

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

Proyecto final

Manual técnico

Computación Gráfica e Interacción Humano –
Computadora

Realizado por:
311243563

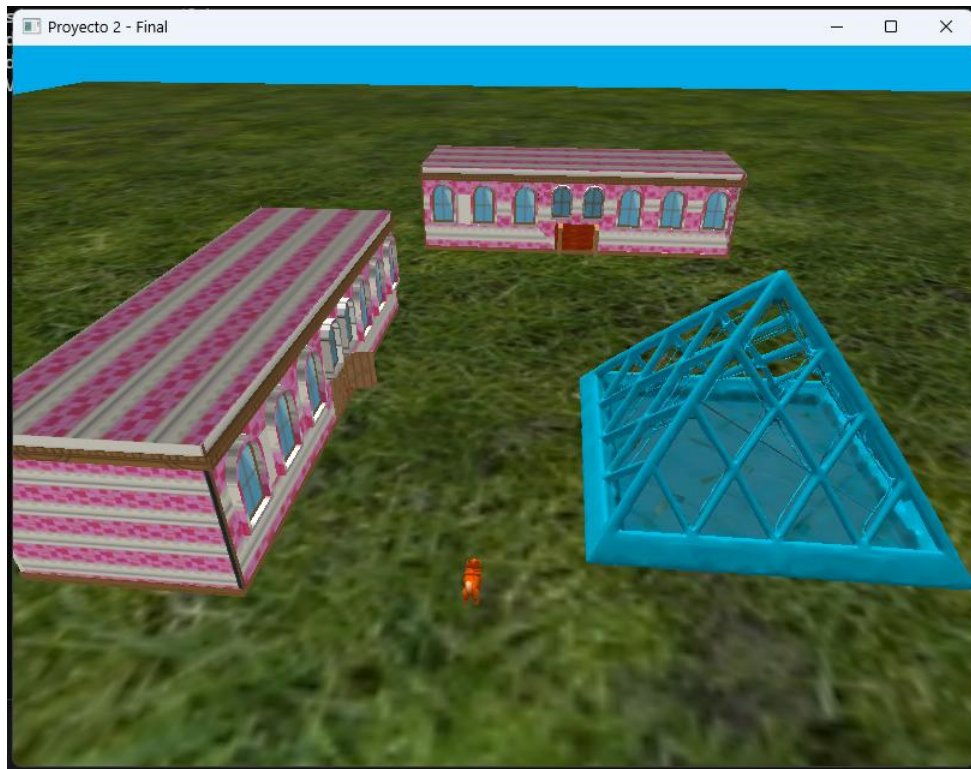
Profesor. Ing. Carlos Aldair Román Balbuena
Grupo 05

Semestre. 2026-1

Manual técnico

Arquitectura del proyecto

La arquitectura del entorno virtual sirve como el nexo central (hub) desde el cual el usuario accede a las distintas salas temáticas del proyecto. El diseño se basa en una reinterpretación estilizada del Museo del Louvre, priorizando una estética "caricaturesca" sobre el realismo, tal como se definió en los objetivos del proyecto. El elemento arquitectónico central de la escena es una representación de la icónica pirámide de cristal del Louvre, que actúa como punto de referencia principal para la navegación del usuario.



En esta implementación, la pirámide central está rodeada por múltiples módulos de edificios que representan las alas del museo. Estos edificios, aunque inspirados en la fachada real, se han modelado de forma simplificada y modular, utilizando un conjunto repetido de assets para optimizar el rendimiento y mantener la coherencia visual del estilo caricaturesco.

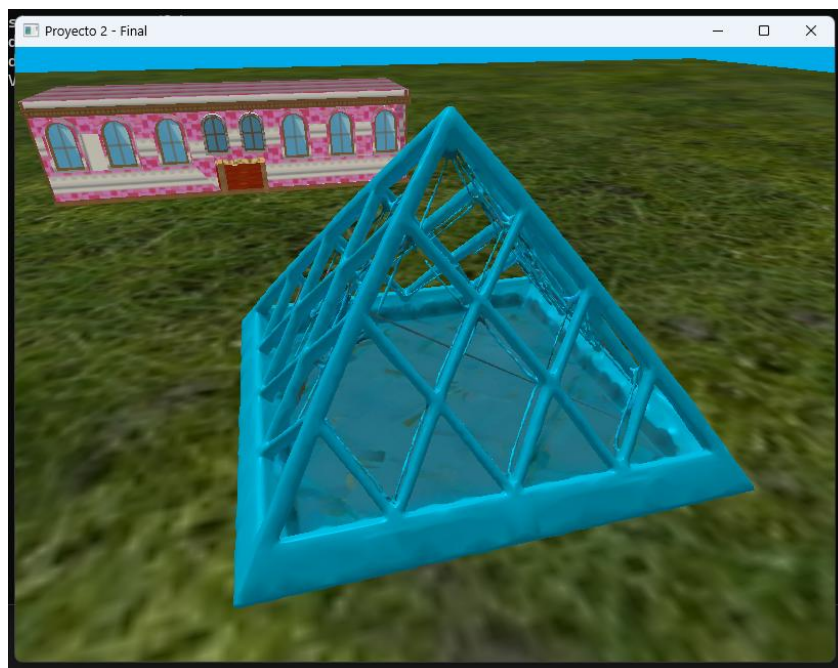
El modelado de los edificios se caracteriza por una geometría simplificada, con texturas limpias que evitan el fotorrealismo. Los elementos arquitectónicos clave, como las ventanas

de arco, se han exagerado ligeramente para alinearse con el enfoque artístico del proyecto. Esta estructura exterior alberga los accesos a los interiores de las cuatro salas temáticas (Antigüedades Egipcias, Pintura Europea), sirviendo como una transición clara entre el espacio de exploración general y las experiencias de galería específicas.

Arquitectura de la Pirámide Central

El elemento arquitectónico principal del entorno es la pirámide central, una representación estilizada de la icónica pirámide de cristal del Louvre. Su diseño 3D se alinea con el enfoque "caricaturesco" general del proyecto, priorizando la forma reconocible y la optimización del rendimiento sobre el realismo. El modelo está compuesto por dos elementos principales: el armazón estructural y las caras de cristal texturizadas. El armazón se ha simplificado en un marco grueso de color sólido que define las cuatro aristas principales y la base de la pirámide. Este enfoque omite deliberadamente el complejo entramado metálico de la estructura real para mantener una silueta limpia y estilizada.

Para las caras de la pirámide, se ha empleado una única textura en lugar de modelar individualmente los múltiples paneles de vidrio. Esta textura simula un cristal de tono azulado y presenta un patrón diagonal repetitivo que crea la ilusión de reflejos o biseles. Esta técnica de texturizado es una optimización clave: reduce drásticamente la carga poligonal y el costo de renderizado, logrando la apariencia deseada de una superficie de vidrio reflectante sin la complejidad geométrica del objeto real.



Arquitectura de las Salas

Los edificios que albergan las salas temáticas constituyen los módulos de galería que rodean la pirámide central. Su diseño se adhiere al enfoque "caricaturesco" del proyecto, sirviendo como una fachada exterior inspirada en la arquitectura del Louvre, pero significativamente simplificada.

La arquitectura de estos módulos se caracteriza por:

- **Geometría Modular:** La estructura es un bloque rectangular alargado y de baja altura, diseñado como un asset modular y repetible. Esta simplificación geométrica es fundamental para la optimización del rendimiento en la escena.
- **Fachada y Texturizado:** La pared principal utiliza una textura de ladrillo de tonos claros (beige/crema). La textura es un patrón uniforme y limpio, que evita el fotorrealismo en favor del estilo visual definido. Un borde decorativo marrón, simulando una cornisa, separa la fachada del tejado.
- **Ventanas:** El edificio presenta dos tipos de ventanas, ambas con un marco marrón grueso y un cristal texturizado en azul opaco, coherente con el material de la pirámide central.
- **Ventanas de Fachada:** Una serie de grandes ventanas de arco en el nivel principal.
- **Buhardillas (Dormers):** Ventanas de arco más pequeñas que emergen del tejado de mansarda, siguiendo su curvatura.
- **Puntos de Acceso:** En el centro de la fachada se define una entrada. Esta área se diferencia por una textura de suelo de baldosas (rojo/rosa) y la inclusión de dos objetos que simulan barreras de acceso, indicando al usuario el punto de ingreso a la sala interior correspondiente.



