UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

DOCUMENTACIÓN PROYECTO FINAL

NOMBRE COMPLETO:

- Gonzalez Cuellar Pablo Arturo 319241013
- Lechuga Castillo Shareny Ixchel 319004252
- Sánchez Medina José Santiago 319246881
- Veraza García Amy Valentina 320490572
- Solís Cisneros Cecilia Ximena 317042333

GRUPO DE LABORATORIO: 02, 03, 04 & 13

GRUPO DE TEORÍA: 04

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 04/12/2025

Contenido

1. Introducción

Este documento presenta el desarrollo del proyecto final de la materia de Computación Gráfica e Interacción Humano-Computadora. El proyecto consiste en un escenario inspirado en las temáticas “Prehispánico” y “Lucha Libre Mexicana”, combinadas con universos fantásticos elegidos previamente por cada uno de los integrantes de este equipo.
El escenario incluye elementos culturales como pirámides, el calendario azteca y un juego de pelota reinterpretado con toques coloridos, así como un ring central. Así pues, todo el escenario estará tematizado con los personajes de Peach, Mickey Mouse y Finn el Humano.
El objetivo del proyecto es integrar los conocimientos adquiridos durante el curso para construir un entorno 3D interactivo con iluminación dinámica, materiales realistas,
animaciones jerárquicas y un sistema de cámaras funcionales, buscando un equilibrio entre la técnica y la creatividad visual de la temática propuesta.

2. Desarrollo

2.1 Geometrías

El escenario fue construido completamente en 3D, combinando modelos creados en Blender con geometrías generadas e importadas a OpenGL.

Entre los elementos modelados se incluyen:

- Personajes elegidos por los integrantes del equipo
  Se utilizaron modelos pre-cargados desde la web, a estos se le tuvieron que hacer ajustes como seccionar el modelo por extremidad, cambiar el eje de rotación por cada extremidad, ajustar la escala, etc.

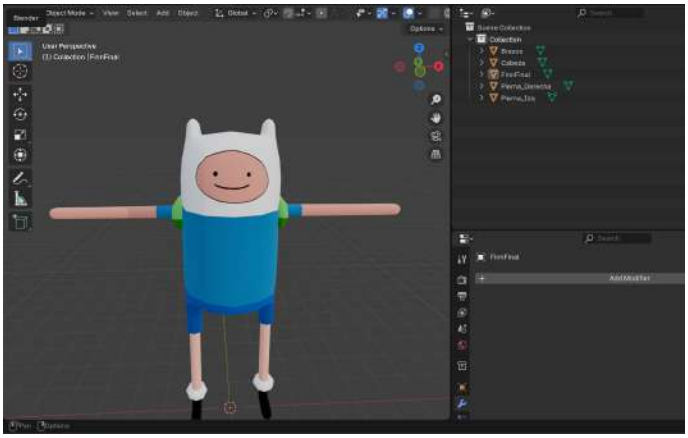

Fig . 1.0 *Modelo Finn*

Fig. 1.1 *Modelo Peach*

Fig. 1.2.1 *Modelo Mickey*

- Pirámides prehispánicas, inspiradas en Chichén Itzá y la Pirámide del Sol, creadas a partir de prismas escalonados.

Fig. 1.3 *Modelo Chichén Itzá*

- Calendario azteca, modelado mediante cilindros y subdivisiones circulares con grabados tallados en textura.

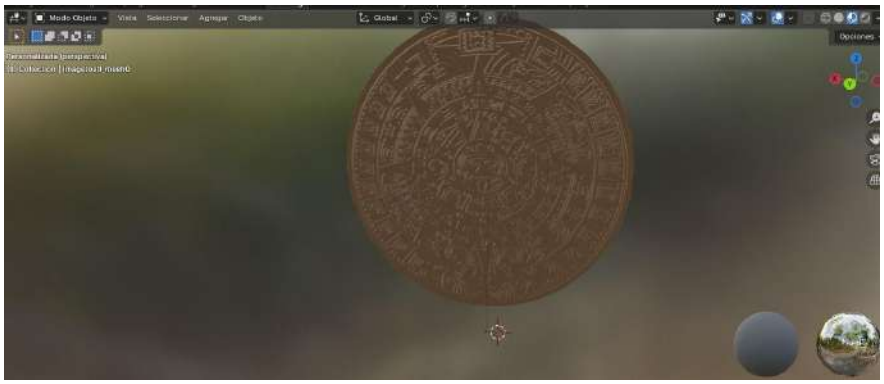

Fig. 1.4 *Modelo Calendario Azteca*

- Ring de lucha libre de Peach, modelado como un cubo central con barandales y postes con jerarquía, ubicado en la plaza principal.

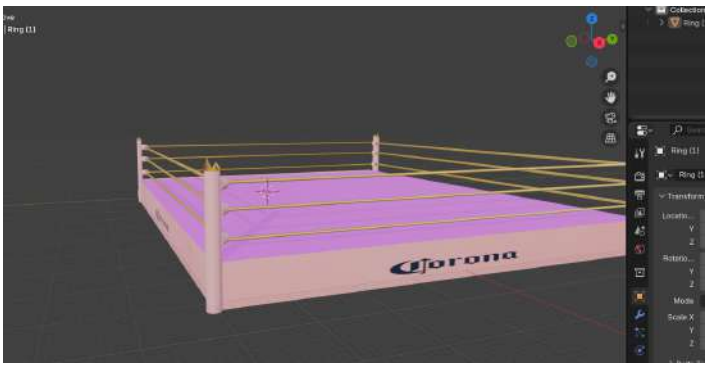

Fig. 1.5 *Modelo Ring*

- Juego de pelota creado a partir de cubos escalonados, cilindros y toroides tematizados

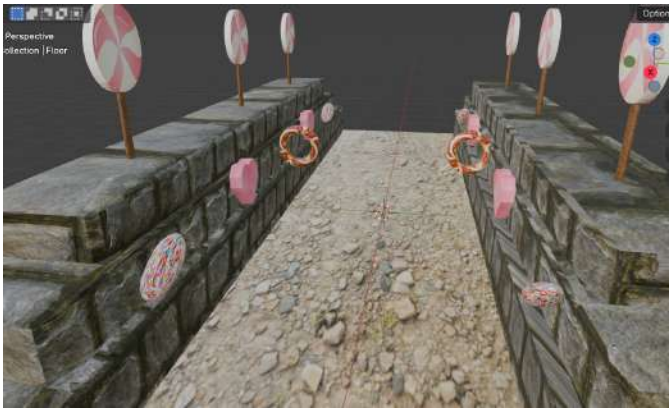

Fig. 1.6 *Modelo Juego de la Pelota*

- Globo aerostático con orejas, compuesto por una esfera principal y semicírculos para las orejas, con una canasta rectangular suspendida

Fig. 1.7 *Modelo Globo Aerostático*

- Bancas y luminarias, formadas por prismas texturizados, ubicadas a lo largo de las rutas de recorrido.

