UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA | DIVISIÓN DE INGENIERÍA ELÉCTRICA

Final Project

Technical Manual

Computación Gráfica e Interacción Humano – Computadora

Prepared by:

311243563

Professor: Ing. Carlos Aldair Román Balbuena

Group: 05

Semester: 2026-1

Technical Manual

Project Architecture

The architecture of the virtual environment serves as the central hub from which the user accesses the different thematic rooms of the project. The design is based on a stylized reinterpretation of the Louvre Museum, prioritizing a “cartoonish” aesthetic over realism, as defined in the project objectives. The central architectural element of the scene is a representation of the iconic glass pyramid of the Louvre, which acts as the main reference point for user navigation.

In this implementation, the central pyramid is surrounded by multiple building modules that represent the museum’s wings. These buildings, although inspired by the real façade, have been modeled in a simplified and modular way, using a repeated set of assets to optimize performance and maintain visual consistency with the cartoonish style.

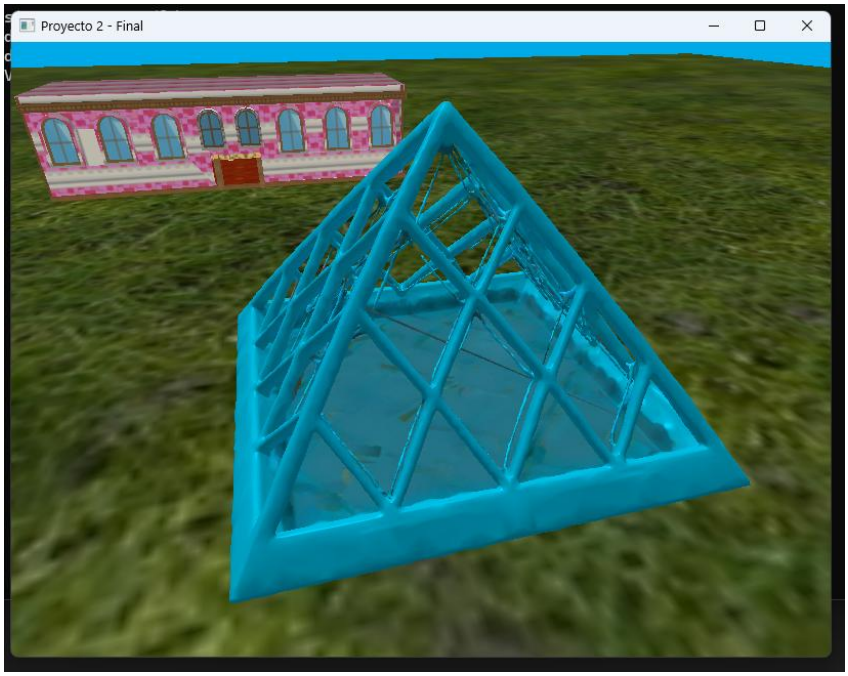
The modeling of the buildings is characterized by simplified geometry with clean textures that avoid photorealism. Key architectural elements, such as arched windows, have been

slightly exaggerated to align with the artistic approach of the project. This exterior structure houses the entrances to the interiors of the four thematic rooms (Egyptian Antiquities, European Painting), serving as a clear transition between the general exploration space and the specific gallery experiences.

Architecture of the Central Pyramid

The main architectural element of the environment is the central pyramid, a stylized representation of the iconic glass pyramid of the Louvre. Its 3D design aligns with the overall “cartoonish” approach of the project, prioritizing recognizable form and performance optimization over realism. The model consists of two main elements: the structural frame and the textured glass faces. The frame has been simplified into a thick solid-colored outline that defines the four main edges and the base of the pyramid. This approach deliberately omits the complex metal lattice of the real structure to maintain a clean and stylized silhouette.

For the pyramid faces, a single texture has been used instead of individually modeling the multiple glass panels. This texture simulates bluish-toned glass and features a repetitive diagonal pattern that creates the illusion of reflections or bevels. This texturing technique is a key optimization: it drastically reduces polygon load and rendering cost, achieving the desired appearance of a reflective glass surface without the geometric complexity of the real object.



Architecture of the Galleries

The buildings that house the thematic rooms constitute the gallery modules surrounding the central pyramid. Their design adheres to the project's "cartoonish" approach, serving as an exterior façade inspired by the Louvre's architecture but significantly simplified.

The architecture of these modules is characterized by:

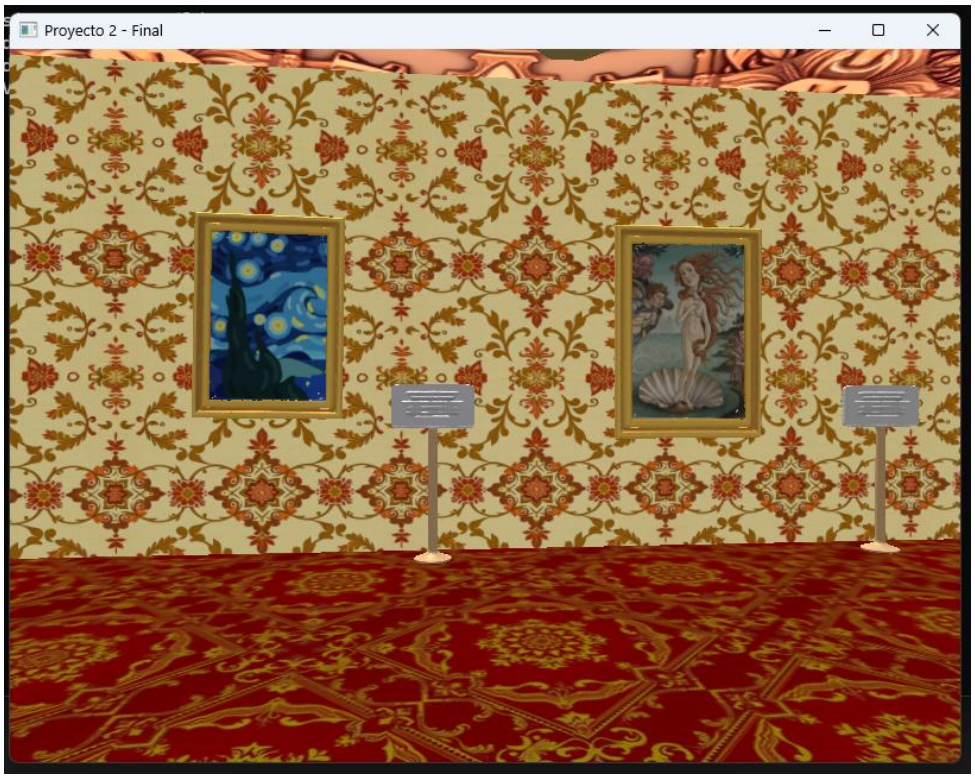
- **Modular Geometry:** The structure is an elongated, low-height rectangular block, designed as a modular and repeatable asset. This geometric simplification is essential for optimizing performance in the scene.
- **Façade and Texturing:** The main wall uses a light-toned brick texture (beige/cream). The texture is a uniform and clean pattern, avoiding photorealism in favor of the defined visual style. A decorative brown border, simulating a cornice, separates the façade from the roof.
- **Windows:** The building features two types of windows, both with thick brown frames and textured opaque blue glass, consistent with the material of the central pyramid.
- **Façade Windows:** A series of large arched windows on the main level.
- **Dormers:** Smaller arched windows emerging from the mansard roof, following its curvature.
- **Access Points:** At the center of the façade, an entrance is defined. This area is distinguished by a tiled floor texture (red/pink) and the inclusion of two objects simulating access barriers, indicating to the user the entry point to the corresponding interior room.