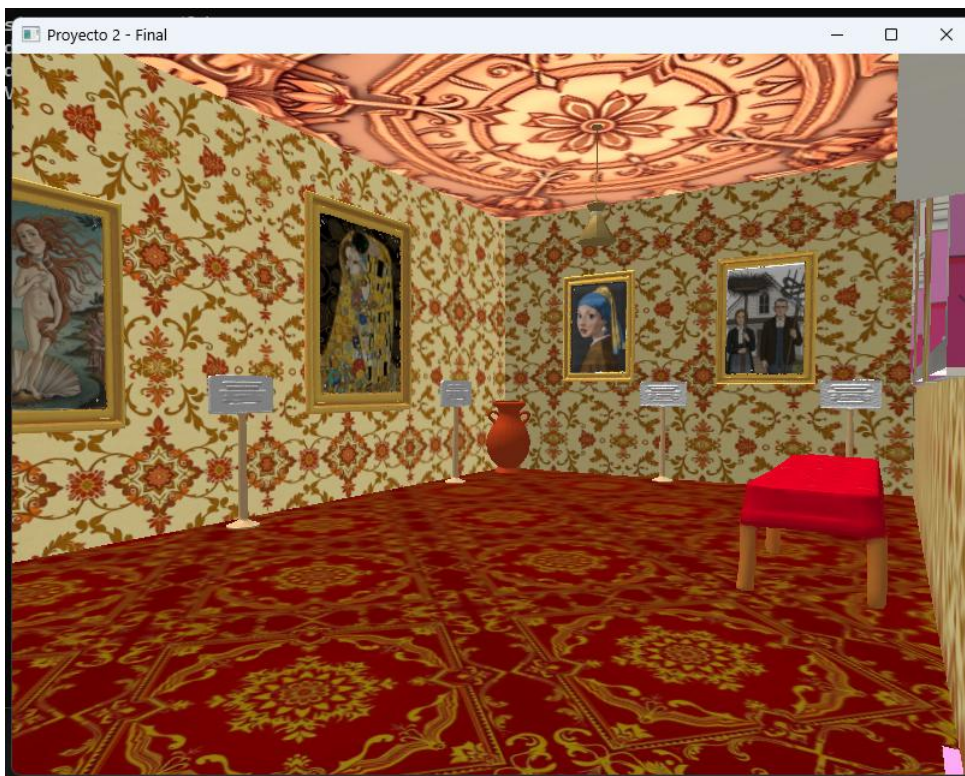
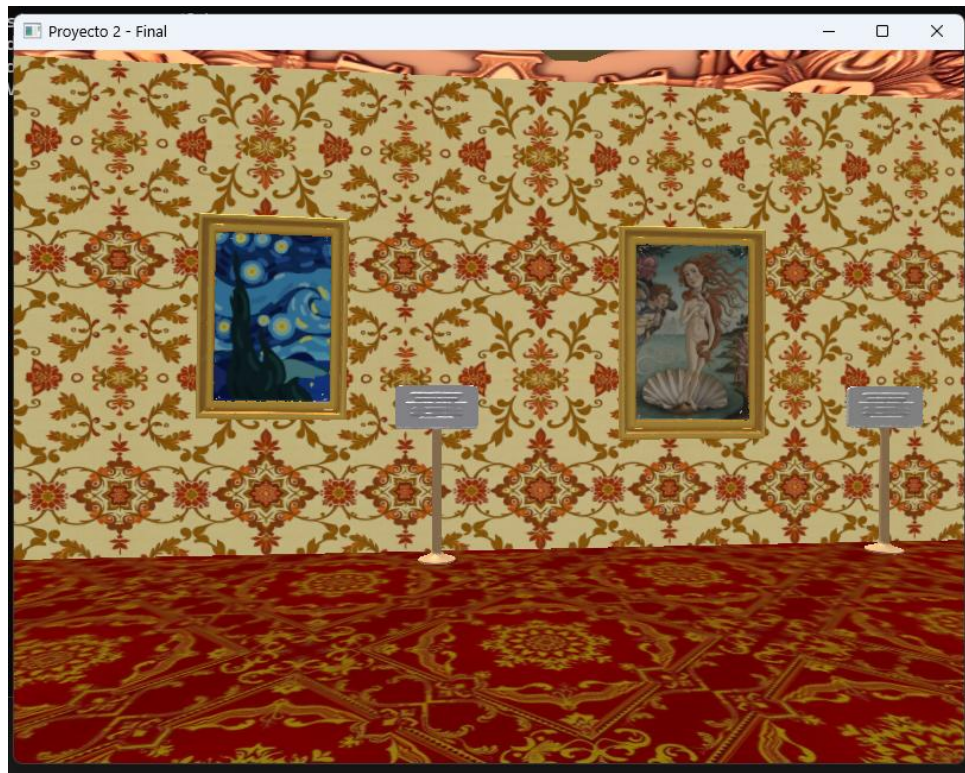
Arquitectura Interior: Sala de Pintura Europea

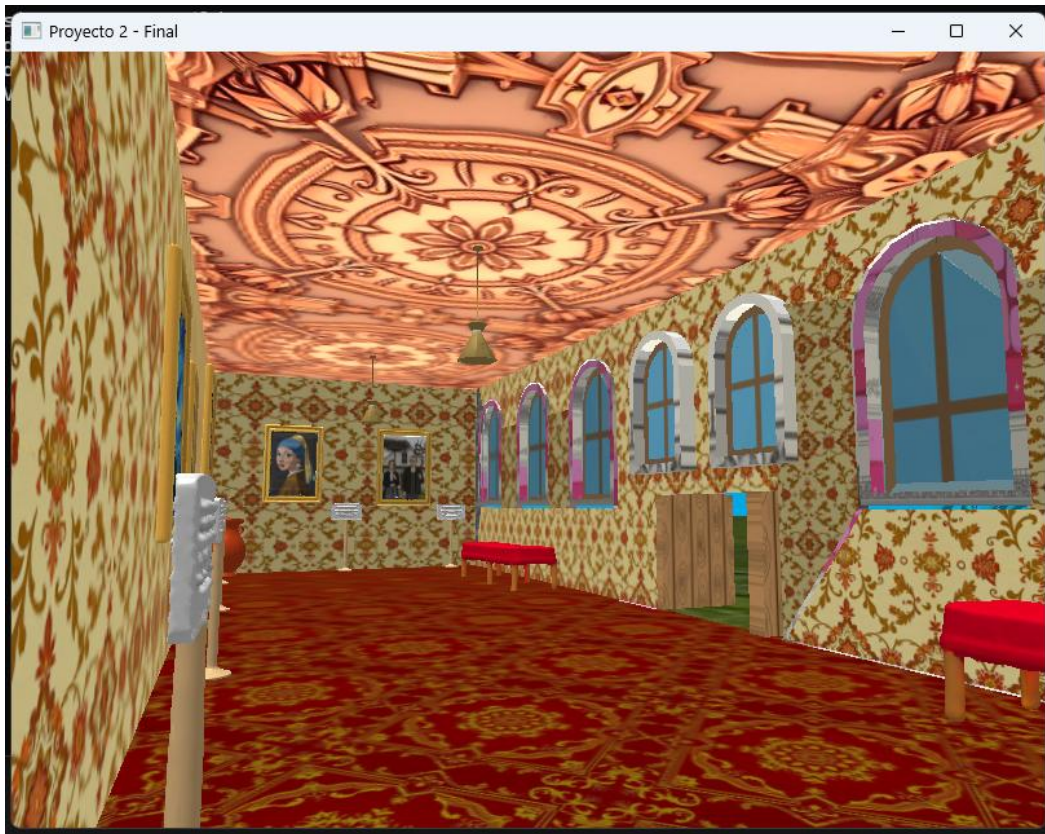
La Sala de Pintura Europea está diseñada para replicar la escala y la atmósfera de una gran galería clásica, como la Grande Galerie del Louvre, manteniendo el nivel de estilización y simplificación visual del proyecto.