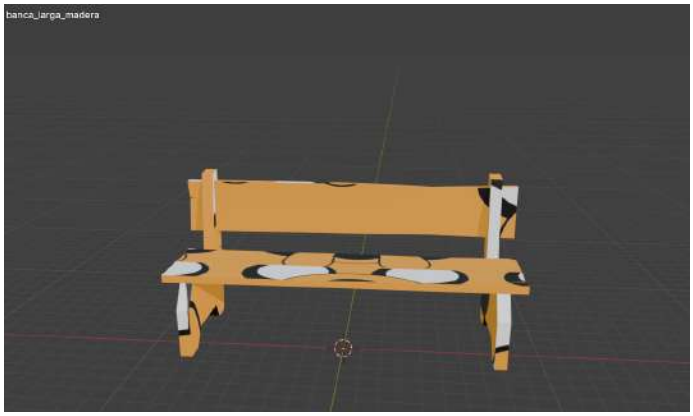

Fig. 1.8 *Modelo Banca*

Todos los modelos fueron escalados y posicionados manualmente para mantener coherencia visual y escala realista respecto al avatar y al entorno.

- Fachada grande, inspirada en ciertos monumentos reconstruidos de la antigua Tenochtitlán:

Fig. 1.9 *Tenochtitlán*

Fig. 2.0 *Fachada de casa modelada*

- Tula:

Fig. 2.1 *Modelo de Tula*

- Árboles:

Fig. 2.2.1 *Árboles modelados*

- Tianguis: inspirándonos en ciertos videojuegos, pensamos en que no podría faltar un mercado aunque fuera pequeño con canastas, chiles, petates, aguacates, jarrones, tendedero, piel de jaguar, etc.

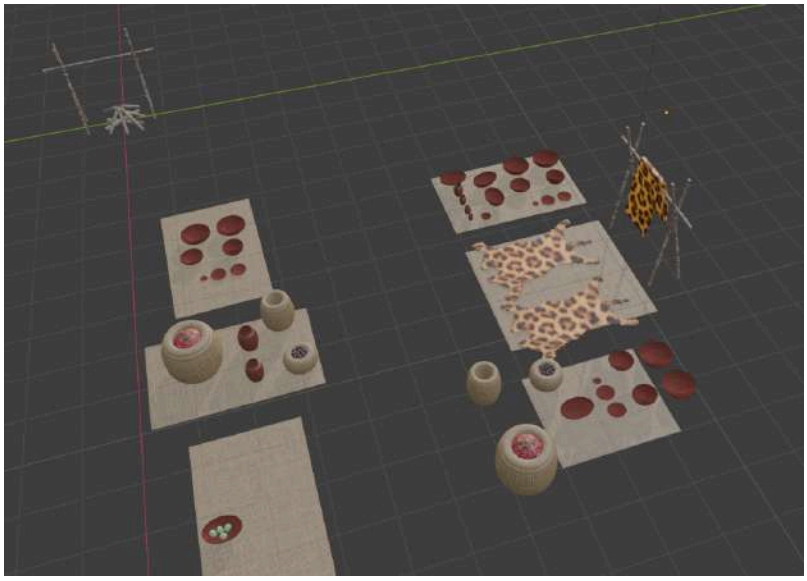

Fig. 2.3 *Tianguis modelado*

- Chozas:

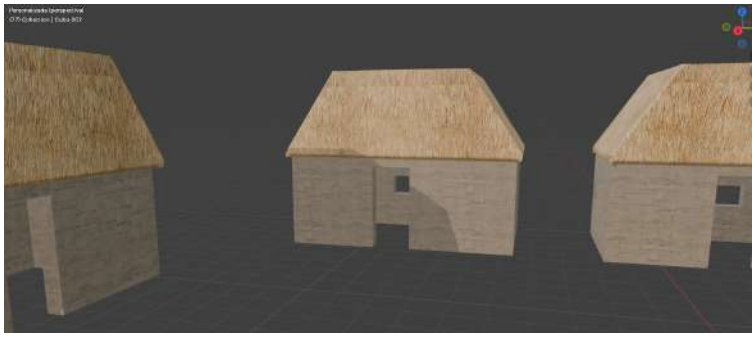

Fig. 2.4 *Modelo de chozas*

Inspiración:

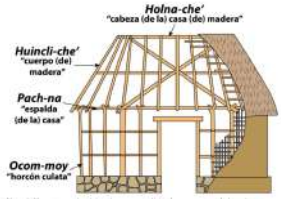

Fig. 2.5 *Inspiración Chozas*

- Objetos en OpenGL con código:

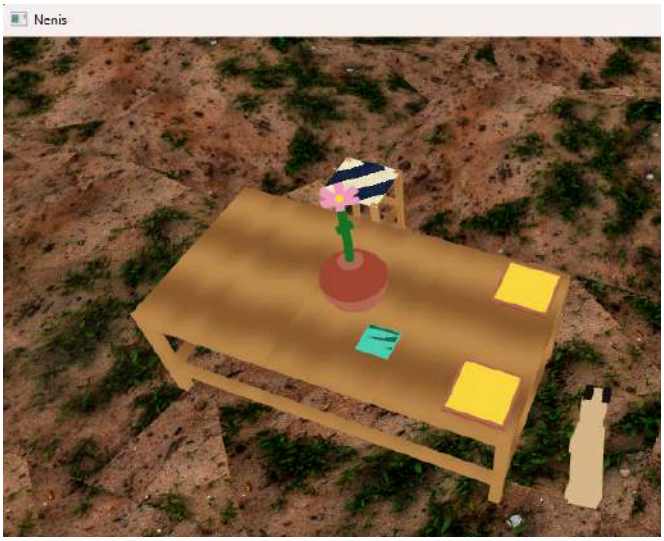

Fig. 2.5.1 Modelos: mesa, máscara, platos, chihuahua, banco, flor y florero.

Fig. 2.5.2 Modelos: hacha y carretilla

Para la versión de Unity:

Fig. 1.5.3 *Modelo Kratos*

Fig. 2.5.4 *Árboles descargados*

- Estatuas

Fig 2.6.1 Estatua de Ehecatl

Fig 2.6.2 Estatua de Ehecatl

Fig 2.6.3 Estatua de Tlaloc

Fig 2.6.4 Estatua de Huitzi

Fig 2.6.5 Estatua de Tezcatlipoca

Fig 2.6.6 Estatua de Quetzalcoatl

Fig 2.6.6 Estatua de Mictlantecutli

2.2 Texturizado

Para la primera versión de OpenGL:

Se aplicaron texturas mediante mapeo UV para simular piedra, arena, metal y otros materiales. Se configuraron propiedades materiales (ambient, diffuse, specular, shininess) para lograr superficies con diferente respuesta a la luz, mejorando el realismo del entorno.

Posee texturas detalladas aplicadas a superficies naturales y arquitectónicas (piedra, arena, madera, metal).

