National Autonomous of Mexico

Faculty of Engineering

Electrical Engineering Division

Career: Computer Engineering

Course: Computer Graphics Human-Computer interaction

(key: 1590)

Project 2. Final

Teacher: Engineer Carlos Aldair Roman Balbuena

Student:

- 318192709 -

Group: 05

Delivery date: May 20, 2025

Semester: 2025-2

index

1. INDEX

This document is part of the final project for the Theory class. The goal of the project was to design, model, and animate a virtual space using 3D design tools like Blender, and later integrate everything into an OpenGL programming environment. The main idea was to recreate a museum inside an ecological park.

During the project, several phases were completed from planning the timeline using a Gantt chart, to building and animating the models. A synthetic camera was implemented, and at least four functional animations were included. The rooms were modeled with a clear artistic vision, and custom objects were created for both the inside and outside areas, trying to make everything look realistic and immersive.

We also worked on extra parts like a cost analysis, technical documentation, a user manual to explain how the system works, and a public repository to share the final result. This document puts together all those elements and shows the full development process of the project.

2. PROJECT JUSTIFICATION

This project came from the need to fully apply what we've learned in computer graphics, 3D modeling, and visual programming. By building a virtual environment in this case, an interactive museum inside an ecological park the goal was to create a space that users could explore, combining modeling, animation, interaction, and logic in code.

The museum wasn't just made to look nice visually, but also to represent the culture of Xochimilco. That's why it includes traditional elements like trajineras and chinampas inside the exhibition hall. There's also a second room, a public restroom, which adds some realistic and practical aspects to the experience.

The challenge was to simulate a realistic environment from both a technical and artistic side. We used synthetic cameras, materials, textures, and OpenGL to bring

everything together, while also focusing on design, references, and a defined visual style. Blender helped us understand the whole 3D pipeline, from modeling to keyframe animation.

On top of that, the project helped us build skills like planning with a Gantt chart, using version control with GitHub, and documenting the process through manuals. Overall, this was a complete learning experience that mixed academic knowledge with tools we can actually use in real work later on.

3. OBJECTIVE

The main goal of this project is to design and develop an interactive virtual museum located in the Xochimilco Ecological Park, inspired by the cultural and natural richness of the area. The virtual space brings together 3D modeling, animations, a synthetic camera, and visual programming using OpenGL.

The museum includes two main areas: a cultural exhibition room and a public restroom. Each space features five custom-modeled objects, created following a consistent artistic style. The project aims to apply everything learned during the course from planning with a Gantt chart to full documentation with both a technical and user manual. It also includes a cost analysis and was published in a working GitHub repository as part of the final delivery.

4. METHODOLOGY

An incremental methodology was used to develop the virtual museum. This approach was ideal for building and organizing the system in a progressive, efficient, and controlled way.

The method allowed the work to be divided into several stages. Each part of the system (modeling, texturing, interaction, sound, etc.) was implemented and tested step by step. This helped to catch errors early, check that each element worked

visually and logically, and make improvements without breaking the rest of the progress.

Main implementation stages:

1. General planning
    - The theme of the museum was defined, the spaces to be modeled were selected, and a list of key objects was created.
2. 3D Modeling
    - Blender was used to model the museum's facade, the interior rooms, and all decorative and functional objects.
3. Texturing
    - Materials and textures were applied using UV mapping in Blender and GIMP.
4. Programming with OpenGL
    - Visual Studio Code was used as the development environment.
    - Libraries like GLAD, GLFW, GLM, and stb_image.h were added to support the rendering and interaction system.
5. Loading models and textures
    - The 3D models were exported as `.fbx` files from Blender and loaded into the project using custom-made classes.
6. Synthetic camera

- A first-person camera was implemented that responds to keyboard and mouse movement, allowing full exploration of the space.

7. Animations and interactions
    - Various interactive features were added:
        - Museum door opens and closes with the H/J keys.
        - Rotating ceiling fans.
        - Bathroom door opens and closes automatically based on proximity.
        - The trajinera spins when the user approaches.
        - A ghost appears when clicking on the book.
8. Visual and sound effects
    - A skybox using a cubemap was integrated, along with a procedural shader to simulate animated water.
    - Ambient background music was added using the irrKlang library.
9. Testing and adjustments
    - Behavior of all models, textures, and animations was tested.
    - Position, lighting, and scale of elements were adjusted to improve the visual experience.

5. Gantt Chart

To organize the development of this project, the work schedule was divided into three Gantt charts, one for each month, covering the full period from March 20th to May 20th. This format helped visualize the project's progress, showing each creative and technical stage more clearly and making it easier to manage time during every phase.

March

The initial part of the project took place in March. This phase focused on organization, planning, and all the basic setup needed to start the development of the virtual museum. In this stage, the general requirements were defined, different ideas for the museum's facade were explored, and a final design was selected as the base for the rest of the project. Also, the rooms and objects that would be included in the museum were chosen, and their general arrangement was planned.

Inicio del proyecto	20-mar-25
Fin del proyecto	20-may-25

PROYECTO FACHADA MUSEO PARQUE ECOLOGICO XOCHIMILCO

	Marzo											
ACTIVIDAD	20-mar-25	21-mar-25	22-mar-25	23-mar-25	24-mar-25	25-mar-25	26-mar-25	27-mar-25	28-mar-25	29-mar-25	30-mar-25	31-mar-25
Revisión de requerimientos												
Idea fachada												
Bocetos y concepto general												
Fachada seleccionada												
Selección de Cuarto y objetos												
Planeación general												
Familiarse software Blender												
Aprender modelado Blender												
Empezar modelado												

April

During the month of April, the focus was on modeling, texturing, and organizing the museum. Work began with the facade and the two main rooms, followed by the creation of several objects like the trajinera, sculptures, doors, and accessories. This stage was essential to shape the structure and atmosphere of the virtual environment before moving on to animations and interactivity.

Inicio del proyecto	20-mar-25		PROYECTO	FACHADA MUSEO PARQUE ECOLOGICO XOCHIMILCO								
Fin del proyecto	20-may-25											
ACTIVIDAD	01-abr-25	02-abr-25	03-abr-25	04-abr-25	05-abr-25	06-abr-25	07-abr-25	08-abr-25	09-abr-25	10-abr-25	11-abr-25	12-abr-25
Modelado fachada												
Ajustes fachada												
Modelado 2 cuartos												
Modelado y textura trajinera												
Modelado y textura banca												
Modelado y textura cuadros												
Modelado y textura esculturas												
Modelado y texturaventilador												
Modelado y textura WC												
Modelado y textura espejo												
Modelado y textura lampara												
Modelado y tectura puertas												
Modelado y textura lavamanos												
Acomodar cuartos con objetos												

	Abril																	
ACTIVIDAD	13-abr-25	14-abr-25	15-abr-25	16-abr-25	17-abr-25	18-abr-25	19-abr-25	20-abr-25	21-abr-25	22-abr-25	23-abr-25	24-abr-25	25-abr-25	26-abr-25	27-abr-25	28-abr-25	29-abr-25	30-abr-25
Modelado fachada																		
Ajustes fachada																		
Modelado 2 cuartos																		
Modelado y textura trajinera																		
Modelado y textura banca																		
Modelado y textura cuadros																		
Modelado y textura esculturas																		
Modelado y texturaventilador																		
Modelado y textura WC																		
Modelado y textura espejo																		
Modelado y textura lampara																		
Modelado y tectura puertas																		
Modelado y textura lavamanos																		
Acomodar cuartos con objetos																		

May

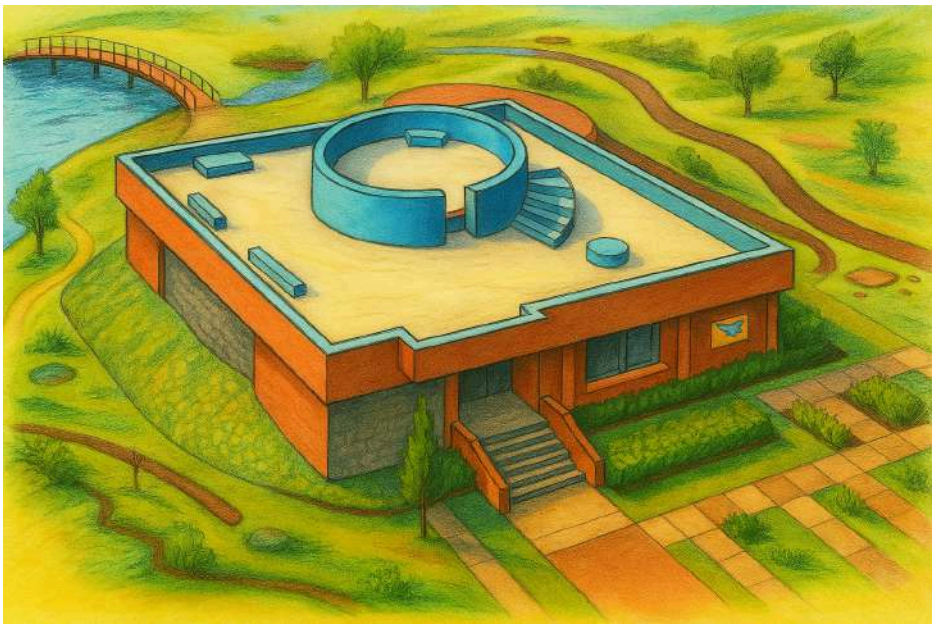
The month of May marked the final phase of the virtual museum's development. This period focused on programming the source code, integrating animations, exporting the complete environment into OpenGL, and generating a working executable.

Other important tasks were also completed during this month, such as creating the GitHub repository, uploading the full project, and finishing the final testing and documentation to ensure everything was ready for submission in the best possible condition.

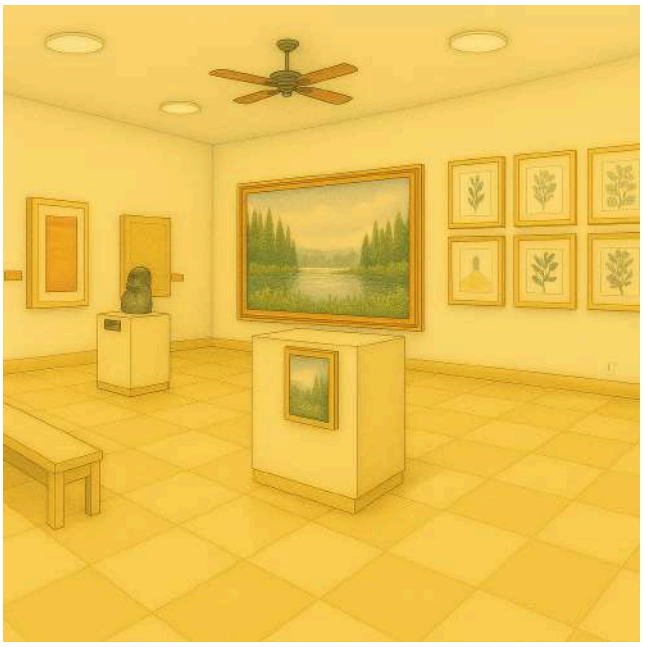
Inicio del proyecto	20-mar-25	PROYECTO	FACHADA MUSEO PARQUE ECOLOGICO XOCHIMILCO								
Fin del proyecto	20-may-25										
											Mayo
ACTIVIDAD		01-abr-25	02-abr-25	03-abr-25	04-abr-25	05-abr-25	06-abr-25	07-abr-25	08-abr-25	09-abr-25	10-abr-25	11-abr-25
Creacion de codigo fuente												
Exportar museo a OpenGL												
ajustes codigo fuente												
Animacion 1												
Animacion 2												
Animacion 3												
Animacion 4												
Modelado entorno para el museo												
textura entorno												
exportar entorno con museo												
creacion cuenta github												
subir modelos github y proyecto												
Pruebas finales												
Documentacion												

6. REFERENCE IMAGES

Facade



Room 1 - Exhibition room



Elements recreated in the exhibition room:

- Paintings
- Bench
- Ceiling fan
- Trajinera
- Sculptures

Room 2 - Museum Sanitary

Elements recreated in the restroom:

- Hanging lamps
- Sink
- Doors
- Toilets
- Mirror

7. TECHNICAL MANUAL

This technical manual provides a detailed overview of the inner workings of the Virtual Museum of the Xochimilco Ecological Park project, developed using open-source technologies. Its goal is to explain the technical components that make the system work, from 3D modeling and texture mapping to the implementation of source code, synthetic camera controls, interactive animations, and sound integration.

The manual covers the tools used, the operating logic, the most important code structures, and key technical features necessary for the project to run correctly and efficiently.

7.1Technical Objective and Scope of the System

The main technical objective of this project was to develop a fully functional and interactive virtual environment that represents the museum located inside the Xochimilco Ecological Park. The goal was not only to visually recreate the space but also to allow the user to interact with various elements in real time—featuring animations, proximity detection, a synthetic camera, and ambient sound.

From a technical perspective, the system covers the following scope:

- Enable real-time 3D visualization of the environment using OpenGL.
- Implement a free-moving synthetic camera controlled by keyboard and mouse.

- Integrate interactive animations such as:
    - Doors that open automatically.
    - A trajinera (boat) that rotates when the user gets close.
    - A ghost that appears after clicking on a book.
- Add ambient sound effects to enhance the immersive experience.
- Use custom 3D models created in Blender and exported in `.fbx` format for integration.

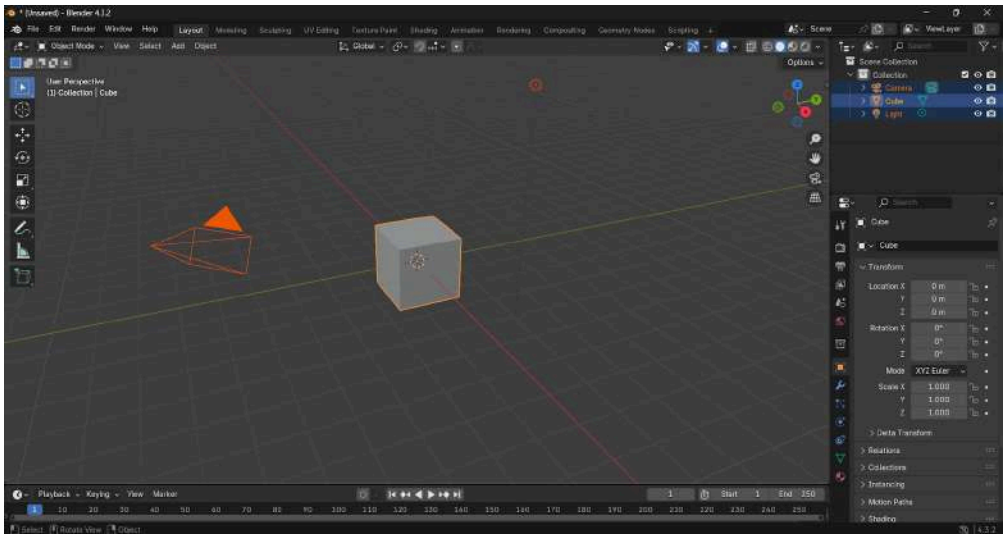
7.2 Modeling

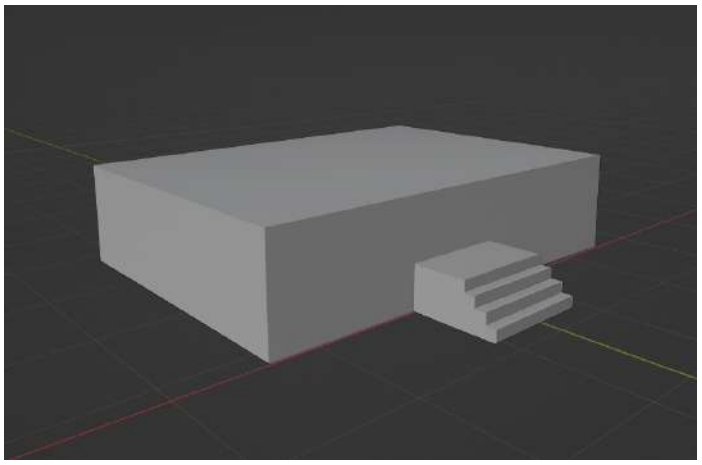
The 3D modeling of the museum was carried out using **Blender**. During this stage, all elements of the virtual environment were built from clean geometry, avoiding unnecessary intersections and maintaining a logical and organized structure within the scene.

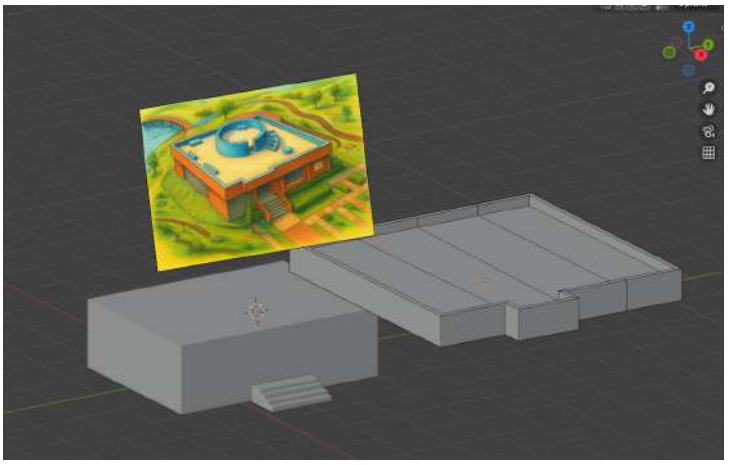
The main components of the environment that were modeled include:

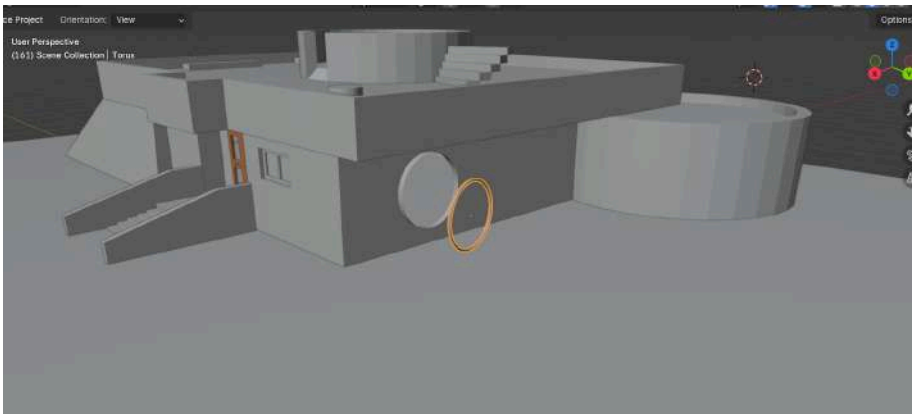
Facade

To build the museum, we started with Blender's default cube. This primitive shape was modified and stretched into a rectangular structure, as shown in the following images. The resulting geometry became the base of our museum's main building.



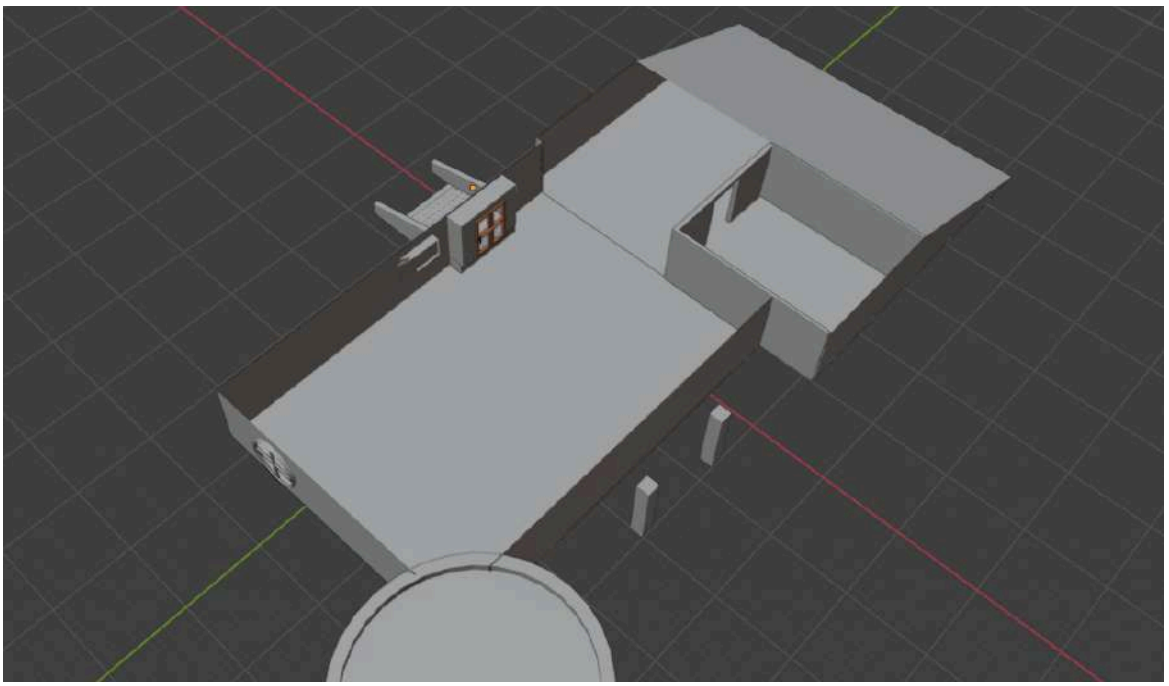





User Perspective
(161) Scene Collection | Torus

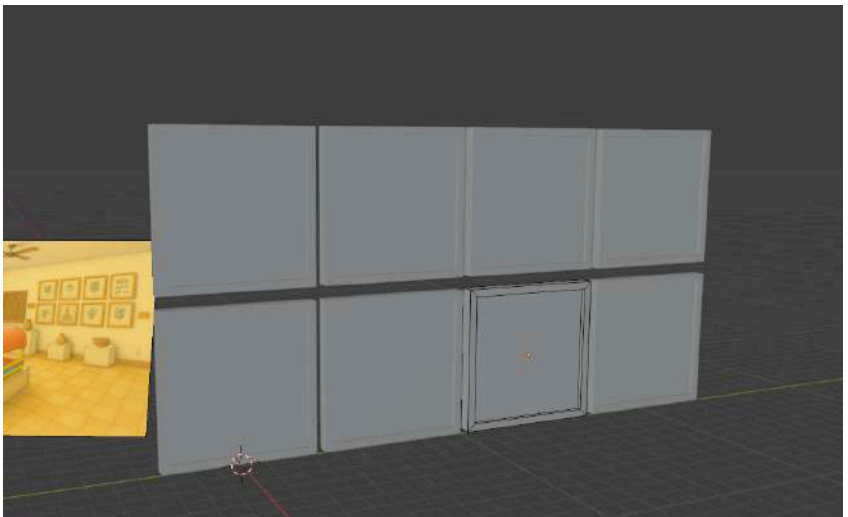
Interior of the museum

You can see the division of the two rooms, which will be used for our museum, the museum's exhibition hall room and later the bathroom.