Estructura y Proporciones: La sala presenta una estructura de corredor alargado y un techo de gran altura, diseñado para generar una sensación de monumentalidad. Las paredes están pintadas en tonos claros (crema/beige), con un dado o zócalo inferior definido por un listón horizontal de color más oscuro (marrón/rojizo).

Iluminación Cenital: La característica arquitectónica más importante es el sistema de iluminación. El techo está dominado por una bóveda de cañón o un arco alto en su eje central, el cual está perforado por una claraboya cenital continua que recorre toda la longitud de la sala. Esta claraboya es la fuente de iluminación principal simulada, lo que es esencial

para la exhibición de pinturas históricas. Su estructura está definida por un simple entramado geométrico.





Arquitectura de Exhibición: Las paredes laterales están articuladas por grandes marcos arqueados integrados a la estructura, de color dorado o latón. Estos marcos sirven para delimitar y enmarcar individualmente las pinturas a gran escala.

Obras Exhibidas: Las pinturas se integran dentro de estos marcos arquitectónicos y presentan sus propios marcos dorados ornamentados, diseñados con geometría simple para mantener la coherencia con el estilo caricaturesco.

Suelo: El suelo es de un color oscuro y uniforme, similar al utilizado en otras salas clásicas, que contribuye a contrastar y enfatizar la luminosidad generada por la claraboya cenital.

El diseño consigue evocar la grandiosidad histórica de una galería de pinturas, a través del uso estratégico de la escala y la simulación de iluminación natural cenital.

Arquitectura Interior: Sala de Antigüedades Egipcias

La Sala de Antigüedades Egipcias está diseñada para evocar la atmósfera de un templo o una sala hipóstila, utilizando elementos arquitectónicos monumentales y una paleta de colores cálidos, todo ello bajo el prisma de la estilización.

Estructura y Proporciones: La sala presenta una planta simétrica, definida por un pasillo central flanqueado por columnas. El espacio tiene una sensación de peso y escala a través de la masividad de sus elementos, lo que es característico de la arquitectura egipcia.

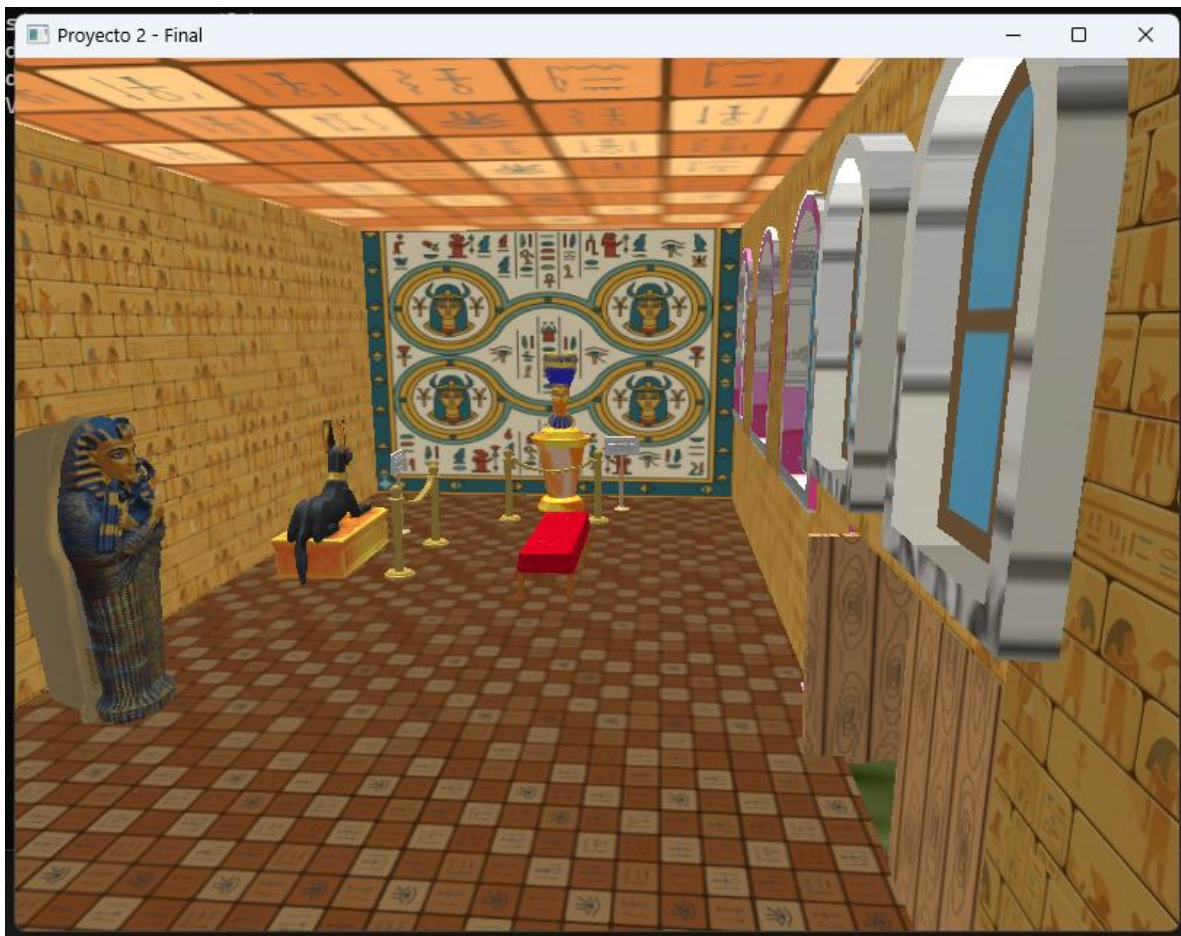
Paleta de Color y Materiales: La arquitectura está dominada por una textura uniforme de color ocre o amarillo arena cálido, que simula el color de la piedra antigua y el entorno desértico. Esta textura se aplica a paredes y columnas por igual. El suelo, en contraste, utiliza una textura oscura y uniforme para anclar la escena visualmente.