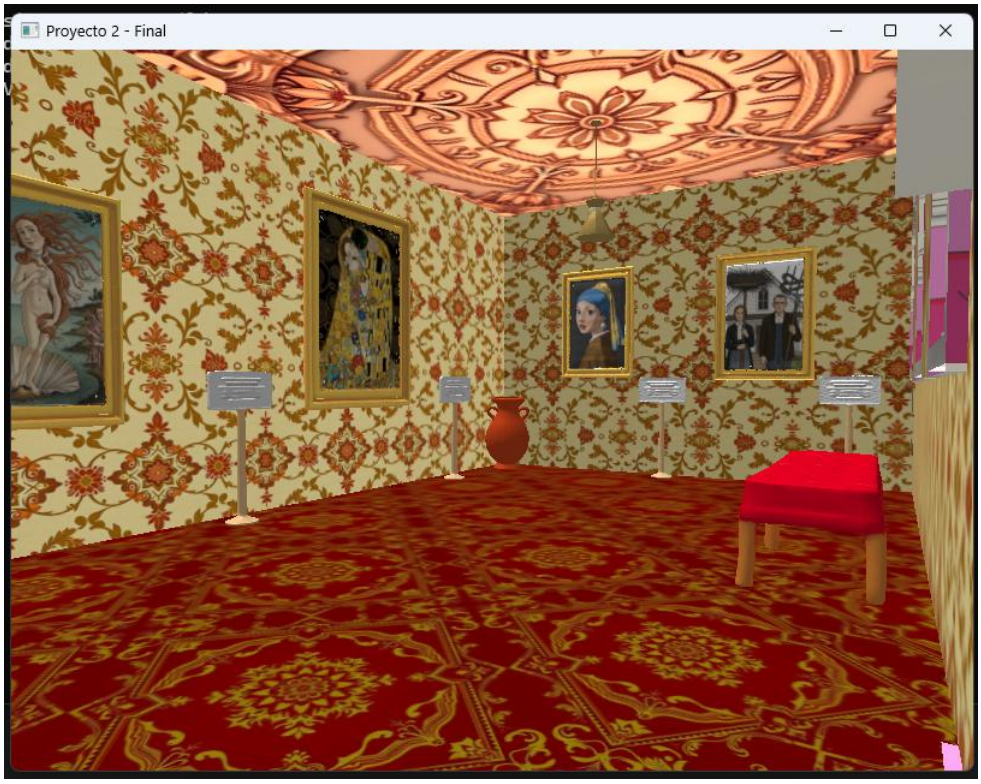
Interior Architecture: European Painting Room

The European Painting Room is designed to replicate the scale and atmosphere of a grand classical gallery, such as the Louvre's Grande Galerie, while maintaining the level of stylization and visual simplification defined for the project.

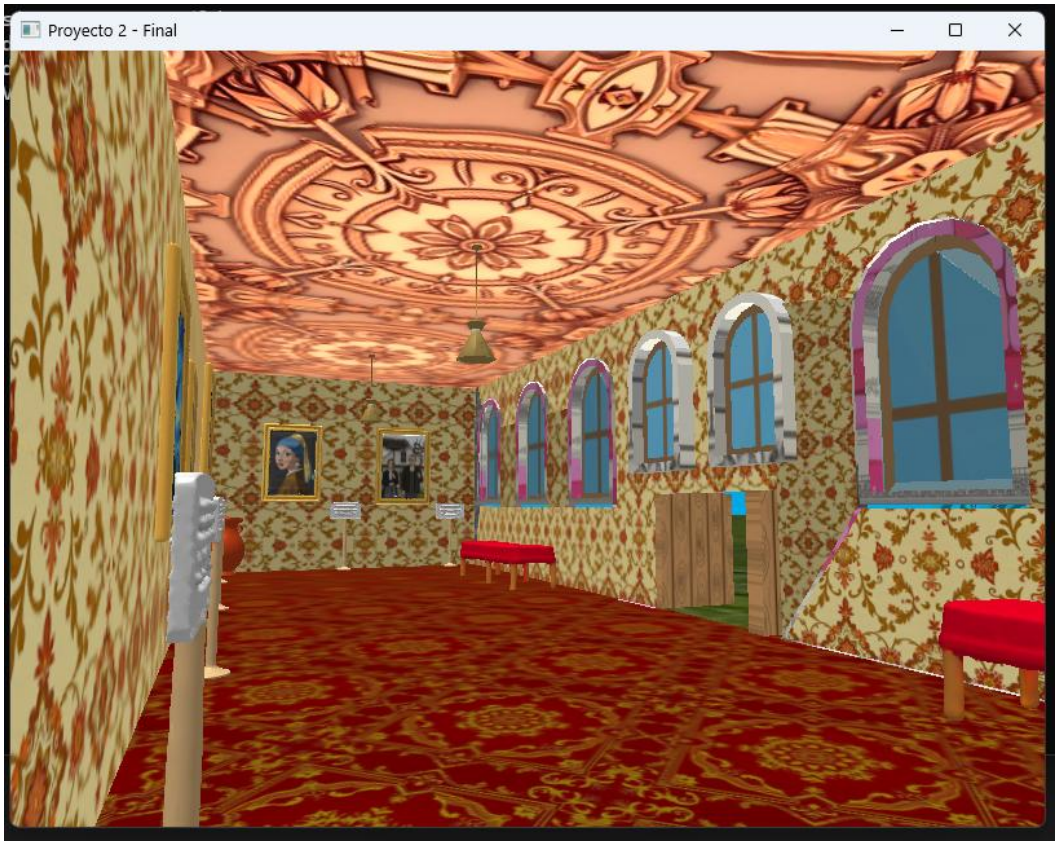
Structure and Proportions: The room features an elongated corridor structure and a high ceiling, designed to evoke a sense of monumentality. The walls are painted in light tones (cream/beige), with a lower dado or baseboard defined by a horizontal strip in a darker color (brown/reddish).