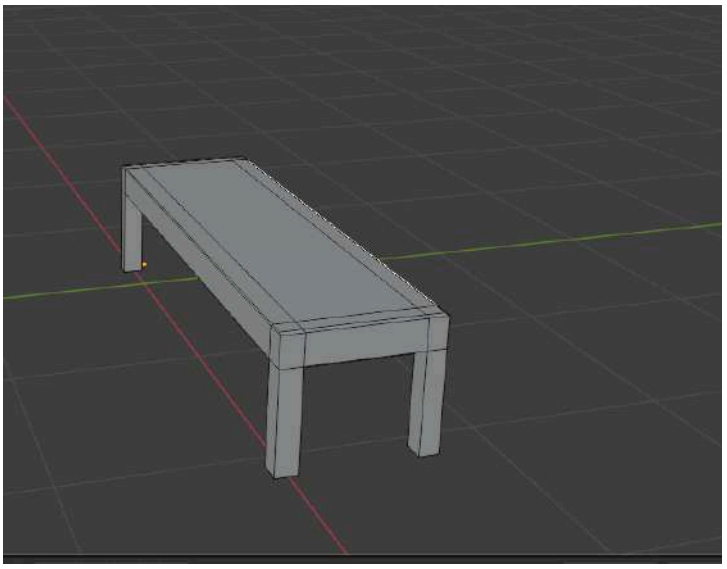


Modeling of the 5 objects for the exhibition room.

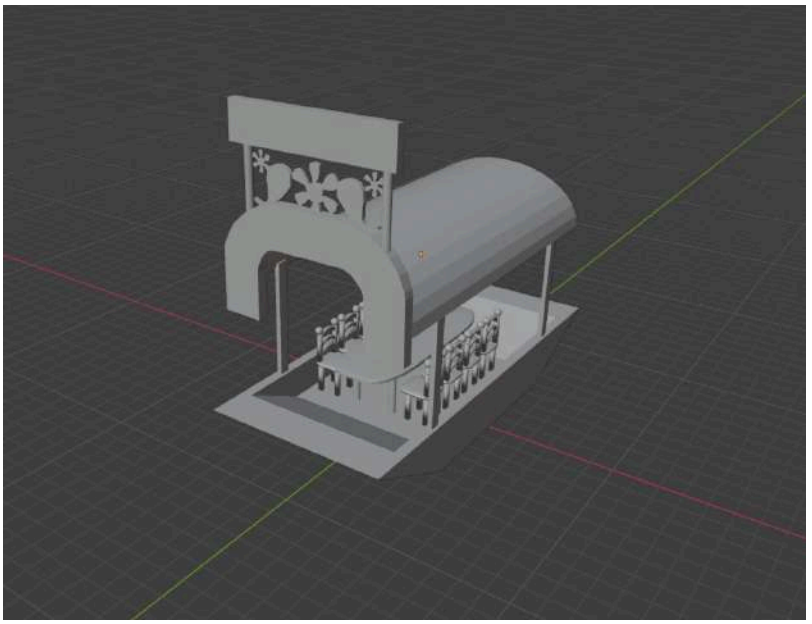
1. Paintings



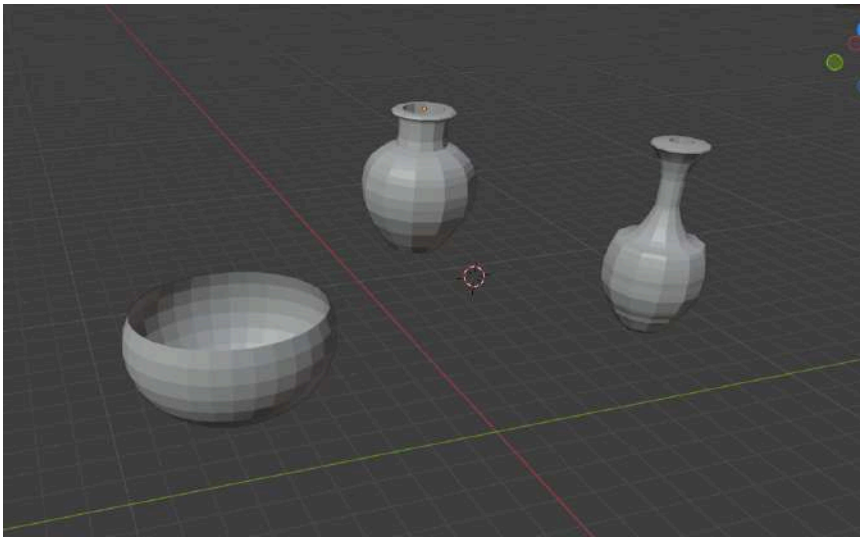
2. Bench



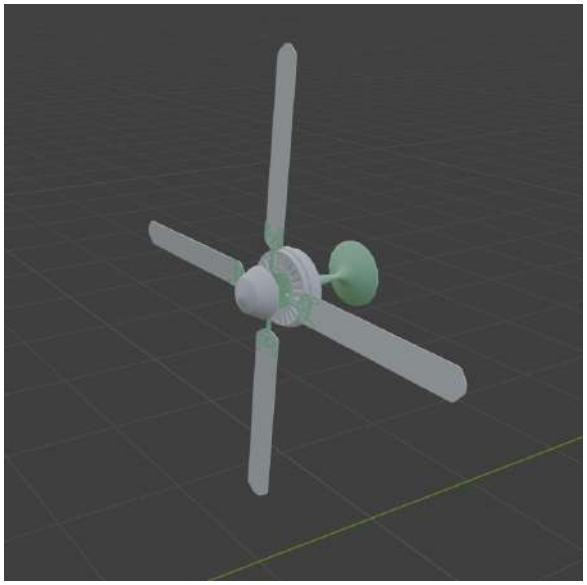
3. Trajinera

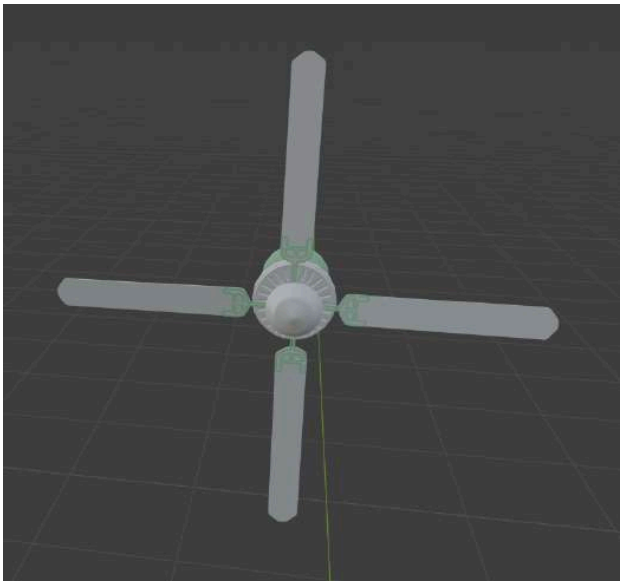


4. Sculptures



5. Ceiling fan

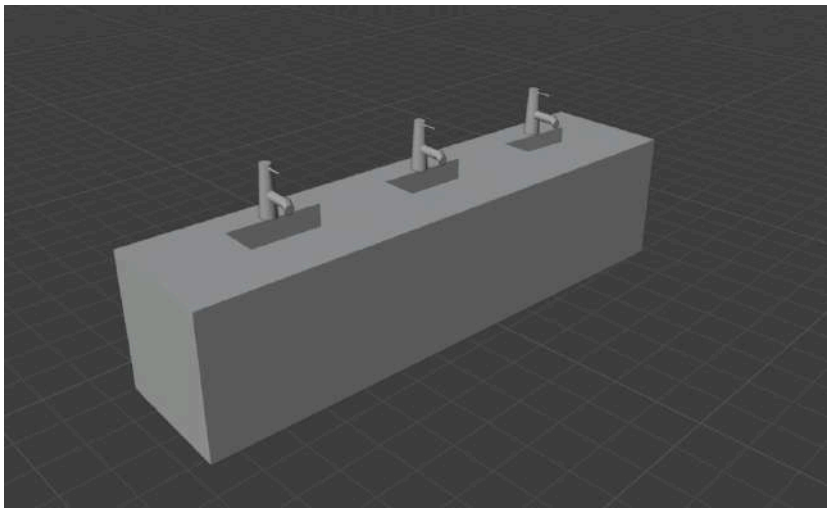


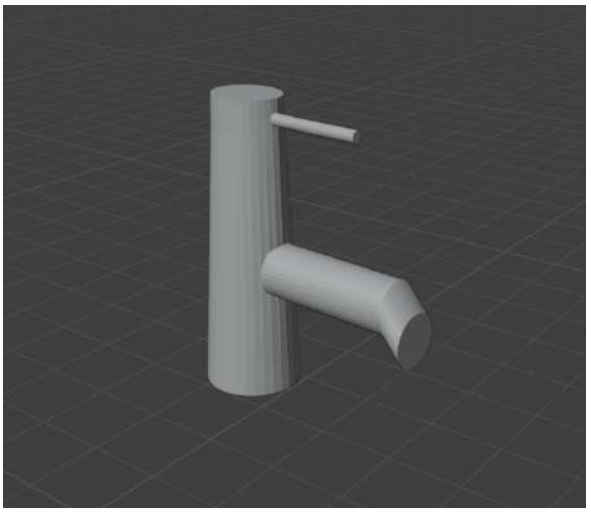


Modeling the 5 objects for the bathroom.

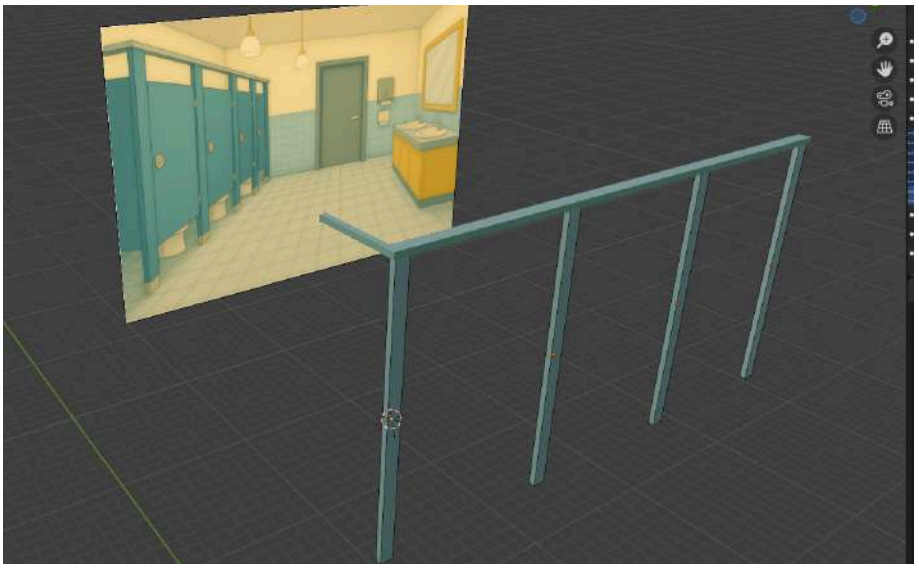
1. Hanging lamps

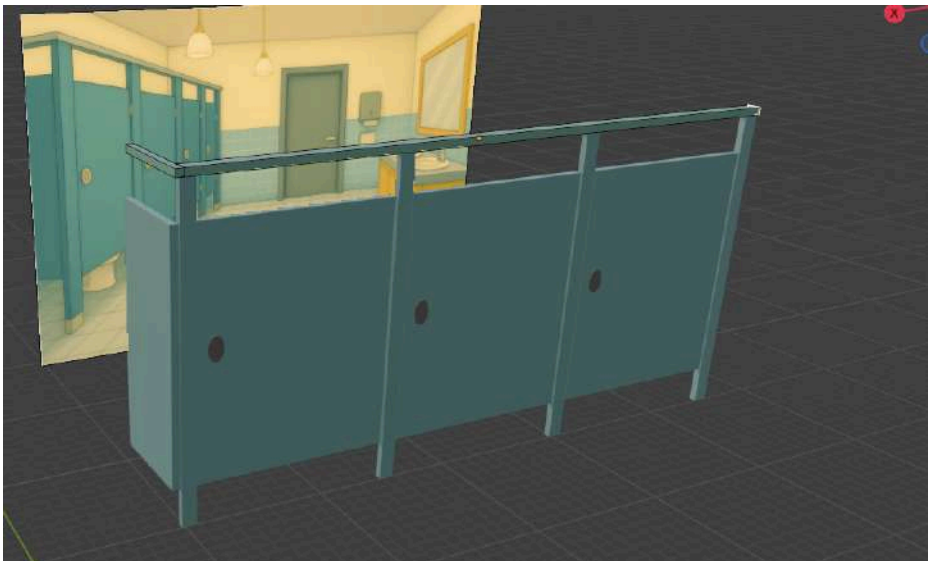
2. Sink



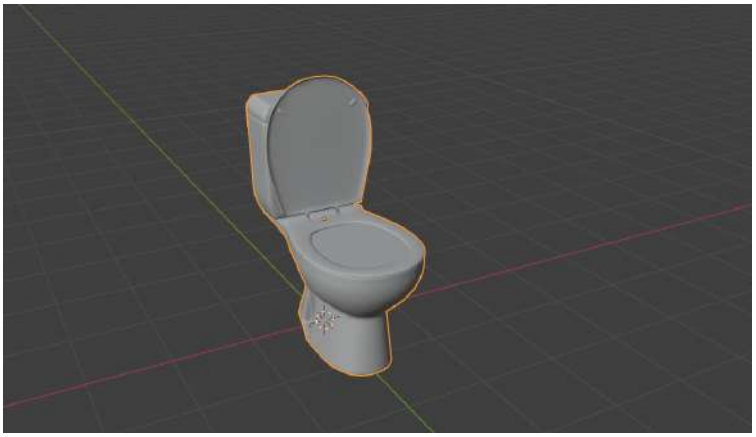


3. Doors

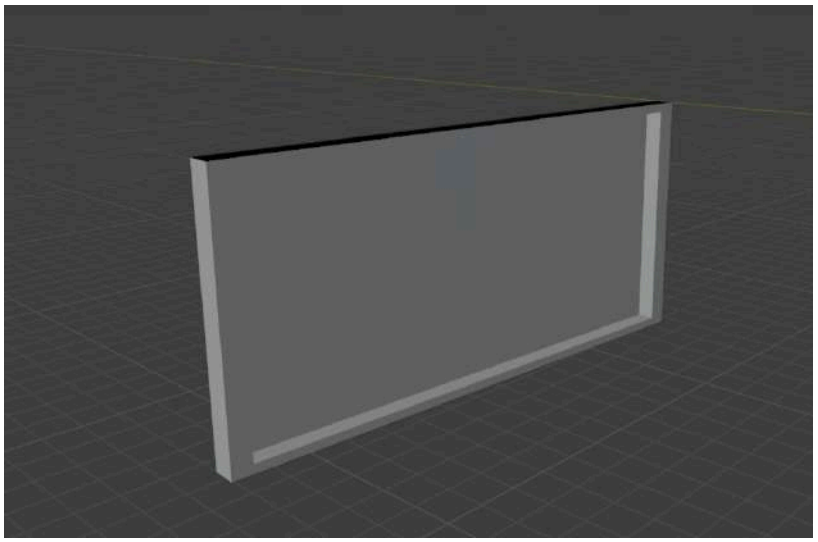




4. WC



5. Mirror

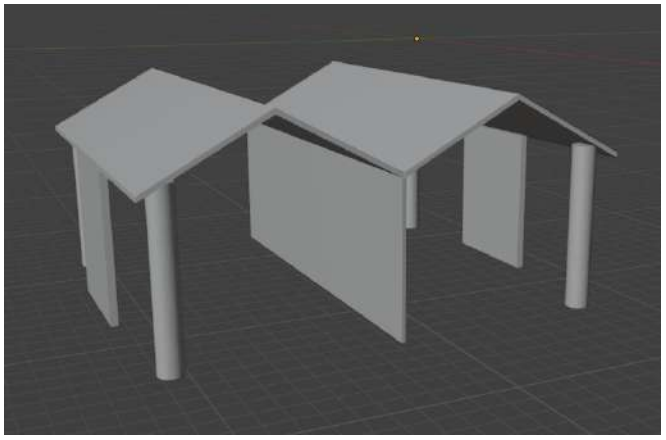


More objects were modeled for our museum complement, as well as the surroundings where our museum will be, so that it doesn't look so empty.