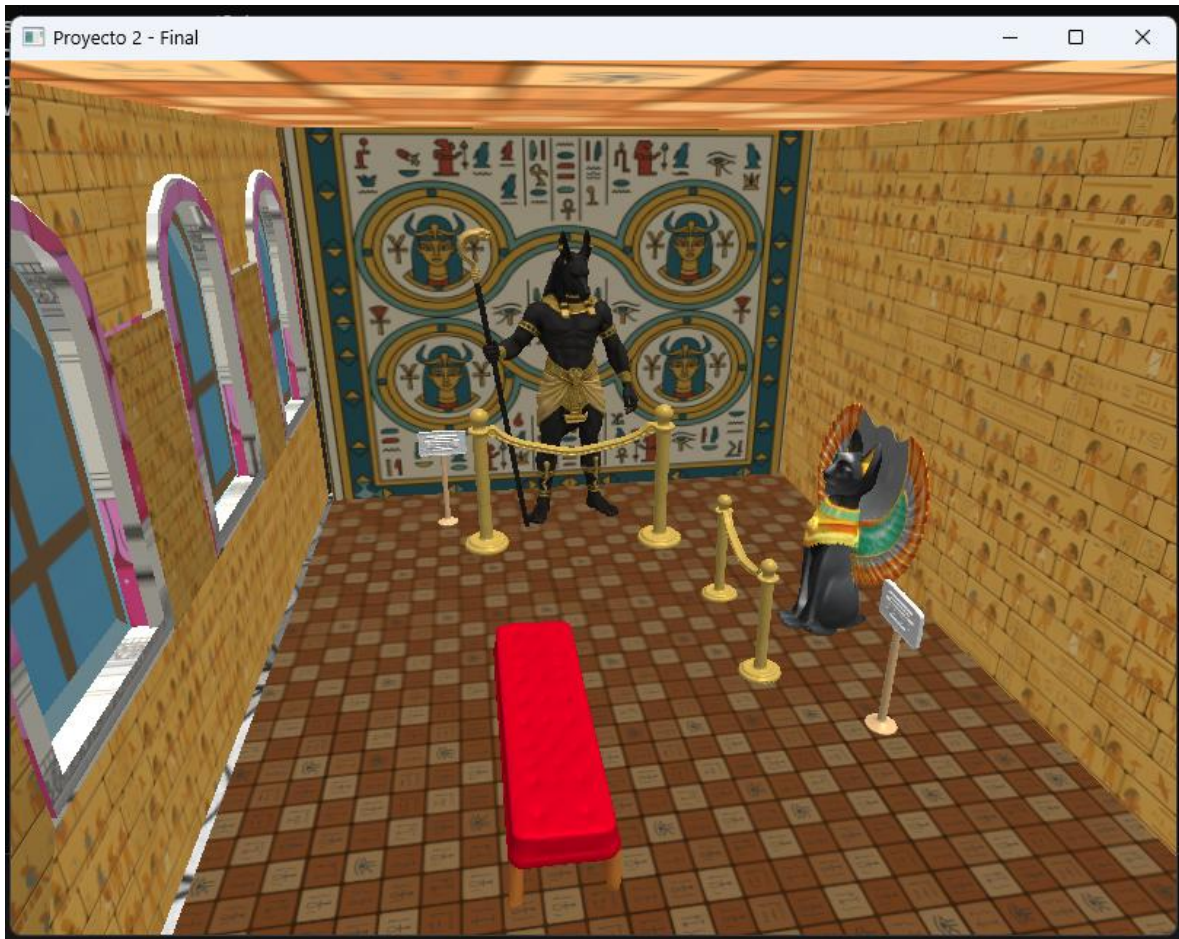
Columnas Monumentales: El rasgo arquitectónico central es una serie de columnas cilíndricas monumentales, modeladas con geometría simplificada. Los capiteles (la parte superior de la columna) están estilizados, sugiriendo los motivos florales (papiro o loto) de la arquitectura faraónica, sin la complejidad detallada del modelado fotorrealista.

Decoración Temática: El detalle decorativo se logra mediante el uso de texturas planas de jeroglíficos simplificados aplicadas directamente sobre las paredes y los fustes de las columnas. Este método permite incorporar la identidad temática de la sala de manera eficiente en términos de rendimiento.

Exhibición: La sala alberga estatuas y sarcófagos de gran escala, modelados con formas geométricas y caricaturescas. El manejo del tráfico de visitantes se implementa mediante el uso consistente de postes dorados (estacionadas) y cuerdas rojas, delimitando las áreas de exhibición.

La arquitectura de esta sala logra su ambientación temática a través de la repetición estructural, el color y la aplicación de texturas icónicas, cumpliendo con la directriz del proyecto de priorizar la expresividad visual estilizada.





El proyecto representa un museo virtual conformado por cuatro salas principales, cada una modelada en 3D e integrada mediante OpenGL y la librería GLM para las transformaciones. Estas salas se cargan a través del sistema de modelos definidos en la función `inicializarListaModelos()`, donde se incluyen los archivos `sala1.obj`, `sala2.obj`, `sala3.obj` y `sala4.obj`. Además, cada sala contiene diferentes objetos como esculturas, engranajes, accesorios y contenedores, los cuales son gestionados mediante un vector de estructuras `ModeloInfo` que almacena el nombre del modelo, su referencia y su estado de carga.

Todas las salas y objetos se dibujan en el bucle principal del renderizado. El piso (`piso.obj`) se carga como base común, y sobre él se posicionan las salas y sus modelos según las transformaciones globales definidas en `traslacionGlobal`. La cámara libre (`Camera`), controlada con teclas WASD y el ratón, permite explorar las cuatro salas de manera fluida. Adicionalmente, la carga progresiva de modelos optimiza el rendimiento, evitando bloqueos al renderizar un entorno extenso con múltiples objetos.

Animación Sarcófago Mágico

La secuencia principal de la animación se gestiona mediante una máquina de estados finitos controlada por la variable `animState`. Esta secuencia se activa a través de un evento de teclado, el cual cambia el estado de reposo al estado de apertura. La lógica interna evalúa el tiempo delta para transicionar secuencialmente entre cinco fases operativas: apertura de la tapa, traslación frontal de la momia para su salida, una pausa temporal, traslación inversa para el retorno y, finalmente, el cierre de la tapa.

El comportamiento cinemático de la tapa del sarcófago implementa una transformación compuesta que combina una traslación lineal en el eje X con una oscilación sinusoidal en el eje Z. El uso de funciones trigonométricas moduladas por la posición de apertura permite simular un efecto de levitación o vibración mecánica mientras la tapa se desliza, calculando la altura en función del radio de apertura. Adicionalmente, se aplica un efecto de vibración o "temblor" calculado con el tiempo de ejecución global para añadir realismo al desplazamiento de la pieza sobre la base.

Por su parte, el movimiento de la momia se basa en un modelado jerárquico donde el torso actúa como nodo padre y las extremidades como nodos hijos. La locomoción utiliza una variable oscilatoria que rota las piernas alternadamente dentro de un rango angular definido, invirtiendo la dirección al alcanzar los límites para simular pasos constantes. Simultáneamente, la rotación de los brazos se ajusta dinámicamente según el estado de la animación; durante la caminata y la pausa, estos rotan en el eje X para simular una pose de persecución, mientras que en reposo mantienen una postura retraída.

Para simular la salida de la momia del sarcófago, se aplica una interpolación lineal sincronizada tanto en posición como en escala. A medida que la entidad avanza frontalmente en el eje Z, el algoritmo incrementa progresivamente su escala global desde un valor unitario hasta un máximo de 1.4, generando un efecto visual de perspectiva forzada y acercamiento dramático. Al invertirse la animación para el retorno, tanto la posición como la escala se interpolan negativamente hasta regresar a los valores originales de confinamiento dentro del sarcófago.

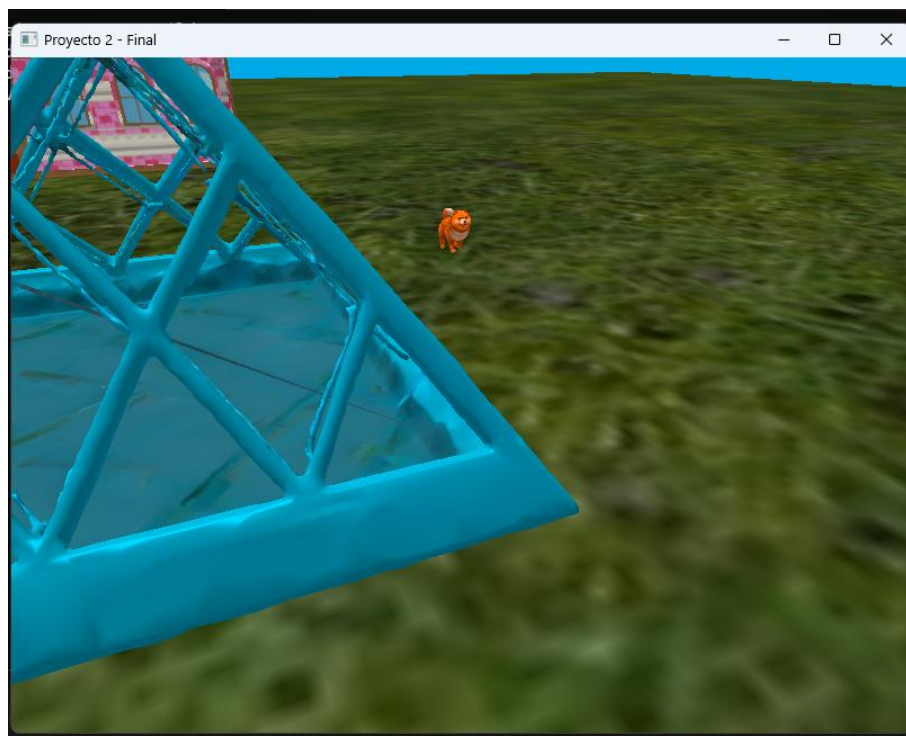
Animación del perro caminando

El desplazamiento del perro por la escena se gestiona mediante una máquina de estados que define una trayectoria rectangular preestablecida. El código divide el recorrido en cuatro fases distintas: avanzar, girar a la izquierda, regresar y girar a la derecha. Dependiendo del