Fig. 2.7 *Choza texturizada a través de mapeo UV*

Fig. 2.8 *Tula texturizada*

Fig. 2.9 *Árboles texturizados con diferentes materiales ad-hoc a la escena*

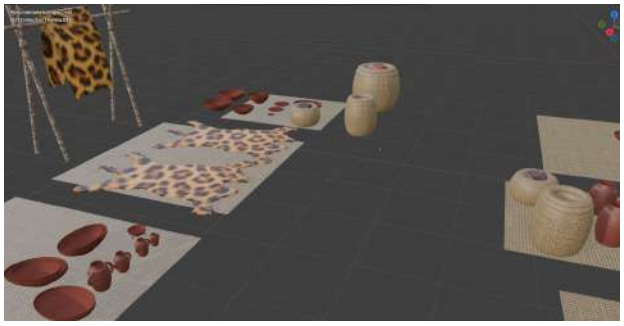

Fig. 3.0 *Tianguis con diferentes texturas*

Lo que más costó trabajo de texturizar cara por cara a través del UV Mapping fue la fachada grande:

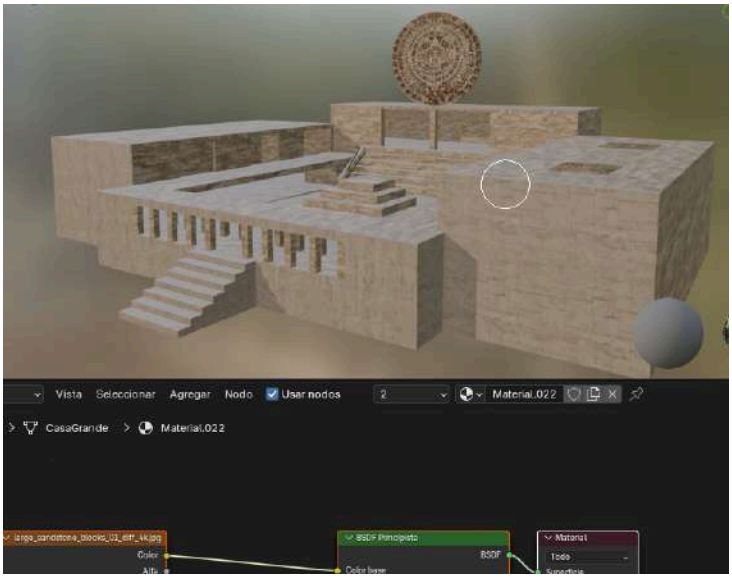

Fig. 3.1 *Fachada Grande texturizada*

Para la segunda versión de OpenGL:

El proceso de texturizado se realizó en Blender, aplicando mapas UV personalizados para evitar distorsiones al exportar los modelos y ajustar ciertas texturas para su correcta ubicación y efecto en el texturizado final. También se usó la herramienta de GIMP para edición de imágenes.

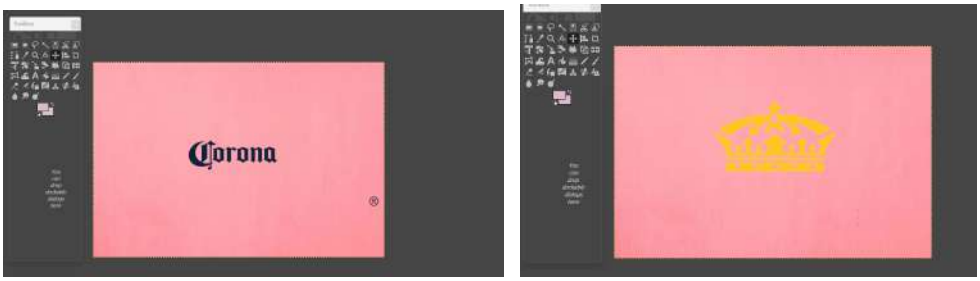

Fig. 3.2 *Proceso de Edición de Imágenes para Texturizado en GIMP*

Se cuidó que las texturas reflejaran materiales reales, pero sin perder el estilo colorido y fantástico del proyecto.

- Pirámides y calendario azteca: textura de piedra tallada con tonos arenosos y grises. Se aplicó un mapa especular suave para simular reflexión en superficies pulidas.

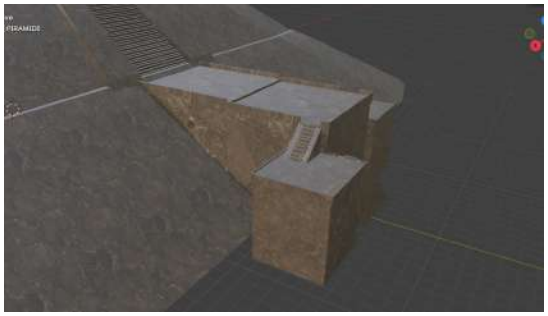

Fig. 3.3 *Superficie Opaca*

- Ring: texturas metálicas y vinílicas con brillo, en tonos rosas y dorados, inspiradas en un espectáculo moderno.

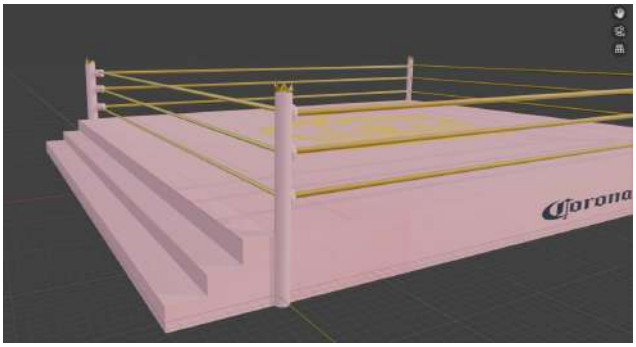

Fig. 3.4 *Texturizado del Ring*

- Zona del juego de la pelota: colores pastel con materiales brillantes para simular dulces o caramelos

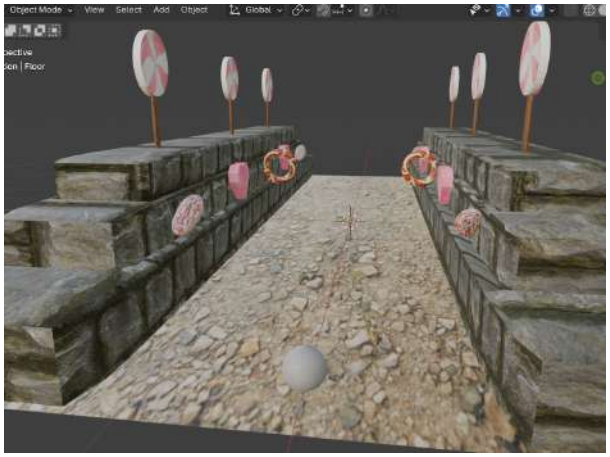

Fig. 3.5 *Texturizado Juego de la Pelota*

- Globo aerostático: tonos rojos, negros y blancos

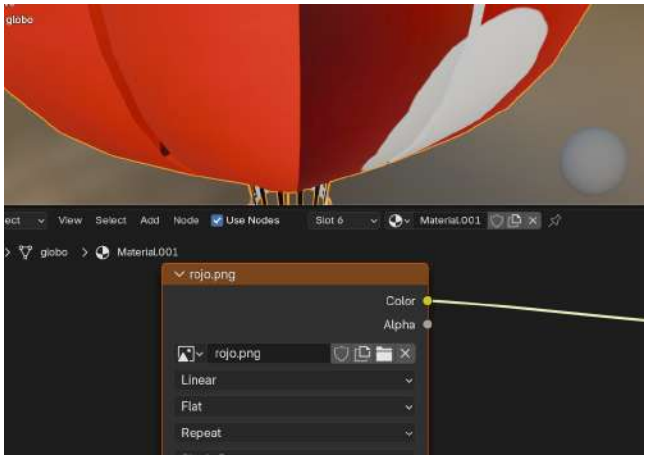

Fig. 3.6 *Texturizado del Globo de Mickey*

- Vegetación y mobiliario: madera para bancas y luminarias, tonos verdes uniformes para copas de árboles, usando shaders simples.

