



UNIVERSIDAD NACIONAL
AUTÓNOMA DE MÉXICO



FACULTAD DE INGENIERÍA

COMPUTACIÓN GRÁFICA E INTERACCIÓN
HUMANO-COMPUTADORA

GRUPO: 5

MANUAL TÉCNICO

NO. CUENTA: 319162538

PROFESOR: CARLOS ALDAIR ROMAN
BALBUENA

SEMESTRE: 2026 - 1

Índice

Introducción	1
Estado del arte actual del proyecto	1
Objetivos	2
Imágenes de referencia.....	2
Diagrama de Gantt.....	5
Diagrama de flujo del software.....	6
Alcance del proyecto.....	7
Tecnologías usadas.....	7
Limitantes	9
Metodología de software aplicada	9
Diccionario de funciones y variables	10
Funciones principales	10
Variables	14
Análisis de costos del proyecto.....	16
Desglose de costos del proyecto.....	16
Costos fijos	17
Costos variables	17
Costo total del proyecto y precio con ganancia.....	17
Impuestos y precio final al cliente.....	18
Observaciones y justificación	18
Conclusiones	19
Referencias	19

Introducción

El presente proyecto tiene como objetivo desarrollar un recorrido virtual de la Torre de los Teen Titans mediante la implementación de técnicas de programación gráfica en OpenGL. A lo largo del proceso se construyó un entorno tridimensional navegable que integra modelos, texturas, iluminación y animaciones, buscando recrear de manera coherente el espacio interior mostrado en la serie. Para mantener la fidelidad visual, se emplearon imágenes de referencia que guiaron la composición del escenario y la disposición de los objetos dentro de la escena.

La creación del ambiente involucró un proceso completo que abarca desde la elaboración de modelos basados en estructuras modulares hasta la integración de materiales, transformaciones y texturas. Se incluyeron múltiples objetos como muebles, lámparas, electrodomésticos y elementos decorativos, así como animaciones programadas, las cuales aportan dinamismo y una experiencia de exploración más atractiva. La cámara en primera persona y el control libre del usuario permiten recorrer el entorno de forma interactiva y en tiempo real.

Este proyecto representa una oportunidad para aplicar los conocimientos adquiridos en el curso de forma práctica, abarcando temas como renderizado, manejo de buffers, iluminación, transformaciones en 3D y control de escena. Además, permite fortalecer habilidades relacionadas con la lógica matemática, la organización del código y la documentación técnica. El resultado final demuestra la capacidad de combinar creatividad y programación para construir un entorno virtual funcional, utilizando herramientas modernas de desarrollo y un enfoque orientado a la construcción de experiencias visuales interactivas.

Estado del arte actual del proyecto

Los recorridos virtuales han evolucionado de ser simples visualizaciones estáticas para convertirse en experiencias inmersivas en tiempo real, impulsadas por avances en gráficos 3D, motores de renderizado y hardware especializado, en su origen, este tipo de recorridos se utilizaban principalmente en arquitectura para mostrar maquetas digitales antes de su construcción; sin embargo, en la actualidad su alcance se ha ampliado hacia áreas como el entretenimiento, educación, museografía, simulaciones de entrenamiento, videojuegos y realidad virtual.

Las tendencias actuales se centran en la creación de entornos interactivos que permitan al usuario explorar espacios con libertad, percibir profundidad, iluminación dinámica y animaciones ambientales, lo que genera una experiencia mucho más cercana a la realidad, tecnologías como renderizado en tiempo real, iluminación física (PBR), sombras dinámicas, navegación estilo FPS y modelado modular se

han convertido en estándares dentro de proyectos de visualización profesional. Asimismo, frameworks y motores como Unreal Engine, Unity y OpenGL han facilitado el desarrollo de recorridos virtuales especializados, brindando herramientas eficientes para implementaciones tanto artísticas como técnicas.