estado activo, se actualizan las coordenadas de posición en los ejes X o Z y se ajusta la orientación del cuerpo a ángulos fijos (0° , 90° , 180° o 270°). El sistema verifica constantemente si el modelo ha alcanzado los límites definidos en el escenario para realizar la transición al siguiente tramo, logrando así un ciclo de movimiento continuo y autónomo.

Para simular la acción de caminar, se emplean funciones senoidales vinculadas al tiempo de ejecución de la aplicación, lo que genera oscilaciones suaves y periódicas. Estas funciones calculan los ángulos de rotación para las patas y la cola en cada fotograma. Para lograr un efecto natural, se aplican desfases matemáticos en las fórmulas de cada extremidad; esto asegura que las patas delanteras y traseras se muevan de manera alternada en lugar de al mismo tiempo, imitando el paso característico de un cuadrúpedo, mientras la cola se balancea con su propio ritmo independiente.

En cuanto a la construcción gráfica, se utiliza un modelo jerárquico donde el cuerpo funciona como el nodo principal o "padre". Al dibujar la escena, primero se calcula la posición y rotación global del torso según la trayectoria actual. Posteriormente, la cabeza, las patas y la cola se dibujan tomando como referencia la ubicación del cuerpo, heredando sus transformaciones. Esto permite que las extremidades sigan al perro por todo el escenario automáticamente, aplicando sobre ellas únicamente sus rotaciones locales para la animación de las articulaciones.



SkyBox

La implementación del SkyBox en el proyecto se realizó con el objetivo de proporcionar un entorno visual inmersivo que simule un fondo tridimensional envolvente. Un skybox consiste en una gran caja cúbica que rodea toda la escena y en la que se aplican texturas en su interior, generando la ilusión de un cielo o paisaje infinito que no se ve afectado por la perspectiva ni por la posición de la cámara. En este proyecto, el skybox se utilizó para ambientar el museo en un entorno cerrado y coherente con la iluminación y la atmósfera del escenario.

```
//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);
```

El proceso de implementación comienza con la definición de los vértices del cubo que forma el skybox. Estos vértices se almacenan en el arreglo skyboxVertices, el cual describe las seis caras del cubo con sus respectivas coordenadas tridimensionales. Cada cara del cubo se renderiza utilizando triángulos, y no se requiere información de normales o coordenadas de textura convencionales, ya que la proyección se calcula en el shader. Este cubo se dibuja en torno al punto de vista del observador, de modo que parece que la cámara siempre se encuentra en el centro del entorno.

A continuación, se crea un Vertex Array Object (VAO) y un Vertex Buffer Object (VBO) específicos para el skybox, mediante las funciones glGenVertexArrays y glGenBuffers. Los vértices del cubo se cargan en la memoria de la GPU utilizando glBufferData, y se configura el atributo de posición en el vertex shader (índice 0) con glVertexAttribPointer. Esta configuración permite que el shader reciba la posición de cada vértice al momento de dibujar el cubo.

Para las texturas del skybox se utiliza un mapa cúbico (cubemap), que es una textura especial compuesta por seis imágenes, una por cada cara del cubo. En el código, las rutas de estas imágenes se almacenan en un vector denominado faces, que incluye las texturas right.jpg, left.jpg, top.jpg, bottom.jpg, back.jpg y front.jpg, ubicadas en la carpeta SkyBox/. Estas imágenes se cargan mediante la función TextureLoading::LoadCubemap(faces), que

genera un identificador de textura OpenGL (`cubemapTexture`) asociado al cubemap. Este recurso se enlaza posteriormente al shader encargado de renderizar el skybox.

El shader utilizado para el skybox se compone de los archivos `SkyBox.vs` (vertex shader) y `SkyBox.frag` (fragment shader). En el vertex shader, los vértices del cubo se transforman únicamente por la matriz de vista (`view`) y la proyección (`projection`), pero eliminando la componente de traslación de la vista para que el skybox siempre se mantenga centrado respecto a la cámara, sin importar su movimiento. De esta manera, el observador puede moverse libremente sin percibir cambios en la distancia aparente del entorno. En el fragment shader, se realiza la muestra de color correspondiente en la textura del cubemap según la dirección del vector del vértice, proyectando correctamente la textura sobre las caras internas del cubo.

Durante el proceso de renderizado, se cambia temporalmente la función de profundidad (`glDepthFunc(GL_LEQUAL)`) para garantizar que el skybox se dibuje detrás de todos los objetos de la escena. Luego, se activa el shader del skybox (`skyboxshader.Use()`), se actualizan las matrices de vista y proyección con los valores de la cámara, y se enlaza la textura del cubemap a la unidad de textura activa. Finalmente, se dibuja el cubo mediante `glDrawArrays(GL_TRIANGLES, 0, 36)`, utilizando los 36 vértices que conforman las seis caras. Al finalizar, se restablece la función de profundidad a su valor original (`glDepthFunc(GL_LESS)`) para que el resto de los objetos se rendericen correctamente sobre el fondo.

Gracias a esta implementación, el skybox logra crear un entorno tridimensional que rodea toda la escena del museo, proporcionando profundidad visual sin aumentar el costo computacional de renderizar un escenario lejano. Este recurso no solo mejora la estética general, sino que también contribuye a la percepción de escala y realismo del entorno virtual.

Tecnologías utilizadas

Implementación de OpenGL y Renderizado

El núcleo del proyecto se gestiona a través del archivo ProyectoFinalMuseo.cpp, el cual utiliza la API de OpenGL para la renderización gráfica en tiempo real. La implementación sigue el pipeline de gráficos programable moderno (Modern OpenGL).

Configuración del Entorno y Contexto

Para interactuar con el sistema operativo y gestionar el contexto de OpenGL, el proyecto no utiliza OpenGL directamente, sino que se apoya en dos bibliotecas de C++ esenciales:

GLFW (Graphics Library Framework): Se utiliza para la creación y gestión de la ventana de la aplicación (glfwCreateWindow), la gestión del contexto de OpenGL (glfwMakeContextCurrent), y la captura de eventos de entrada del usuario (teclado y mouse), como se ve en las funciones KeyCallback y MouseCallback.

GLEW (OpenGL Extension Wrangler): Se inicializa con glewInit(). Su función es fundamental para cargar los punteros a las funciones de las extensiones modernas de OpenGL (versiones 3.3+), permitiendo el acceso a características como shaders, buffers de vértices y arrays de vértices.

