UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTY OF ENGINEERING
DIVISION OF ELECTRICAL ENGINEERING
COMPUTER ENGINEERING

COMPUTER GRAPHICS AND HUMAN-COMPUTER INTERACTION

FINAL PROJECT DOCUMENTATION

Gónzalez Cuellar Pablo Arturo 319241013

Lechuga Castillo Shareny Ixchel 319004252

Sánchez Medina José Santiago 319246881

Veraza García Amy Valentina 320490572

Solís Cisneros Cecilia Ximena 317042333

LAB GROUP: 02, 03, 04 & 13
THEORY GROUP: 04

SEMESTER: 2026-1

DEADLINE: 12/04/2025

Table of Contents

1. Introduction

This document outlines the development of the final project for the course **Computer Graphics and Human-Computer Interaction**. The project consists of a 3D environment inspired by the themes *Pre-Hispanic Culture* and *Mexican Lucha Libre*, blended with fantasy universes previously selected by each team member.

The scene features culturally significant elements such as pyramids, the Aztec calendar, and a reinterpretation of the traditional ball game, enhanced with vibrant visual touches. A central wrestling ring also serves as a key focal point. The entire environment is themed around the characters **Princess Peach**, **Mickey Mouse**, and **Finn the Human**.

The objective of the project is to apply the knowledge acquired throughout the course to design and build an interactive 3D environment incorporating dynamic lighting, realistic materials, hierarchical animations, and a functional camera system—striving for a balance between technical execution and creative visual storytelling.

2.1 Geometries

The scene was built entirely in 3D, combining models created in **Blender** with geometries generated and imported into **OpenGL**.

Among the modeled elements are the characters chosen by the team members. Preloaded models sourced from the web were used as a base. These models required several modifications, including:

- Segmenting the model by limb
- Adjusting the rotation axis for each limb
- Rescaling the models appropriately

These adjustments were essential to ensure proper integration with the animation system and maintain visual consistency within the 3D environment.

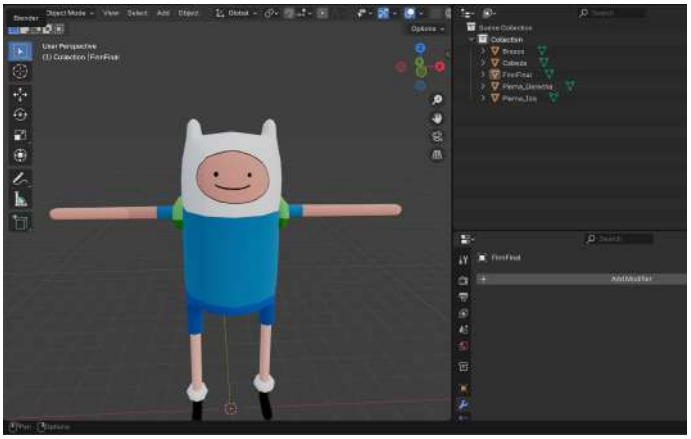

Fig. 1.0 – Finn Model

Fig. 1.1 – Peach Model

Fig. 1.2 – Mickey Model

- Pre-Hispanic pyramids, inspired by Chichén Itzá and the Pyramid of the Sun, were created using stepped prism structures.

Fig. 1.3 – Chichén Itzá Model

- **Aztec calendar**, modeled using cylinders and circular subdivisions, with engraved details represented through texture mapping.

Fig. 1.4 – Aztec Calendar Model

- **Peach's wrestling ring**, modeled as a central cube with rails and hierarchical posts, positioned in the main plaza.

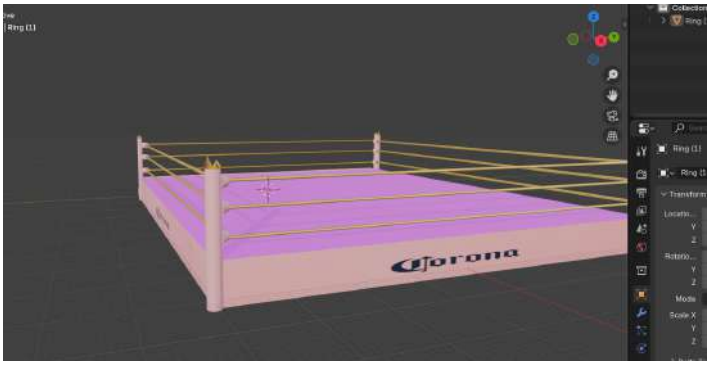

Fig. 1.5 – Wrestling Ring Model

- **The Mesoamerican ball game** was created using scaled cubes, cylinders, and themed toroids to replicate architectural and decorative elements.

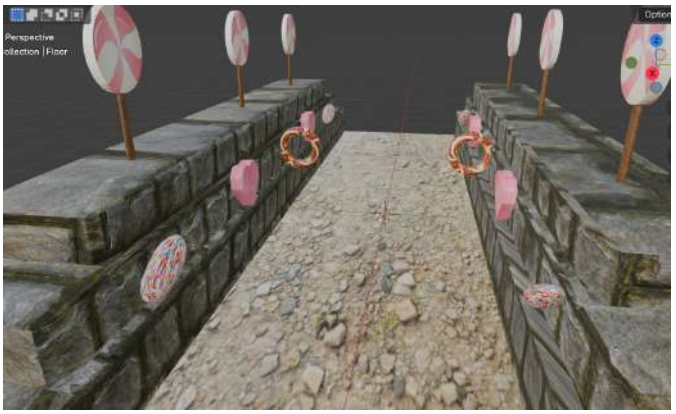

Fig. 1.6 – Ball Game Model

- A **hot air balloon with ears**, composed of a main sphere and semicircles representing the ears, with a rectangular basket suspended below.

Fig. 1.7 – Hot Air Balloon Model

- Benches and streetlights were constructed using textured prisms and placed along the walkthrough paths to enhance environmental detail and realism.

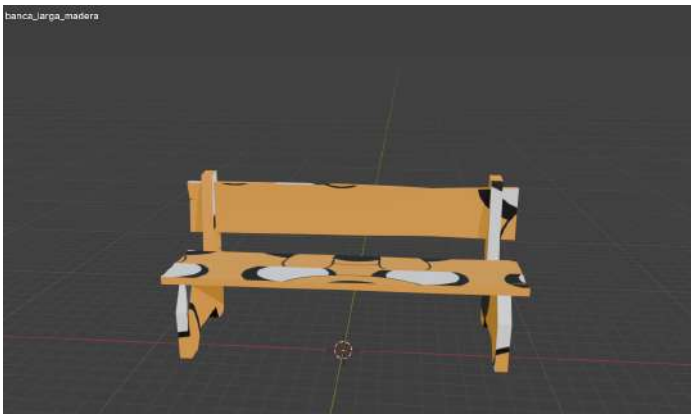

Fig. 1.8 – Bench Model

All models were manually scaled and positioned to maintain visual consistency and a realistic sense of scale relative to both the avatar and the environment.

- A large facade was also included, inspired by reconstructed monuments from ancient Tenochtitlán.