En este contexto, el proyecto presentado “*una recreación interactiva del interior de la Torre de los Teen Titans*” se fundamenta en dichos principios, aplicando técnicas contemporáneas de renderizado, animación y control de cámara para simular un recorrido realista dentro de un ambiente ficticio. El uso de modelado modular mediante cubos texturizados, iluminación direccional y puntual, shaders personalizados y animaciones en tiempo real coloca este trabajo dentro de la línea de desarrollo utilizada actualmente en prototipos de videojuegos y experiencias virtuales, además, la incorporación de imágenes de referencia permite mantener consistencia visual y coherencia estética con el material original, siguiendo prácticas profesionales de diseño digital.

Este proyecto no solo busca mostrar una reconstrucción visual del espacio, sino también aplicar conceptos vigentes en el estado del arte de la simulación 3D, integrando interactividad, navegación libre y elementos animados que enriquecen la experiencia del usuario, con ello, se alinea a los estándares actuales de los recorridos virtuales, demostrando cómo estas tecnologías pueden emplearse para recrear espacios tanto reales como ficticios con un enfoque moderno, técnico y funcional.

Objetivos

- Aplicar los conocimientos del curso para desarrollar una recreación 3D de la fachada y dos cuartos interiores de la Torre de la serie *Teen Titans Go*, utilizando OpenGL para construir las escenas. Se debe asegurar que la representación virtual sea lo más fiel posible a las imágenes de referencia, incluyendo la ambientación correspondiente a cada espacio.
- Integrar un flujo de trabajo de modelado externo con Blender y con la renderización en OpenGL, modelando un mínimo de 10 objetos, 5 en cada cuarto usando OpenGL y blender.
- Implementar un sistema de animaciones coherentes y generadas por código para dar vida a la escena, desarrollando 4 animaciones requeridas y 1 extra.

Imágenes de referencia

A continuación, se presentan las referencias visuales que sirvieron como guía base para asegurar que la ambientación y la estructura arquitectónica fueran lo más fieles posible a la serie original.

Referencia de la fachada

El exterior del proyecto se centra en la estructura en forma de "T" característica del edificio, situada sobre una colina, esta imagen sirvió de base para el modelado de la arquitectura externa y el entorno circundante.

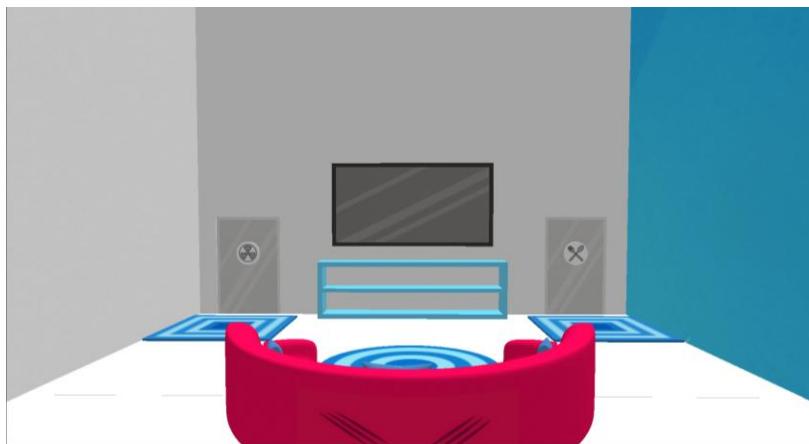


Referencias de los cuartos a recrear

Para el diseño de interiores, el alcance del proyecto se delimitó a dos habitaciones principales, recreando la disposición del mobiliario y la iluminación específica de cada zona.

Sala de estar





Objetos recreados:

1. Mesa del televisor
2. Televisor
3. Alfombras
4. Cojines
5. Sillon

Comedor o cocina



**Objetos recreados:**

1. Asientos
2. Mesa
3. Estufa
4. Refrigerador
5. Mueble de cocina

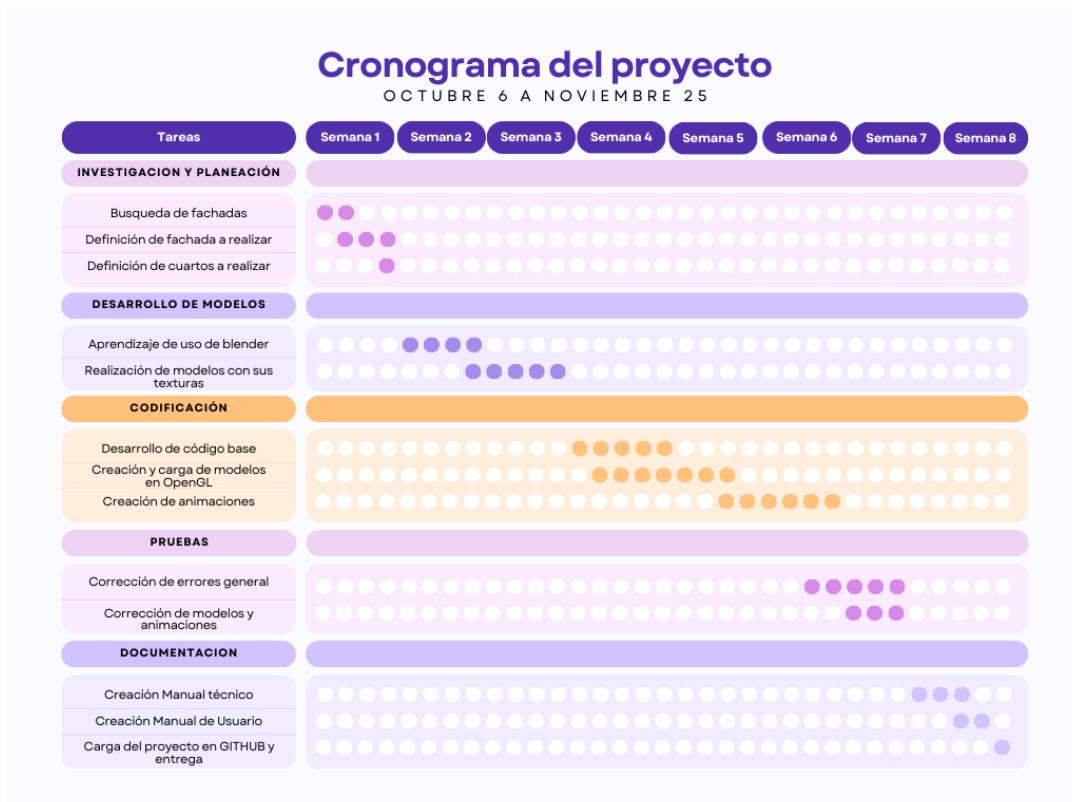
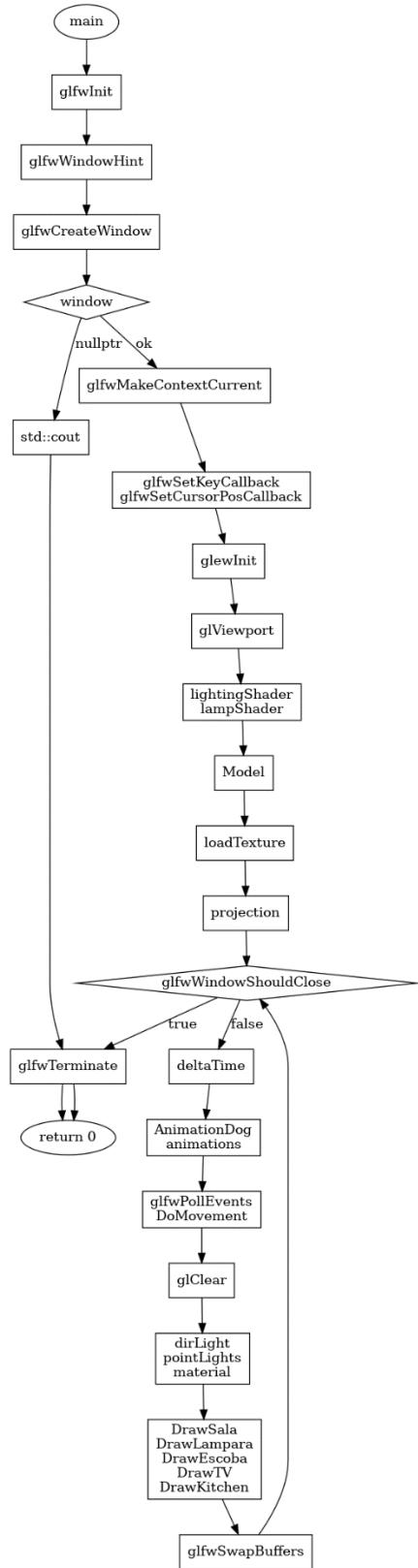
Diagrama de Gantt