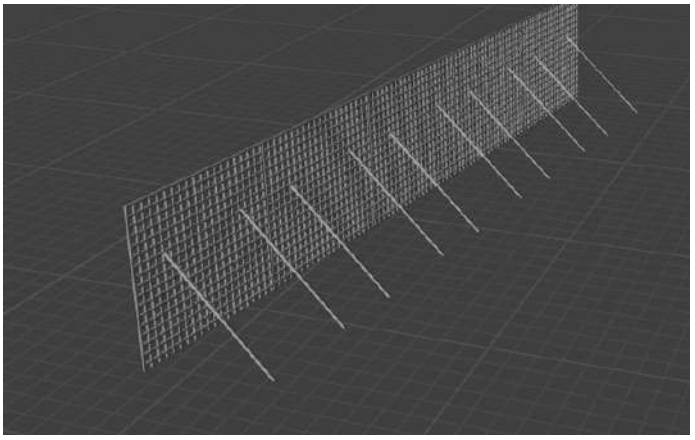
- Eagle

- Sing

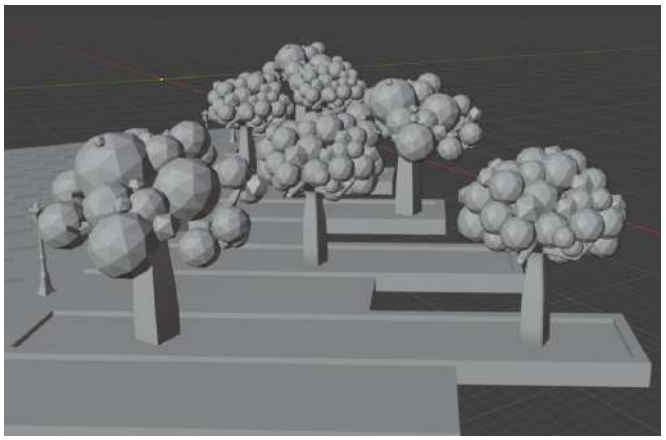
- Park entrance



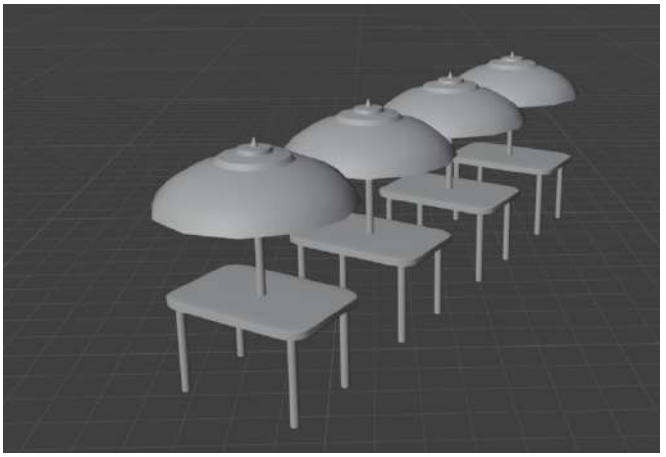
- Bars



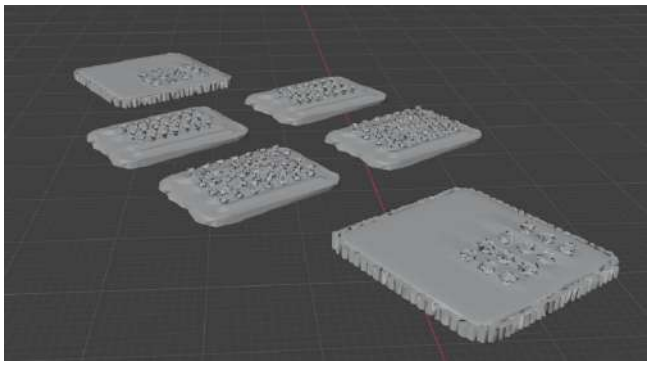
- Trees



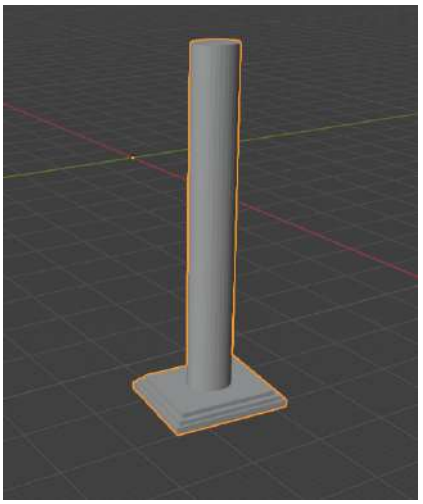
- Tables



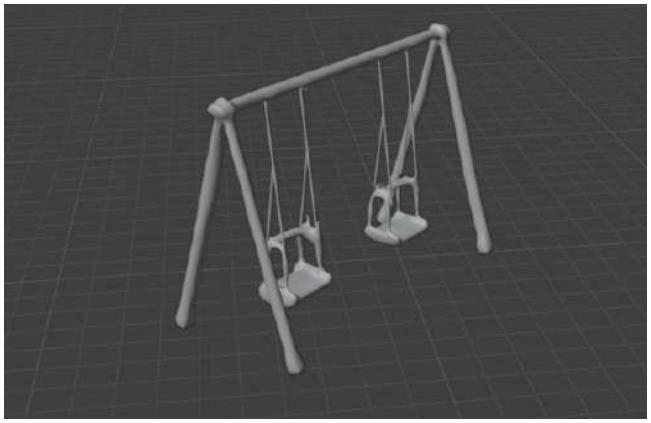
- Chinampas



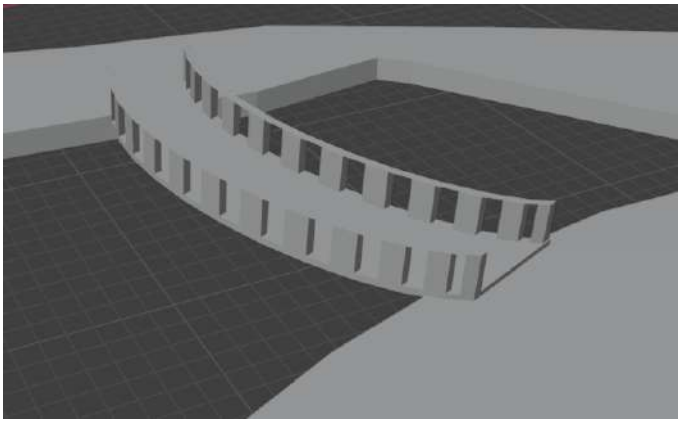
- Tower



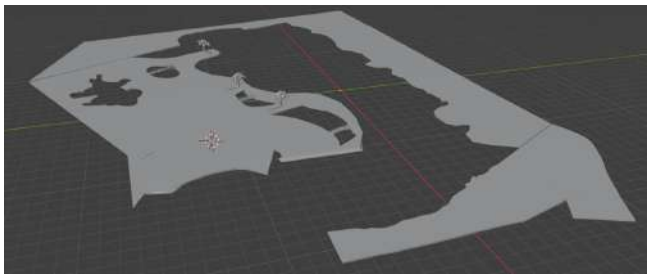
- Swings



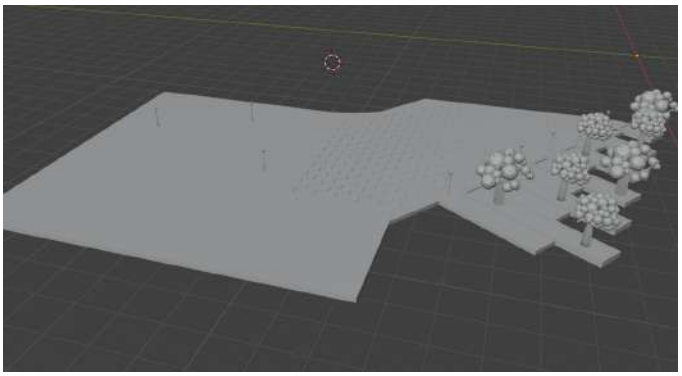
- Bridges



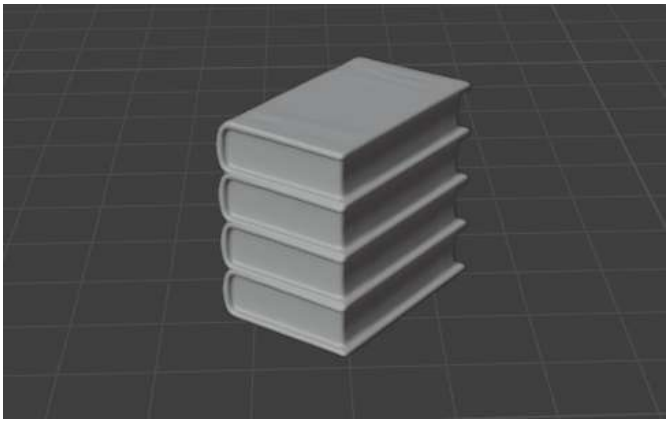
- Land



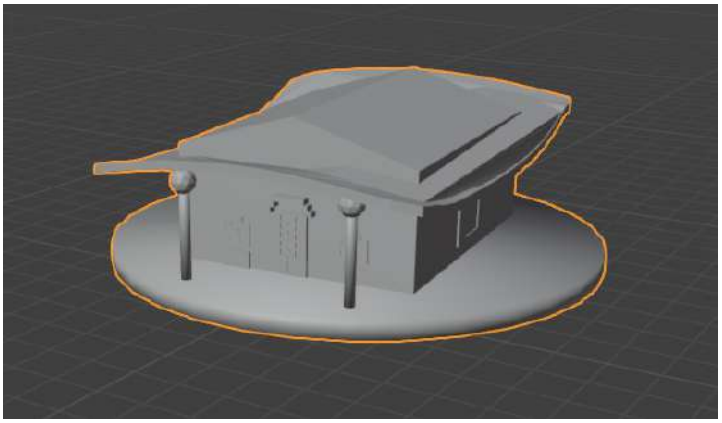
-Terrace



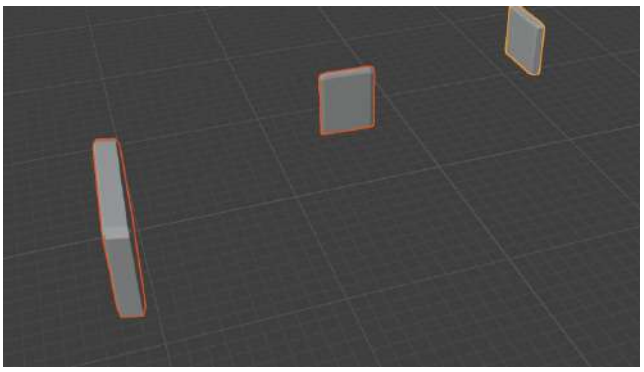
- Books



- School



- Murals



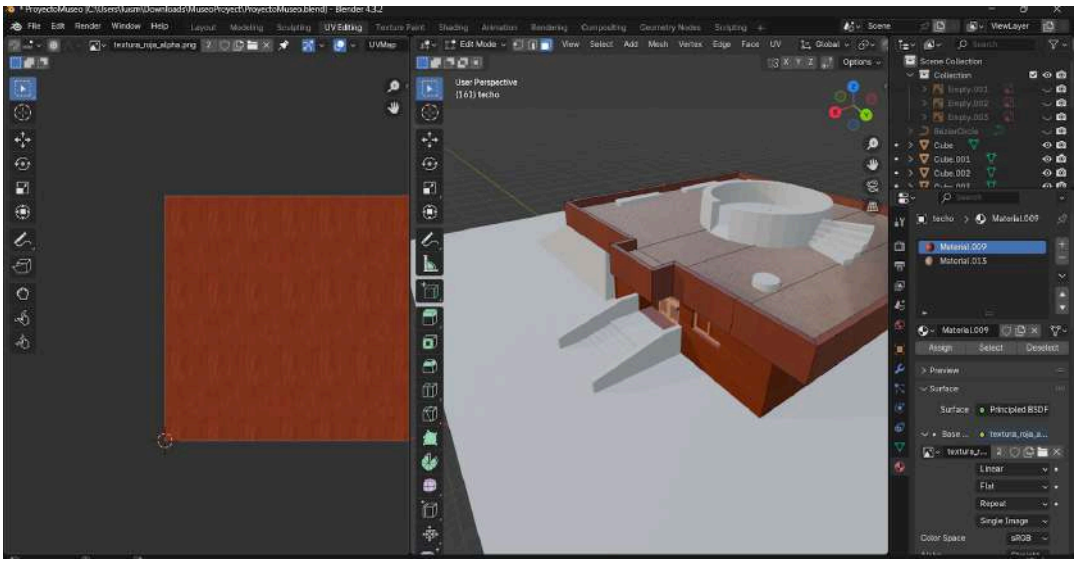
- Ghost

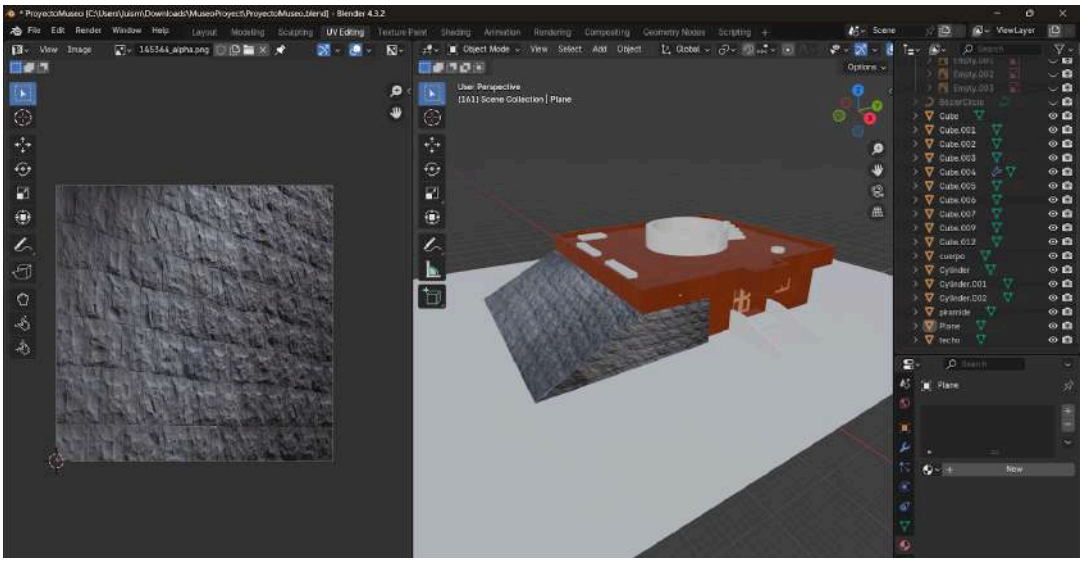
7.3 Texturiing and UV Mapping

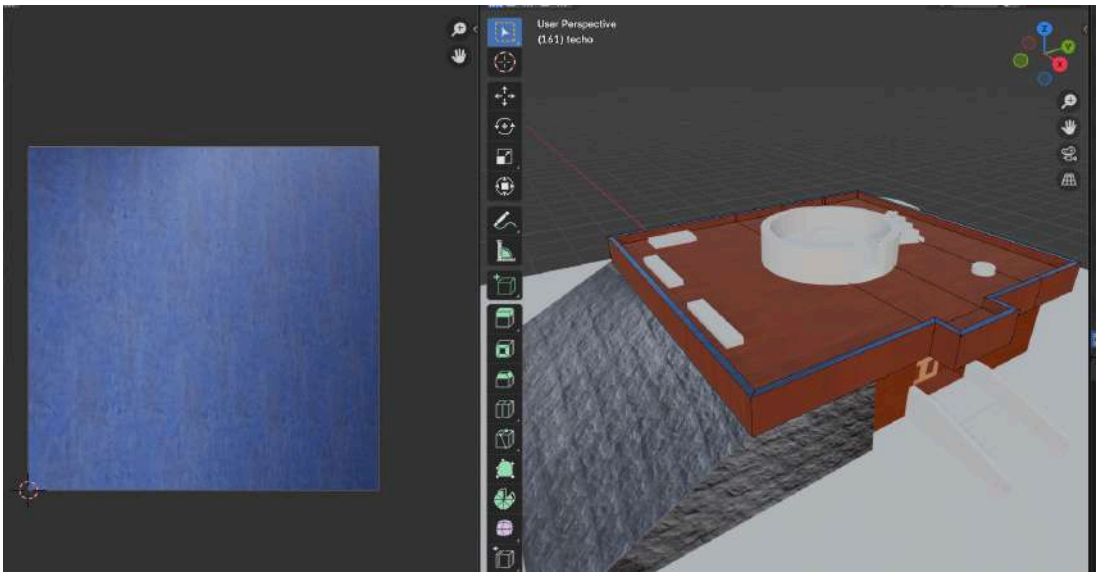
The texturing process was carried out using Blender software, with the goal of applying visually coherent and functional materials to each modeled object. Manual UV mapping was used for greater control over texture distribution, avoiding stretching, excessive repetitions, or scaling issues.

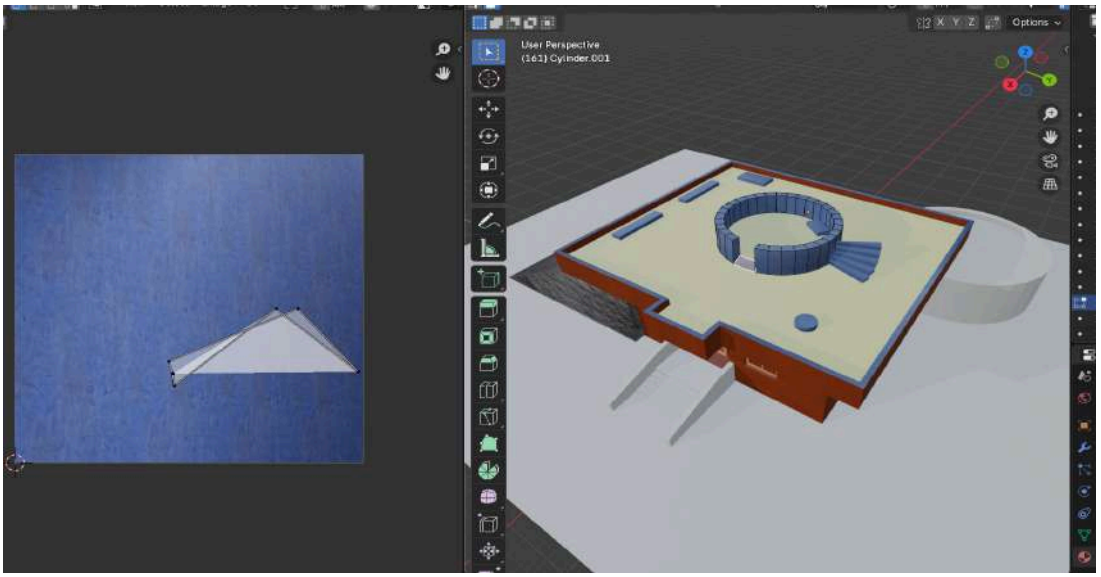
The textures used were images in .png and .jpg formats, edited with GIMP to reduce weight and improve contrast without losing quality. Resolutions between 512x512 px were used. The texturing process is shown below.