Gestión de Shaders y Programas

- El proyecto implementa un pipeline programable mediante el uso de shaders GLSL (OpenGL Shading Language). Se cargan múltiples programas de shaders para diferentes propósitos:
- lightingShader: Es el programa principal, cargado desde Shader/lighting.vs (Vertex Shader) y Shader/lighting.frag (Fragment Shader). Este shader complejo gestiona todos los cálculos de iluminación de la escena.
- lampShader: Un shader más simple usado para renderizar las representaciones visuales (cubos) de las luces puntuales, dándoles un color sólido.
- skyboxshader: Un shader específico para renderizar el fondo del cielo (Skybox), que utiliza una técnica de mapeo de cubos (Cubemap).
- La activación de un programa se realiza mediante la llamada lightingShader.Use() antes de enviar los uniformes y dibujar los objetos correspondientes.

Modelos, Buffers y Atributos

La geometría de la escena se gestiona de dos maneras:

- **Modelos Complejos:** Se utiliza una clase personalizada Model para cargar archivos de geometría (ej. Models/museo.obj, Models/cuerpo.obj). Esta clase (definida en Model.h, no visible pero inferida) es responsable de crear internamente los Vertex Array Objects (VAO), Vertex Buffer Objects (VBO) y Element Buffer Objects (EBO) necesarios para almacenar la geometría del modelo y enviarla eficientemente a la GPU.
- **Geometría Primitiva:** Para el Skybox y las lámparas, la geometría se define explícitamente en el código (ej. skyboxVertices[]). Se configuran manualmente los VAO y VBO (glGenVertexArrays, glGenBuffers, glBufferData) y se especifican los atributos de vértice (glVertexAttribPointer, glEnableVertexAttribArray) para indicar a OpenGL cómo interpretar los datos (posición, normales, etc.).

Bucle de Renderizado (Game Loop)

- El while (!glfwWindowShouldClose(window)) es el corazón del programa. En cada fotograma, realiza las siguientes operaciones de OpenGL:
- **Limpieza de Buffers:** Se limpian los buffers de color y profundidad (glClearColor, glClear) para preparar el siguiente fotograma.
- **Habilitación de Pruebas:** Se activa el test de profundidad (glEnable(GL_DEPTH_TEST)) para asegurar que los objetos se dibujen en el orden correcto de oclusión (lo que está más cerca tapa a lo que está más lejos).
- **Gestión de Uniforms:** Antes de dibujar, se envían datos cruciales (uniforms) a los shaders.

Transformaciones (Matrices MVP)

- El sistema de coordenadas se gestiona mediante la biblioteca GLM (OpenGL Mathematics). En cada fotograma, se calculan y envían a los shaders las tres matrices principales:
- **Proyección (Projection):** Se calcula una sola vez usando glm::perspective(). Define el tipo de cámara (perspectiva) y el campo de visión.

- Vista (View): Se obtiene de la clase `camera.GetViewMatrix()`. Esta matriz se actualiza en `DoMovement()` y `MouseCallback()` para simular el movimiento y la orientación de la cámara del observador.
- Modelo (Model): Esta es la matriz más dinámica. Se recalcula para cada objeto (o parte de objeto) a dibujar. Las animaciones (como las de la momia y el reloj en la función `Animation()`) funcionan actualizando variables (`mummyLegRot`, `tapaPosX`, `anguloPendulo`) que luego se usan para crear matrices de modelo únicas con `glm::translate`, `glm::rotate` y `glm::scale`. Esta matriz se envía al shader con `glUniformMatrix4fv(modelLoc, ...)` justo antes de llamar a `TuModelo.Draw()`.

Sistema de Iluminación (Uniforms)

- El `lightingShader` implementa un modelo de iluminación complejo (probablemente Phong) que recibe múltiples fuentes de luz a través de uniforms:
- Luz Direccional (`dirLight`): Una luz global que simula el sol.
- Luces Puntuales (`pointLights`): Un array de `NR_POINT_LIGHTS` (17) luces. El código define sus posiciones (`pointLightPositions`) y gestiona sus colores (`pointLightColors`). La tecla `SPACE` activa o desactiva estas luces cambiando su color entre `COLOR_YELLOW_ON` y `COLOR_OFF`.
- Foco (`spotLight`): Una luz de linterna que sigue la posición y dirección de la cámara.

Técnicas de Dibujado

- Dibujo de Modelos: La mayoría de los objetos (`SalasMuseo`, `MummyBody`, reloj, etc.) se dibujan llamando a su función `TuModelo.Draw(lightingShader)`. Esta función interna se encarga de vincular el VAO del modelo y ejecutar la llamada `glDrawElements` o `glDrawArrays`.
- Dibujo del Skybox: Se utiliza una técnica especial al final del bucle. Se deshabilita la escritura de profundidad (`glDepthFunc(GL_LEQUAL)`) y se usa el `skyboxshader` junto con una textura Cubemap (`GL_TEXTURE_CUBE_MAP`) para dibujar un cubo que siempre aparece en el fondo, creando la ilusión de un entorno infinito.

Software de Modelado y Edición 3D

Blender: Se empleó como la suite de creación 3D principal. Sus funciones críticas en el proyecto incluyeron:

- Modelado Poligonal: Creación de las mallas (meshes) de alta y baja densidad para los entornos, objetos y personajes.
- Mapeado UV (UV Unwrapping): Desplegado de las coordenadas de las mallas 3D a un espacio 2D para la correcta proyección de texturas.
- Texturizado y Materiales: Asignación y configuración de los mapas de textura (difusos, normales, etc.) y propiedades de materiales para definir la apariencia visual de los activos.

Herramientas Asistidas por IA para Generación de Activos

Para optimizar los tiempos de producción, se utilizaron plataformas web basadas en inteligencia artificial para la generación rápida de activos.

- imgto3d.ai: Esta plataforma se utilizó para la generación de modelos 3D. Su función fue convertir imágenes de referencia 2D en mallas 3D base, agilizando la creación de objetos específicos para las salas del entorno.
- Tripo AI Studio: Este servicio web se empleó para el proceso de texturizado asistido. Permitió la generación y aplicación rápida de materiales y texturas a los modelos 3D, facilitando la iteración y finalización de la apariencia visual de los activos.

Cronograma detallado de actividades

Fase	Actividad	Semanas	Fechas
1. Planeación	Definición del tema y recolección de referencias visuales (PDF).	Semana 1	06 Oct - 12 Oct
	Configuración del entorno (GLFW, GLEW, SOIL2) y estructura base del código.	Semana 1	06 Oct - 12 Oct
2. Escenario	Modelado/Búsqueda de assets (Sala, Sarcófago, Momia, Perro).	Semana 2	13 Oct - 19 Oct
	Implementación del Skybox (Cubemap) y carga de texturas.	Semana 2	13 Oct - 19 Oct
3. Iluminación	Implementación de Shaders (lighting.vs, lamp.vs).	Semana 3	20 Oct - 26 Oct
	Configuración de fuentes de luz (Direccional, Point Lights, SpotLight).	Semana 3	20 Oct - 26 Oct
4. Lógica (Perro)	Desarrollo de modelado jerárquico del perro (Torso -> Patas/Cola).	Semana 4	27 Oct - 02 Nov
	Programación de la trayectoria rectangular (Máquina de estados 0-3).	Semana 4	27 Oct - 02 Nov
5. Lógica (Momia)	Programación de interacción con teclado (Input M, N).	Semana 5	03 Nov - 09 Nov
	Implementación de la cinemática del Sarcófago (Apertura/Levitación).	Semana 5	03 Nov - 09 Nov
6. Refinamiento	Integración de interpolación de escala y movimiento de la momia.	Semana 6	10 Nov - 16 Nov
	Ajuste de velocidades (deltaTime) y corrección de límites de animación.	Semana 6	10 Nov - 16 Nov
7. Entrega	Pruebas finales de integración, limpieza de código y comentarios.	Semana 7	17 Nov - 24 Nov
	Documentación y elaboración del Manual Técnico.	Semana 7	17 Nov - 24 Nov

Estimación de costo aproximado

La siguiente estimación presupuestal desglosa los costos asociados al desarrollo del sistema gráfico "Museo Egipcio & Louvre". El cálculo se basa en una dedicación estimada de 140 horas hombre (promedio de 20 horas semanales durante 7 semanas) y tarifas de mercado estándar para un perfil de Programador Gráfico Junior.