Fig. 3.7 *Ambientación*

- Personajes NPC’s: Se incluyeron modelos para NPC’s distribuidos a lo largo de la escena.

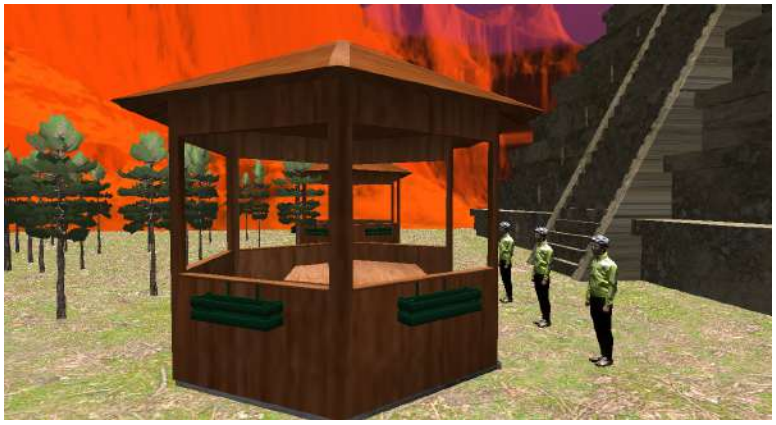

Fig. 3.8.1 *Modelos NPC's*

Para la versión de Unity:

Es importante recalcar que se arrastraban las texturas a los objetos en Unity, un punto a favor de éste, era cuestión de saber qué texturas corresponden exactamente, incluso con conocimiento adicional solo basto de algunos ajustes en cuestión de materiales desde Unity y exportando en .fbx desde blender.

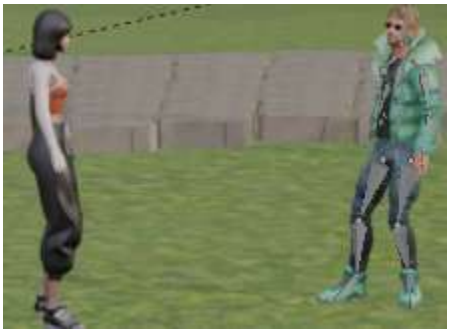

Fig. 3.8.2 *Modelos NPC's*

Fig. 3.8.3

2.3 Recorrido total

Para la primera versión de OpenGL:

El proyecto contempla el desarrollo de un entorno virtual 3D completo e interactivo, implementado en OpenGL, que permita al usuario recorrer y observar un espacio cultural inspirado en elementos prehispánicos y lucha libre mexicana. El sistema incluye:

- Escena 3D completa con estructuras, decoraciones y ambientación temática.

- Navegación en primera persona, utilizando teclado y mouse, con desplazamiento libre por el entorno.

- Interacción básica mediante teclas para activar animaciones, cambiar iluminación o consultar puntos de interés.

Fig. 3.9

Para la segunda versión de OpenGL:

El usuario puede desplazarse por todo el escenario con tres tipos de cámara:

- Cámara de tercera persona, que sigue al avatar activo.
- Cámara aérea, que muestra una vista superior del mapa.

- Cámara libre o isométrica, que permite explorar el entorno manualmente.

Estas cámaras pueden alternarse mediante la tecla **V**, dando una experiencia inmersiva y múltiple..

El movimiento del usuario se gestiona usando las teclas W, A, S, D y el desplazamiento del mouse.

Fig. 4.0 *Modo 1ra Persona*

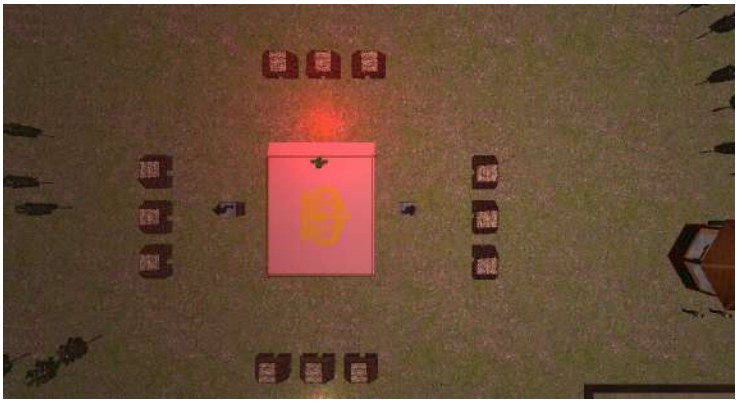

Fig. 4.1 *Modo Vista Aérea*

Fig. 4.2 *Modo Libre*

2.4 Iluminación y materiales

Para la primera versión de OpenGL:

Iluminación dinámica, incluyendo luz direccional, luces puntuales y linterna tipo “spotlight”.

Fig. 4.3 Iluminación y vista en primera persona

Fogata animada con flicker e iluminación

```
// ====== ANIMACIÓN DEL FUEGO ======
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);
```

El parámetro t usa glfwGetTime() para obtener el tiempo en segundos desde que inició el programa. Con un seno de alta frecuencia se calcula flick, que oscila entre 0 y 1. Esa oscilación modifica la escala del modelo para simular que las llamas crecen y se encogen. Además, se pasan al shader uniforms que representan la posición, el color y la intensidad variable de la luz de la fogata.

En el fragment shader, fireFlicker se puede usar para aumentar o disminuir el brillo difuso y especular de la luz, logrando una iluminación dinámica que cambia fotograma a fotograma.

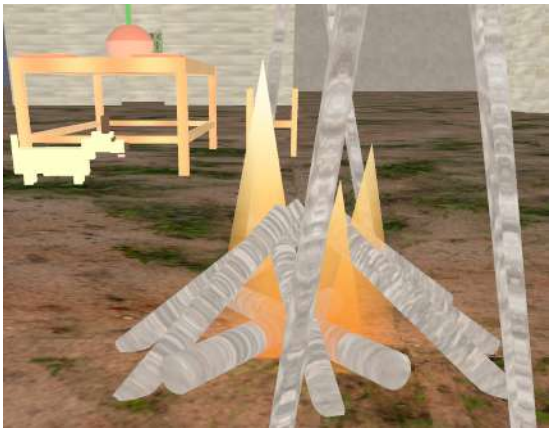

Fig. 4.4 Fogata

Ciclo día/noche y luz direccional del sol

```
// ====== CÁLCULO DE LA DIRECCIÓN DEL SOL ======
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));
```

Se usa una variable timeOfDay que avanza lentamente con el tiempo y se limita a un ciclo de 0 a 2π. Aplicando coseno y seno se obtiene un vector que describe una trayectoria circular en el cielo. Ese vector se normaliza y se pasa al shader como sunDir. En el shader, sunDir se usa para calcular la luz direccional del sol (o la luna, dependiendo de la etapa del ciclo).