Overhead Lighting: The most important architectural feature is the lighting system. The ceiling is dominated by a barrel vault or a high arch along its central axis, which is perforated by a continuous overhead skylight running the entire length of the room. This skylight serves as the main simulated light source, essential for the exhibition of historical paintings. Its structure is defined by a simple geometric framework.


Proyecto 2 - Final


Proyecto 2 - Final


Proyecto 2 - Final


Proyecto 2 - Final

Exhibition Architecture:

The side walls are articulated with large arched frames integrated into the structure, finished in gold or brass. These frames serve to delimit and individually frame the large-scale paintings.

Displayed Works:

The paintings are integrated within these architectural frames and feature their own ornate golden frames, designed with simple geometry to maintain consistency with the cartoonish style.

Floor:

The floor is a dark, uniform color, similar to that used in other classical rooms, which helps create contrast and emphasize the brightness generated by the overhead skylight.

The design successfully evokes the historical grandeur of a painting gallery through the strategic use of scale and the simulation of natural overhead lighting.

Interior Architecture: Egyptian Antiquities Room

The Egyptian Antiquities Room is designed to evoke the atmosphere of a temple or hypostyle hall, using monumental architectural elements and a warm color palette, all under the lens of stylization.

Structure and Proportions:

The room features a symmetrical layout defined by a central aisle flanked by columns. The space conveys a sense of weight and scale through the massiveness of its elements, characteristic of Egyptian architecture.

Color Palette and Materials:

The architecture is dominated by a uniform texture in warm ochre or sandy yellow tones, simulating the color of ancient stone and the desert environment. This texture is applied equally to walls and columns. In contrast, the floor uses a dark, uniform texture to visually anchor the scene.

Monumental Columns:

The central architectural feature is a series of monumental cylindrical columns modeled with simplified geometry. The capitals (the top part of the column) are stylized, suggesting the floral motifs (papyrus or lotus) of Pharaonic architecture without the detailed complexity of photorealistic modeling.

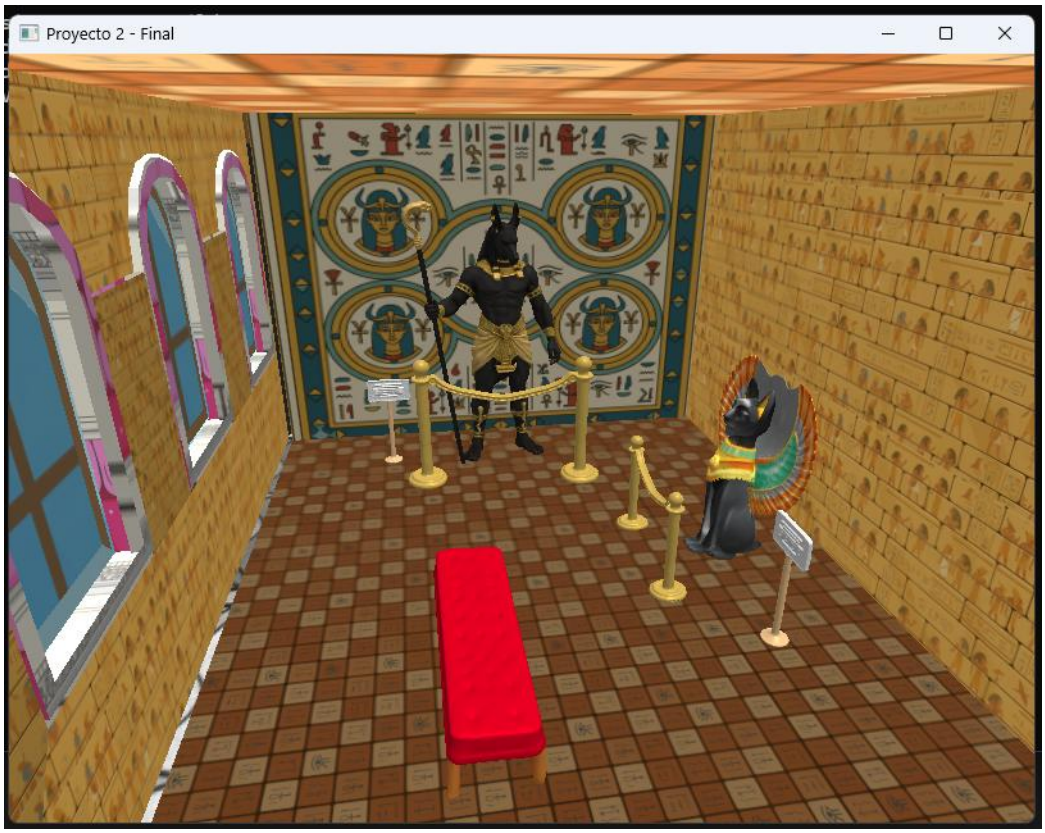
Thematic Decoration:

Decorative detail is achieved through the use of flat textures of simplified hieroglyphs applied directly to the walls and column shafts. This method efficiently incorporates the thematic identity of the room while optimizing performance.

Exhibition:

The room houses large-scale statues and sarcophagi, modeled with geometric and cartoonish forms. Visitor traffic management is implemented through the consistent use of golden stanchions and red ropes, delimiting the exhibition areas.

The architecture of this room achieves its thematic ambiance through structural repetition, color, and the application of iconic textures, fulfilling the project's directive to prioritize stylized visual expressiveness.