Facade texture

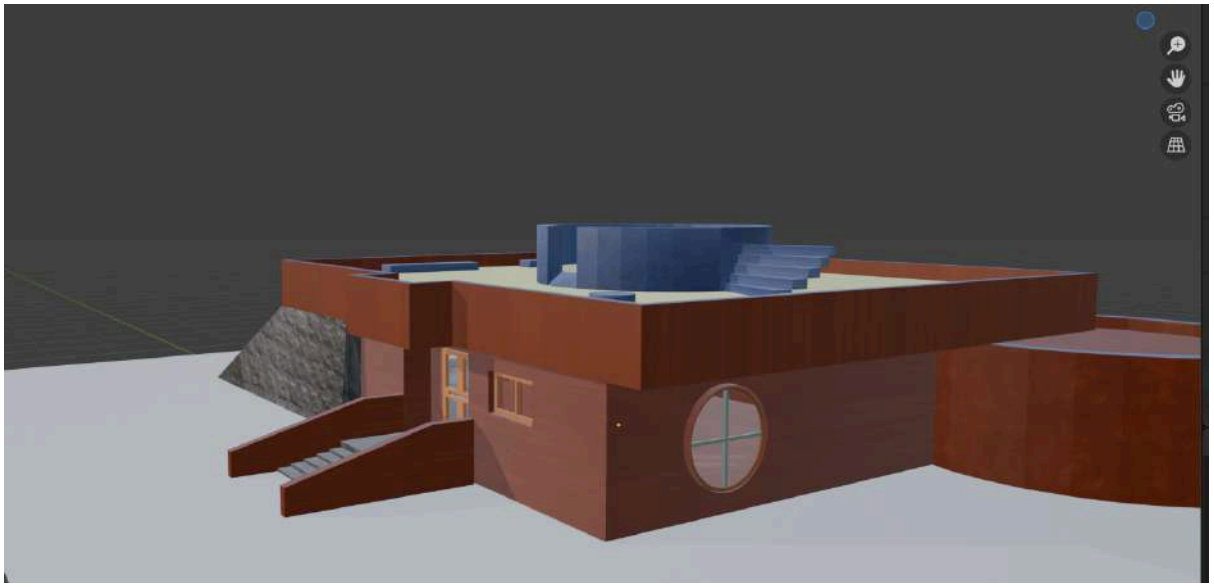





User Perspective
(161) techo

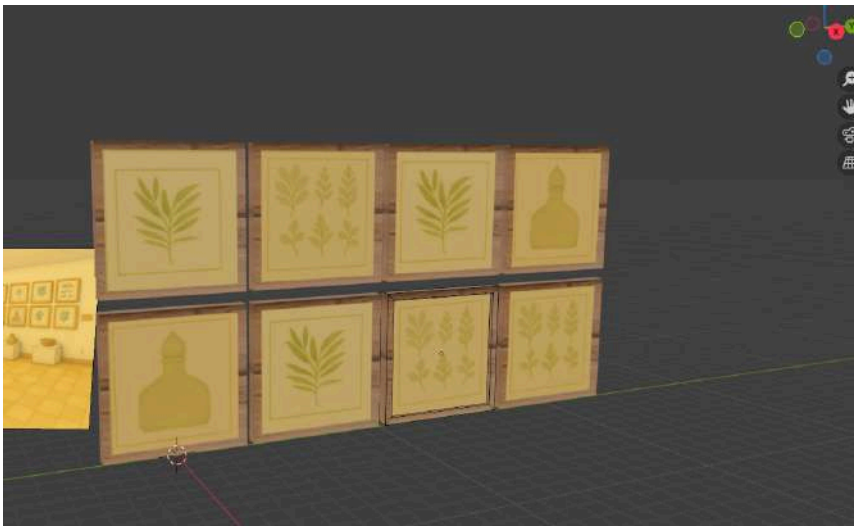

User Perspective
(161) Cylinder.001

This is how the facade would look with its corresponding textures

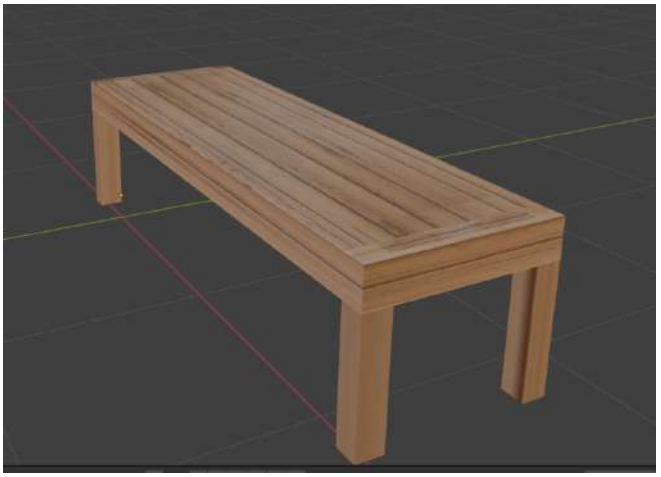


Texture objets

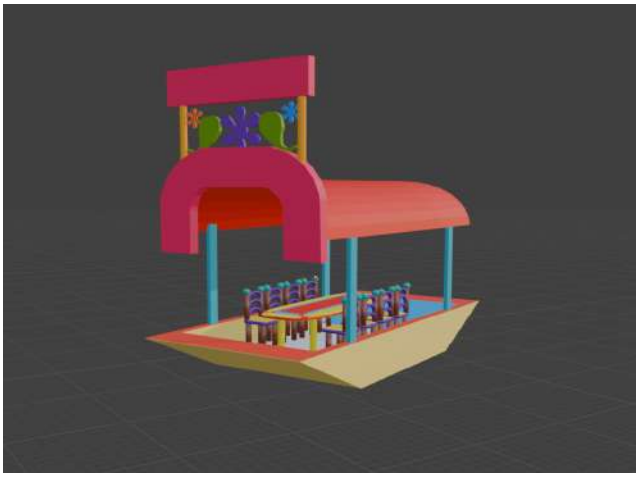
1. Paintgs



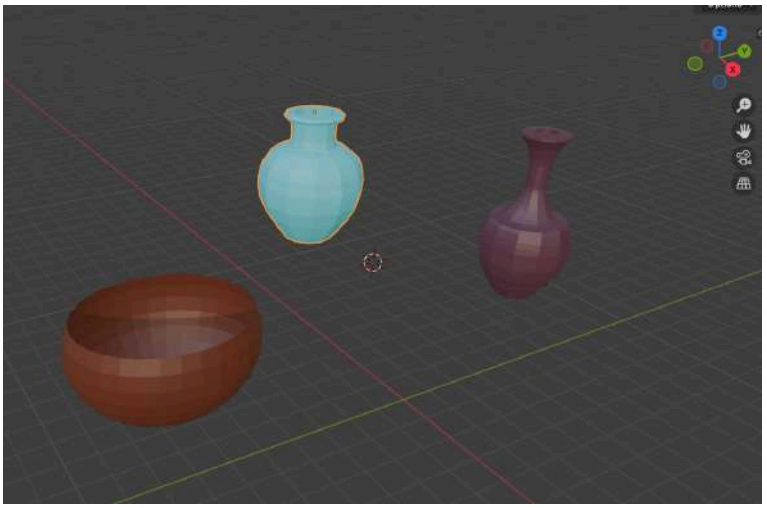
2. Banking



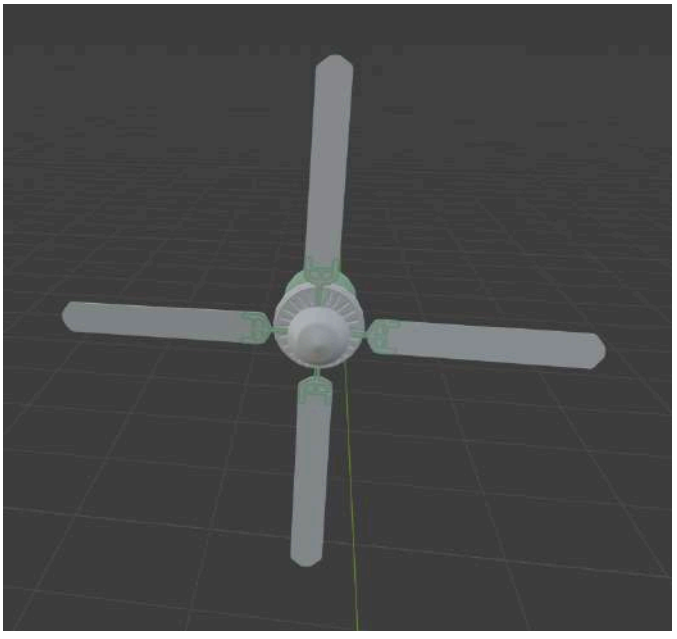
3. Trajinera



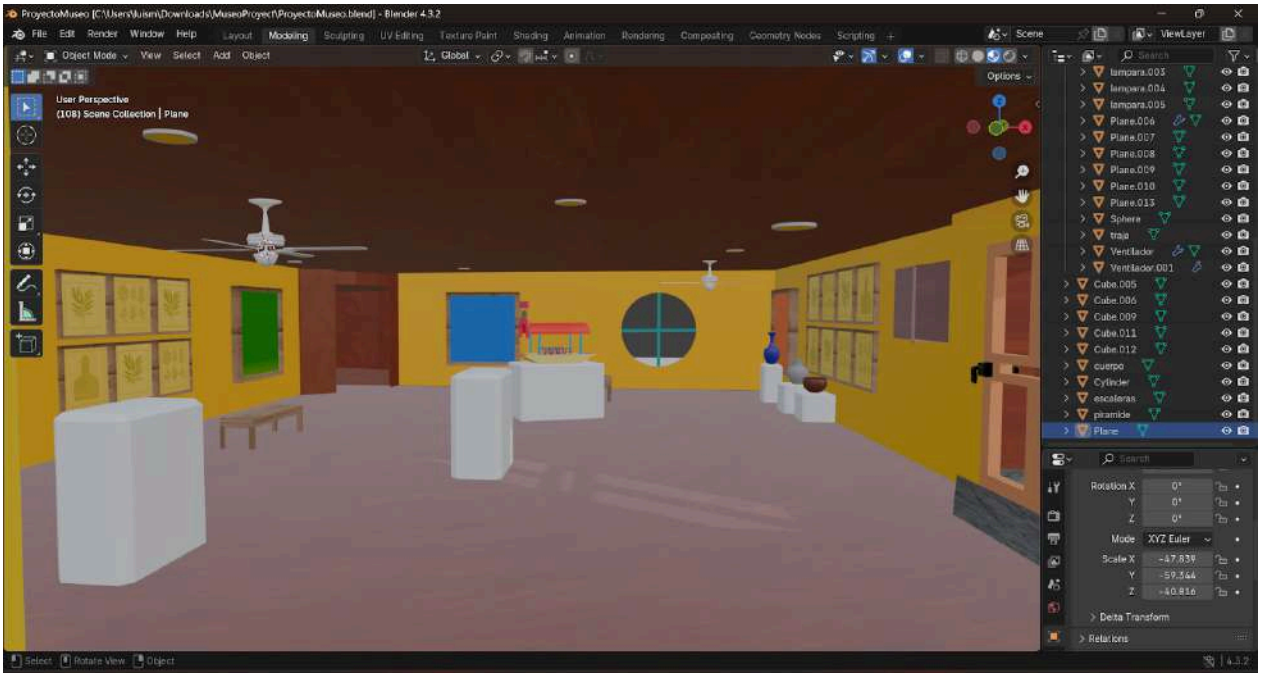
4. Sculptures



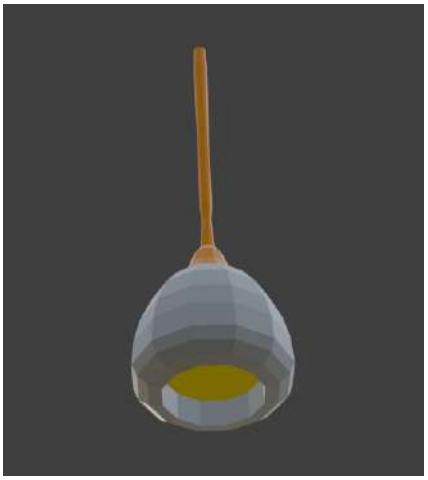
5. Fan



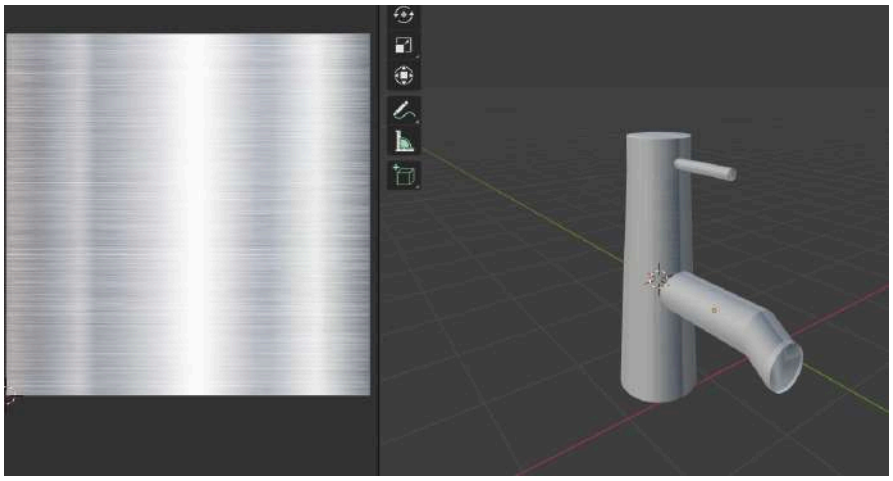
Room with the 5 modeled objects and their texture



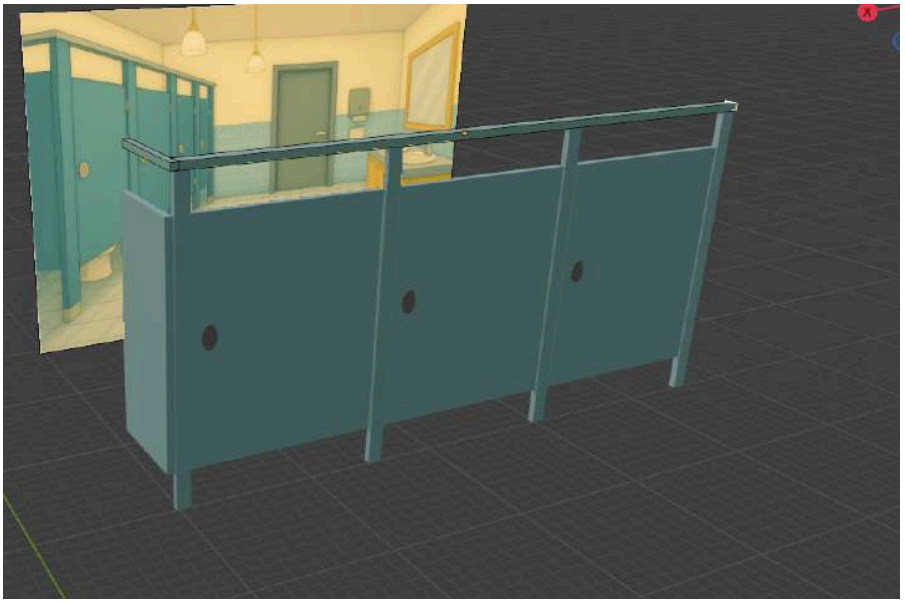
6. Lamps



7. Sink

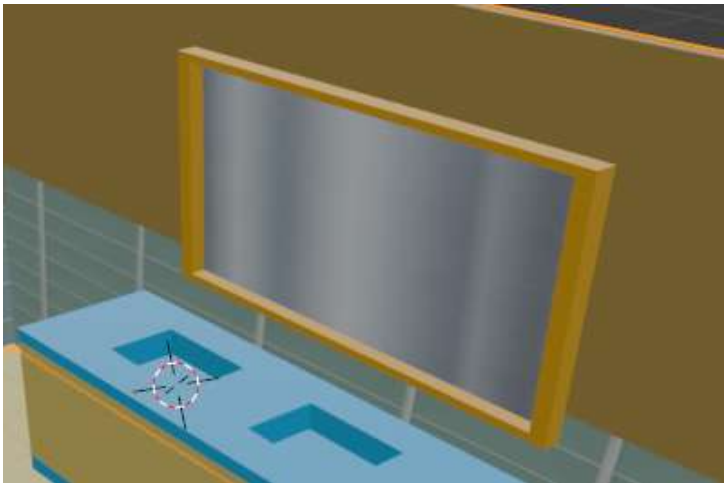


8. Doors

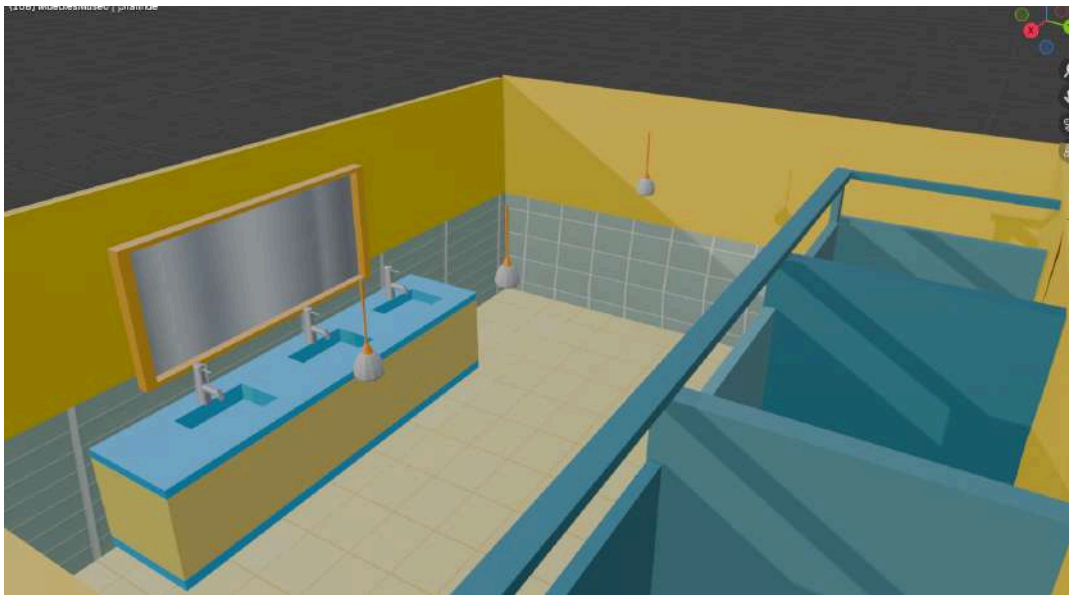


9. WC

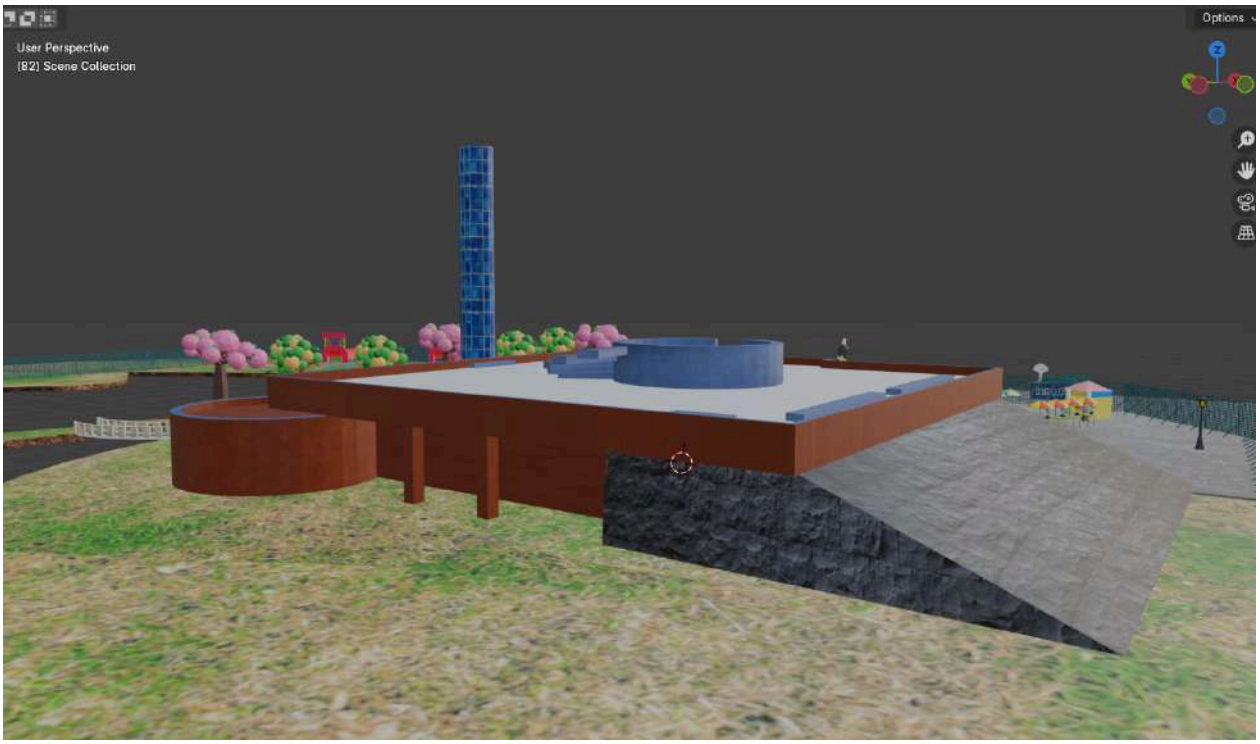
10. Mirror



bath with objects



The museum would finally look like this, after its modeling and corresponding textures

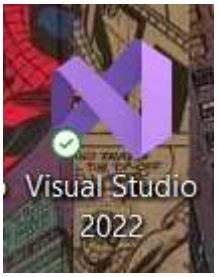


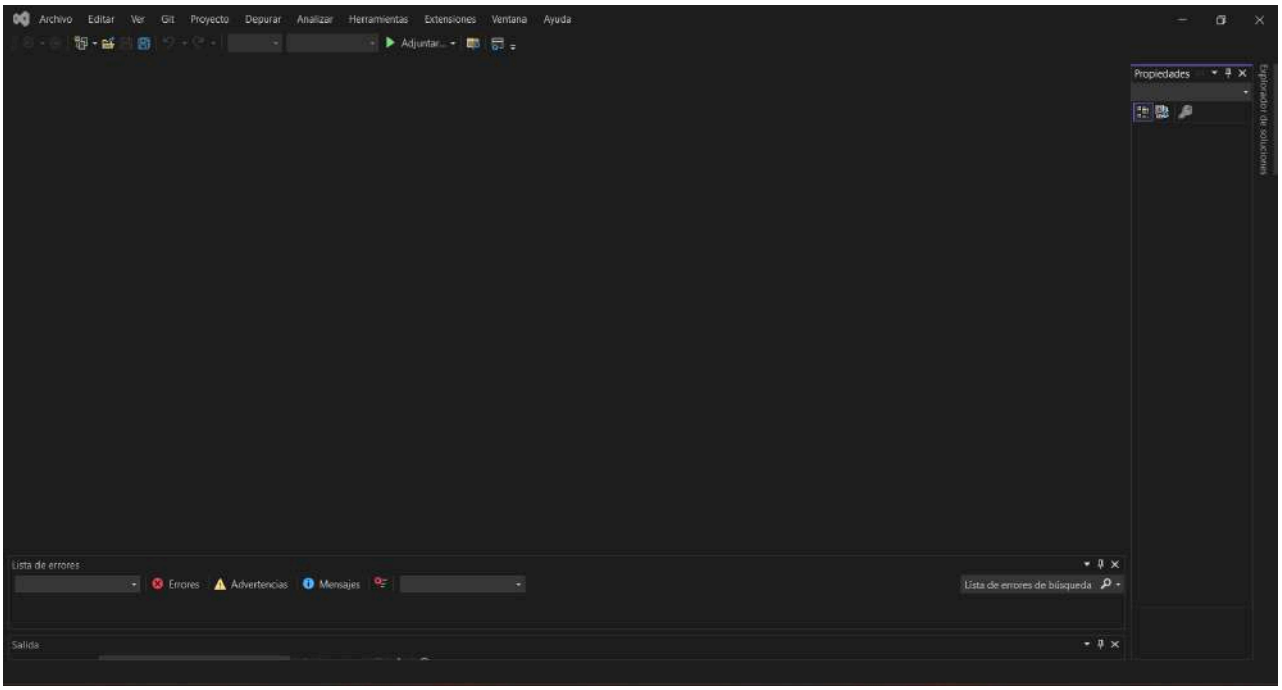
7.4 Herramientas, librerías y estructura del código

Free software tools and libraries were used to develop the museum, facilitating both 3D modeling and C++ programming with OpenGL. The main tools used and the internal organization of the project are detailed below.

Tool used

Visual Studio Code: This was the development environment used to write, compile, and organize the project's source code. It was used with a clear folder structure that allowed for the separation of code, models, textures, and sounds. In addition, extensions were integrated to facilitate writing in C++ and navigating between files.





Libraries integrated into the project

- **OpenGL**: Main library for rendering the 3D environment in real time.
- **GLFW**: Allows you to create windows, detect keyboard and move the mouse
- **GLAD**: OpenGL function loader.
- **GLM**: To handle vectors and matrices in 3D.
- **stb_image.h**: To load textures in .png or .jpg format.

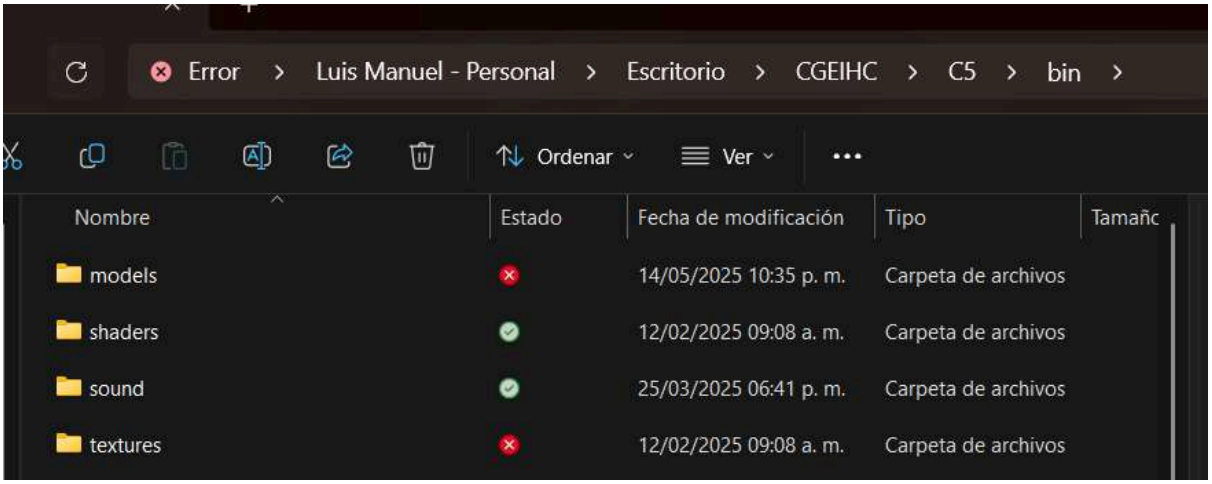
```
1
2
3   #include <iostream>
4   #include <stdlib.h>
5
6   // GLAD: Multi-Language GL/GLES/EGL/GLX/WGL Loader-Generator
7   // https://glad.dav1d.de/
8   #include <glad/glad.h>
9
10  // GLFW: https://www.glfw.org/
11  #include <GLFW/glfw3.h>
12
13  // GLM: OpenGL Math library
14  #include <glm/glm.hpp>
15  #include <glm/gtc/matrix_transform.hpp>
16  #include <glm/gtc/type_ptr.hpp>
17
18  // Model loading classes
19  #include <shader_m.h>
20  #include <camera.h>
21  #include <model.h>
22  #include <animatedmodel.h>
23  #include <material.h>
24  #include <light.h>
25  #include <cubemap.h>
26
27  #include <irrKlang.h>
```

Project folder structure

The project was organized into specific folders that group together the graphic and audio resources and models used in the construction of the virtual museum. This organization keeps the project organized and facilitates access to files during programming.