Fig. 1.9 - *Tenochtitlán*

Fig. 2.0 – Modeled House Facade

- Tula:

Fig. 2.1 – Tula Model

- Trees:

Fig. 2.2 – Modeled Trees

- **Marketplace (Tianguis):**

  Inspired by various video games, we envisioned the scene with a small open-air market, even if modest in size. The market includes elements such as baskets, chili peppers, woven mats (*petates*), avocados, jars.

Fig. 2.3 – Modeled Marketplace (Tianguis)

- Hut:

Fig. 2.4 – Hut Models

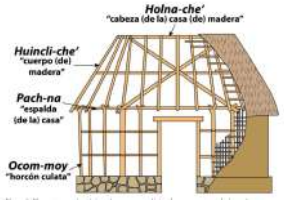

Fig. 2.5 – Hut Inspiration Reference

- Objects Created in OpenGL via Code:

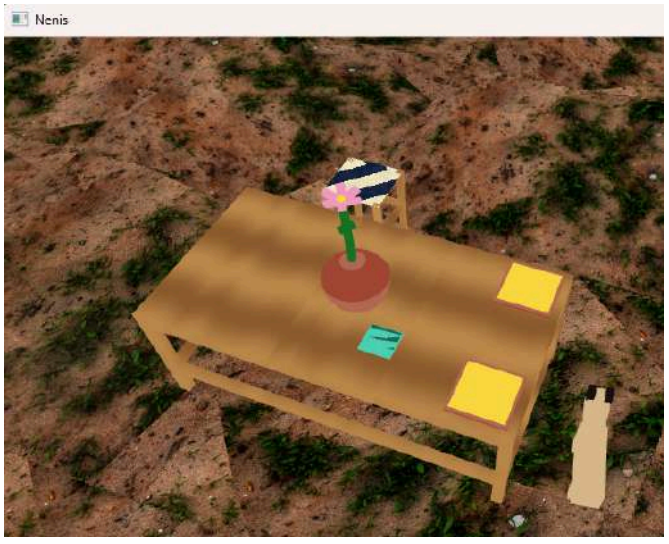

Fig. 2.5.1 – Models: Table, Mask, Plates, Chihuahua, Bench, Flower, and Vase

Fig. 2.5.2 – Models: Axe and Wheelbarrow

Unity Version:

Fig. 1.5.3 - *Kratos model*

Fig. 2.5.4 – Downloaded Tree Models Used in the Scene

- Statues

Fig 2.6.1 Ehecatl

Fig 2.6.2 Ehecatl

Fig 2.6.3 Tlaloc

Fig 2.6.4 Huitzi

Fig 2.6.5 Tezcatlipoca

Fig 2.6.6 Quetzalcoatl

Fig 2.6.6 Mictlantecutli

2.2 Texturing

For the first OpenGL version:

For the first OpenGL version, UV mapping was used to apply textures that simulate materials such as stone, sand, wood, and metal. Material properties—including ambient, diffuse, specular, and shininess—were configured to give each surface a unique response to lighting, thereby enhancing the realism of the environment.

Detailed textures were applied to both natural and architectural surfaces, contributing to a more immersive visual experience.

Fig. 2.6 – Hut Textured Using UV Mapping

Fig. 2.7 – Tula Monument Textured

Fig. 2.8 – Trees Textured with Scene-Appropriate Materia

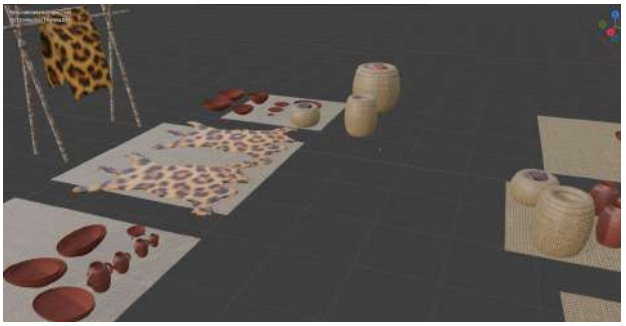

Fig. 2.9 – Marketplace (Tianguis) with Various Textures Applied

The most challenging object to texture using UV mapping, face by face, was the large facade.

This model required precise unwrapping and careful alignment of textures to maintain visual coherence across each architectural element, making it one of the most time-consuming parts of the texturing process:

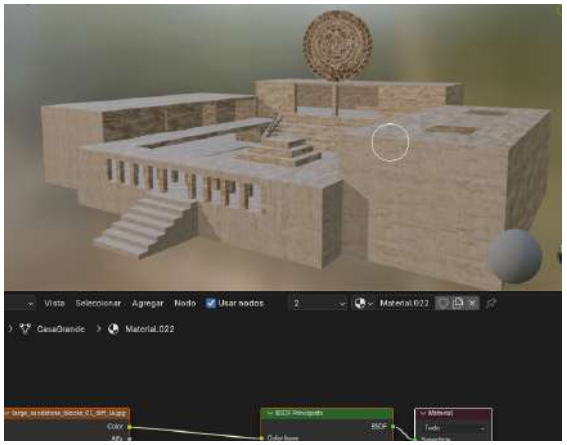

Fig. 3.0 – Large Facade with Applied Textures

Second OpenGL Version:

In the second OpenGL version, the texturing process was carried out in Blender, using custom UV maps to prevent distortion during model export and to fine-tune texture placement and visual impact.

To ensure accurate alignment and consistency in the final textures, specific adjustments were made directly in Blender. Additionally, the GIMP image editor was used to modify and optimize texture files, allowing for better control over color, detail, and transparency where needed.

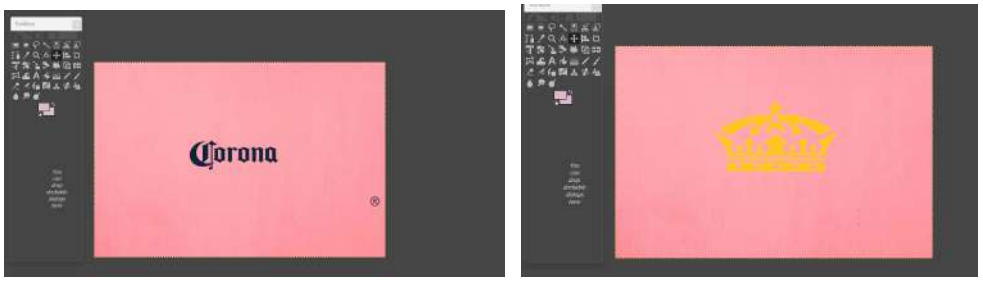

Fig. 3.1 – Image Editing Process for Texturing in GIMP

Textures were carefully selected to reflect real-world materials while preserving the project's colorful and fantastical aesthetic.