The project represents a virtual museum composed of four main rooms, each modeled in 3D and integrated using OpenGL and the GLM library for transformations. These rooms are loaded through the model system defined in the inicializarListaModelos() function, which includes the files sala1.obj, sala2.obj, sala3.obj, and sala4.obj. Additionally, each room contains different objects such as sculptures, gears, accessories, and containers, which are managed through a vector of ModeloInfo structures that store the model name, its reference, and its loading state.

All rooms and objects are drawn in the main rendering loop. The floor (piso.obj) is loaded as a common base, and on top of it, the rooms and their models are positioned according to the global transformations defined in traslacionGlobal. The free camera (Camera), controlled with WASD keys and the mouse, allows smooth exploration of the four rooms. Furthermore, progressive model loading optimizes performance, preventing freezes when rendering an extensive environment with multiple objects.

Magic Sarcophagus Animation

The main animation sequence is managed through a finite state machine controlled by the animState variable. This sequence is triggered by a keyboard event, which changes the state from idle to opening. The internal logic evaluates the delta time to sequentially transition through five operational phases: lid opening, forward translation of the mummy for its exit, a temporary pause, reverse translation for return, and finally, lid closure.

The kinematic behavior of the sarcophagus lid implements a composite transformation that combines linear translation along the X-axis with sinusoidal oscillation along the Z-axis. The use of trigonometric functions modulated by the opening position simulates a levitation or mechanical vibration effect while the lid slides, calculating the height based on the opening radius. Additionally, a vibration or “shaking” effect is applied, calculated using the global runtime to add realism to the piece’s movement over the base.

The mummy’s movement is based on hierarchical modeling where the torso acts as the parent node and the limbs as child nodes. Locomotion uses an oscillatory variable that alternately rotates the legs within a defined angular range, reversing direction upon reaching the limits to simulate constant steps. Simultaneously, arm rotation dynamically adjusts according to the animation state; during walking and pause, they rotate along the X-axis to simulate a chasing pose, while in idle they maintain a retracted posture.

To simulate the mummy’s exit from the sarcophagus, synchronized linear interpolation is applied to both position and scale. As the entity moves forward along the Z-axis, the algorithm progressively increases its global scale from a unit value to a maximum of 1.4, creating a visual effect of forced perspective and dramatic approach. When the animation reverses for the return, both position and scale are negatively interpolated until they return to the original confinement values inside the sarcophagus.

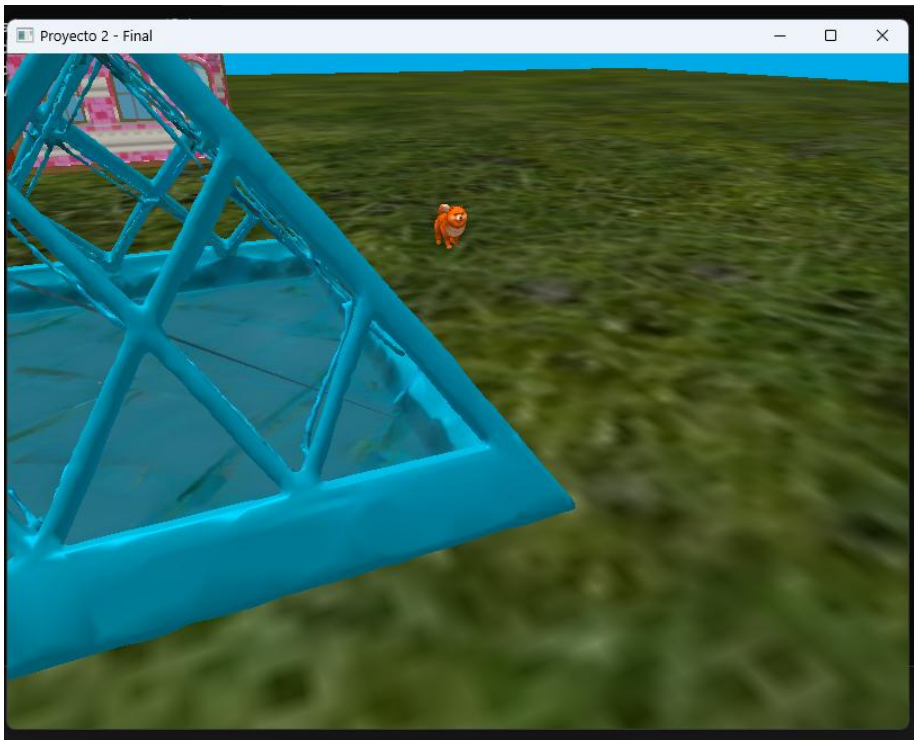
Walking Dog Animation

The dog’s movement through the scene is managed by a state machine that defines a predefined rectangular trajectory. The code divides the path into four distinct phases: move forward, turn left, return, and turn right. Depending on the active state, position coordinates on the X or Z axes are updated, and the body orientation is adjusted to fixed angles (0°, 90°, 180°, or 270°). The system constantly checks whether the model has reached the defined

scene limits to transition to the next segment, achieving a continuous and autonomous movement cycle.

To simulate walking, sinusoidal functions linked to the application's runtime are used, generating smooth and periodic oscillations. These functions calculate rotation angles for the legs and tail in each frame. To achieve a natural effect, mathematical phase shifts are applied in the formulas for each limb; this ensures that the front and rear legs move alternately rather than simultaneously, imitating the characteristic gait of a quadruped, while the tail swings with its own independent rhythm.

Regarding graphical construction, a hierarchical model is used where the body functions as the main or "parent" node. When drawing the scene, the global position and rotation of the torso are calculated first according to the current trajectory. Subsequently, the head, legs, and tail are drawn based on the body's location, inheriting its transformations. This allows the limbs to follow the dog throughout the scene automatically, applying only their local rotations for joint animation.