The main folders are described below:

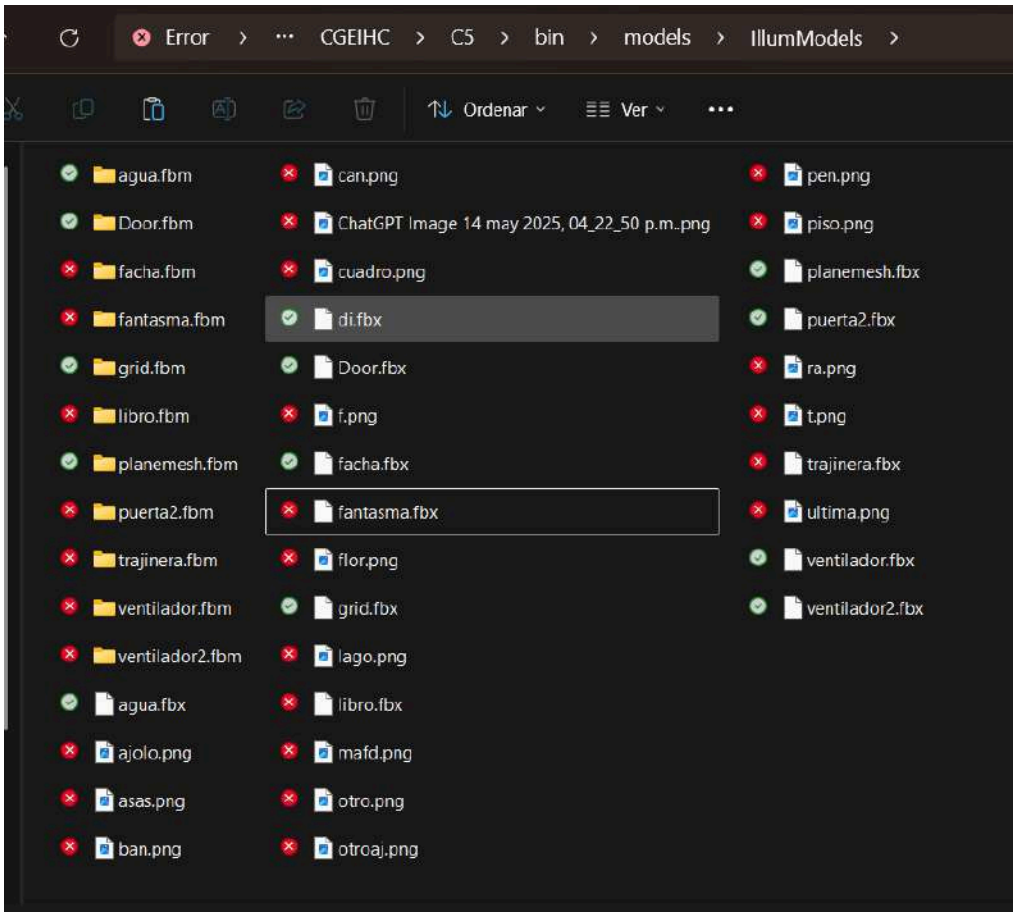
- **/models**: Contains the models exported from Blender in .fbx format along with their textures.
- **/textures**: Folder where the cubemap of our project is located, to give it a more realistic background
- **/shaders**: Includes vertex and fragment shader files (if using custom lighting or effects).
- **/sound**: Stores the ambient sound files used.



7.5 Loading models and textures into the system

Once the modeling and texturing process was complete in Blender, the models were exported in .fbx format for integration into OpenGL. The system was responsible for loading these files, interpreting them, and rendering them in real time.

The 3D models were created in Blender and subsequently exported in . fbx format, due to its compatibility with hierarchical structures, embedded textures and OpenGL-compatible axes. These files are stored in the /models/IllumModels folder.



In the code, model loading is done using the following instructions

```
// Carga la información del modelo
Model   *house;
Model   *door;
Model   *moon;
Model   *gridMesh;
Model* fan1;
Model* fan2;
Model* door_bathroom;
Model* trajinera;
Model* book_model;
Model* ghost_model;
```

```
house = new Model("models/IllumModels/facha.fbx");
door = new Model("models/IllumModels/di.fbx");
fan1 = new Model("models/IllumModels/ventilador.fbx");
fan2 = new Model("models/IllumModels/ventilador2.fbx");
gridMesh = new Model("models/IllumModels/grid.fbx");
door_bathroom = new Model("models/IllumModels/puerta2.fbx");
trajinera = new Model("models/IllumModels/trajinera.fbx");
ghost_model = new Model("models/IllumModels/fantasma.fbx");
book_model = new Model("models/IllumModels/libro.fbx"); //
```

Cube Map

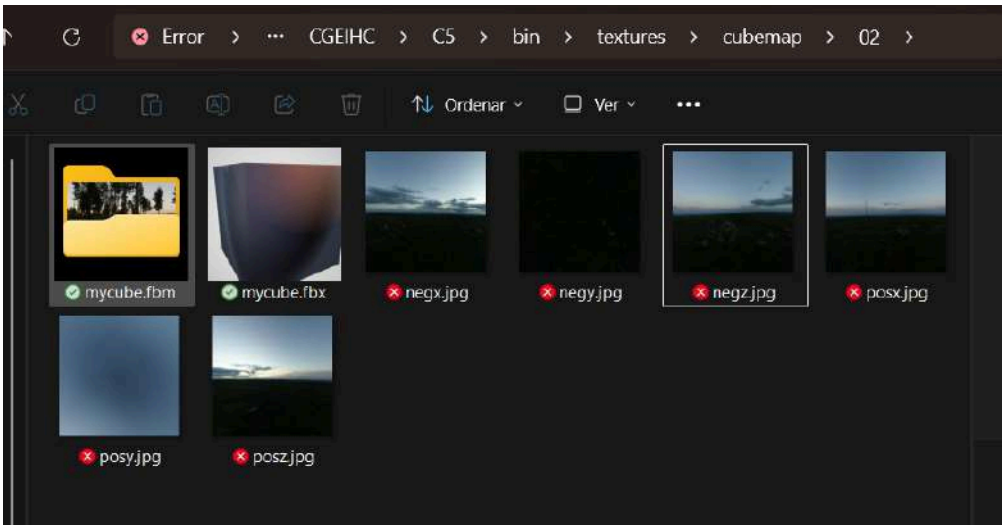
To visually set the museum's surroundings, a cube map was integrated as a sky background, providing depth and realism to the scene. This cube map was loaded using a custom CubeMap class, and the individual images were placed in the /textures/cubemap/02 folder.

6 images were used, one for each face of the cube:

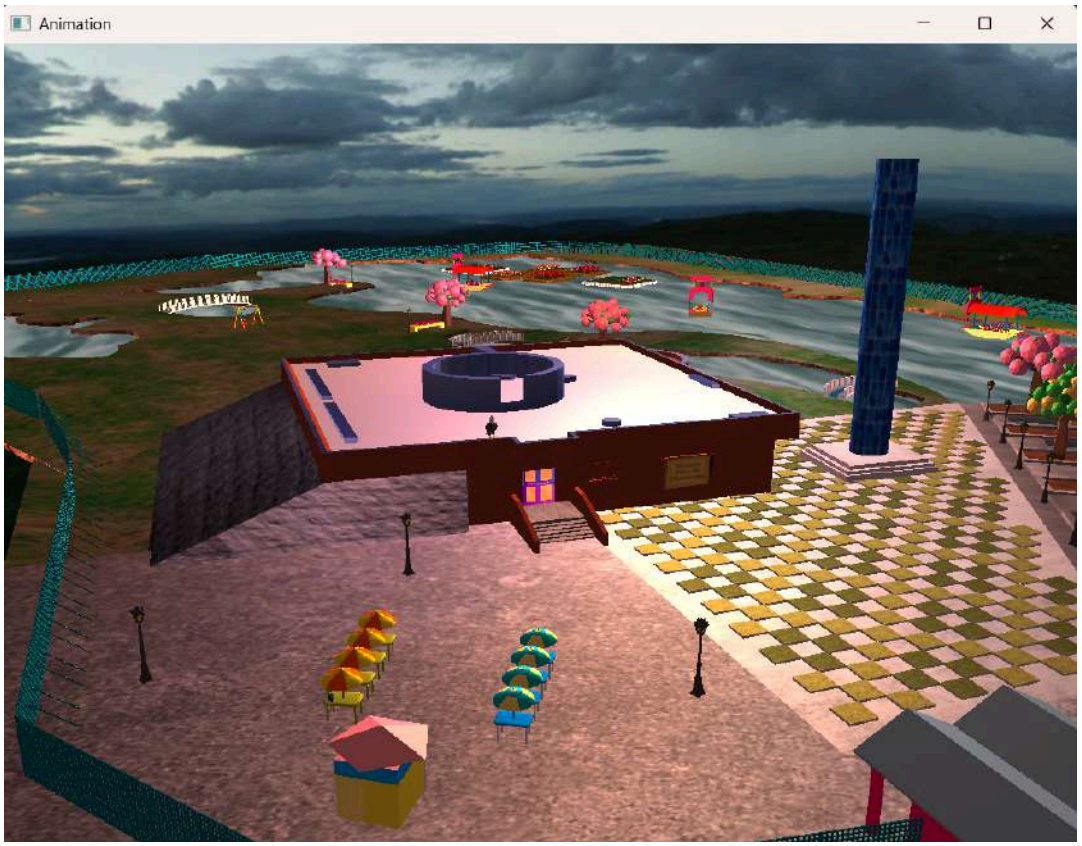
```
// Cubemap
vector<std::string> faces
{
    "textures/cubemap/02/posx.jpg",
    "textures/cubemap/02/negx.jpg",
    "textures/cubemap/02/posy.jpg",
    "textures/cubemap/02/negy.jpg",
    "textures/cubemap/02/posz.jpg",
    "textures/cubemap/02/negz.jpg"
};
mainCubeMap = new CubeMap();
mainCubeMap->loadCubemap(faces);
```

These images are applied as a texture to a cube mesh that surrounds the entire scene. The cubemap is displayed before the rendering of the

3D models, ensuring that they always remain in the background without interfering with the museum objects.



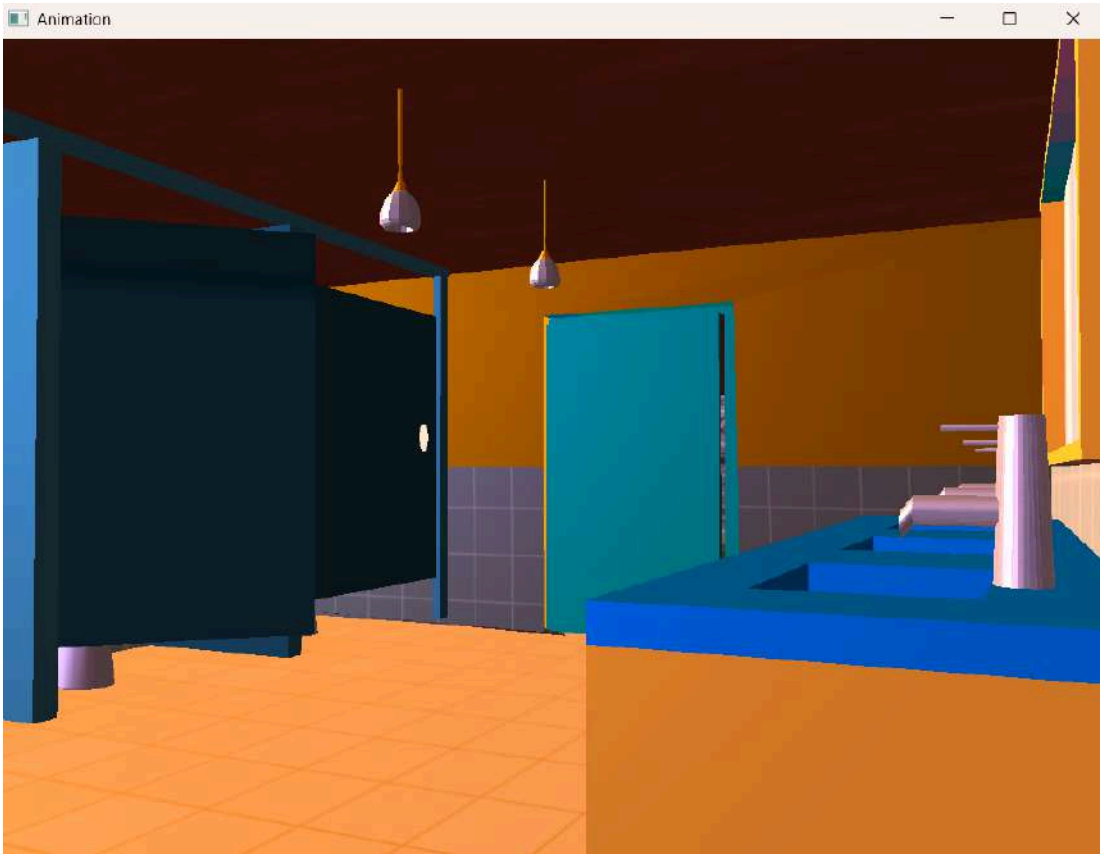
When running we can see our models with their textures loaded in OpenGL

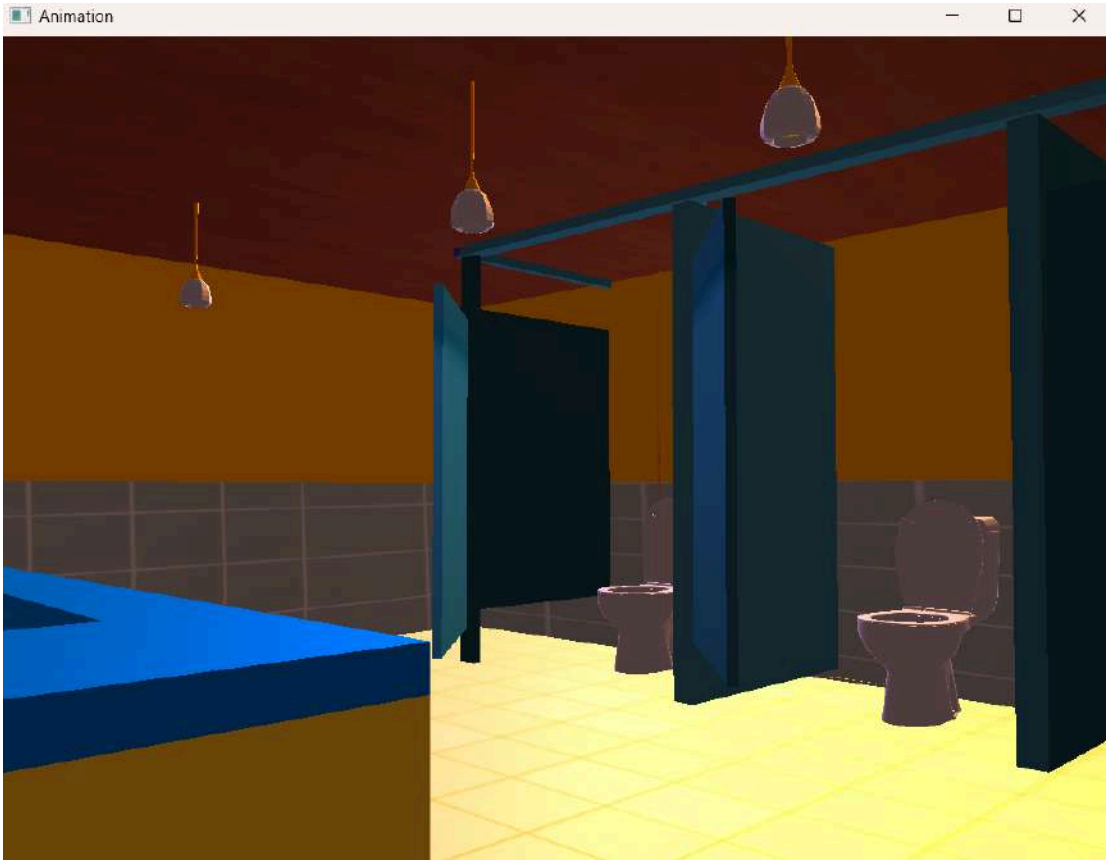


Museum inside

Room 1

Room 2 bath


Animation


Animation

7.6 Implementation of the synthetic camera

The project includes a first-person synthetic camera implemented manually using the GLM library for vector and matrix manipulation, and managed by a custom class called Camera. This camera allows the user to move freely around the museum with precise and natural controls.

```
#include <shader_m.h>
#include <camera.h>
#include <model.h>
#include <animatedmodel.h>
#include <material.h>
#include <light.h>
#include <cubemap.h>
```

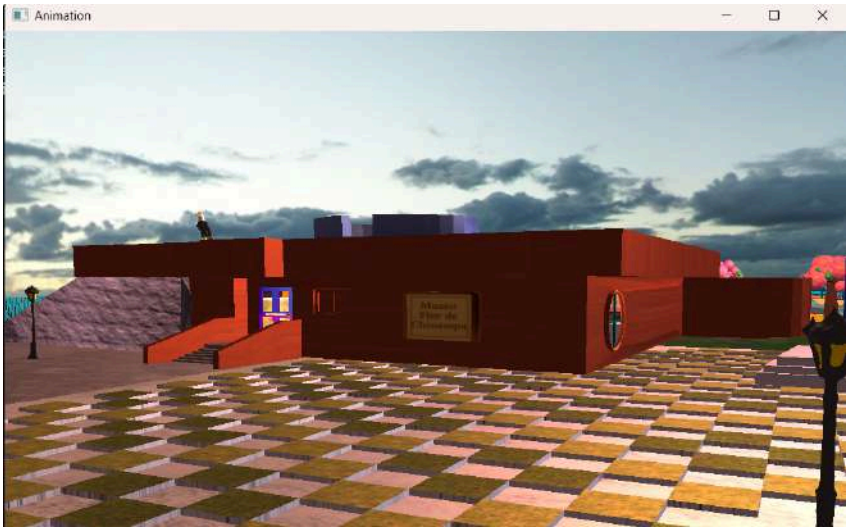
Initialized the main first-person camera with a custom position and orientation:

```
Camera camera(glm::vec3(210.75f, -4.3073f, 108.293f),
    glm::vec3(0.0f, 1.0f, 0.0f), -185.0f, -10.0f);
```

The mouse_callback() callback controls the horizontal and vertical rotation of the view:

```
void scroll_callback(GLFWwindow* window, double xoffset, double yoffset)
{
    camera.ProcessMouseScroll((float)yoffset);
}
```

We can get this view



7.7 Animations

To achieve the animation of each object we used, it was necessary to export them in . fbx as mentioned, then move their position so that they matched where they should go.

1. Main door of the museum

   The museum's main door is animated by rotating around its vertical axis using GLM transformations. The user can open it by pressing H or close it with J, which modifies the door_rotation variable, used to rotate the model during rendering, in real time.

```
model = glm::mat4(1.0f);
model = glm::translate(model, door_position);
model = glm::rotate(model, glm::radians(door_rotation), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(-90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
mLightsShader->setMat4("model", model);
door->Draw(*mLightsShader);
```

```
if (glfwGetKey(window, GLFW_KEY_H) == GLFW_PRESS)
    door_rotation += 1.f;
if (glfwGetKey(window, GLFW_KEY_J) == GLFW_PRESS)
    door_rotation -= 1.f;
```

2. Fans

   Both fans rotate using a continuous, time-based animation (deltaTime).