- For example, the pyramids and the Aztec calendar use a carved stone texture in sandy and gray tones. A subtle specular map was applied to simulate light reflection on smoother, polished surfaces—enhancing the realism without compromising the stylized look:

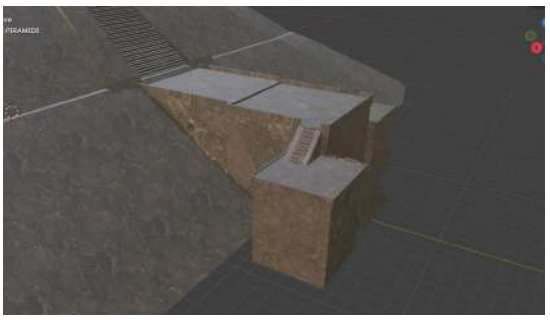

Fig. 3.2 – Opaque Surface Material Example

- Ring Texturing: This section refers specifically to the wrestling ring, which features metallic and vinyl-like textures with a glossy finish. The materials were designed in pink and gold tones, inspired by the aesthetics of a modern, show-style spectacle. This choice reinforces the vibrant and fantastical visual theme of the project.

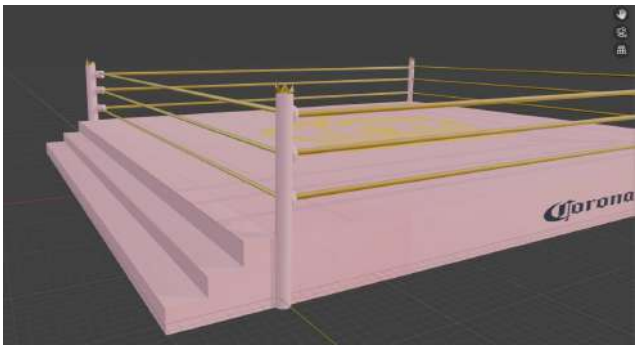

Fig. 3.3 – Ring Texturing

- Ball Game Area Texturing: The ball game zone features pastel colors and shiny materials designed to simulate the appearance of candy or sweets. This stylistic choice contributes to the project's playful and fantastical tone, blending cultural references with a whimsical, imaginative atmosphere.

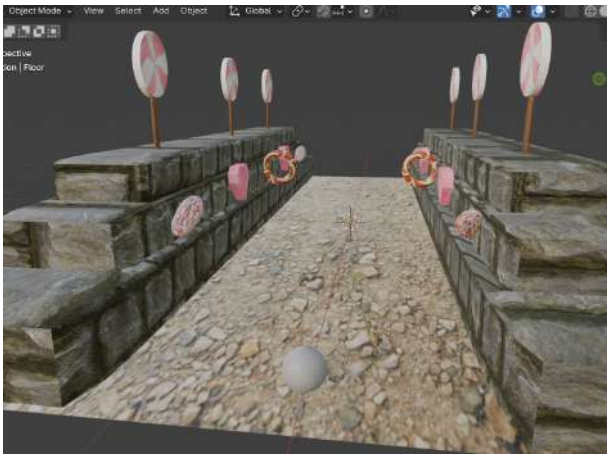

Fig. 3.4 – Ball Game Area Texturing

- The hot air balloon was textured using a color palette of red, black, and white, chosen to give it a bold, eye-catching appearance that aligns with the character it represents and the overall visual style of the project.

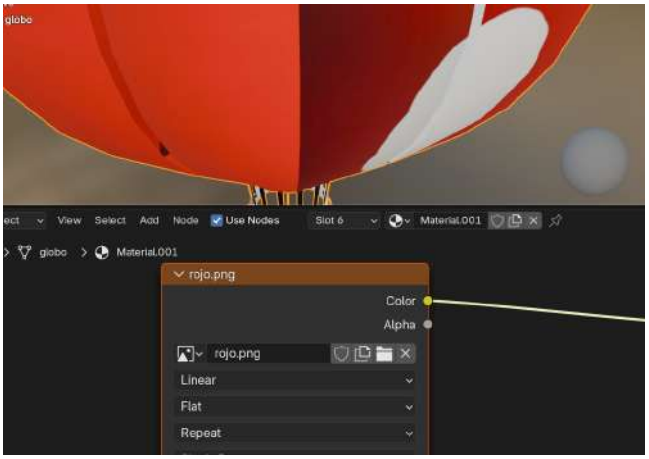

Fig. 3.5 – Texturing of Mickey's Hot Air Balloon

- Vegetation and furniture were textured using wood materials for benches and streetlights, while uniform green tones were applied to tree canopies. These elements used simple shaders to maintain stylistic consistency and optimize rendering performance.

Fig. 3.6 – Scene Atmosphere and Environment

- NPC Characters: Several non-playable character (NPC) models were included and strategically placed throughout the scene. These characters contribute to the overall atmosphere by adding life, narrative context, and visual variety to the environment.

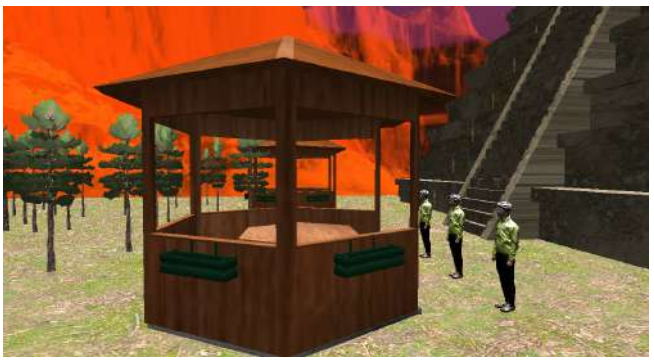

Fig. 3.7 – NPC Models

Unity Version – Texturing:

It's important to note that in Unity, textures could be applied simply by dragging them onto objects. This was a clear advantage, as it only required knowing which texture matched each model.

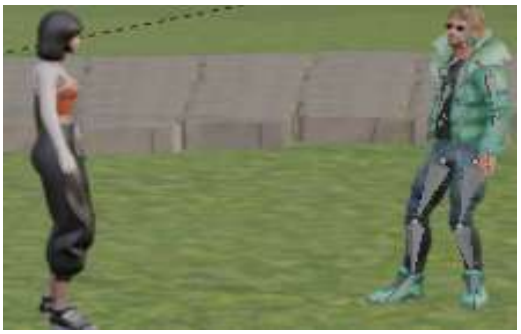

Fig. 3.8.2

Fig. 3.8.3

2.3 Complete Walkthrough

First OpenGL Version:

The project includes the development of a fully interactive 3D virtual environment implemented in OpenGL, allowing users to explore a culturally inspired space based on Pre-Hispanic elements and Mexican Lucha Libre.

The system features:

- A complete 3D scene with architectural structures, themed decorations, and environmental elements that reflect the chosen cultural and fantastical inspirations.

- First-person navigation using keyboard and mouse input, enabling free movement throughout the environment.

- Basic interaction through key inputs, allowing the user to:

    Trigger animations.

    Change lighting conditions.

    Access or highlight points of interest.

This version of the walkthrough focuses on providing an immersive experience while integrating the foundational elements of interaction and visual storytelling.

Fig. 3.8 – First-Person Walkthrough in the 3D Environment

Second OpenGL Version:

In the second version of the OpenGL project, users can explore the entire scene through **three different camera modes**:

- **Third-Person Camera:** Follows the active avatar, maintaining a dynamic view from behind the character.

- **Aerial Camera:** Provides a top-down view of the entire map, useful for spatial orientation and overview.

- **Free or Isometric Camera:** Allows manual exploration of the environment from customizable angles.

These camera modes can be toggled by pressing the V key, offering a versatile and immersive navigation experience.