SkyBox

The implementation of the SkyBox in the project was carried out with the goal of providing an immersive visual environment that simulates a surrounding three-dimensional

background. A skybox consists of a large cubic box that encloses the entire scene, with textures applied to its interior, creating the illusion of an infinite sky or landscape that is unaffected by perspective or camera position. In this project, the skybox was used to set the museum within an enclosed environment consistent with the lighting and atmosphere of the scene.

```
//Skybox
GLuint skyboxVBO, skyboxVAO;
glGenVertexArrays(1, &skyboxVAO);
glGenBuffers(1, &skyboxVBO);
glBindVertexArray(skyboxVAO);
glBindBuffer(GL_ARRAY_BUFFER, skyboxVBO);
glBufferData(GL_ARRAY_BUFFER, sizeof(skyboxVertices), &skyboxVertices, GL_STATIC_DRAW);
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(GLfloat), (GLvoid*)0);
```

The implementation process begins with defining the vertices of the cube that forms the skybox. These vertices are stored in the skyboxVertices array, which describes the six faces of the cube with their respective three-dimensional coordinates. Each face of the cube is rendered using triangles, and no normal or conventional texture coordinate information is required, as the projection is calculated in the shader. This cube is drawn around the observer's viewpoint so that the camera always appears to be at the center of the environment.

Next, a specific Vertex Array Object (VAO) and Vertex Buffer Object (VBO) are created for the skybox using the functions glGenVertexArrays and glGenBuffers. The cube's vertices are loaded into GPU memory using glBufferData, and the position attribute in the vertex shader (index 0) is configured with glVertexAttribPointer. This setup allows the shader to receive the position of each vertex when drawing the cube.

For the skybox textures, a cubemap is used, which is a special texture composed of six images, one for each face of the cube. In the code, the paths to these images are stored in a vector called faces, which includes the textures right.jpg, left.jpg, top.jpg, bottom.jpg, back.jpg, and front.jpg, located in the SkyBox/ folder. These images are loaded using the function TextureLoading::LoadCubemap(faces), which generates an OpenGL texture identifier (cubemapTexture) associated with the cubemap. This resource is then linked to the shader responsible for rendering the skybox.

The shader used for the skybox consists of the files SkyBox.vs (vertex shader) and SkyBox.frag (fragment shader). In the vertex shader, the cube's vertices are transformed

only by the view and projection matrices, but the translation component of the view is removed so that the skybox always remains centered relative to the camera, regardless of its movement. This way, the observer can move freely without perceiving changes in the apparent distance of the environment. In the fragment shader, the corresponding color sample is taken from the cubemap texture based on the vertex vector direction, correctly projecting the texture onto the inner faces of the cube.

During the rendering process, the depth function is temporarily changed (glDepthFunc(GL_LEQUAL)) to ensure that the skybox is drawn behind all objects in the scene. Then, the skybox shader is activated (skyboxshader.Use()), the view and projection matrices are updated with the camera values, and the cubemap texture is bound to the active texture unit. Finally, the cube is drawn using glDrawArrays(GL_TRIANGLES, 0, 36), utilizing the 36 vertices that make up the six faces. At the end, the depth function is restored to its original value (glDepthFunc(GL_LESS)) so that the rest of the objects render correctly over the background.

Thanks to this implementation, the skybox creates a three-dimensional environment that surrounds the entire museum scene, providing visual depth without increasing the computational cost of rendering a distant scenario. This resource not only improves the overall aesthetics but also contributes to the perception of scale and realism in the virtual environment.

Technologies Used

OpenGL Implementation and Rendering

The core of the project is managed through the file ProyectoFinalMuseo.cpp, which uses the OpenGL API for real-time graphical rendering. The implementation follows the modern programmable graphics pipeline (Modern OpenGL).

Environment and Context Setup

To interact with the operating system and manage the OpenGL context, the project does not use OpenGL directly but relies on two essential C++ libraries:

- **GLFW (Graphics Library Framework):** Used for creating and managing the application window (glfwCreateWindow), handling the OpenGL context

(glfwMakeContextCurrent), and capturing user input events (keyboard and mouse), as seen in the KeyCallback and MouseCallback functions.

- **GLEW (OpenGL Extension Wrangler):** Initialized with glewInit(). Its role is crucial for loading pointers to modern OpenGL extension functions (version 3.3+), enabling access to features such as shaders, vertex buffers, and vertex arrays.

Shader and Program Management

The project implements a programmable pipeline using GLSL shaders (OpenGL Shading Language). Multiple shader programs are loaded for different purposes:

- **lightingShader:** The main program, loaded from Shader/lighting.vs (Vertex Shader) and Shader/lighting.frag (Fragment Shader). This complex shader handles all lighting calculations for the scene.
- **lampShader:** A simpler shader used to render visual representations (cubes) of point lights, giving them a solid color.
- **skyboxShader:** A specific shader for rendering the skybox background, which uses a cubemap technique.

Program activation is performed by calling lightingShader.Use() before sending uniforms and drawing the corresponding objects.

Models, Buffers, and Attributes

The scene's geometry is managed in two ways:

- **Complex Models:** A custom Model class is used to load geometry files (e.g., Models/museo.obj, Models/cuerpo.obj). This class (defined in Model.h, not visible but inferred) is responsible for internally creating the Vertex Array Objects (VAO), Vertex Buffer Objects (VBO), and Element Buffer Objects (EBO) needed to store the model's geometry and efficiently send it to the GPU.
- **Primitive Geometry:** For the Skybox and lamps, the geometry is explicitly defined in the code (e.g., skyboxVertices[]). VAOs and VBOs are manually configured (glGenVertexArrays, glGenBuffers, glBufferData), and vertex attributes are specified

(glVertexAttribPointer, glEnableVertexAttribArray) to indicate to OpenGL how to interpret the data (position, normals, etc.).

Rendering Loop (Game Loop)

The while (!glfwWindowShouldClose(window)) loop is the heart of the program. In each frame, it performs the following OpenGL operations:

- **Buffer Clearing:** Color and depth buffers are cleared (glClearColor, glClear) to prepare for the next frame.
- **Test Enabling:** Depth testing is activated (glEnable(GL_DEPTH_TEST)) to ensure objects are drawn in the correct occlusion order (closer objects hide farther ones).
- **Uniform Management:** Before drawing, crucial data (uniforms) are sent to the shaders.

Transformations (MVP Matrices)

The coordinate system is managed using the GLM library (OpenGL Mathematics). In each frame, the three main matrices are calculated and sent to the shaders:

- **Projection:** Calculated once using glm::perspective(). Defines the camera type (perspective) and field of view.
- **View:** Obtained from camera.GetViewMatrix(). This matrix is updated in DoMovement() and MouseCallback() to simulate the observer's camera movement and orientation.
- **Model:** The most dynamic matrix. It is recalculated for each object (or part of an object) to be drawn. Animations (such as those for the mummy and the clock in the Animation() function) work by updating variables (mummyLegRot, tapaPosX, anguloPendulo) that are then used to create unique model matrices with glm::translate, glm::rotate, and glm::scale. This matrix is sent to the shader with glUniformMatrix4fv(modelLoc, ...) just before calling TuModelo.Draw().