   Rotation is activated with the T key and stopped with G.

```
glm::mat4 model_fan = glm::mat4(1.0f);
model_fan = glm::translate(model_fan, fan1_position);
model_fan = glm::rotate(model_fan, glm::radians(-90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model_fan = glm::rotate(model_fan, glm::radians(fan_rotation), glm::vec3(0.0f, 0.0f, 1.0f));

mLightsShader->setMat4("model", model_fan);
fan1->Draw(*mLightsShader);
```

```
if (glfwGetKey(window, GLFW_KEY_T) == GLFW_PRESS) {
    fan_rotate = true;
    fan2_rotate = true;
}
if (glfwGetKey(window, GLFW_KEY_G) == GLFW_PRESS) {
    fan_rotate = false;
    fan2_rotate = false;
}
```

3. Trajinera

   If the user is within 10 units, the trajinera_rotation variable is incremented, causing it to rotate constantly as long as it remains nearby.

```
float dist_trajinera = glm::distance(camera.Position, trajinera_position);

    if (dist_trajinera < 10.0f) {
        trajinera_rotation += 60.0f * deltaTime;
        if (trajinera_rotation > 360.0f) trajinera_rotation -= 360.0f;
    }
```

```
glm::mat4 model_trajinera = glm::mat4(1.0f);
model_trajinera = glm::translate(model_trajinera, trajinera_position);
model_trajinera = glm::rotate(model_trajinera, glm::radians(trajinera_rotation), glm::vec3(0.0f, 1.0f, 0.0f));
model_trajinera = glm::rotate(model_trajinera, glm::radians(-90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
mLightsShader->setMat4("model", model_trajinera);
trajinera->Draw(*mLightsShader);
```

4. Phantom Apparition

   The ghost appears over the book when the show_book_info variable is activated. Its upward movement is achieved by increasing popup_height over time (deltaTime). The model is then translated along the Y axis, rotated, and scaled to display correctly on the screen.

```
if (show_book_info) {
    popup_height += deltaTime * 1.5f;
    if (popup_height > 1.5f) popup_height = 1.5f;

    glm::vec3 ghost_pos = book_position + glm::vec3(0.0f, popup_height + 1.0f, 0.0f);
    glm::mat4 ghost_model_mat = glm::mat4(1.0f);
    ghost_model_mat = glm::translate(ghost_model_mat, ghost_pos);
    ghost_model_mat = glm::rotate(ghost_model_mat, glm::radians(-90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
    ghost_model_mat = glm::scale(ghost_model_mat, glm::vec3(0.45f));

    mLightsShader->setMat4("model", ghost_model_mat);
    ghost_model->Draw(*mLightsShader);
}
```

5. Bath door opens alone

   The bathroom door opens automatically when the user approaches within 10 units of the distance, and closes when they move away. This is achieved

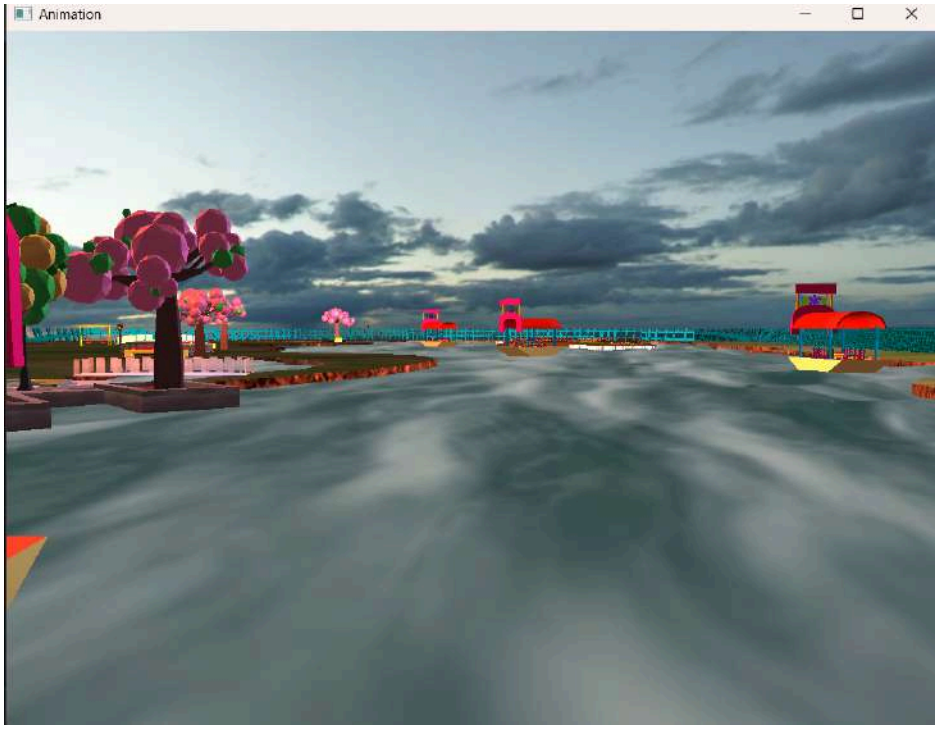
measuring the distance with glm::distance and adjusting the door_bath_rotation angle on each frame using deltaTime

```
float dist = glm::distance(camera.Position, door_bath_position);

// Abrir si está cerca
if (dist < 10.0f && door_bath_rotation < 100.0f) {
    door_bath_rotation += 60.0f * deltaTime;
}
// Cerrar si ya se alejó
else if (dist >= 10.0f && door_bath_rotation > 0.0f) {
    door_bath_rotation -= 60.0f * deltaTime;
}
```

6. Lake movement

The water surface is animated using a procedural shader that simulates wave motion. Each frame increments a time variable (wavesTime), which is sent to the shader as a uniform to dynamically modify the vertices and create the wave effect.

```
wavesShader->setFloat("time", wavesTime);
wavesShader->setFloat("radius", 5.0f);
wavesShader->setFloat("height", 5.0f);

gridMesh->Draw(*wavesShader);
wavesTime += 0.0003;
```



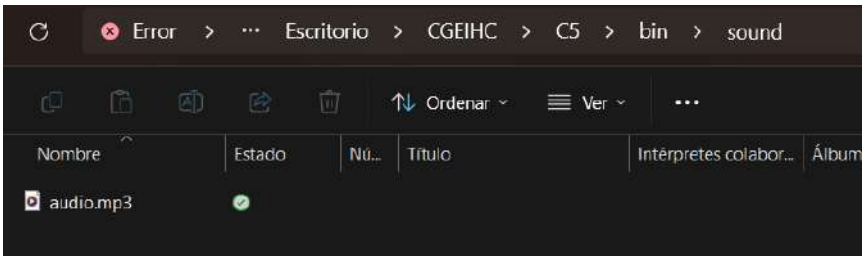
7.8 Loading and implementing sound

The system plays ambient sound from the start of the program using IrrKlang.

```
#include <irrKlang.h>
using namespace irrklang;
```

```
SoundEngine->play2D("sound/audio.mp3", true);
```

The true parameter indicates that it plays continuously while the application is running.

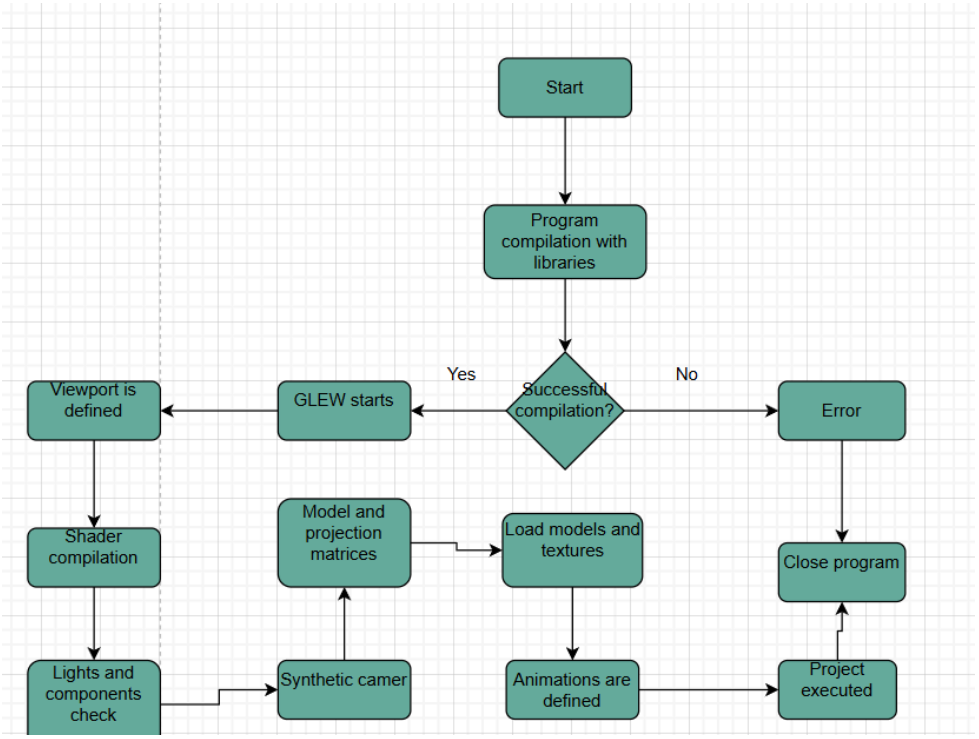


7.9 System flow diagram

The flowchart represents the logical architecture of the program lifecycle, from initialization to shutdown. It begins with compilation of the program along with the necessary libraries (GLFW, GLAD, GLM, irrKlang), followed by compilation validation. In the event of an error, the system terminates execution in a controlled manner.

If the compilation is successful, the graphical environment is configured: GLEW initialization, VAO and VBO definition, shader compilation, and viewport definition. Next, the lighting systems are configured, and the synthetic camera is declared to establish the rendering perspective.

Next, the model, view, and projection matrices are defined, and the 3D models in .fbx format are loaded, along with their corresponding textures. Once the resources are loaded, the animation system is initialized, and the main rendering loop begins, where the interactive elements of the environment, such as the museum door, the fans, the trajinera, the bathroom door, and the pop-up effect of the book and the ghost, are continuously updated.



7.10 Dictionary of functions and key variables

Main functions

Function	Description
main()	Starts the program by calling `Start()`, and then enters the `Update()` loop to keep running the application until the user closes the window.
Start()	Initializes GLFW, GLAD, shaders, lights, camera, models, textures, cubemap and sound
Update()	Main loop. Processes input, updates animations, renders models and effects.
processInput()	Captures and executes keyboard inputs (movement, actions, camera, selection).
mouse_callback()	Controls rotation of the camera in first person through mouse movement.
scroll_callback()	Control the camera zoom usign the mouse wheel.
mouse_button_callback()	Detects use clicks. Triggers the ghost to appear when the book is clicked.
SetLightUniform*()	Functions that load light parameters into the shader.

key system variables

Variable	Tipo	Descripción / Función
camera	Camera	Camera in first person. Controlled with keyboard and the mouse
camera3rd	Camera	Camera in the third person
deltaTime	float	Time between frames, used for animations sofft.
lastFrame	float	Timestamp of the previous frame.
door_rotation	float	Museum door opening angle (H/J keys)
fan_rotation	float	Fan rotation 1 (T/G keys)
fan2_rotation	float	Fan rotation 2.
door_bath_rotation	float	Bathroom door angle (automatic by closeness)
trajinera_rotation	float	Rotation of the traijinera as it approaches.
book_position	glm::vec3	Position of the book on the scene
popup_height	float	Height of appearance of the ghost
show_book_info	bool	Activates ghost animation
water_position	glm::vec3	Position of the plane that simulates water
wavesTime	float	Time value in the procedural shader water.

SoundEngine	ISoundEngine*	Engine of audio of irrklang
window	GLFWwindow*	Window main of the system
mainCubeMap	CubeMap*	sky cubemap surrounding the scene
gLights	std::vector<Light>	Vector og lights in the scene.

USER MANUAL

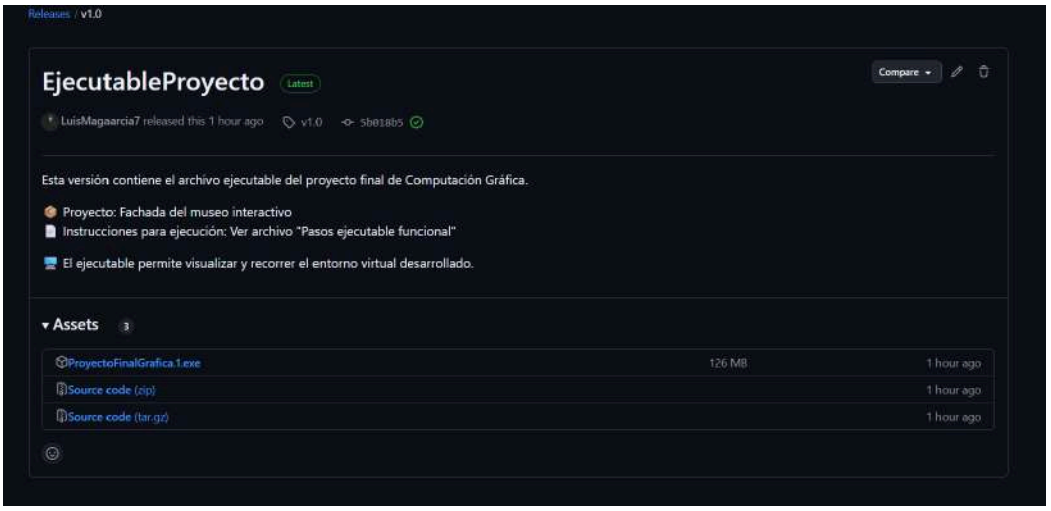
This user manual guides visitors through the Xochimilco Ecological Park Museum, a three-dimensional environment developed using computer graphics technologies. This space has been designed to offer an immersive and interactive experience, allowing them to explore content related to the natural, cultural, and agricultural heritage of the Xochimilco region.

Recommended Requirements

- **Operating system:** Windows 10 or 11 (Home or Pro)
- **Processor:** Intel Core i7 (9th gen or higher) / AMD Ryzen 5 3600 or higher
- **RAM memory:** 16 GB or more
  Graphics card: NVIDIA GeForce GTX 1660 Ti / RTX 2060 or AMD Radeon RX 6600 or higher
- **Free storage:** At least 1 GB on an SSD (NVMe preferred)
  Screen resolution: Full HD (1920x1080) or higher, preferably with a 60 Hz or higher monitor
- **Peripherals:** Optical mouse and physical keyboard for better control
- **Audio:** Speakers or headphones to enjoy ambient sound and interactions

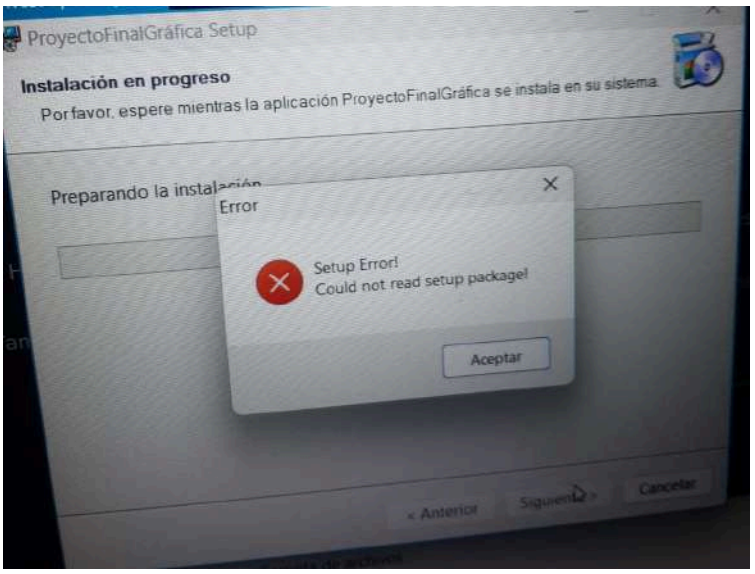
Project executable.

To run the project program, you must enter the following link:
https://github.com/LuisMagaarcia7/ProyectoComputaci-nGr-fica/releases/tag/v1.0



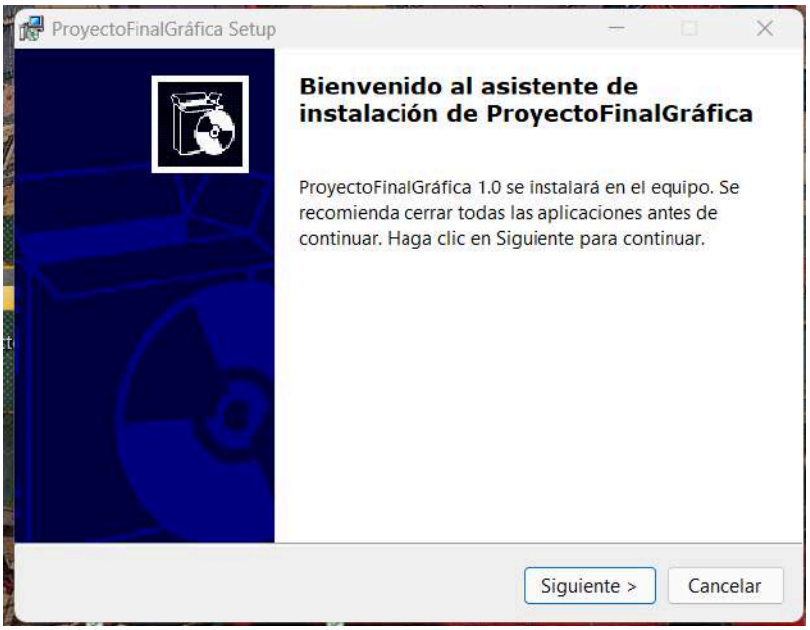
Within the link we can see the executable,We only download the .exe file .exe **“ProyectoFinalGráfica.1.exe”**

Once downloaded successfully, it is recommended to run it as an administrator and not have any antivirus activated, as this may prevent the executable from downloading properly, as shown in the image:

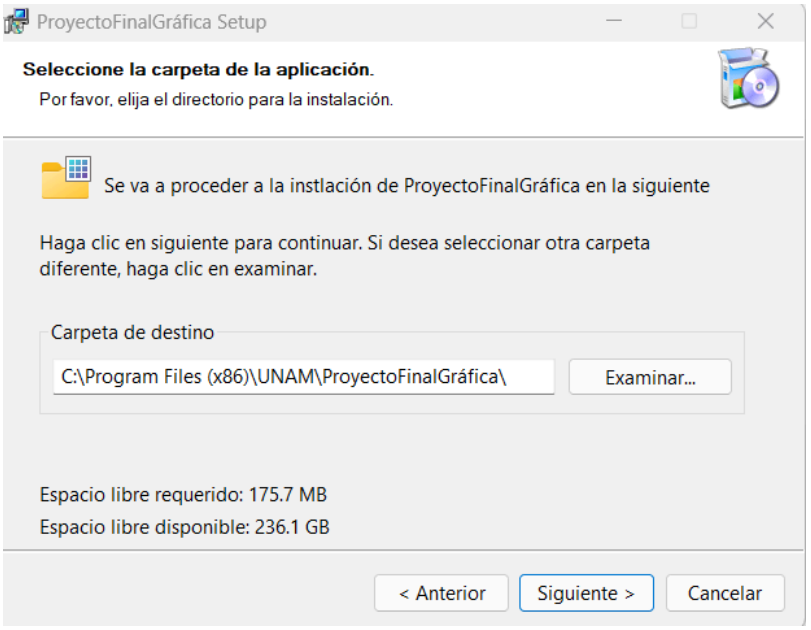


After running the .exe we follow the following steps

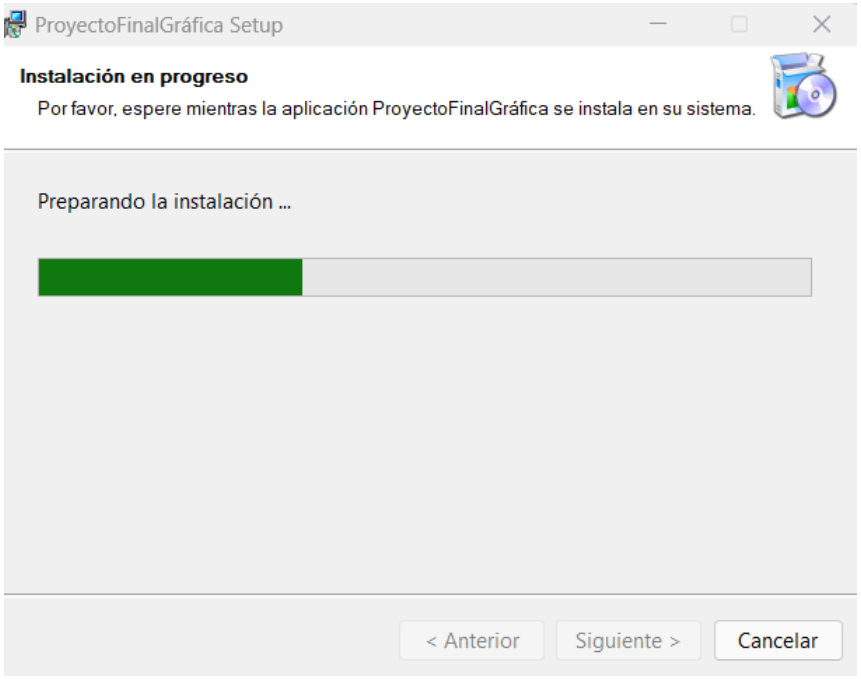
Click next



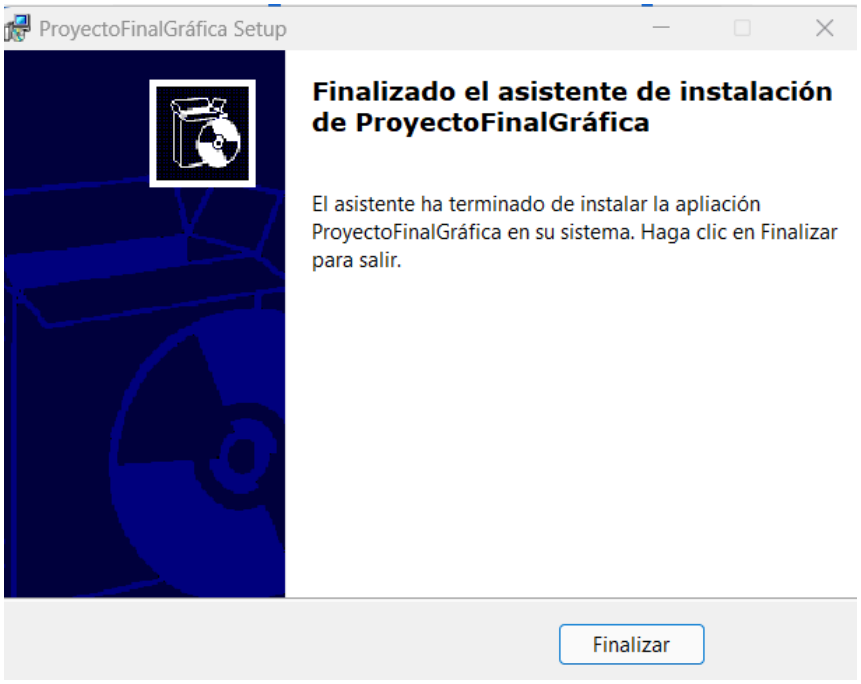
We choose the directory where the program will be installed, in this case, mine is the following, we give the following