User movement is controlled with the W, A, S, D keys, while the mouse is used for directional viewing and fine control, creating a smooth and responsive interaction system.

Fig. 3.9 – First-Person Camera Mode

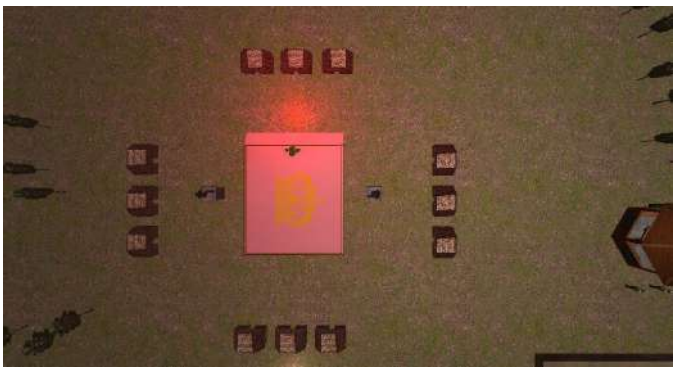

Fig. 4.0 – Aerial Camera Mode

Fig. 4.1 – Free/Isometric Camera Mode

2.4 Lighting and Materials

First OpenGL Version:

The first OpenGL version includes dynamic lighting, which enhances the visual realism and interactivity of the environment. The lighting system incorporates:

- Directional light to simulate sunlight and global illumination.
- Point lights to represent localized light sources such as lamps or torches.
- A spotlight (flashlight-style) mounted to the user's perspective for immersive first-person exploration.

These lighting types interact with the material properties of the models, producing varied effects such as highlights, shadows, and specular reflections, depending on the surface characteristics.

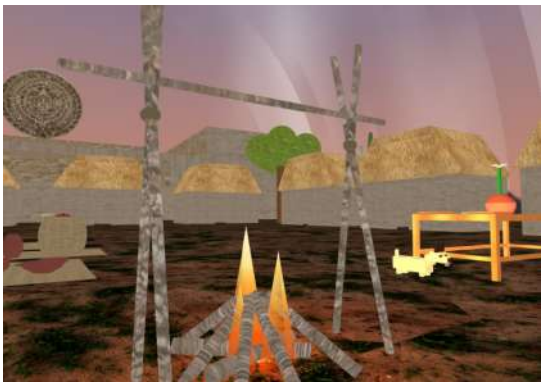

Fig. 4.2 – Lighting and First-Person View

Fire Animation and Lighting

```
// ====== IRE ANIMATION ======
float t = glfwGetTime();
float flick = 0.5f + 0.5f * sin(t * 6.0f);
float fireScale = 0.6f + 0.15f * flick;

glm::mat4 fireModel(1.0f);
fireModel = glm::translate(fireModel, gCampFirePos);
fireModel = glm::scale(fireModel, glm::vec3(fireScale, fireScale * 1.8f, fireScale));

glUniformMatrix4fv(glGetUniformLocation(shader.Program, "model"), 1, GL_FALSE,
glm::value_ptr(fireModel));
FuegoCocina.Draw(shader);

// Parámetros de luz de la fogata
glUniform3f(glGetUniformLocation(shader.Program, "firePos"),   gCampFirePos.x,
gCampFirePos.y, gCampFirePos.z);
glUniform3f(glGetUniformLocation(shader.Program, "fireColor"), 1.0f, 0.6f, 0.2f);
glUniform1f(glGetUniformLocation(shader.Program, "fireFlicker"), flick);
```

The parameter t uses glfwGetTime() to obtain the number of seconds elapsed since the program started. Using a high-frequency sine function, a value called flick is computed to oscillate between 0 and 1. This oscillation dynamically adjusts the scale of the fire model, simulating the natural pulsing of flames as they grow and shrink.

In addition, several uniforms are passed to the shader, including:

- The position of the light source (the campfire)
- Its color
- A variable intensity driven by the flick value

Within the fragment shader, the fireFlicker value can be used to modulate the diffuse and specular brightness of the light. This creates frame-by-frame changes in illumination, simulating a lively and dynamic flame behavior that reacts visually in real time.

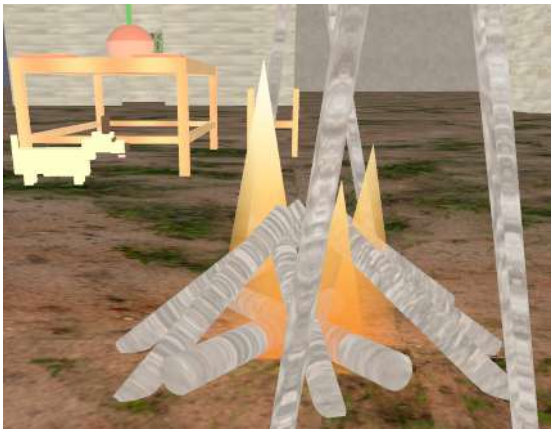

Fig. 4.3 – Campfire with Animated Flicker and Lighting Effects

Day/Night Cycle and Directional Sunlight

```
// ====== UN DIRECTION CALCULATION ======
float timeOfDay = fmod(glfwGetTime() * 0.05f, 2.0f * 3.14159f);
```

```
glm::vec3 sunDir = glm::normalize(glm::vec3(cos(timeOfDay), sin(timeOfDay), 0.2f));

glUniform3fv(glGetUniformLocation(shader.Program, "sunDir"), 1, glm::value_ptr(sunDir));
```

A variable called timeOfDay is used to simulate the passage of time. It increases gradually over runtime and is constrained to a cycle from 0 to 2π, representing a full day/night rotation.

By applying cosine and sine functions to timeOfDay, a direction vector is computed to describe a circular trajectory across the sky. This vector is then normalized and passed to the shader as sunDir.

Within the shader, sunDir is used to compute the directional lighting of the sun—or the moon, depending on the current stage of the cycle. This dynamic lighting affects shading, shadows, and the overall visual atmosphere of the scene.

Second OpenGL Version:

In the second OpenGL version, a full day/night cycle was implemented using a directional light to simulate the sun. Both the intensity and color of the light vary over time, creating smooth transitions from daytime to nighttime.

During nighttime, point lights (PointLight) are activated to simulate ambient illumination from streetlamps and environmental light sources, while spotlights (SpotLight) are used to highlight key areas such as:

- The wrestling ring
- The hot air balloon

The materials of objects in the scene respond to lighting conditions in real time, contributing to enhanced visual realism and atmospheric immersion.

Lamp and Spotlight Management

Multiple light sources are defined and distributed strategically throughout the scene:

- Along the main paths
- Near huts (chozas)
- At fixed coordinates simulating lanterns or torches

Each light source is initialized with its:

- Position
- Color
- Diffuse intensity
- Ambient intensity

A keyboard control system is implemented, allowing users to toggle lamps on and off interactively. When the variable spotLights_On is set to true, all designated lights increase their intensity, simulating nighttime activation.

The lighting system dynamically updates the intensity values of all registered lamps using methods such as:

SetDiffuseIntensity();

SetAmbientIntensity();

These methods ensure that lighting conditions adjust in real time, based on both time of day and user input.