Concepto	Descripción	Costo Estimado (MXN)
Recursos Humanos	Desarrollo, Diseño Lógico e Integración	\$35,000.00
Infraestructura	Amortización de Hardware (PC de Desarrollo)	\$2,500.00
Licenciamiento	Software y Herramientas (Visual Studio, etc.)	\$0.00*
Gastos Operativos	Electricidad e Internet (Prorrateado)	\$1,200.00
TOTAL	Costo Total del Proyecto	\$38,700.00

Dado que el proyecto fue realizado por una sola persona, el desarrollador asumió múltiples roles técnicos durante el ciclo de vida del software. El costo se calcula en base a una tarifa por hora de \$250.00 MXN.

- Gestión y Planeación (10 Horas): Definición de alcances, búsqueda de referencias visuales (Museo del Louvre, Sala Egipcia) y cronograma.
- Programación Gráfica - C++/OpenGL (80 Horas): Implementación del *pipeline* de renderizado, shaders, iluminación (Phong/Blinn-Phong), skybox y configuración de bibliotecas (GLEW, GLFW, GLM).
- Lógica y Animación (30 Horas): Desarrollo de máquinas de estados para la momia, cinemática del sarcófago y modelado jerárquico del perro.
- Integración y QA (20 Horas): Carga de modelos OBJ, mapeo de texturas, depuración de errores y pruebas de rendimiento.

Cálculo: 140 horas \times \$250.00 MXN/hora = \$35,000.00 MXN

Infraestructura y Herramientas

- Equipo de Cómputo: Se considera la depreciación de una estación de trabajo de gama media (CPU i5/i7, GPU dedicada, 16GB RAM) durante los dos meses de uso intensivo.

- *Valor estimado del equipo:* \$25,000 MXN.
 - *Depreciación imputada al proyecto (10%):* \$2,500.00 MXN.
- Software Utilizado:
 - Microsoft Visual Studio Community 2022 (Licencia Gratuita).
 - Bibliotecas Open Source: OpenGL, GLFW, GLEW, SOIL2, GLM (Licencias MIT/Zlib).
 - Blender (para revisión de modelos 3D - Open Source).

Documentación del aprendizaje autónomo de herramientas nuevas

311243563

Para poder llevar a cabo el proyecto de la galería 3D, tuve que enfrentarme a un proceso de aprendizaje autónomo en varias herramientas que iban desde la gestión de código hasta la creación de activos digitales.

Mi primera tarea fue establecer un entorno de desarrollo funcional y colaborativo. Para esto, tuve que aprender a utilizar Visual Studio como mi Entorno de Desarrollo Integrado (IDE) principal, enfocándome en la configuración de las librerías gráficas necesarias para el proyecto. Paralelamente, aprendí a manejar GitHub para el control de versiones. El desafío inicial no fue solo subir el código, sino entender cómo configurar correctamente el archivo .gitignore para un proyecto de Visual Studio. Al principio, subí archivos temporales de compilación que ralentizaron el repositorio. Resolví este problema investigando y aplicando una plantilla de .gitignore estándar, lo que me permitió mantener el control limpio y funcional del código fuente. Este aprendizaje fue fundamental para la gestión de las tareas de programación y ensamblaje de la escena.

La creación de los objetos de la galería (muebles, el sarcófago, etc.) fue un desafío técnico que abordé con un flujo de trabajo híbrido. Para generar una base de modelos y texturas rápidamente, utilicé dos herramientas de IA. Empecé a usar imgto3d.ai para convertir mis imágenes de referencia en mallas 3D básicas. Luego, utilicé studio.tripo3d.ai para obtener texturas rápidas de materiales genéricos. Estas herramientas me sirvieron para un prototipado muy rápido.

Sin embargo, los modelos generados por IA a menudo presentaban topologías complejas y el texturizado automático no me permitía el control de detalle necesario para el proyecto. Por esta razón, fue indispensable aprender a usar Blender para el refinamiento y, crucialmente, para el texturizado con imágenes personalizadas mediante UV Mapping.

El UV Mapping fue el concepto más difícil que tuve que dominar. Comprendí que esta técnica era esencialmente "desenvolver" mi modelo 3D en un plano 2D para que la imagen de textura se proyectara correctamente. Investigué múltiples tutoriales sobre cómo marcar las "costuras" (seams). Mi principal obstáculo era decidir dónde cortar el modelo para que la textura no se viera estirada o rota. Practiqué con modelos simples hasta que logré entender la lógica del unwrapping. Este proceso de aprendizaje me permitió importar los

modelos base de la IA, limpiarlos, y luego aplicar texturas personalizadas (como los jeroglíficos específicos o la dirección de la veta de la madera) con la precisión exacta que requería la estética final de la galería. Finalmente, este dominio me permitió llevar a cabo las tareas de "Modelado 3D de objetos" y "Creación y aplicación de texturas" de manera efectiva.

Conclusiones

La fase de desarrollo confirmó la viabilidad técnica del proyecto a pesar del cronograma acelerado. Se implementó una estrategia híbrida para la creación de activos 3D: se utilizó la inteligencia artificial (imgto3d.ai y studio.tripo3d.ai) para el prototipado rápido, y luego se empleó Blender para el indispensable refinamiento técnico, en especial el UV Mapping. Este enfoque permitió mantener la calidad visual y la precisión requerida para las texturas sin sacrificar la eficiencia. Además, la integración final en el entorno de Visual Studio, gestionada mediante GitHub, demostró la robustez de la arquitectura. El sistema implementado gestiona fluidamente la navegación del usuario y las animaciones clave (como el sarcófago y el cubo), probando que la solución es estable para futuras expansiones.

El impacto social del proyecto radica principalmente en la democratización del acceso a la cultura. Al ser una experiencia completamente virtual, se eliminan las barreras geográficas y económicas, haciendo que las exposiciones de alto valor cultural sean accesibles a una audiencia global sin necesidad de desplazamiento. La naturaleza interactiva e inmersiva de la galería potencia el engagement del usuario, traducándose en una experiencia educativa superior y una mayor retención de información en comparación con los formatos pasivos. Finalmente, el proyecto establece un modelo de referencia viable para la curación de contenido a distancia, fomentando la colaboración entre especialistas sin importar su ubicación.

Desde una perspectiva económica, el proyecto presenta una optimización de costos significativa. La eliminación de los gastos asociados a la infraestructura física incluyendo construcción, seguros, seguridad y mantenimiento minimiza el capital inicial y operativo. La dependencia de herramientas de bajo coste o código abierto (Blender, GitHub) y la aceleración del Time-to-Market gracias al flujo de trabajo con IA han maximizado la eficiencia de los recursos. Esto se traduce en un modelo de negocio altamente escalable, ya que el coste marginal de añadir nuevos objetos o salas virtuales es mínimo comparado con la expansión de un museo tradicional.

El proyecto de galería virtual se posiciona como una solución de bajo impacto ambiental. La eliminación de la infraestructura física significa que el proyecto no genera residuos de construcción y no requiere el uso intensivo de energía para la climatización, iluminación o seguridad de un edificio de exposición. Además, el acceso remoto a la galería contribuye

directamente a la mitigación de la huella de carbono generada por el transporte masivo de visitantes y de las propias obras de arte. En esencia, la implementación tecnológica promueve un uso más sostenible y eficiente de los recursos, al sustituir la necesidad de infraestructura material por una solución digital.