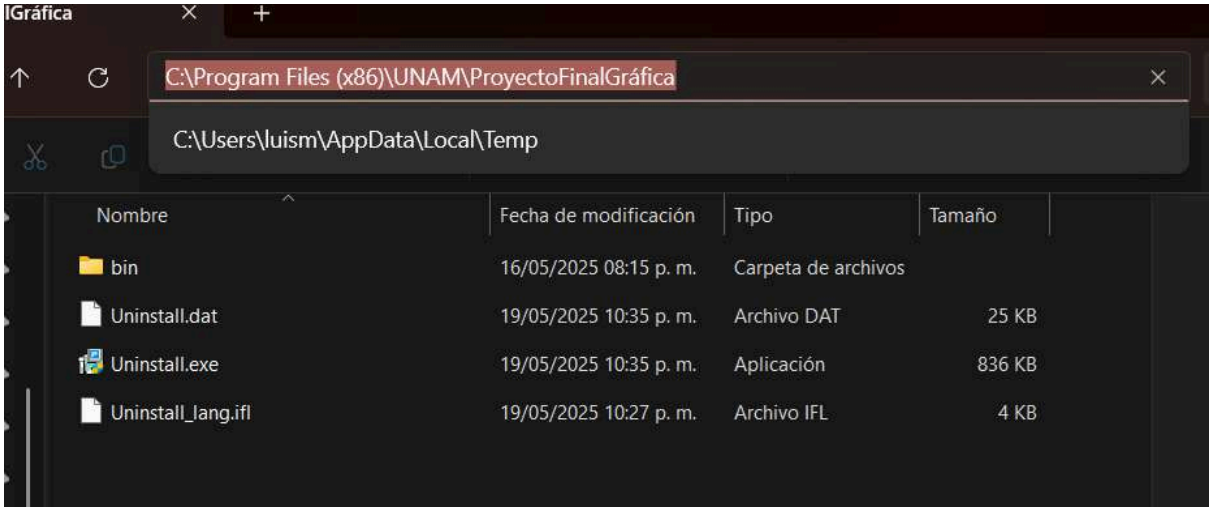
Installation begins



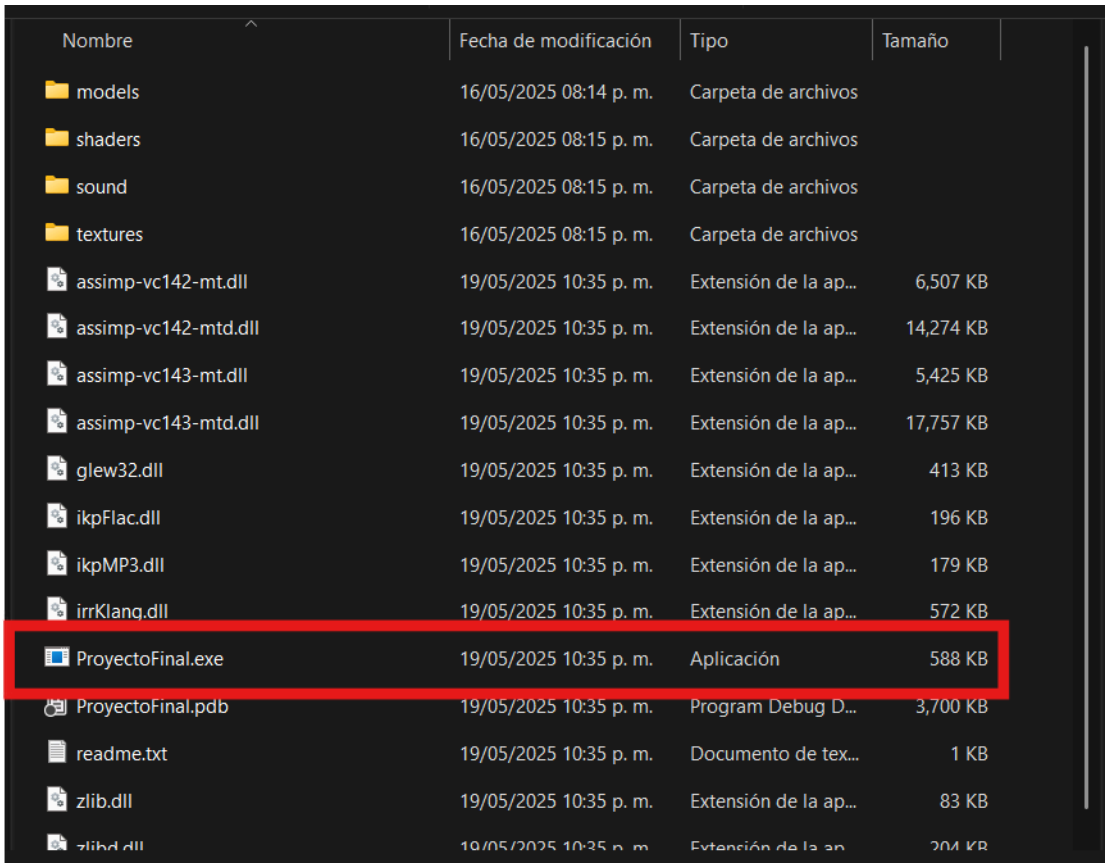
We finish



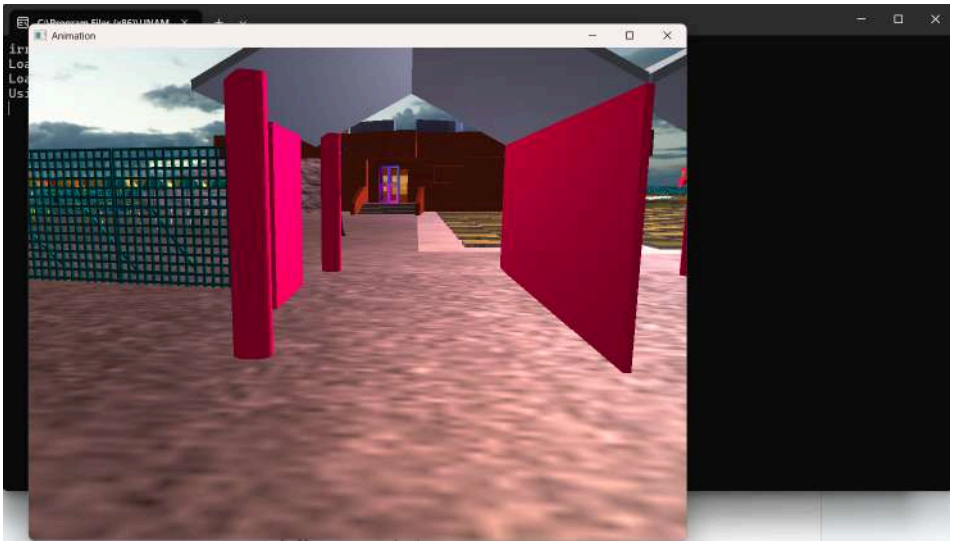
Once the executable is installed, we navigate to the directory where we chose to install it. We can see that it also has an uninstaller, in case the user wants to uninstall the executable. Then, we click on the "bin" folder.



In the bin folder, we can see various folders and files, but the one we are interested in is the one that will not open the project, we click “FinalProject.exe” this will open the project.



We can see how the executable opens correctly, so we can begin navigating through our final project.

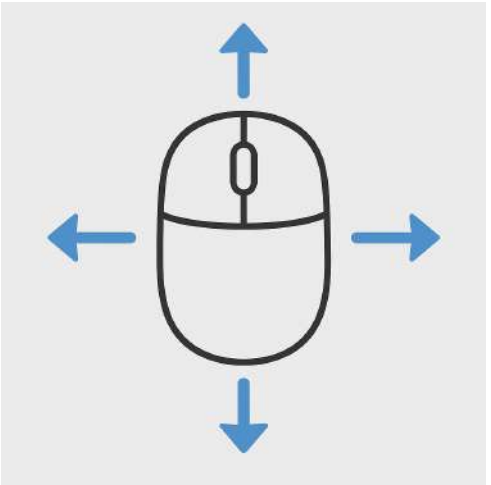


1. **Move around the environment**

   To be able to move around the environment it is necessary to use the W, D, Q A, S, and E keys on our keyboard.

   - W → Forward
   - S → Back
   - D → Movement of the right
   - A → Movement of the left
   - Q → Down
   - E → Upload

In addition to keyboard panning, the camera is controlled by mouse movement, allowing the user to freely view in all directions.



- **Move the mouse to the right or left:** rotates the view horizontally (camera flip).

- **Move the mouse up or down:** Changes the vertical angle of the camera.

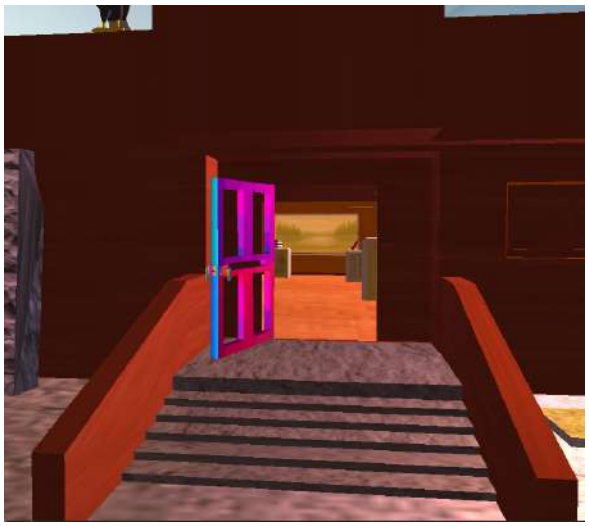
2. **Open and close the Museum door**

   For this first animation, we can observe the museum door, to open and close the door, it is necessary to use the H and J keys on the keyboard.

- J → open door
- H → close door

Normal door

Open door by pressing “J”



Door closed by pressing “H”

3. Operate museum fans

To activate the 2 fans inside the museum, you need to press the T and G keys on your keyboard.

- T → Activate the fans (they will start spinning)
- G → Stop fan movement (Fan rotation stops)

By pressing the T key, they will start to spin, if we want to stop the fans from spinning we press the G key.

4. **Movement of the trajinera**

Inside the museum, visitors will find a decorative trajinera as part of the room dedicated to the agricultural and lake culture of Xochimilco.

This trajinera features an interactive animation that is automatically activated based on the user's visual proximity:

- When the user approaches the trajinera with the camera, it begins to rotate slowly, allowing its details to be observed from different angles without the need for additional interaction.
- As you move away from the trajinera the animation it stops autocatically, simulating a natural pause in its movement.

Trajinera without approach

As we approach the trajinera from any angle it begins to rotate, allowing us to appreciate it better.

As we move away from the trajinera, we can see that it stops turning, implying that we will no longer appreciate it.

5. **Ghost animation**

Inside the museum, we can see three sections of books. The animation is directly on these books here, the orange ones.

This interaction aims to add a moment of surprise and fantasy to the museum environment, without altering the educational content of the rest of the tour.

- **Action to activate it:** left mouse click on the books
- **Trigger condition:** only activated if the used points and clicks on them
- **Result:** the ghost appears and performs a predefined animation

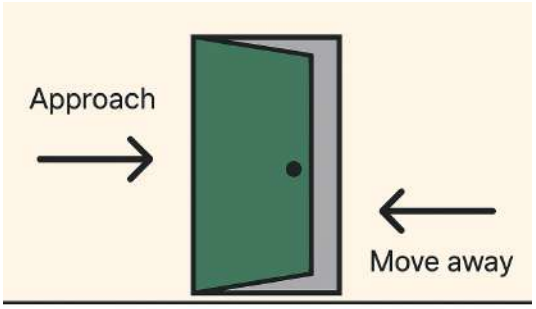
We can observe the books, when we click on the books we can see how the animation of a ghost appears.

6. Abrir y cerrar puerta del baño

Within the museum grounds, visitors will find an automatic door that provides access to the restroom area. This door features a completely autonomous opening and closing system based on user proximity, allowing for seamless access without the need for manual interaction.

The actions work as follows:

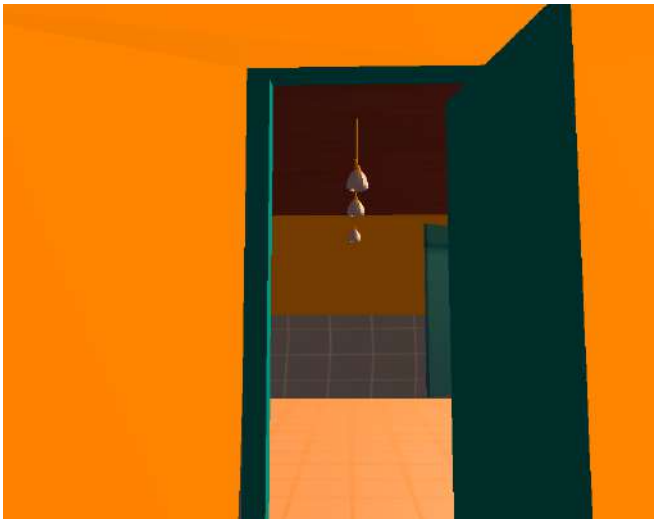
- When approaching the door from the outside, it opens automatically, allowing access to the bathroom
- Once the user has entered, the door closes automatically behind him.
- To exit the bathroom, simply approach the door again from the inside; it will open automatically



The door opens automatically as we approach it, and closes automatically as we move awa

Normal door

Door opens by itself when I approach



Door closes when entering the bath

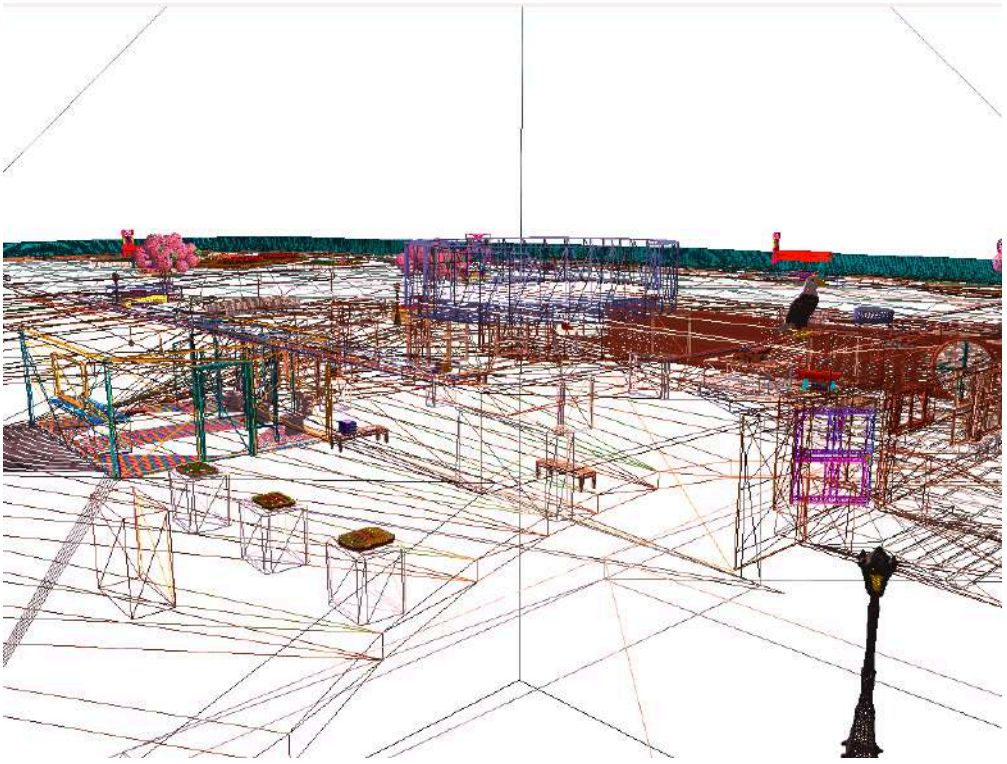
7. See the geometry of the museum and park

To fully appreciate the geometry of the museum and park, the user can activate the mesh display mode by pressing the M key.

This mode shows the internal structure of the models, allowing you to observe:

- The spatial distribution of objects
- The polygonal construction of the elements
- The technical organization of each room and the natural enviroment

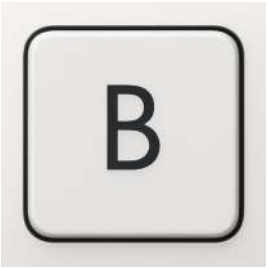
Pressing the N key returns you to the normal environment mode.



8. **See points of the Museum and park**

The system allows you to activate a special mode that displays all geometric points (vertices) of the museum and park. This allows you to accurately observe how the objects are distributed and their detailed structure in 3D space.

- B key → Activate point view.
- N key → Return to normal.



Cost analysis

Fixed Costs

The following table shows the essential elements used throughout the project, including computer equipment, software, and basic services. Most of the tools were free, allowing for resource optimization, and electricity and internet costs were also considered.

Category	Element	Approximate cost (MXN)	Grades
Hardware	Personal laptop (HP Victus)	$14,000	Equipment for home own, 16 GB RAM, Ryzen 5, SSD and GPU integrated.
Software	Blender	$0	Free software for 3D modeling
	Visual Studio Code / OpenGL	$0	Free IDE + code graphical
	stb_image.h / GLM / GLFW	$0	free libraries for loading textures, cameras and windows
	GitHub	$0	Free platform
	Audacity / GIMP	$0	Software libre para edición de audio e imágenes
Servicios	Electricidad (2 meses)	$750	Dear by consumption in the team of development

	Internet (2 meses)	$500	Connection necessary for documentation

Variable Costs

This table presents the costs associated with elements created specifically for this project, such as the museum facade, modeled objects, and sound design. Although many assets were developed in-house, an estimated commercial value is assigned to reflect their actual contribution. The sound was generated using open-source tools and therefore did not represent an additional expense.

Category	Element	Approximate cost (MXN)	Grades
Facade Modeling	Main facade	$900	Equivalent commercial approximate
3D Modeling	Objects modeled	$3,600	$180 c/u
Sound	Effects environmental and objects	$0	Engravings with tools free

Labour

Category	Activity	time my dear	cost	Subtotal
Development	Programming and interaction	60 hours	$100/hour	$6,000
Documentation	Manual technical, manual os user	10 hours	$100/hour	$1,000

Graphic design	Textures and generation of images	20 hours	$100/hour	$2,000

Estimated Total Cost

The cumulative total for the project, including fixed, variable, and labor costs, is presented below. This figure represents a general estimate of the virtual museum's actual production value, considering both material resources and time invested.

Concept	Cost (MXN)
Hardware and services	$15,250
Modeling and graphic resources	$4,500
Labor (dev + docs + design)	$9,000
Total estimated project	$28,750

Suggested Retail Price

Based on the total estimated cost of the project, a profit margin of 25% is calculated to establish a competitive and fair price.