Fig. 4.6 – PointLight Examples Used in the Environment

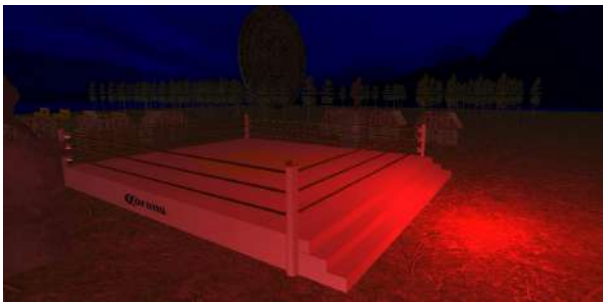

Fig. 4.7 – SpotLight Examples

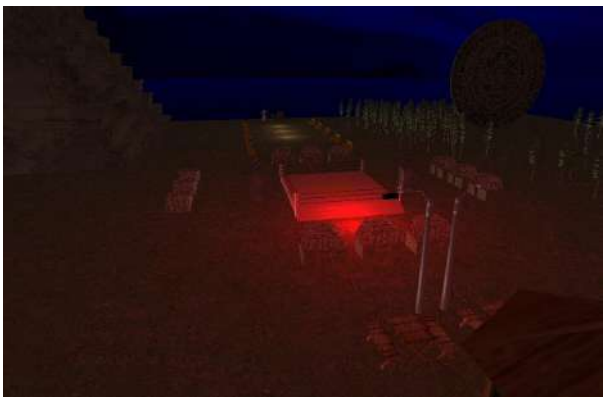

Fig. 4.8 – SpotLight Examples

Unity Version – Lighting

In Unity, we implemented both **sunlight** and **object-based lighting** such as **lamps**. This can be seen below:

Fig. 4.9.1 – Lighting Setup with Sun and Lamps

Fig. 4.9.2

2.5 Animations

First OpenGL Version:

In the first OpenGL version, lightweight animations were added using the deltaTime variable. These include object movements and rotations, allowing dynamic elements to enhance the user's perception of the environment.

Users can freely explore the entire scene in first-person mode, observing lighting, material properties, scale, and spatial depth, simulating an immersive exploration experience.

When sin(timeOfDay) is positive, the sun is considered above the horizon (daytime). When it becomes negative, it signals nighttime, which triggers:

- Star activation
- Increased intensity of the campfire lighting
- Dimming of global ambient light

This logic creates a realistic environmental shift, enhancing immersion and time-based dynamics.

Fig. 4.9 – Day/Night Cycle Transition

Axe Animation

The axe animation was implemented using a hierarchical transformation system applied directly in the OpenGL rendering pipeline. The goal was to simulate a repetitive striking motion, aligning with the scene's Pre-Hispanic theme.

To achieve this effect:

- An oscillating rotation based on the sin() function and the time value from glfwGetTime() was used to simulate a cyclical up-and-down motion.
- The rotation pivot was placed at the end of the axe's handle, so the blade performs the motion while the handle remains stationary.

This design creates a natural and believable animation, as if the axe were being swung by a person or functioning within a ceremonial mechanism.

The model transformation followed the standard sequence used across the project:

1. Translation – to position the axe in the scene
2. Rotation – to generate the swinging motion
3. Scaling – for consistent proportions

Fig. 5.0 *Axe*

Chihuahua Animation

To enhance realism and bring life to the environment, a Chihuahua model was animated with natural and repetitive motions. The main goal was to give the character a dynamic and scene-consistent behavior, using simple yet expressive movement to enrich the 3D world.

The animation was built using a hierarchical structure, allowing different parts of the dog's body to be animated independently yet synchronously. The animation system combines:

- Alternating rotations
- Controlled translations
- Real-time coordination between limbs

At the core of the system is a cyclical leg motion, created using time-dependent trigonometric functions (sin() / cos()), with time retrieved via glfwGetTime(). Each leg rotates around its local axis in a staggered pattern:

- Front legs move forward while
- Back legs move slightly backward — and vice versa

This produces the illusion of a continuous walking cycle.

Thanks to the hierarchical model, where the body serves as the root node and the limbs are child nodes, all motions are preserved in a visually cohesive and fluid manner.

Fig. 5.1 – Animated Chihuahua Walking Cycle

Ball Game

The main movement of the ball in the "Juego de la Pelota" scene is driven by a sine-based or parabolic function, using the elapsed time retrieved via glfwGetTime(). This function generates a continuous vertical displacement that mimics the natural motion of a bouncing object.

The motion is carefully calibrated:

- Amplitude and frequency were fine-tuned to produce a smooth and perceptible bounce

- The animation avoids exaggerated motion, maintaining visual coherence with the rest of the scene

- The ball follows a smooth upward curve, reaches a peak height, and then descends more quickly, repeating this pattern indefinitely.

Horizontal Movement Integration

In addition to the vertical bounce, a controlled horizontal movement was added to simulate the ball's progression across the court. This was achieved through either:

- A light linear variation.
- A subtle sine function.

This combination produces the illusion of forward motion while bouncing, effectively simulating real-world ball behavior in a traditional Mesoamerican ballgame context.

The transformation logic follows the standardized structure used throughout the project:

1. Translation – sets the ball's position based on time

2. Rotation – optionally simulates spin or rotation

3. Scaling – applies the final size of the model

This ensures consistency and maintainability across animated elements in the scene.

Fig. 5.2 – Animated Ballgame Scene with Simulated Bounce Motion

Second OpenGL Version:

In the second version of the OpenGL project, both basic and complex animations were implemented. These animations are either:

- Triggered manually via keyboard input, or
- Automatically executed using time-based or keyframe-driven systems.

Types of Animations Implemented

Basic Animations:

Triggered by the user (e.g., turning lights on/off).

Immediate visual feedback for interaction.

Complex Animations:

Hot air balloon: Uses keyframe animation, allowing continuous movement playback.

Day/Night cycle: Simulates the sun's motion across the sky.

Avatar walking: Includes a walking cycle based on user input.

NPC movement: Adds dynamic ambient life to the environment.

Hot Air Balloon – Keyframe Animation & Manual Control

The hot air balloon features a hybrid animation system:

Manual control:

Controlled via keys I, K, J, L and arrow keys ↑ ↓ for movement along the 3D axes.

Keyframe system:

Press G to save the current position as a keyframe.

Press K to play back the saved animation.

Press C to load keyframes from the file globo_keyframes.txt.

This allows users to record and replay flight paths, combining real-time control and animation playback.

Fig. 5.3 – Hot Air Balloon Keyframe Animation in Action

The environment becomes brighter when the simulated sun is above the horizon and darker when it falls below. This creates a dynamic lighting effect that mimics the

passage of time and affects materials, shadows, and atmospheric lighting across the entire scene.

Fig. 5.6 – Procedural Sky Implementation for Day/Night Cycle

- Jake's Paddle Animation: A unique interactive animation was added to the ballgame area:
When the player presses the J key, the character Jake strikes the ball, initiating a parabolic motion that simulates a realistic bounce or hit.

This interaction adds both interactivity and responsiveness to the scene, and enhances the feeling of a real-time reaction between the player and objects in the environment.