Para la segunda versión de OpenGL:

Se implementó un ciclo de día y noche con una luz direccional que simula el sol, variando su intensidad y color en función del tiempo. Las luminarias del escenario utilizan luces de tipo puntual (PointLight) que se encienden durante la

noche, mientras que focos tipo Spotlight iluminan zonas específicas como el ring y el globo aerostático. Los materiales responden a la luz para lograr realismo visual.

Se definen varias lámparas distribuidas estratégicamente a lo largo del camino, en zonas específicas como las chozas y en coordenadas fijas simulando el encendido de faroles durante la noche.

Cada luz se inicializa con su posición, color y niveles de intensidad difusa y ambiental. Posteriormente, se implementa un control mediante teclado, que permite activar o desactivar las lámparas..

Al detectar que la variable spotLights_On está activa, las lámparas aumentan su intensidad simulando su encendido nocturno. El sistema aplica los valores de intensidad a todas las lámparas registradas mediante los métodos SetDiffuseIntensity() y SetAmbientIntensity(), permitiendo que la iluminación varíe dinámicamente en tiempo real durante la ejecución del programa.

Fig. 4.7 *Luces tipo PointLight*

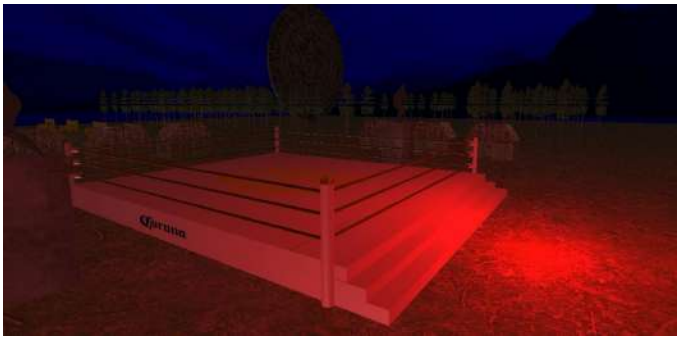

Fig. 4.8 Luces tipo SpothLight

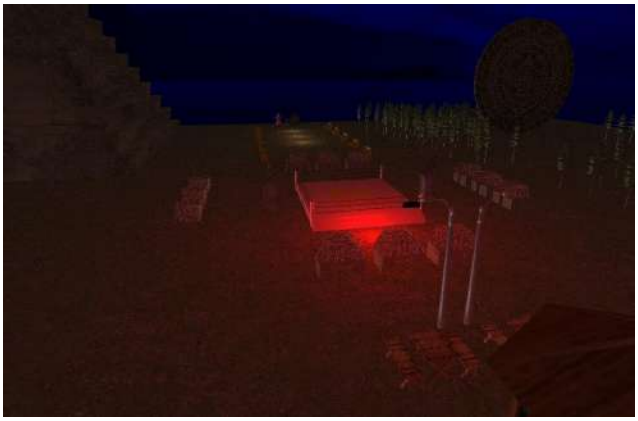

Fig. 4.9 *Luces tipo SpothLight*

First version:

• Directional, point, and spotlight lighting

• Animated bonfire using flicker

• Day/night cycle using trigonometric sun movement

Second version:

• Full dynamic day/night cycle

• Point lights act as streetlights at night

• Spotlights for ring and balloon

• Keyboard control for lights

Para Unity:

implementamos iluminación tanto del Sol, como en objetos: lámparas. Esto se ve a continuación:

Fig. 4.9.1

Fig. 4.9.2

2.5 Animaciones

Para la primera versión de OpenGL:

Se añadieron animaciones ligeras usando deltaTime, como movimientos y rotaciones de objetos. El usuario puede recorrer todo el entorno libremente observando iluminación, materiales, dimensiones y profundidad espacial, simulando una experiencia de exploración en primera persona.

Cuando sin(timeOfDay) es positivo, el sol está por encima del horizonte (es de día). Cuando es negativo, se puede considerar que es de noche y activar estrellas, aumentar la intensidad de la fogata y atenuar la luz ambiental global.

Fig. 5.0 *Ciclo día/noche*

Animación Hacha

La animación del hacha se implementó mediante un sistema de transformaciones jerárquicas aplicadas directamente en el modelo 3D dentro del pipeline de OpenGL. El objetivo fue simular un movimiento repetitivo de golpe, coherente con la temática prehispánica del escenario. Para lograrlo, se utilizó una rotación oscilatoria dependiente del tiempo, basada en la función sin() y en el valor retornado por glfwGetTime(). De esta manera, el hacha realiza un ciclo continuo de descenso y ascenso, lo que genera la apariencia de un golpe rítmico.

El pivote de rotación se posicionó en el extremo del mango, de modo que la hoja del hacha sea la parte que describe el movimiento mientras que el mango

permanece fijo. Esto permite que el movimiento sea más natural, ya que la animación se comporta como lo haría un arma real sujetada por una persona o colocada en una estructura ceremonial. La transformación del modelo respeta la secuencia habitual utilizada en todo el proyecto: traducción para posicionar el hacha en la escena, rotación para generar el movimiento y posteriormente el escalado.

Fig. 5.1 *Hacha*

Perro

Reproducir movimientos naturales y repetitivos dentro del escenario 3D. El objetivo principal fue dotar al modelo de un comportamiento dinámico y

coherente con la escena, simulando un movimiento simple pero expresivo que aportara vida al entorno. Para ello, se combinaron rotaciones alternadas, desplazamientos controlados y una estructura jerárquica que permite animar distintas partes del cuerpo del perro de manera independiente pero sincronizada.

La base de la animación consiste en un movimiento cíclico de las patas, generado mediante funciones trigonométricas dependientes del tiempo obtenido con glfwGetTime(). De esta forma, cada pata rota sobre su eje local siguiendo un patrón alternado: mientras las patas delanteras avanzan, las traseras retroceden ligeramente, simulando un paso continuo. La rotación se aplica respetando la jerarquía del modelo, donde el cuerpo funciona como nodo principal y las extremidades como nodos secundarios. Esto produce un movimiento fluido, manteniendo la cohesión visual del modelo.

Fig. 5.2 Chihuahua animado

Juego de la pelota

El movimiento principal de la pelota está basado en una función senoidal o parabólica obtenida mediante el uso de glfwGetTime(). Esta función permite generar un desplazamiento vertical continuo que imita el ascenso y descenso natural de un objeto en rebote. La amplitud del movimiento y la frecuencia del ciclo fueron ajustadas para que el bote resultara fluido, perceptible, pero no exagerado, manteniendo la coherencia visual con el resto del escenario. De esta manera, la pelota sube con una curva suave, alcanza un punto de máxima

elevación y luego desciende con mayor rapidez, repitiendo este patrón de forma indefinida.