Bibliografía

1. Amazon. (s.f.). Sarcófago [Imagen]. Recuperado de <https://m.media-amazon.com/images/I/71PcWupl0RL.jpg>
Licencia: No especificada.
2. Free3D. (s.f.). Mummy [Modelo 3D]. Recuperado de <https://free3d.com/es/modelo-3d/mummy-7958.html>
Licencia: Descarga gratuita (Free3D).
3. Freepik. (s.f.). Planta verde en maceta roja [Vector premium]. Recuperado de 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
Licencia: Freepik Premium.
4. Freepik. (s.f.). Marco oro vintage [Vector premium]. Recuperado de https://www.freepik.es/vector-premium/marco-oro-vintage_2205924.htm
Licencia: Freepik Premium.
5. Freepik. (s.f.). Conjunto marco fotos iluminado realista [Vector gratis]. Recuperado de https://www.freepik.es/vector-gratis/conjunto-marco-fotos-iluminado-realista_20289082.htm
Licencia: Freepik Free.
6. Freepik. (s.f.). Maqueta realista exposición galería imágenes pared [Vector gratis]. Recuperado de https://www.freepik.es/vector-gratis/maqueta-realista-exposicion-galeria-imagenes-pared-ilustracion-vector-sofa-visitantes_33770595.htm
Licencia: Freepik Free.
7. Freepik. (s.f.). Luces estudio realistas diseño fondo vacío [Vector gratis]. Recuperado de https://www.freepik.es/vector-gratis/luces-estudio-realistas-diseno-fondo-vacio_10016499.htm
Licencia: Freepik Free.
8. Sketchfab. (s.f.). Pixel art voxel art person with a hat [Modelo 3D]. Recuperado de <https://sketchfab.com/3d-models/pixel-artvoxel-art-person-with-a-hat-13aa4bfb0d2f4573af0154dec3b2c269>
Licencia: Gratuito según plataforma Sketchfab.

9. Freepik. (s.f.). Columnas antiguas y modernas [Vector gratis]. Recuperado de https://www.freepik.es/vector-gratis/antiguas-duricas-cilindricas-ordenes-jonicas-modernas-columnas-cubicas-marmol_4393674.htm
Licencia: Freepik Free.
10. Freepik. (s.f.). Ilustración isométrica museo Louvre [Imagen IA premium]. Recuperado de https://www.freepik.es/imagen-ia-premium/ilustracion-isometrica-lowpoly-museo-louvre-su-iconica-piramide-vidrio_266441597.htm
Licencia: Freepik Premium.
11. Freepik. (s.f.). Fachada histórica Palacio Louvre [Imagen IA premium]. Recuperado de https://www.freepik.es/imagen-ia-premium/ilustracion-vectorial-fachada-historica-palacio-louvre_249038071.htm
Licencia: Freepik Premium.
12. Tripo3D. (s.f.). Glass pyramid with diamond grid facade [Modelo 3D]. Recuperado de <https://studio.tripo3d.ai/workspace/overview?project=2aae6996-e3d4-4b43-ba9a-9f65d0df70b6>
Licencia: Gratuito (Tripo3D).
13. Freepik. (s.f.). Fondo piedra mármol para diseño 3D [Vector premium]. Recuperado de https://www.freepik.es/vector-premium/fondo-piedra-marmol-usos-diseno-ilustracion-3d_10510617.htm
Licencia: Freepik Premium.
14. Vecteezy. (s.f.). Mona Lisa icono dibujos animados [Vector]. Recuperado de <https://es.vecteezy.com/arte-vectorial/52103629-mona-lisa-icono-dibujos-animados-estilo-icono-blanco-antecedentes>
Licencia: Gratuito con atribución (Vecteezy).
15. Pinterest. (s.f.). El Grito [Imagen]. Recuperado de <https://mx.pinterest.com/pin/7318418133384987/>
Licencia: Uso limitado, ver condiciones (Pinterest).
16. Freepik. (s.f.). Noche estrellada Van Gogh [Vector gratis]. Recuperado de https://www.freepik.es/vector-gratis/ilustracion-pintura-van-gogh-diseno-plano_29725492.htm
Licencia: Freepik Free.
17. Freepik. (s.f.). Escritura sumeria asiria cuneiforme acadia [Vector gratis]. Recuperado de https://www.freepik.es/vector-gratis/escritura-sumeria-asiria-cuneiforme-acadia_15754391.htm

Licencia: Freepik Free.

18. WikiArt. (s.f.). Cash Register (1917) [Pintura]. Recuperado de <https://www.wikiart.org/es/amadeo-de-souza-cardoso/cash-register-1917>

Licencia: Uso educativo permitido (WikiArt).

19. WikiArt. (s.f.). La Composizione TA (1916) [Pintura]. Recuperado de <https://www.wikiart.org/es/carlo-carra/la-composizione-ta-1916>

Licencia: Uso educativo permitido (WikiArt).

20. WikiArt. (s.f.). Silence (1915) [Pintura]. Recuperado de <https://www.wikiart.org/es/georges-ribemont-dessaigues/silence-1915>

Licencia: Uso educativo permitido (WikiArt).

21. WikiArt. (s.f.). Balance [Pintura]. Recuperado de <https://www.wikiart.org/es/francispicabia/balance>

Licencia: Uso educativo permitido (WikiArt).

22. WikiArt. (s.f.). Ambiguous Figures [Pintura]. Recuperado de <https://www.wikiart.org/es/max-ernst/ambiguous-figures-1-copper-plate-1-zinc-plate-1-rubber-cloth>

Licencia: Uso educativo permitido (WikiArt).

23. WikiArt. (s.f.). Bicycle Wheel (1913) [Pintura]. Recuperado de <https://www.wikiart.org/es/marcel-duchamp/bicycle-wheel-1913>

Licencia: Uso educativo permitido (WikiArt).

24. WikiArt. (s.f.). The Hill [Pintura]. Recuperado de <https://www.wikiart.org/es/manray/the-hill>

Licencia: Uso educativo permitido (WikiArt).

25. Freepik. (s.f.). Estatua Nefertiti reina mujer faraón [Vector gratis]. Recuperado de https://www.freepik.es/vector-gratis/estatua-nefertiti-reina-mujer-faraon-antiguo-egipto-ilustracion-vectorial-dibujos-animados_11433881.htm

Licencia: Freepik Free.

26. Free3D. (s.f.). Rectangular grass patch [Modelo 3D]. Recuperado de <https://free3d.com/es/modelo-3d/-rectangular-grass-patch--205749.html>

Licencia: Descarga gratuita (Free3D).

27. Free3D. (s.f.). Small glass pyramid [Modelo 3D]. Recuperado de <https://free3d.com/es/modelo-3d/small-glas-pyramid-9315.html>

Licencia: Descarga gratuita (Free3D).

28. TurboSquid. (s.f.). Jackal god of the Egyptian [Modelo 3D]. Recuperado de <https://www.turbosquid.com/es/3d-models/3d-jackal-god-of-the-egyptian-3d-printable-1778764>
Licencia: Puede requerir compra (TurboSquid).
29. Freepik. (s.f.). Ilustración isométrica interior museo exposición pintura [Vector gratis]. Recuperado de https://www.freepik.es/vector-gratis/ilustracion-isometrica-color-interior-museo-exposicion-pintura_3791816.htm
Licencia: Freepik Free.
30. Freepik. (s.f.). Marcos chimenea y lámpara [Vector premium]. Recuperado de https://www.freepik.es/vector-premium/marcos-chimenea-chimenea-lampara_242532203.htm
Licencia: Freepik Premium.
31. Imgto3D. (s.f.). Conversión de imágenes a modelos 3D [Sitio web]. Recuperado de <https://www.imgto3d.ai/es>
32. Tripo3D. (s.f.). Generador de modelos 3D a partir de imágenes [Sitio web]. Recuperado de <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.

Repositorio de GitHub:

[SandraLaparra/311243563_ProyectoFinalTeoria_GP05: Proyecto Final de la materia Computación Gráfica Teoría](#)

En este link de Drive se descarga la carpeta de “Modelos” en formato .obj:

https://drive.google.com/drive/folders/1ykp-Z47fhd6Pcpj6ks_qpUEx2K-0pEJ?usp=drive_link