The bounce trajectory was modeled using trigonometric or quadratic functions based on time, producing a smooth upward and downward motion with lateral displacement, reinforcing the idea of a reactive paddle game.

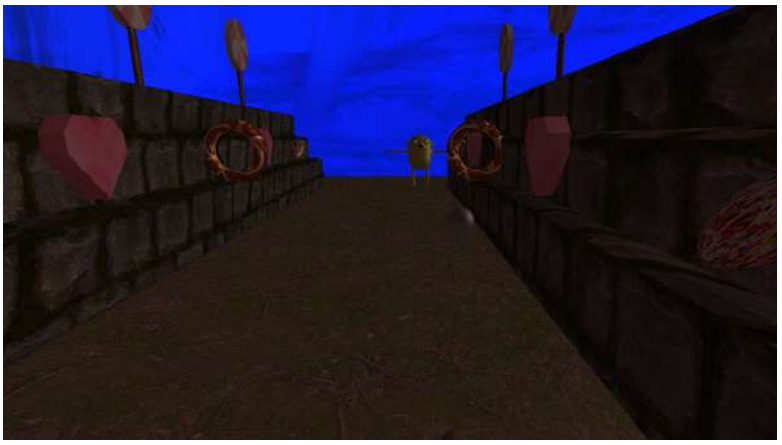

Fig. 5.7 – Ballgame Scene with Paddle Animation (Jake Interaction)

Unity Version: Implemented Animations

In the Unity version, we implemented two animations:

Kratos walking freely in the scene
(Fig. 1.5.3)

A statue rising from the floor in front of the main facade
(Fig. 5.8.1)

Fig. 5.8.1

4. Testing and Results

Scene View

First OpenGL Version

During testing of the first OpenGL version, a series of successful validations were performed. Key observations and results include:

- The user navigation system worked efficiently, allowing smooth movement throughout the 3D environment.
- The lighting system functioned as expected, with dynamic changes based on user interaction and time cycles.
  Animations were executed smoothly, with no noticeable lag or stuttering.

- All 3D models loaded correctly with their corresponding textures and materials, preserving both realism and stylistic consistency.

These results confirm that the core systems (rendering, input, lighting, animation, and model integration) operated stably in this early version of the project.

Fig. 6.1 – Rendered Scene: Avatar Navigation and Lighting Test

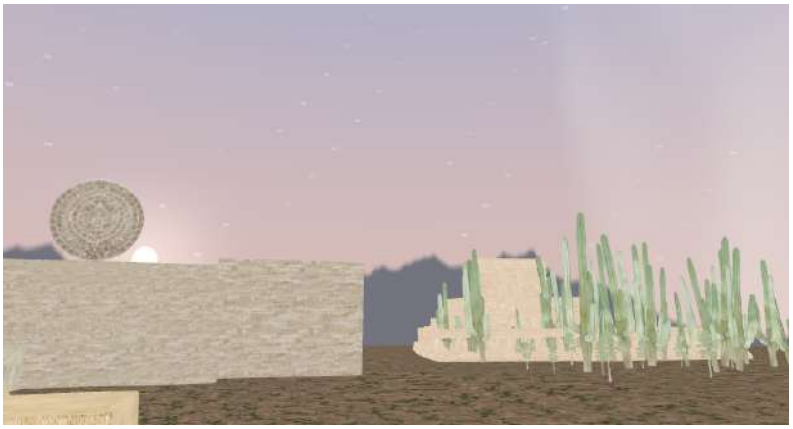

Fig. 6.2 – Visual Validation of Textures and Model Integration

Second OpenGL Version:

During the testing phase of the second OpenGL version, the following components were successfully validated:

Camera systems: All three implemented camera modes (third-person, aerial, and free) functioned correctly and switched seamlessly via user input.

Dynamic lighting: The day/night cycle, spotlight activation, and ambient lighting performed as expected in real time.

Animations: Both time-based and user-triggered animations (e.g., balloon keyframes, NPC movement, avatar walking) ran smoothly without visual artifacts.

Scene interaction: Object manipulation and interactive elements (e.g., key-triggered lights, paddle action) responded promptly and without delay.

Fig. 6.3 – Camera System Functionality Test: Third-Person and Aerial Views

Fig. 6.4 – Lighting & Animation Performance Snapshot (Nighttime Scene)

Fig. 6.5 – Model and Texture Loading Validation in Full Scene

Unity Version: Visual Improvements & Scene Handling

During the development of the Unity version, we observed a notable improvement in visual quality compared to the original OpenGL implementation.

Fig. 6.7 – Unity Scene Render: Enhanced Lighting and Material Quality

Fig. 6.8

Among the three versions developed, the Unity implementation was the most faithful and fluid in rendering our models.

4. Conclusions

This project proved to be a comprehensive and challenging experience, providing a practical opportunity to apply the concepts learned throughout the semester in Computer Graphics and Human-Computer Interaction.

During development, numerous elements were successfully integrated, including:

- Textured 3D models
- Dynamic lighting
- Keyframe-based animations
- Keyboard-controlled interaction
- Multiple camera systems
- Day/night cycle simulation

These components collectively contributed to an environment with a high level of interaction and realism.

Among the most notable features were:

- The keyframe-animated hot air balloon
- The ballgame with basic physics
- The implementation of three distinct camera modes: third-person, aerial, and free-view

All of these fulfilled the project requirements outlined in the course guidelines.

Despite the extensive codebase and time limitations, teamwork allowed the successful integration of three distinct thematic universes—Mario Bros, Mickey Mouse, and Adventure Time—each featuring custom models and unique animations within a single cohesive scene.

This project served as an excellent opportunity to reinforce key skills in modeling, texturing, and animation using OpenGL, and demonstrated the importance of:

- Organization
- Collaborative work
- Planning and task distribution

These aspects were essential to achieving a functional and visually appealing 3D environment. With additional time, further optimization of animations and performance improvements could be made.

The "Pre-Hispanic Market" Scene

The development of the "Pre-Hispanic Market" scene in OpenGL successfully combined several core topics from the course:

- Procedural geometry
- Model loading using Assimp
- Advanced texturing
- Dynamic lighting
- Interactive 3D camera controls

The scene integrated both:

Artist-created models in Blender (e.g., pyramids, huts, corn, horse, ballgame court, market props).

Code-generated models (e.g., table, chair, vase, flower, campfire, jade mask, voxel dog, wheelbarrow, axe, animated ball)

This demonstrates the possibility of building complex environments by blending artistic assets with fully parametric, programmatically generated content.

The procedural sky implementation, with its day/night cycle, sun direction calculations, and an embedded shader for sky and grass rendering (featuring anti-tiling techniques), resulted in a rich, dynamic visual atmosphere—all without relying on pre-baked skyboxes.

Using noise functions, FBM (fractal Brownian motion), and custom shaders for:

- Clouds
- Stars
- Sun
- Moon

The campfire lighting system, triggered via the F key, integrated:

A point light for model shading (external shader)
An emissive and local light contribution within the procedural shader

This setup strengthened understanding of:

How to combine multiple light sources (directional and point).

How to attenuate light based on distance.

How to reuse lighting data (position and color) across different shaders.