Además del desplazamiento vertical, se incorporó un movimiento horizontal controlado para simular el desplazamiento de la pelota a lo largo de la cancha. Este movimiento se calcula con una variación lineal o senoidal ligera, lo cual crea la ilusión de que la pelota avanza mientras rebota, reproduciendo así el comportamiento que tendría durante un juego real. El modelo de transformaciones aplicado sigue el orden característico del proyecto: traslación para establecer la posición de la pelota en cada instante, rotación opcional para sugerir giro y finalmente el escalado general del modelo.

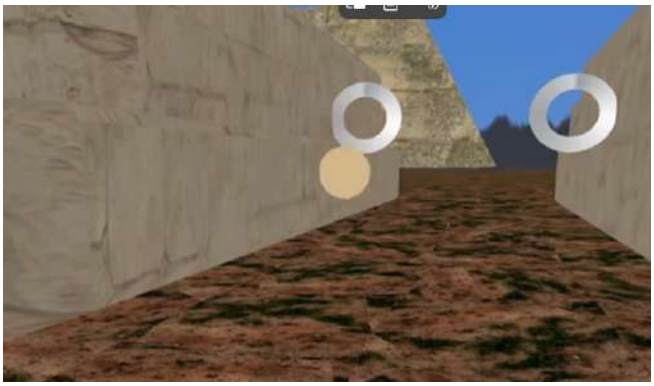

Fig. 5.3 Juego de la pelota animado

Para la segunda versión de OpenGL:

El proyecto cuenta con animaciones básicas activadas mediante teclado (por ejemplo, encender luces) y animaciones complejas con movimiento continuo, como el globo aerostático que utiliza un movimiento guardando keyframes para reproducir una animación o el simulado del día/noche con el sol. También se implementó una animación de caminata para el avatar y movimiento de NPCs. Las animaciones del escenario fueron diseñadas para ser controladas directamente por el usuario mediante teclado:

- Globo aerostático: control con teclas I, K, J, L y flechas ↑↓. Permite moverse en tres ejes. Tecla **G** guarda el keyframe actual, **K** reproduce la animación y **C** carga los keyframes desde el txt generado *globo_keyframes.txt* .

Fig. 5.4 *Animación Globo Aerostático*

-Simulación día/noche: El escenario se ilumina cuando el “sol” se posiciona sobre la escena y se oscurece cuando éste está por debajo..

Fig. 5.7 Cielo procedural

- Juego de pelota: el personaje Jake golpea una pelota al presionar J, generando un movimiento parabólico que simula rebote.

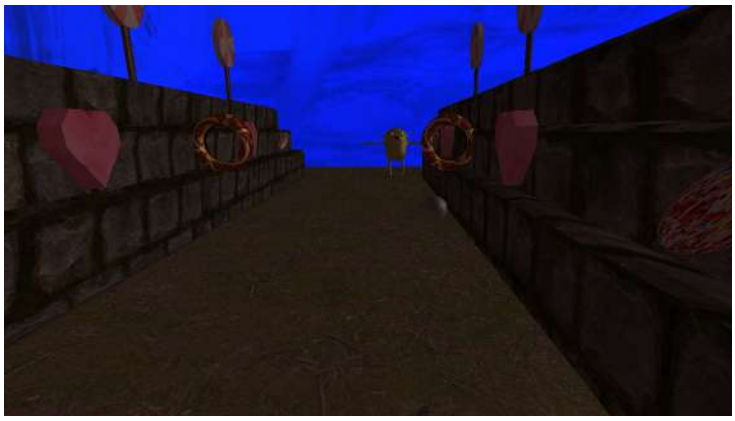

Fig. 5.8 Juego de la pelota con paletas

Para la versión de unity:

Implementamos dos animaciones, una de Kratos caminando libremente en la escena mediante un vacio ligado a el modelo de Kratos (Fig. 1.5.3).

Y otra donde la estatua sale del piso de la fachada grande:

Fig. 5.8.1

4. Pruebas y resultados

Vista de escena

Para la primera versión de OpenGL:

Tuvimos una serie de pruebas exitosas, al corroborar el eficiente funcionamiento del desplazamientos del usuario a través del entorno de la escena creada, además, la implementación de iluminación y animaciones fueron fluidas, no se trabó. Los modelos fueron cargados correctamente con sus texturas.

Fig. 6.2

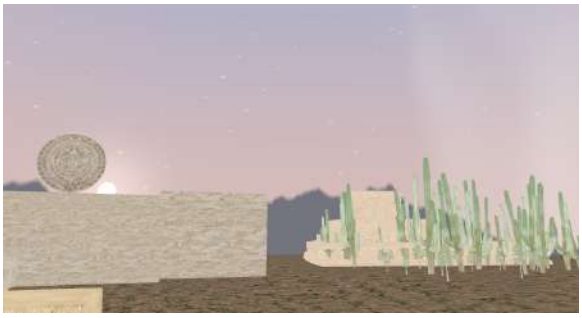

Fig. 6.3

Para la segunda versión de OpenGL:

Durante las pruebas se verificó el correcto funcionamiento de las cámaras, la iluminación dinámica, las animaciones y la interacción con el escenario.

Se realizaron pruebas de carga de modelos y texturas, así como de rendimiento general para asegurar una experiencia fluida.

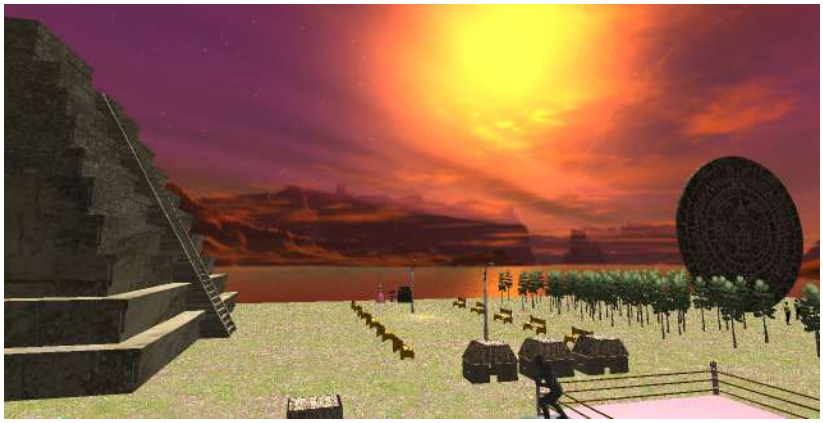

Fig. 6.4

Fig. 6.5

Fig. 6.6

Para Unity:

Notamos cómo mejoró la calidad visual a comparación de OpenGL, además de la rapidez y facilidad de cargar la escena.

Fig. 6.7

Fig. 6.8

De las tres versiones, esta fue la más fiel y fluida a nuestros modelos.