Lighting System (Uniforms)

The lightingShader implements a complex lighting model (likely Phong) that receives multiple light sources through uniforms:

- **Directional Light (dirLight):** A global light simulating the sun.
- **Point Lights (pointLights):** An array of NR_POINT_LIGHTS (17) lights. The code defines their positions (pointLightPositions) and manages their colors (pointLightColors). The SPACE key toggles these lights by changing their color between COLOR_YELLOW_ON and COLOR_OFF.
- **Spotlight (spotLight):** A flashlight that follows the camera's position and direction.

Drawing Techniques

- **Model Drawing:** Most objects (SalasMuseo, MummyBody, clock, etc.) are drawn by calling their TuModelo.Draw(lightingShader) function. This internal function binds the model's VAO and executes the glDrawElements or glDrawArrays call.
- **Skybox Drawing:** A special technique is used at the end of the loop. Depth writing is disabled (glDepthFunc(GL_LEQUAL)), and the skyboxShader is used along with a Cubemap texture (GL_TEXTURE_CUBE_MAP) to draw a cube that always appears in the background, creating the illusion of an infinite environment.

3D Modeling and Editing Software

Blender: Used as the main 3D creation suite. Its critical functions in the project included:

- **Polygonal Modeling:** Creation of high- and low-density meshes for environments, objects, and characters.
- **UV Mapping (UV Unwrapping):** Unfolding the coordinates of 3D meshes into a 2D space for correct texture projection.
- **Texturing and Materials:** Assignment and configuration of texture maps (diffuse, normal, etc.) and material properties to define the visual appearance of assets.

AI-Assisted Tools for Asset Generation

To optimize production time, web platforms based on artificial intelligence were used for rapid asset generation:

- **imgto3d.ai:** This platform was used for generating 3D models. Its function was to convert 2D reference images into base 3D meshes, speeding up the creation of specific objects for the environment's rooms.
- **Tripo AI Studio:** This web service was employed for assisted texturing. It enabled quick generation and application of materials and textures to 3D models, facilitating iteration and finalization of the visual appearance of assets.

Detailed Activity Schedule

Phase	Activity	Weeks	Dates
1. Planning	Definition of the theme and collection of visual references (PDF).	Week 1	Oct 06 – Oct 12
	Environment setup (GLFW, GLEW, SOIL2) and base code structure.	Week 1	Oct 06 – Oct 12
2. Scene	Modeling/Search for assets (Room, Sarcophagus, Mummy, Dog).	Week 2	Oct 13 – Oct 19
	Implementation of Skybox (Cubemap) and texture loading.	Week 2	Oct 13 – Oct 19
3. Lighting	Implementation of Shaders (lighting.vs, lamp.vs).	Week 3	Oct 20 – Oct 26
	Configuration of light sources (Directional, Point Lights, SpotLight).	Week 3	Oct 20 – Oct 26
4. Logic (Dog)	Development of hierarchical modeling for the dog (Torso → Legs/Tail).	Week 4	Oct 27 – Nov 02
	Programming of rectangular trajectory (State machine 0–3).	Week 4	Oct 27 – Nov 02
5. Logic (Mummy)	Programming of keyboard interaction (Input M, N).	Week 5	Nov 03 – Nov 09
	Implementation of sarcophagus kinematics (Opening/Levitation).	Week 5	Nov 03 – Nov 09
6. Refinement	Integration of scale interpolation and mummy movement.	Week 6	Nov 10 – Nov 16
	Adjustment of speeds (deltaTime) and correction of animation limits.	Week 6	Nov 10 – Nov 16
7. Delivery	Final integration tests, code cleanup, and comments.	Week 7	Nov 17 – Nov 24
	Documentation and preparation of the Technical Manual.	Week 7	Nov 17 – Nov 24

Approximate Cost Estimation

The following budget estimate breaks down the costs associated with the development of the graphic system “Egyptian & Louvre Museum.” The calculation is based on an estimated dedication of 140 man-hours (an average of 20 hours per week over 7 weeks) and standard market rates for a Junior Graphics Programmer profile.

Concept	Description	Estimated Cost (MXN)
Human Resources	Development, Logical Design, and Integration	$35,000.00
Infrastructure	Hardware Depreciation (Development PC)	$2,500.00
Licensing	Software and Tools (Visual Studio, etc.)	$0.00*
Operating Expenses	Electricity and Internet (Prorated)	$1,200.00
TOTAL	Total Project Cost	$38,700.00

Since the project was carried out by a single person, the developer assumed multiple technical roles throughout the software lifecycle. The cost is calculated based on an hourly rate of **$250.00 MXN**:

- **Management and Planning (10 Hours):** Definition of scope, search for visual references (Louvre Museum, Egyptian Hall), and scheduling.
- **Graphics Programming – C++/OpenGL (80 Hours):** Implementation of the rendering pipeline, shaders, lighting (Phong/Blinn-Phong), skybox, and configuration of libraries (GLEW, GLFW, GLM).
- **Logic and Animation (30 Hours):** Development of state machines for the mummy, sarcophagus kinematics, and hierarchical modeling of the dog.
- **Integration and QA (20 Hours):** Loading OBJ models, texture mapping, debugging, and performance testing.

Calculation: 140 hours × $250.00 MXN/hour = **$35,000.00 MXN**

Infrastructure and Tools

- **Computer Equipment:** Depreciation of a mid-range workstation (CPU i5/i7, dedicated GPU, 16GB RAM) during two months of intensive use.

    - Estimated equipment value: **$25,000 MXN**
    - Depreciation allocated to the project (10%): **$2,500.00 MXN**
- **Software Used:**
    - Microsoft Visual Studio Community 2022 (Free License).
    - Open Source Libraries: OpenGL, GLFW, GLEW, SOIL2, GLM (MIT/Zlib Licenses).
    - Blender (for reviewing 3D models – Open Source).

Documentation of Self-Learning of New Tools

311243563

To carry out the 3D gallery project, I had to go through a self-learning process in several tools, ranging from code management to the creation of digital assets.