These techniques maintained visual consistency and contributed to a cohesive lighting experience throughout the scene.

Individual Conclusions

Cecilia: Personally, I really enjoyed working on this project because I was able to integrate what I learned both in the laboratory and in the theory class. Learning Blender was crucial for the development of this project, as well as fully understanding the functionalities of OpenGL, which we later incorporated when adapting our entire work to Unity.

The most time-consuming part was modeling, texturing, and animating—both in code and with tools such as OpenGL. However, I found it amazing to work on a project where we combined curiosity and creativity, because modeling our scenes and giving them color and lighting truly felt like playing Minecraft.

Of course, there are still details that could be improved, but I was satisfied with the results of the project, since we integrated three interesting areas: history, architecture, and computer graphics and I would also add video games, because when the user interacts with objects and animations, it feels like a game.

Valentina: The random instances of trees, cacti, and cornfields, with exclusions around important areas (the table, the camp, the pyramids, Tula, the central zone), allowed us to practice controlled scattering techniques over large terrains, maintaining visual composition and avoiding overlapping.

The inclusion of the animated voxel dog, the bouncing ball, and the controllable wheelbarrow added playful and interactive elements that make the scene not just a "static render," but a small game-like environment where the user can explore with the camera and observe localized animations.

Overall, the project achieved its goal of integrating in a single environment: complex models, procedural geometry, advanced texturing techniques (including anti-tiling and anisotropy), time-dependent dynamic lighting, and a first-person interaction system. In addition, it served as a complete pipeline experience: from modeling and exporting, to loading assets, organizing the scene, designing shaders, and finally documenting and analyzing the software architecture.

Pablo: This project allowed us to create a lively, interactive 3D world, blending Mexican traditions such as pyramids and street markets with characters from some of our favorite cartoons, such as Mario Bros and Adventure Time. The most interesting and challenging aspect of this project was that we managed to make the setting have, for example, a day and night cycle, with lamps that turn on by themselves when the environment changes and an interactive ball game. In addition, elements such as the hot air balloon controlled by keyframes make it feel like a small step towards what a video game is.

Shareny: Developing this project was a comprehensive challenge, as we had to bring together many different techniques in one place. We combined objects that we modeled ourselves in Blender with other imported elements that brought the scene to life and complemented it. We also focused specifically on visual details, applying textures that look like real stone or wood and animations, such as the ball bouncing in a natural way. In the end, we managed to make everything work together seamlessly, allowing the user to interact with and navigate the environment in different ways.

Santiago: Thanks to our joint efforts, we were able to build a visually appealing stage that combines themes as diverse as wrestling and the pre-Hispanic world in a single setting. We managed to include everything from a ring with shiny metallic textures to an interactive mini-game like ball games, making sure that the lighting and materials looked good at all times. Although it was a challenge to combine our different areas of expertise and what each of us did best, in the end, we managed to build a stage that included everything we had seen in class, allowing the user to explore an interactive 3D world.

5. Teamwork

Diagrama de Gantt - Proyecto 3D OpenGL (Incluye Modelado Blender)				Semana 1						
				1	2	3	4	5	6	7
Actividad	Responsable	Inicio	Fin							
Instalación VS, OpenGL, GLEW, GLFW, Assimp	Amy	S1, d1	S1, d7							
Organización de proyecto	Ceci	S1, d3	S2, d2							
Modelado en Blender	Equipo	S2, d1	S3, d3							
Exportación y corrección UV en Blender	Amy	S3, d4	S3, d7							
Render + ventana + contexto	Ceci	S2, d3	S2, d7							
Integración modelos Blender → OpenGL (Assimp)	Amy	S4, d1	S4, d7							
Shader Phong + luces	Ceci	S5, d1	S6, d3							
Texturas y blending	Amy	S4, d3	S4, d7							
Cámara FPS + animación	Ceci	S7, d1	S7, d7							
Estrellas + efectos + theme MX	Equipo	S7, d3	S8, d2							
Pruebas + optimización	Equipo	S8, d1	S8, d4							
Documentación + exposición	Equipo	S8, d3	S8, d7							

Table 1.1 – Gantt Chart: Initial Phase of Development

Covers early stages such as project planning, task delegation, and base model creation.

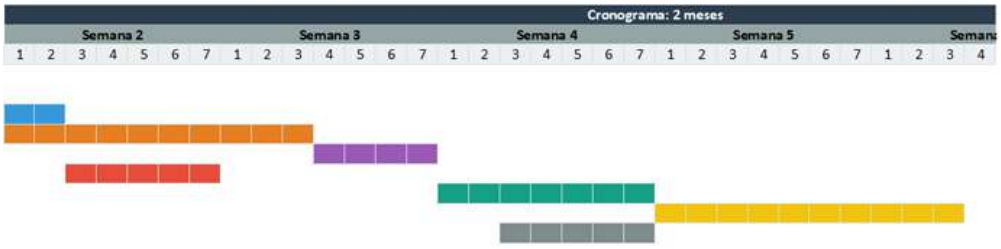

Table 1.2 – Gantt Chart: Mid-Development

Includes geometry refinement, animation systems, texturing pipeline, and input handling.

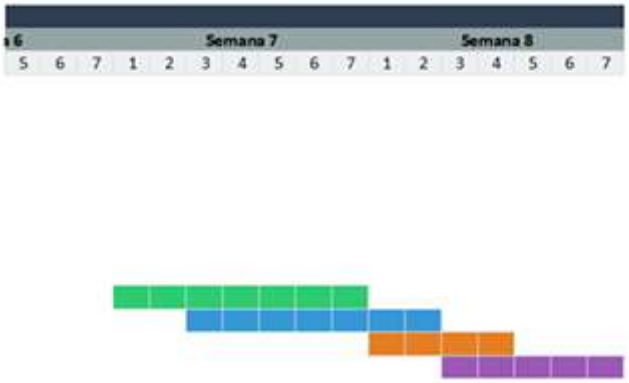

Table 1.3 – Gantt Chart: Final Integration and Testing

For the general project:

Valentina contributed an outstanding combination of technical and artistic skills. During the OpenGL phase, she modeled structures such as Tula, various types of vegetation, and multiple objects created directly through code. She also developed one of the most advanced visual systems of the project: the animated procedural sky. This system included time-based color transitions, atmospheric changes, simulation of day/night cycles, and a coherent integration with directional lighting. She also implemented hierarchical animations such as the axe and the voxel dog, which required understanding node structures, time-dependent rotations, and local transformations. This technical experience was crucial in Unity, where she adapted her Blender and OpenGL models to the engine by correcting UVs, maintaining material integrity, smoothing normals, and optimizing geometry to avoid performance issues. Valeria also played an active role in the lighting setup in Unity, adjusting shadows, HDR parameters, and the general chromatic coherence of the map, in addition to distributing vegetation and decorative objects to create natural transitions between thematic zones.