Concepto	Costo (MXN)
Base cost of the project	$28,750
Profit margin (25%)	$7,187
Final sale price	$35,937

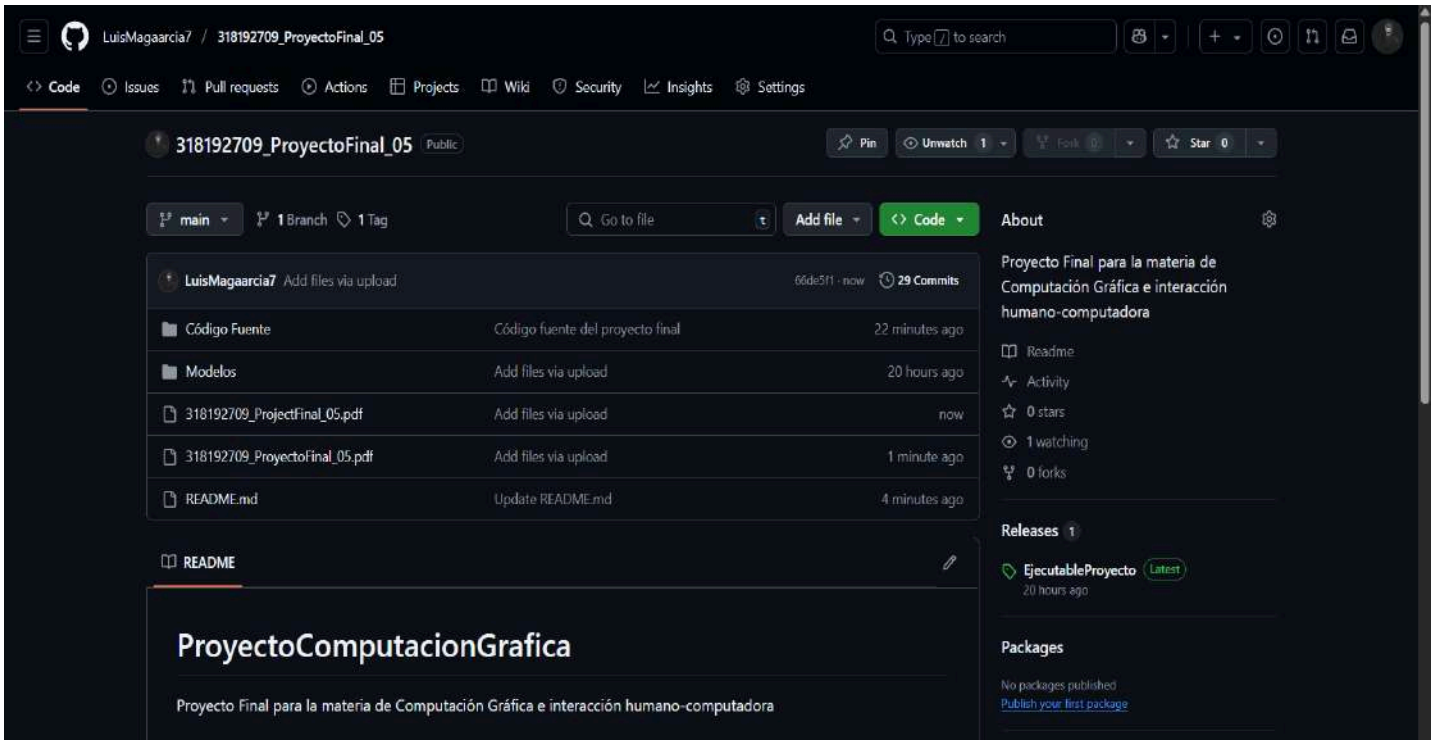
GITHUB REPOSITORY

Link:

https://github.com/LuisMagaarcia7/ProyectoComputaci-nGr-fica/tree/main

The entire project is located in the GitHub repository:

- Models used
- Source code
- Executable (located in releases)
- Documentation in spanish
- Documentation in english

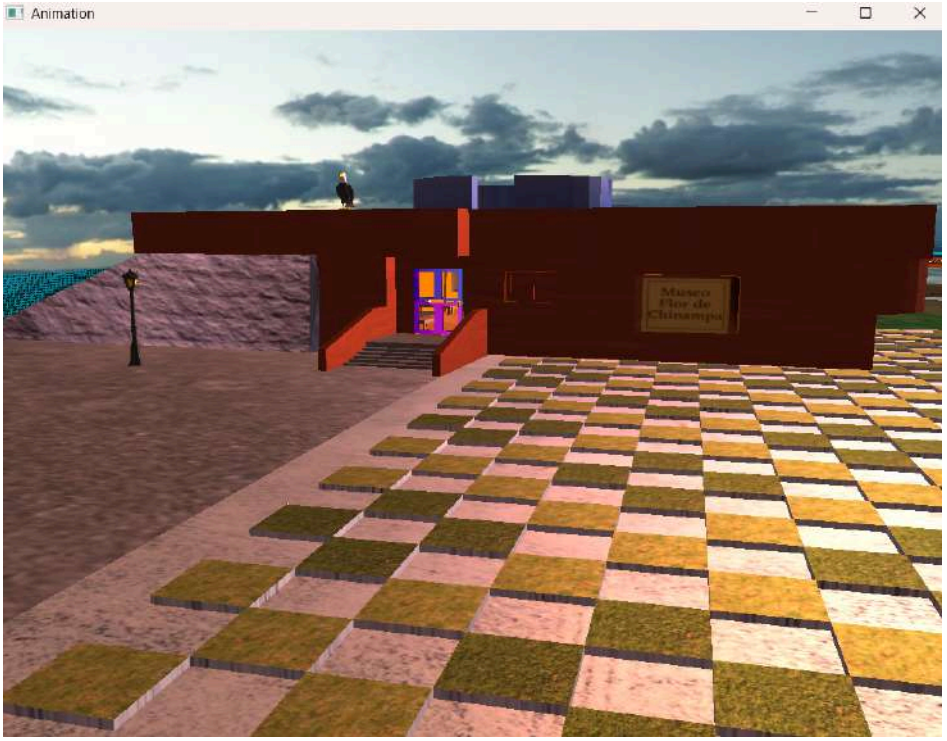


FACADE DEMONSTRATION VIDEO

https://drive.google.com/file/d/1YgZOBDug4wkQrNEltVO0BVuHVG84w4yE/view?usp=sharing

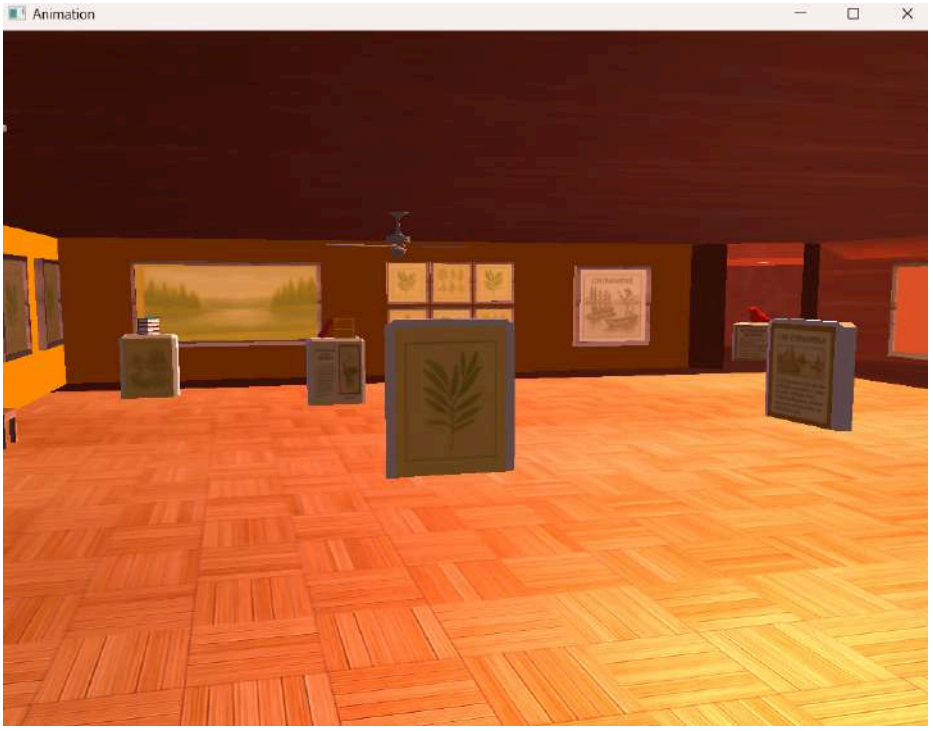
Final Result

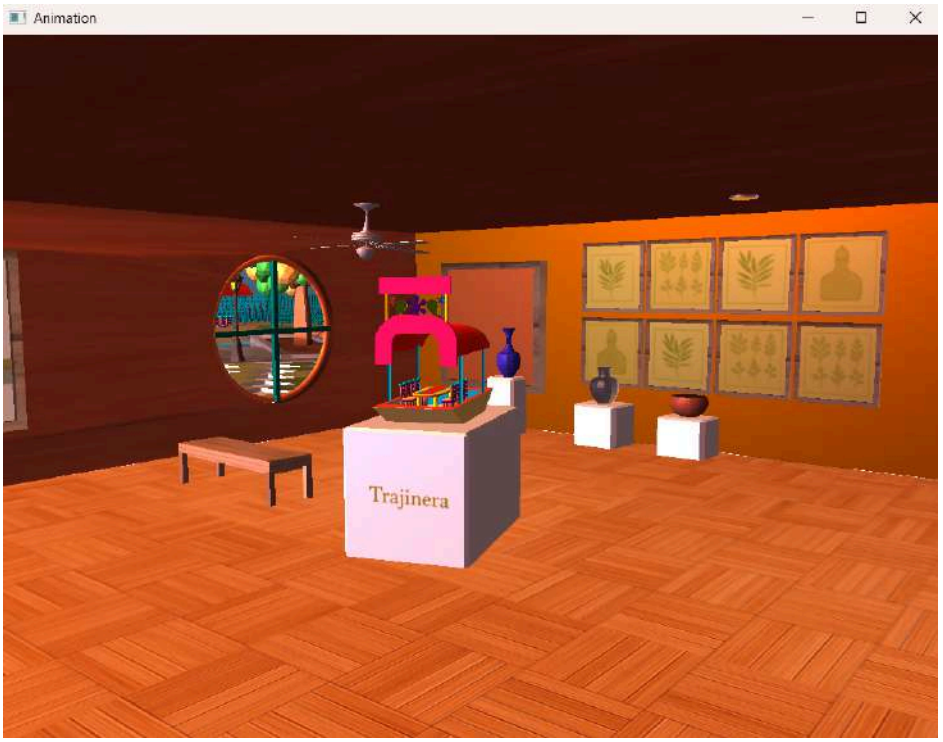
In this section, screenshots of the project will be uploaded in its final stage, showing how everything worked on in the executable looks.

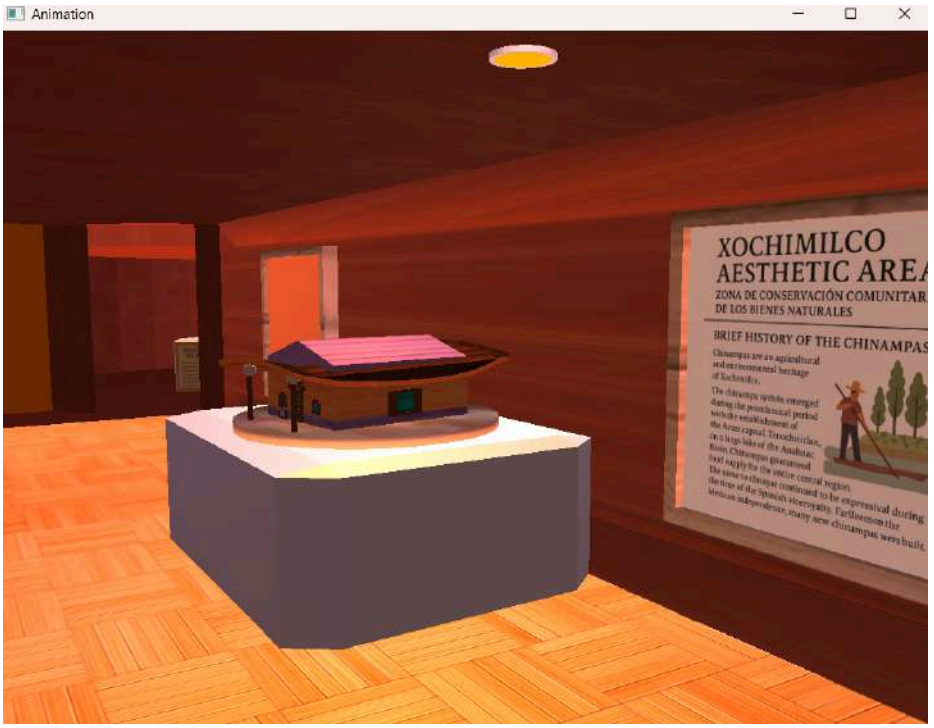



Animation


Animation

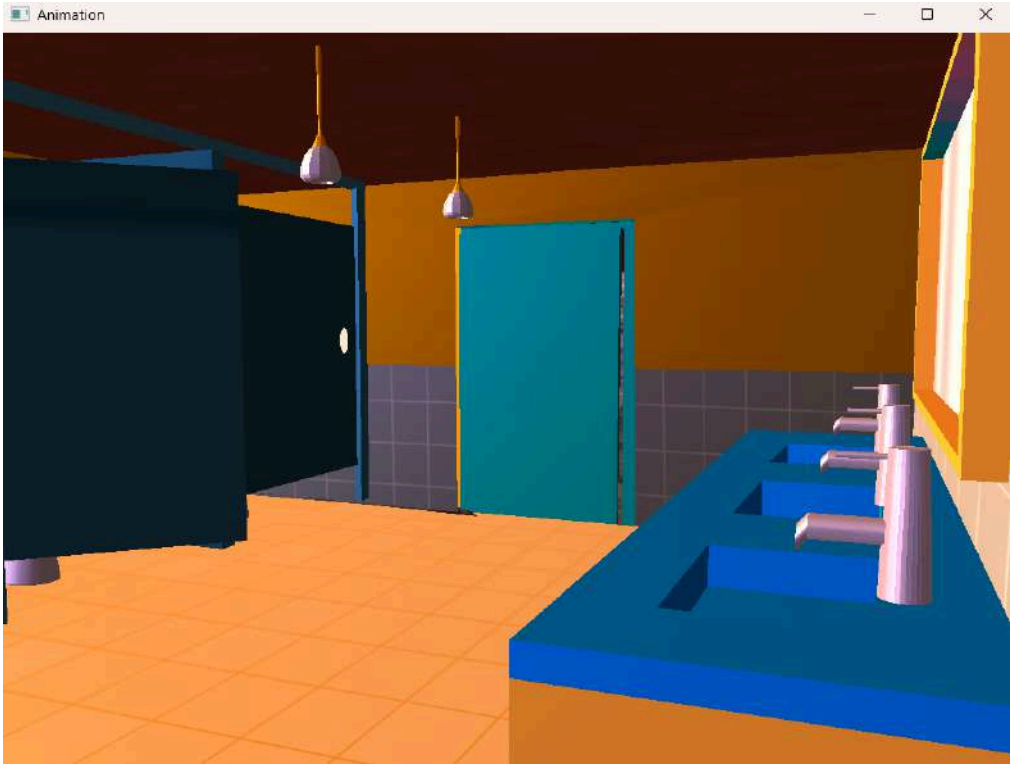

Animation

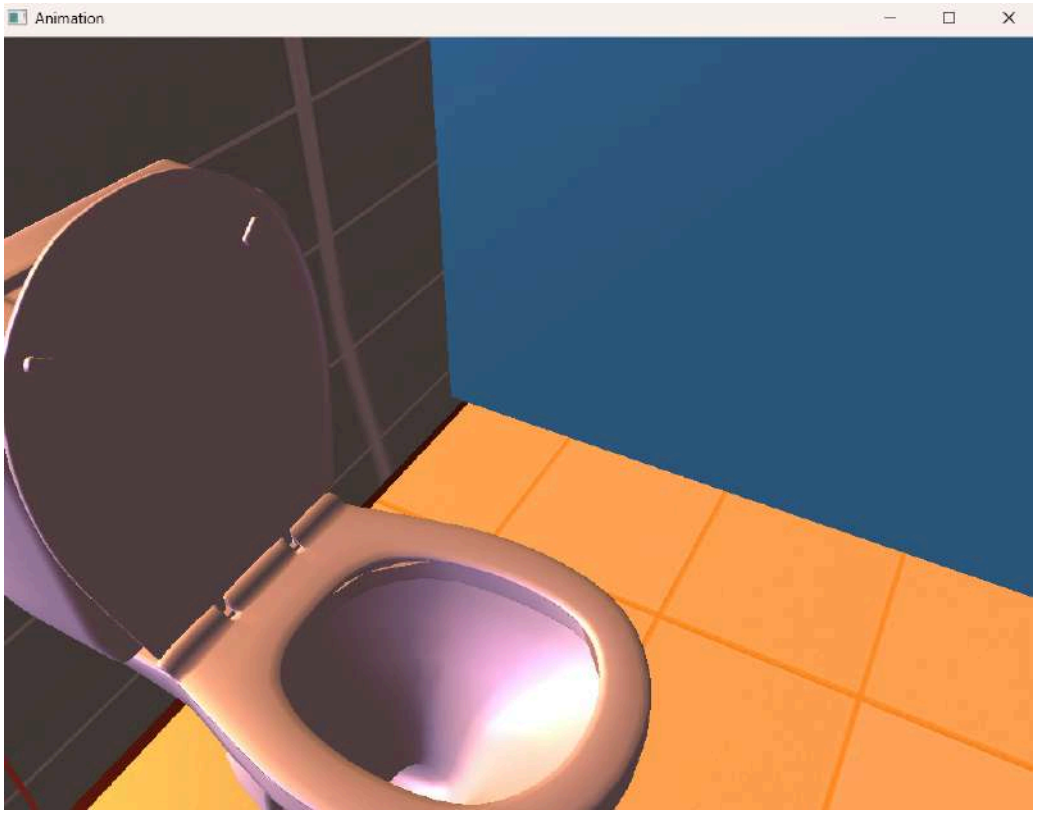

Animation
Trajinera

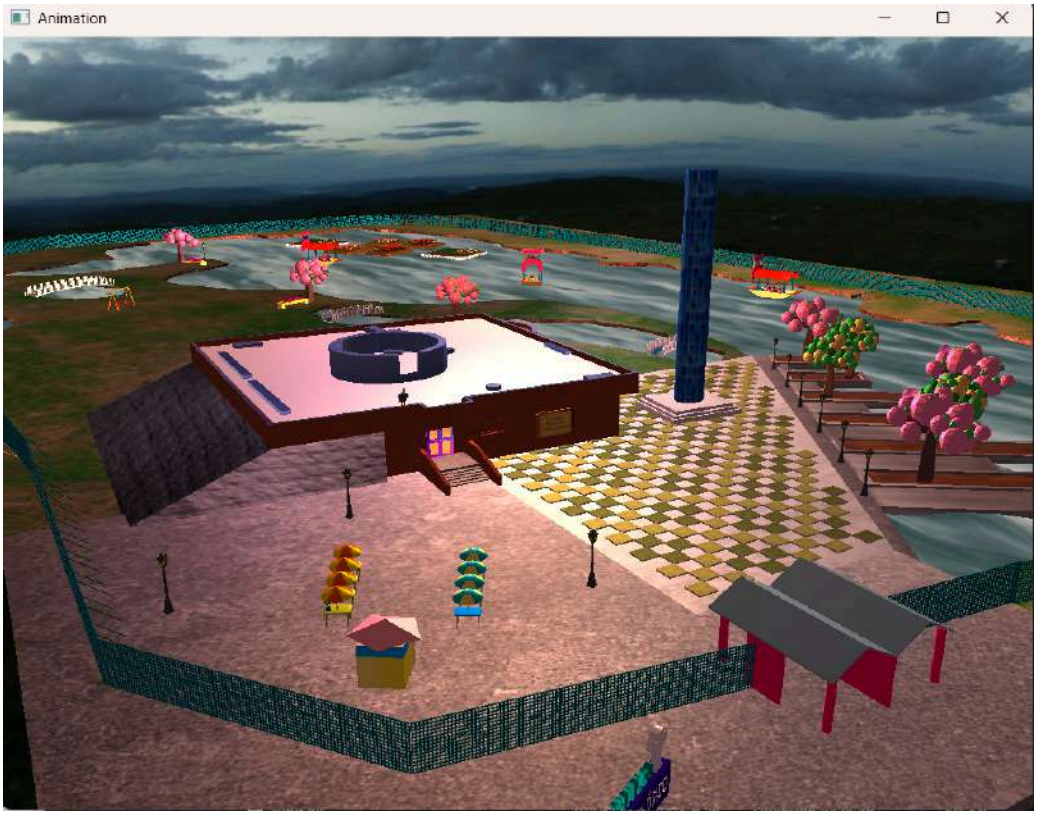

Animation
XOCHIMILCO
AESTHETIC AREA
ZONA DE CONSERVACIÓN COMUNITAR
DE LOS BIENES NATURALES
BRIEF HISTORY OF THE CHINAMPAS


Animation


Animation


Animation


Animation


Animation

Conclusions

This has been one of the longest and most challenging projects I've completed in my academic journey. However, it’s also been one of the most fascinating, not just because of its complexity, but because it gave me the opportunity to fully apply everything I’ve learned in the Computer Graphics course, both in theory and practice. It wasn’t just about meeting a requirement—it was about understanding and implementing a full graphics pipeline, from conceptualization to technical execution.

Throughout the development, I faced the challenge of learning how to model objects from scratch in Blender, managing my time to complete each stage, and applying techniques we covered in class, such as UV mapping, texturing, the synthetic camera, lighting, loading .fbx models, shader implementation, and integrating ambient sound. Every component—from the museum’s façade to the interactive elements and animations—was developed based on what I learned, always aiming for both visual and functional consistency.

One of the parts I enjoyed the most was seeing how everything started to come together: the models with their textures, the lighting effects, and the reactive animations, like the trajinera spinning as you approach or the ghost appearing when you interact with the book. All of this was made possible thanks to OpenGL, which, along with its libraries (GLFW, GLAD, GLM, stb_image, irrKlang), allowed us to build a complete and visually appealing 3D environment. The logic implemented for the camera control, interaction keys, and visual and sound responses resulted in a very rewarding experience.

Beyond the technical development, this project helped me realize the amount of work involved in creating an interactive graphic environment. It gave me a new perspective on the world of video games and simulations, and made me appreciate how complex it can be to make something as “simple” as opening a door or turning on a light feel natural.