My first task was to establish a functional and collaborative development environment. For this, I had to learn how to use Visual Studio as my main Integrated Development Environment (IDE), focusing on configuring the graphic libraries required for the project. At the same time, I learned to use GitHub for version control. The initial challenge was not just uploading the code but understanding how to properly configure the .gitignore file for a Visual Studio project. At first, I uploaded temporary build files that slowed down the repository. I solved this problem by researching and applying a standard .gitignore template, which allowed me to keep the source code clean and functional. This learning was essential for managing programming tasks and assembling the scene.

Creating the gallery objects (furniture, sarcophagus, etc.) was a technical challenge that I approached with a hybrid workflow. To quickly generate a base of models and textures, I used two AI tools. I started using imgto3d.ai to convert my reference images into basic 3D meshes. Then, I used studio.tripo3d.ai to obtain quick textures for generic materials. These tools helped me achieve very fast prototyping.

However, the models generated by AI often had complex topologies, and automatic texturing did not give me the level of detail control needed for the project. For this reason, it was essential to learn Blender for refinement and, crucially, for texturing with custom images through UV Mapping.

UV Mapping was the most difficult concept I had to master. I understood that this technique was essentially “unwrapping” my 3D model into a 2D plane so that the texture image would project correctly. I researched multiple tutorials on how to mark the “seams.” My main obstacle was deciding where to cut the model so that the texture would not appear stretched or broken. I practiced with simple models until I managed to understand the logic of unwrapping. This learning process allowed me to import the base models generated by AI, clean them, and then apply custom textures (such as specific hieroglyphs or the direction of wood grain) with the exact precision required for the gallery’s final aesthetic. Finally,

mastering this process enabled me to effectively carry out the tasks of “3D Object Modeling” and “Creation and Application of Textures.”

Conclusions

The development phase confirmed the technical feasibility of the project despite the accelerated schedule. A hybrid strategy was implemented for creating 3D assets: artificial intelligence tools (imgto3d.ai and studio.tripo3d.ai) were used for rapid prototyping, followed by Blender for essential technical refinement, especially UV Mapping. This approach allowed maintaining visual quality and the precision required for textures without sacrificing efficiency. Additionally, the final integration in the Visual Studio environment, managed through GitHub, demonstrated the robustness of the architecture. The implemented system smoothly handles user navigation and key animations (such as the sarcophagus and the cube), proving that the solution is stable for future expansions.

The social impact of the project lies mainly in democratizing access to culture. Being a fully virtual experience eliminates geographical and economic barriers, making high-value cultural exhibitions accessible to a global audience without the need for physical travel. The interactive and immersive nature of the gallery enhances user engagement, resulting in a superior educational experience and greater information retention compared to passive formats. Finally, the project establishes a viable reference model for remote content curation, fostering collaboration among specialists regardless of their location.

From an economic perspective, the project presents significant cost optimization. Eliminating expenses associated with physical infrastructure—including construction, insurance, security, and maintenance—minimizes initial and operational capital. The reliance on low-cost or open-source tools (Blender, GitHub) and the acceleration of time-to-market thanks to the AI-assisted workflow maximized resource efficiency. This translates into a highly scalable business model, as the marginal cost of adding new objects or virtual rooms is minimal compared to expanding a traditional museum.

The virtual gallery project positions itself as a low environmental impact solution. Eliminating physical infrastructure means the project does not generate construction waste and does not require intensive energy use for climate control, lighting, or security in an exhibition building. Furthermore, remote access to the gallery directly contributes to reducing the carbon footprint generated by mass transportation of visitors and artworks. In essence, the

technological implementation promotes a more sustainable and efficient use of resources by replacing the need for material infrastructure with a digital solution.

References

1. **Amazon.** (n.d.). *Sarcophagus [Image].* Retrieved from https://m.media-amazon.com/images/I/71PcWupI0RL.jpg
   License: Not specified.

2. **Free3D.** (n.d.). *Mummy [3D Model].* Retrieved from https://free3d.com/es/modelo-3d/mummy-7958.html
   License: Free download (Free3D).

3. **Freepik.** (n.d.). *Green plant in red pot [Premium Vector].* Retrieved from https://www.freepik.es/vector-premium/planta-verde-maceta-roja-icono-3d-planta-interior-o-planta-maceta-flor-ilustracion-vectorial-3d-jardin-domestico-sobre-fondo-blanco-jardineria-naturaleza-decoracion-concepto-botanica_42332064.htm
   License: Freepik Premium.

4. **Freepik.** (n.d.). *Vintage gold frame [Premium Vector].* Retrieved from https://www.freepik.es/vector-premium/marco-oro-vintage_2205924.htm
   License: Freepik Premium.

5. **Freepik.** (n.d.). *Set of realistic illuminated photo frames [Free Vector].* Retrieved from https://www.freepik.es/vector-gratis/conjunto-marco-fotos-iluminado-realista_20289082.htm
   License: Freepik Free.

6. **Freepik.** (n.d.). *Realistic mockup of gallery wall image exhibition [Free Vector].* Retrieved from https://www.freepik.es/vector-gratis/maqueta-realista-exposicion-galeria-imagenes-pared-ilustracion-vector-sofa-visitantes_33770595.htm
   License: Freepik Free.

7. **Freepik.** (n.d.). *Realistic studio lights empty background design [Free Vector].* Retrieved from https://www.freepik.es/vector-gratis/luces-estudio-realistas-diseno-fondo-vacio_10016499.htm
   License: Freepik Free.

8. **Sketchfab.** (n.d.). *Pixel art voxel art person with a hat [3D Model].* Retrieved from https://sketchfab.com/3d-models/pixel-artvoxel-art-person-with-a-hat-

13aa4bfb0d2f4573af0154dec3b2c269
License: Free according to Sketchfab platform.

