement	Movement in x axis + of PointLight right side front
8	PointLight Movement	Movement in x axis - of PointLight right side front

## Functions

Each of the functions used is detailed and a brief description of its use is added.

Name	Description
<code>glfwMakeContextCurrent (GLFWwindow *window)</code>	Makes the specified window context current to the calling thread.
<code>glfwGetFramebufferSize (window, &amp;SCREEN_WIDTH, &amp;SCREEN_HEIGHT);</code>	Change the size of a window.
<code>glfwSetKeyCallback (window, KeyCallback)</code>	Sets the key callback of the specified window, which is called when a key is pressed, repeated, or released.
<code>glViewport(0, 0, SCREEN_WIDTH, SCREEN_HEIGHT);</code>	Sets the viewport.
<code>glfwWindowShouldClose(window)</code>	Returns the value of the close flag of the specified window.
<code>glAreTexturesResident (GLsizei n, const GLuint *textures, GLboolean *residences);</code>	Determine if the specified texture objects reside in texture memory.
<code>glArrayElement (GLint i);</code>	Specifies the array elements used to represent a vertex.
<code>glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);</code>	Specifies pixel arithmetic.
<code>glAlphaFunc (GLenum func, GLclampf ref);</code>	Allows the application to set the alpha testing function.
<code>glBegin (GLenum mode);</code>	They delimit the vertices of a primitive or a group of similar primitives.
<code>glBindTexture (GLenum target, GLuint texture);</code>	Allows you to create a named texture bound to a texture target.
<code>glBitmap (GLsizei width, GLsizei height, GLfloat xorig, GLfloat yorig, GLfloat xmove, GLfloat ymove, const GLubyte *bitmap);</code>	Draw a bitmap.
<code>glfwSwapBuffers(window);</code>	Swap the front and back buffers of the specified window when rendering with OpenGL
<code>glDrawArrays(GL_TRIANGLES, 0, 36);</code>	Specifies multiple primitives to render.



<code>glfwTerminate();</code>	Destroys all remaining windows and cursors, restores modified gamma ramps, and frees any other allocated resources.
<code>glTexCoord1d (GLdouble s);</code>	Sets the current texture coordinates.
<code>glfwInit();</code>	Initializes the GLFW library.
<code>glfwGetTime();</code>	Returns the value of the GLFW timer.
<code>glfwCreateWindow(WIDTH, HEIGHT, "Iluminacion 2", nullptr, nullptr);</code>	Create a window and its OpenGL context.
<code>glfwPollEvents();</code>	Processes only those events that are already in the event queue and then returns immediately.
<code>glTranslated (GLdouble x, GLdouble y, GLdouble z);</code>	Multiplies the current matrix by a translation matrix.
<code>glDisable(GL_BLEND);</code>	They disable OpenGL features.
<code>glClear (GLbitfield mask);</code>	Clears buffers at preset values.
<code>glDrawElements (GLenum mode, GLsizei count, GLenum type, const void *indices);</code>	Forces the execution of OpenGL functions in finite time.
<code>glFlush (void);</code>	Forces the execution of OpenGL functions in finite time.
<code>glLightf (GLenum light, GLenum pname, GLfloat param);</code>	Returns light source parameter values.
<code>glRotated (GLdouble angle, GLdouble x, GLdouble y, GLdouble z);</code>	Multiplies the current matrix by a rotation matrix.
<code>glClear (GLbitfield mask);</code>	Specifies a plane to which all geometry is clipped.
<code>glColorMaterial (GLenum face, GLenum mode);</code>	Causes a material color to keep track of the current color.
<code>glEndList (void);</code>	Create or replace a list to display.
<code>glNormal3b (GLbyte nx, GLbyte ny, GLbyte nz);</code>	Sets the current normal vector.
<code>glEnable(GL_BLEND);</code>	They enable OpenGL features.
<code>glScaled (GLdouble x, GLdouble y, GLdouble z);</code>	They multiply the current matrix by a general scaling matrix.
<code>glVertex2d (GLdouble x, GLdouble y);</code>	Specifies a vertex.
<code>glfwGetKey(window, GLFW_KEY_I)</code>	Returns the last reported state for the specified key to the specified window.
<code>glfwSetWindowShouldClose(window, GL_TRUE);</code>	Sets the value of the close flag for the specified window.
<code>glIndexf (GLfloat c);</code>	Sets the current color index.
<code>glAccum (GLenum op, GLfloat value);</code>	It works on the build buffer.

## 11 Costs

To determine the final cost of the project, the costs of labor, services and materials used in a period of 7 weeks of development of all the activities necessary for proper operation are considered.

CONCEPT	AMOUNT
FIXED COSTS	
Internet	\$ 700.00
Electricity	\$ 290.00
Salaries	\$ 35,000.00
Equipment maintenance	\$ 420.00
Software and PC	\$ 700.00
Subtotal	\$37,110.00
VARIABLE COSTS	
Materials	\$ 500.00
Supplies	\$ 150.00
Subtotal	\$ 650.00
Subtotal	\$ 37,760.00
IVA 16%	\$6,042.00
Utility	\$6,570.00
TOTAL	\$50,370.00

## 12 GitHub link

All project files are located on GitHub. In addition, you can download the tablet, at the following link: [https://github.com/JocelynPR/313231706\\_ProyectoFinal\\_GPO05](https://github.com/JocelynPR/313231706_ProyectoFinal_GPO05)

## 13 References

Microsoft. (s.f.). Visual Studio. Recovered on September 14, 2022 from <https://visualstudio.microsoft.com/es/vs/older-downloads/>

Alex. G. (January 2004). Programming with OpenGL. Recovered on September 30, 2022 from [https://lc.fie.umich.mx/~rochoa/Manuales/OPEN\\_GL/TUTORIAL.pdf](https://lc.fie.umich.mx/~rochoa/Manuales/OPEN_GL/TUTORIAL.pdf)

Microsoft. (s.f.). Introduction to OpenGL. Recovered on November 17, 2022 from <https://learn.microsoft.com/es-es/windows/win32/opengl/introduction-to-opengl>

GLFW. (s.f.). Functions OpenGL. Recovered on November 30, 2022 from [https://www.glfw.org/docs/3.1/group\\_\\_window.html#ga37bd57223967b4211d60ca1a0bf3c832](https://www.glfw.org/docs/3.1/group__window.html#ga37bd57223967b4211d60ca1a0bf3c832)