4. Conclusiones

Este proyecto resultó una experiencia muy completa y desafiante, ya que permitió aplicar de manera práctica los conocimientos adquiridos durante el semestre en el área de Computación Gráfica e Interacción

Humano-Computadora. A lo largo del desarrollo, se integran múltiples elementos, como el uso de modelos 3D texturizados, iluminación dinámica, animaciones por keyframes y control por teclado, además del cambio de cámaras y ciclos de día y noche, lo cual hizo que el entorno tuviera un alto nivel de interacción y realismo. Entre los aspectos más destacados se encuentran el globo aerostático animado con keyframes, el juego de pelota con física básica y la implementación de distintas cámaras (tercera persona, aérea y libre), cumpliendo con los objetivos planteados en los lineamientos del proyecto. A pesar de que el código fue extenso y el tiempo limitado, el trabajo en equipo permitió avanzar en cada universo temático Mario Bros, Mickey Mouse y Hora de Aventura integrando modelos propios y animaciones individuales dentro de un mismo escenario. En conclusión, este proyecto fue una excelente oportunidad para reforzar los conocimientos en modelado, texturizado y animación en OpenGL, demostrando la importancia de la organización, el trabajo colaborativo y la planeación para lograr un entorno funcional y visualmente atractivo, aunque con más tiempo se podrían optimizar las animaciones y mejorar el rendimiento general de la escena.

El desarrollo de la escena “tianguis prehispánico” en OpenGL permitió integrar en un solo proyecto varios de los temas centrales de la materia de Computación Gráfica: geometría procedimental, carga de modelos con Assimp, iluminación dinámica, texturizado avanzado y cámara interactiva en 3D. La combinación de elementos modelados en Blender (pirámides, chozas, maíz, caballo, juego de pelota, props del tianguis) con objetos generados por código (mesa, silla, florero, flor, fogata, máscara de jade, perrito voxel, carretilla, hacha, pelota animada) demuestra que es posible construir un entorno complejo mezclando contenido artístico y contenido totalmente paramétrico.

La implementación del cielo procedural con ciclo día/noche, el cálculo de la dirección del sol y el uso de un shader embebido para el cielo y el pasto texturizado con anti-tiling, permitió obtener una atmósfera visualmente rica sin depender de skyboxes prehorneados. El uso de ruido, FBM y funciones

personalizadas para nubes, estrellas, sol y luna dio como resultado un fondo dinámico que cambia en el tiempo de forma suave y coherente.

Por otro lado, el sistema de iluminación de la fogata, controlado mediante la tecla F, integró tanto una luz puntual para los modelos (shader externo) como una contribución emisiva y de iluminación local en el shader procedural. Esto reforzó el entendimiento de cómo combinar varias fuentes de luz (direccional y puntual), cómo atenuarlas con la distancia y cómo reutilizar la misma información (posición y color de la fogata) en distintos shaders para mantener consistencia visual en toda la escena.

Conclusiones individuales:

Cecilia: A mí en lo personal me gustó mucho trabajar en este proyecto, ya que integré lo que aprendí en el laboratorio y lo de teoría, aprender blender fue crucial para el desarrollo de este proyecto, además de comprender bien las funcionalidades de OpenGL, que a su vez, integramos al adaptar todo nuestro trabajo a Unity. La parte más tardada fue modelar, texturizar y animar, tanto en código como con herramientas como OpenGL. Pero realmente me pareció increíble poder tener un proyecto donde combinamos la curiosidad y la creatividad, ya que se sintió como jugar minecraft al modelar nuestras escenas y darles color e iluminación. Claro que quedan detalles que mejorar, pero quedé satisfecha con los resultados del proyecto, al integrar tres áreas interesantes: Historia, arquitectura y computación gráfica. Y añadiría también: videojuegos. Ya que, cuando el usuario interactúa con objetos y animaciones, se siente como un videojuego.

Valentina: Las instancias aleatorias de árboles, cactus y maizales, con exclusiones alrededor de áreas importantes (mesa, campamento, pirámides, Tula, zona central), permitieron practicar técnicas de dispersión controlada en

grandes terrenos, manteniendo composición visual y evitando solapamientos. La inclusión del perrito voxel animado, la pelota con rebote y la carretilla controlable con teclas añadió elementos lúdicos e interactivos que hacen que la escena no solo sea un “render estático”, sino un pequeño entorno casi de videojuego, donde el usuario puede explorar con la cámara y observar animaciones locales.

En conjunto, el proyecto cumplió el objetivo de integrar en un mismo escenario: modelos complejos, geometría procedural, técnicas de texturizado (incluyendo anti-tiling y anisotropía), iluminación dinámica dependiente del tiempo y un sistema de interacción en primera persona. Además, sirvió como experiencia completa de pipeline: desde modelado y exportación, pasando por carga, organización de la escena y diseño de shaders, hasta la documentación y análisis de la arquitectura del software.

Pablo: Este proyecto nos permitió crear un mundo en 3D vivo e interactivo, mezclando tradiciones mexicanas como las temáticas de las pirámides y el tianguis, con personajes de universos de algunas de nuestras caricaturas preferidas como Mario Bros y Hora de Aventura. Lo más interesante y al igual desafiante de este proyecto, fue que logramos que el escenario tenga, por ejemplo, un ciclo de día y de noche, las lámparas se encienden solas al cambiar el ambiente y el juego interactivo de la pelota. Además, elementos como el globo aerostático controlado por keyframes hace que se sienta como un pequeño acercamiento a lo que es un videojuego.

Shareny:El desarrollo de este proyecto fue un reto muy completo, ya que tuvimos que unir muchas técnicas diferentes en un solo lugar. Combinamos objetos que modelamos propiamente en Blender, con otros elementos importados que le dieron total vida y complemento al escenario. También, hubo un enfoque específico en los detalles visuales, aplicando texturas que parecen piedra o madera real y animaciones, como el rebote de la pelota que sigue un movimiento natural. Al final, logramos que todo funcionara en conjunto sin problemas, permitiendo al usuario interactuar con el entorno y navegar en él de distintas formas

Santiago: Gracias al trabajo en conjunto, pudimos construir un escenario muy atractivo visualmente que une temáticas tan distintas como la lucha libre y el mundo prehispánico en un solo escenario. Logramos incluir desde un ring con texturas metálicas brillantes hasta un minijuego interactivo como el juego de la pelota, cuidando que la iluminación y los materiales se vieran bien en todo momento. Aunque fue un reto unir diversas áreas de conocimiento que teníamos y en las que mejor nos desempeñábamos cada uno, al final, logramos construir un escenario que incluye todo lo visto en clases permitiendo al usuario explorar un mundo en 3D interactivo.