Cecilia was the team member who contributed the most artistic and cultural content to the entire project. In OpenGL, she modeled and textured highly complex elements such as the pre-Hispanic market, the huts, the large facade inspired by Tenochtitlan, the woven mats, and an extensive variety of handcrafted objects. The detailed UV mapping process she performed on the facade and the market required precision and hours of work, especially when adjusting textures to prevent distortions. She also developed the campfire animation with dynamic lighting, including the flicker effect that influenced nearby objects, integrating shaders and time-dependent calculations. When migrating to Unity, Cecilia successfully adapted her models by rebuilding materials, adjusting scales, and distributing her elements across complete areas of the map, creating thematic zones that enriched the visual narrative of the scene. She implemented point lighting in key areas such as the market, highlighting cultural elements, and collaborated in organizing colliders to ensure smooth navigation for the user. Her work ensured that in both OpenGL and Unity, the environment had identity, depth, and an unmistakable aesthetic cohesion.

Shareny played a significant role in both OpenGL and Unity. During the first stage, she modeled elements such as Peach’s ring and the Aztec calendar, and actively participated in character separation and the initial implementation of complex models within the scene. She also contributed to the design of the original layout, the planning of spatial distribution, and the programming of the hot-air balloon movement, which required handling translations, scaling, and keyboard control. This experience provided her with a clear understanding of how models behave in a three-dimensional environment, knowledge that was essential in Unity. There, she was responsible for integrating and adjusting many of the main models of the scene, optimizing scales, correcting rotations, and adapting Blender materials so they preserved their visual quality. She also helped organize hierarchies within the engine, reviewed collisions, and contributed to the final debugging process, ensuring each area of the map was visually balanced and functionally correct. Her ability to detect visual inconsistencies and her previous work with transformations and textures in OpenGL greatly facilitated the final Unity stage.

Santiago had one of the most technically important roles in the project and contributed throughout both stages. In OpenGL, he implemented the three main cameras (third-person, aerial, and isometric), developed the dynamic day/night skybox, configured point lights activated by the temporal cycle, and programmed the full ball-game animation with simplified physics, collisions, and horizontal movement. This knowledge was fundamental for the Unity version, where he configured the project's main camera system, adjusting distance, follow behavior, and transitions between views. He also worked on the global lighting through HDR skybox settings, directional light calibration, shadows, and environmental parameters, applying his prior understanding of dynamic lighting. Additionally, he implemented avatar movement, interaction scripts, global map colliders, and reviewed materials to ensure proper visualization under different lighting conditions. His technical mastery allowed the Unity scene to feel smooth, controllable, and visually balanced.

Pablo contributed some of the most advanced systems of the project, standing out in both OpenGL and Unity. In the first stage, he modeled elements such as the market, benches, and interactive objects. He also developed complex animations, such as the arch with scrolling text and the phased door-opening system, using interpolations, hierarchies, and time-dependent calculations. He carried out large-scale integration of trees, lamps, benches, and statues, organizing the scene with great attention to detail. In Unity, this knowledge translated into scripting interaction logic, object activation, dynamic prefab configuration, and animation adjustments within the engine. He also optimized both his own and others' models, adjusted collisions, organized complete zones of the map, and participated in the final debugging of the scene, ensuring performance and stability. His previous experience with hierarchical animations and internal logic in OpenGL enabled him to recreate complex behaviors in Unity, resulting in a robust and functional interactive environment.

6. References

Angel, E., & Shreiner, D. (2016). *Interactive Computer Graphics: A Top-Down Approach with OpenGL* (7th ed.). Addison-Wesley.

- OpenGL Architecture Review Board. (2013). *OpenGL Programming Guide* (8th ed.). Addison-Wesley.

- Sketchfab. (2019, April 8). Sketchfab. Sketchfab. https://sketchfab.com/3d-models/finn-the-human-adventure-time-68bb055e68844da4878dc3ae7ec0b28a
- Sketchfab. (2020, January 10). Sketchfab. Sketchfab. https://sketchfab.com/3d-models/jake-the-dog-f8a17c174f244e11af1fb668895ddcae
- Sketchfab. (2017, July 2). Sketchfab. Sketchfab. https://sketchfab.com/3d-models/marceline-the-vampire-queen-78b2b8e2ae1849fa9bfbd234c6ce51ca
- Sketchfab. (2020, August 19). Sketchfab. Sketchfab. https://sketchfab.com/3d-models/makoto-yukiminato-arisato-persona-3-85308017127c43e991e646fa5c32b71d
- Sketchfab. (2024, May 22). Sketchfab. Sketchfab. https://sketchfab.com/3d-models/persona-formal-53209dc94ac949118b58f66b0fcfba63
- Sketchfab. (2023, January 4). Sketchfab. Sketchfab. https://sketchfab.com/3d-models/braulio-fe3405e250d14381adaba1a6e590a319
- Sketchfab. (2021, February 22). Sketchfab. Sketchfab. https://sketchfab.com/3d-models/mickey-mouse-fbx-8cfc790775f14c8fbbf386fef1352065
- Sketchfab. (2024, October 30). Sketchfab. Sketchfab. https://sketchfab.com/3d-models/disney-color-and-play-minnie-mouse-a31fe6b8b47b48a7a30d8444ee324ab3

- Sketchfab. (2018, April 9). Sketchfab. Sketchfab. https://sketchfab.com/3d-models/goofy-b3c7e30a83fd431a985a53e19d35052f
- Sketchfab. (2017, November 19). Sketchfab. Sketchfab. https://sketchfab.com/3d-models/pluto-df0a99f11cf44182a037c0fc78322044
- Sketchfab. (2020, December 7). Sketchfab. Sketchfab. https://sketchfab.com/3d-models/piramide-e9afe46f46c5424182090d075465fbb5
- Sketchfab. (2024, November 12). Sketchfab. Sketchfab. https://sketchfab.com/3d-models/diferentes-arboles-d61df886e18a41f2acf8dd116a6d29ac
- Sketchfab. (2024, November 30). Sketchfab. Sketchfab. https://sketchfab.com/3d-models/peach-adult-08590a05863e492fb28de2472a296f00

- Sketchfab. (2023, April 18). Sketchfab. Sketchfab. https://sketchfab.com/3d-models/toad-1687c9bd6b5944b0bbfe548f5f9d07a7
- Sketchfab. (2024, November 19). Sketchfab. Sketchfab. https://sketchfab.com/3d-models/mario-and-luigi-new-super-mario-bros-8979f2b3a4e7464b912a2567e5790a3f
- Sketchfab. (2021, May 15). Sketchfab. Sketchfab. https://sketchfab.com/3d-models/yoshi-58eb9ae7cbd3436eb7b5fde84faca10c
- https://www.turbosquid.com/es/3d-models/pergola-gazebo-arbour-3d-1357036
- https://www.turbosquid.com/es/3d-models/garden-furniture-3d-1841667
- Sketchfab. (2023, September 3). Sketchfab. Sketchfab. https://sketchfab.com/3d-models/lampara-20fa6c63508141a98c281aa94640e134

GitHub Repositories

First OpenGL Implementation:
https://github.com/CeciliaXimenaSolisCisneros/ProyectoCompuGrafica

Second OpenGL Implementation:

https://github.com/PablinCu02/Proyecto_CGEIHC

Unity Version:

https://github.com/SantiagoSanchezMedina/ProyectoFinal-

Project Video Demonstration

Video Showcase: https://youtu.be/qDD2HFZ3GEc