9. **Freepik.** (n.d.). *Ancient and modern columns [Free Vector].* Retrieved from https://www.freepik.es/vector-gratis/antiguas-duricas-cilindricas-ordenes-jonicas-modernas-columnas-cubicas-marmol_4393674.htm
   License: Freepik Free.
10. **Freepik.** (n.d.). *Isometric illustration of Louvre Museum [Premium AI Image].* Retrieved from https://www.freepik.es/imagen-ia-premium/ilustracion-isometrica-lowpoly-museo-louvre-su-iconica-piramide-vidrio_266441597.htm
    License: Freepik Premium.
11. **Freepik.** (n.d.). *Historic façade of Louvre Palace [Premium AI Image].* Retrieved from https://www.freepik.es/imagen-ia-premium/ilustracion-vectorial-fachada-historica-palacio-louvre_249038071.htm
    License: Freepik Premium.
12. **Tripo3D.** (n.d.). *Glass pyramid with diamond grid façade [3D Model].* Retrieved from https://studio.tripo3d.ai/workspace/overview?project=2aae6996-e3d4-4b43-ba9a-9f65d0df70b6
    License: Free (Tripo3D).
13. **Freepik.** (n.d.). *Marble stone background for 3D design [Premium Vector].* Retrieved from https://www.freepik.es/vector-premium/fondo-piedra-marmol-usos-diseno-ilustracion-3d_10510617.htm
    License: Freepik Premium.
14. **Vecteezy.** (n.d.). *Mona Lisa cartoon icon [Vector].* Retrieved from https://es.vecteezy.com/arte-vectorial/52103629-mona-lisa-icono-dibujos-animados-estilo-icono-blanco-antecedentes
    License: Free with attribution (Vecteezy).
15. **Pinterest.** (n.d.). *The Scream [Image].* Retrieved from https://mx.pinterest.com/pin/7318418133384987/
    License: Limited use, see Pinterest conditions.
16. **Freepik.** (n.d.). *Starry Night by Van Gogh [Free Vector].* Retrieved from https://www.freepik.es/vector-gratis/ilustracion-pintura-van-gogh-diseno-plano_29725492.htm
    License: Freepik Free.

17. **Freepik.** (n.d.). *Sumerian-Assyrian cuneiform writing [Free Vector].* Retrieved from https://www.freepik.es/vector-gratis/escritura-sumeria-asiria-cuneiforme-acadia_15754391.htm
License: Freepik Free.
18. **WikiArt.** (n.d.). *Cash Register (1917) [Painting].* Retrieved from https://www.wikiart.org/es/amadeo-de-souza-cardoso/cash-register-1917
License: Educational use permitted (WikiArt).
19. **WikiArt.** (n.d.). *La Composizione TA (1916) [Painting].* Retrieved from https://www.wikiart.org/es/carlo-carra/la-composizione-ta-1916
License: Educational use permitted (WikiArt).
20. **WikiArt.** (n.d.). *Silence (1915) [Painting].* Retrieved from https://www.wikiart.org/es/georges-ribemont-dessaignes/silence-1915
License: Educational use permitted (WikiArt).
21. **WikiArt.** (n.d.). *Balance [Painting].* Retrieved from https://www.wikiart.org/es/francis-picabia/balance
License: Educational use permitted (WikiArt).
22. **WikiArt.** (n.d.). *Ambiguous Figures [Painting].* Retrieved from https://www.wikiart.org/es/max-ernst/ambiguous-figures-1-copper-plate-1-zinc-plate-1-rubber-cloth
License: Educational use permitted (WikiArt).
23. **WikiArt.** (n.d.). *Bicycle Wheel (1913) [Painting].* Retrieved from https://www.wikiart.org/es/marcel-duchamp/bicycle-wheel-1913
License: Educational use permitted (WikiArt).
24. **WikiArt.** (n.d.). *The Hill [Painting].* Retrieved from https://www.wikiart.org/es/man-ray/the-hill
License: Educational use permitted (WikiArt).
25. **Freepik.** (n.d.). *Nefertiti statue queen woman pharaoh [Free Vector].* Retrieved from https://www.freepik.es/vector-gratis/estatua-nefertiti-reina-mujer-faraon-antiguo-egipto-ilustracion-vectorial-dibujos-animados_11433881.htm
License: Freepik Free.
26. **Free3D.** (n.d.). *Rectangular grass patch [3D Model].* Retrieved from https://free3d.com/es/modelo-3d/-rectangular-grass-patch--205749.html
License: Free download (Free3D).

27. **Free3D.** (n.d.). *Small glass pyramid [3D Model].* Retrieved from https://free3d.com/es/modelo-3d/small-glas-pyramid-9315.html
License: Free download (Free3D).
28. **TurboSquid.** (n.d.). *Jackal god of the Egyptian [3D Model].* Retrieved from https://www.turbosquid.com/es/3d-models/3d-jackal-god-of-the-egyptian-3d-printable-1778764
License: May require purchase (TurboSquid).
29. **Freepik.** (n.d.). *Isometric illustration interior museum painting exhibition [Free Vector].* Retrieved from https://www.freepik.es/vector-gratis/ilustracion-isometrica-color-interior-museo-exposicion-pintura_3791816.htm
License: Freepik Free.
30. **Freepik.** (n.d.). *Fireplace frames and lamp [Premium Vector].* Retrieved from https://www.freepik.es/vector-premium/marcos-chimenea-chimenea-lampara_242532203.htm
License: Freepik Premium.
31. **Imgto3D.** (n.d.). *Image-to-3D model conversion [Website].* Retrieved from https://www.imgto3d.ai/es
32. **Tripo3D.** (n.d.). *3D model generator from images [Website].* Retrieved from https://studio.tripo3d.ai/home

Acquired Experience and Project Management

311243563

Throughout the development of this 3D virtual gallery project, I gained extensive hands-on experience in managing a complex digital production pipeline. I successfully implemented a hybrid workflow that combined rapid asset generation tools with professional software for detailed development and refinement.

One of the most technically challenging aspects was texturing in Blender, particularly UV mapping. This process required significant self-learning and practice to accurately unwrap complex models and apply custom image textures without distortion. Additionally, constructing the museum's architectural layout within the development environment posed a major challenge, demanding precise attention to model scaling, scene lighting, and the integration of navigation logic.

At first, the sheer volume of tasks and the tight deadline felt overwhelming. However, the custom compressed schedule we created using a Gantt chart proved invaluable. By proportionally adjusting timelines for each activity, I was able to break the project into manageable daily goals. This structured approach allowed me to organize and prioritize modeling, texturing, animation, and coding tasks effectively, ultimately enabling me to complete the entire project within the accelerated timeframe.

GitHub Repository

SandraLaparra/311243563_ProyectoFinalTeoria_GP05: Proyecto Final de la materia Computación Gráfica Teoría

In this Drive link, you can download the “Models” folder in .obj format:

https://drive.google.com/drive/folders/1ykp-Z47fhd6Pcpj6ks_qpUSEx2K-0pEJ?usp=drive_link