5. Trabajo en equipo

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							

Tabla 1.1 Diagrama de Gantt primera parte

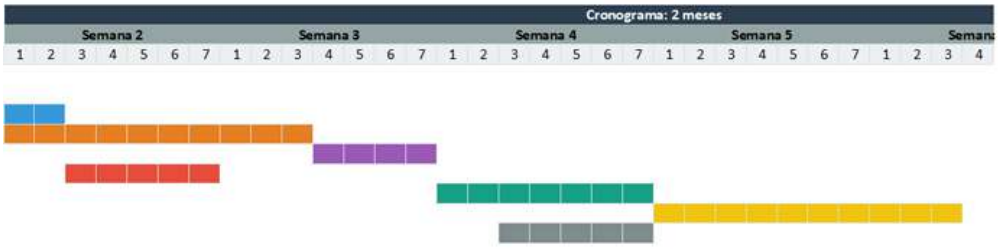

Tabla 1.2 Segunda parte de la Tabla 1.1

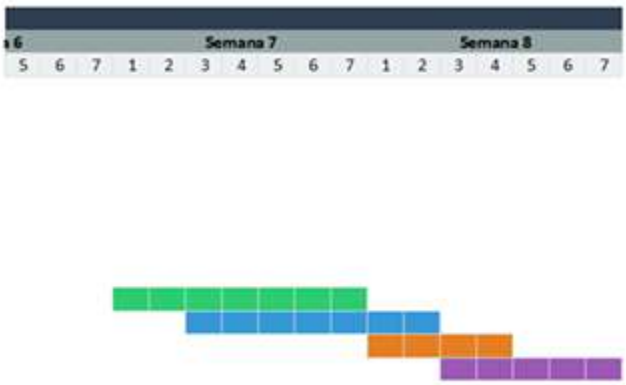

Tabla 1.3 Tercera parte de la Tabla 1.1

Valentina aportó una combinación destacada de habilidades técnicas y artísticas. En la fase de OpenGL, modeló estructuras como Tula, vegetación variada y múltiples objetos creados desde código, además de desarrollar uno de los sistemas visuales más avanzados del proyecto: el cielo procedural animado. Este sistema incluía transiciones de color basadas en tiempo, cambios atmosféricos, simulación de ciclos día/noche y una integración coherente con la iluminación direccional. También implementó animaciones jerárquicas como el hacha y el perro voxel, que requerían comprender estructuras de nodos, rotaciones dependientes del tiempo y transformaciones locales. Esta experiencia técnica fue clave en Unity, donde se dedicó a adaptar sus modelos de Blender y OpenGL al motor, corrigiendo UVs, manteniendo la integridad de materiales, suavizando normales y optimizando geometrías para evitar caídas en el rendimiento. Valeria también participó activamente en la iluminación del entorno en Unity, ajustando sombras, parámetros HDR y la coherencia cromática del mapa, además de distribuir vegetación y objetos decorativos para crear transiciones visuales naturales entre zonas temáticas.

Cecilia fue la integrante que más contenido artístico y cultural aportó al proyecto completo. En OpenGL modeló y texturizó elementos de alta complejidad, como el tianguis prehispánico, las chozas, la fachada inspirada en Tenochtitlán, los petates y una extensa variedad de objetos artesanales. El detallado proceso de UV Mapping que realizó en la fachada y en el tianguis requirió precisión y horas de trabajo, especialmente al ajustar texturas para evitar distorsiones. También desarrolló la animación de la fogata con iluminación dinámica, incluyendo el efecto de flicker que afectaba objetos cercanos, integrando shaders y cálculos dependientes del tiempo. Al migrar a Unity, Cecilia adaptó correctamente sus modelos, reconstruyó materiales, ajustó escalas y distribuyó sus elementos en zonas completas del mapa, creando espacios temáticos completos que enriquecían la narrativa visual del escenario. Implementó iluminación puntual en áreas como el mercado, destacando objetos culturales, y colaboró en la organización de colliders para asegurar la navegación fluida del usuario. Su trabajo hizo que tanto en OpenGL como en Unity el escenario tuviera identidad, profundidad y una cohesión estética inconfundible.

Shareny desempeñó un rol significativo tanto en OpenGL como en Unity. Durante la primera etapa, modeló elementos como el ring de Peach, el calendario azteca y participó activamente en la separación de personajes y la implementación inicial de modelos complejos dentro de la escena. También colaboró en el diseño del croquis original, la planificación de la distribución y la programación del movimiento del globo aerostático, lo cual implicó manejar traslaciones, escalas y control por teclado. Esta experiencia le permitió tener un entendimiento claro de cómo se comportaban los modelos en un entorno tridimensional, conocimiento que fue esencial en Unity, donde se encargó de integrar y ajustar muchos de los modelos principales del escenario, optimizando escalas, corrigiendo rotaciones y adaptando materiales importados desde Blender para que conservaran su calidad visual. Asimismo, colaboró en la organización de jerarquías dentro del motor, en la revisión de colisiones y en la

depuración final, asegurando que cada zona del mapa estuviera equilibrada visualmente y funcionara correctamente. Su habilidad para detectar errores visuales y su experiencia en OpenGL con transformaciones y texturas facilitaron enormemente la etapa final de Unity.

Santiago tuvo uno de los roles técnicos más importantes del proyecto y estuvo presente a lo largo de las dos etapas. En OpenGL implementó las tres cámaras principales (tercera persona, aérea e isométrica), desarrolló el skybox dinámico de día/noche, configuró luces puntuales que se activaban según el ciclo temporal y programó la animación completa del juego de pelota con físicas simplificadas, colisiones y movimiento horizontal. Estos conocimientos fueron fundamentales para la versión en Unity, donde configuró el sistema de cámaras del proyecto final, ajustando distancias, comportamiento de seguimiento y transición entre vistas. También trabajó en la iluminación global mediante skybox HDR, la calibración de luz direccional, sombras y parámetros ambientales, aplicando su comprensión previa de iluminación dinámica. Asimismo, implementó el movimiento del avatar, scripts de interacción, colliders generales del mapa y revisó materiales para asegurar correcta visualización bajo distintas condiciones de luz. Su dominio técnico permitió que la escena en Unity se sintiera fluida, controlable y visualmente balanceada.

Pablo contribuyó con algunos de los sistemas más avanzados del proyecto, destacando tanto en OpenGL como en Unity. En la primera etapa modeló elementos como el mercado, bancos y objetos interactivos, además de desarrollar animaciones complejas como el arco con texto desplazable y las puertas con fases de apertura, usando interpolaciones, jerarquías y cálculos dependientes del tiempo. También integró masivamente árboles, lámparas, bancas y estatuas, organizando la escena con gran atención al detalle. En Unity, este conocimiento se trasladó al desarrollo de scripts de interacción, activación

de elementos, configuración de prefabs dinámicos y ajuste de animaciones dentro del motor. Asimismo, optimizó modelos propios y ajenos, ajustó colisiones, organizó zonas completas del mapa y colaboró en la depuración final de la escena, cuidando rendimiento y estabilidad. Su experiencia previa con animaciones jerárquicas y lógica interna en OpenGL permitió recrear comportamientos complejos en Unity, logrando un entorno interactivo robusto y funcional.

6. Bibliografía

- 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

ANEXOS:

Link a GitHub de la primera implementación en OpenGL:
https://github.com/CeciliaXimenaSolisCisneros/ProyectoCompuGrafica

Link a GitHub de la segunda implementación en OpenGL:

https://github.com/PablinCu02/Proyecto_CGEIHC

Link a GitHub de la versión en Unity:

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

Link a vídeo: https://youtu.be/qDD2HFZ3GEc