Diagrama de flujo del software



Alcance del proyecto

El presente proyecto tiene como alcance el desarrollo de una aplicación gráfica funcional en OpenGL que recrea una ambientación 3D basada en la serie Teen Titans Go, delimitándose a lo siguiente:

1. **Desarrollo del Entorno:** La creación de una escena 3D que representa fielmente la fachada exterior y dos cuartos interiores de la Torre de los Teen Titans, basándose en imágenes de referencia.
2. **Modelado y Texturizado:** La creación e integración de al menos 10 objetos 3D únicos y evaluables modelados en OpenGL, junto con Blender y texturizados adecuadamente.
3. **Implementación de Animaciones:** La programación de 4 animaciones generadas por código que sean coherentes con el entorno.
4. **Software Funcional:** La entrega de un archivo ejecutable funcional construido sobre el código base proporcionado por el profesor.
5. **Cámara y vista:** El movimiento de cámara fue implementado de forma fluida y en tiempo real, ofreciendo libertad para explorar el espacio desde diferentes ángulos y perspectivas, mejorando la navegación y la interacción del usuario con el recorrido.

Tecnologías usadas

En el desarrollo del proyecto se emplearon diversas tecnologías y herramientas que permitieron crear un recorrido virtual interactivo con modelos 3D, animaciones y un entorno visual coherente. A continuación, se presenta una descripción de cada una y su función dentro del proceso de producción:

OpenGL (Open Graphics Library)

OpenGL fue la tecnología principal del proyecto, utilizada para implementar el renderizado en tiempo real, el manejo de buffers, cámara, iluminación dinámica y procesamiento de shaders. Su enfoque de bajo nivel permitió un control detallado sobre la pipeline gráfica, lo cual fue esencial para la creación de animaciones, efectos de luz y estructuras tridimensionales.

Modelado y texturizado en blender

Blender se empleó para crear las referencias visuales y practicar el modelado de la estructura interior. Aunque el proyecto final usa modelos construidos modularmente con cubos en OpenGL, Blender funcionó como herramienta de apoyo para comprender proporciones, volúmenes y composición espacial basándose en videos y tutoriales. Sí cuenta como herramienta técnica del proyecto.

ChatGPT (Asistencia técnica, documentación y generación de texturas)

ChatGPT se utilizó como herramienta auxiliar para:

- Asesoría en corrección de shaders
- Depuración de iluminación
- Optimización de funciones y animaciones
- Generación o edición de texturas simples
- Explicación de conceptos de OpenGL

Su función se enmarca dentro de las herramientas modernas de apoyo al desarrollo, similares a un IDE inteligente o asistente técnico.



Material de referencia (Imágenes y videos de YouTube)

El uso de videos e imágenes de referencia para modelar la Torre de los Teen Titans es una práctica estándar en entornos 3D y de animación. No es una “tecnología”, pero sí es una herramienta metodológica que respalda la coherencia visual del proyecto.



Limitantes

A pesar de los logros alcanzados, la implementación de este se enfrentaron ciertas limitaciones que afectaron en cierta medida el desarrollo hasta el final:

1. **Recursos técnicos:** La capacidad de procesamiento y memoria disponible en las estaciones de trabajo condicionó la complejidad de las texturas y el número de luces dinámicas que se pudieron implementar sin afectar el rendimiento en tiempo real.
2. **Herramientas y entorno:** Aunque Blender y OpenGL ofrecieron amplias posibilidades, algunas funcionalidades avanzadas como shaders personalizados o efectos visuales complejos no se exploraron a fondo debido a la curva de aprendizaje y el tiempo disponible.
3. **Limitaciones de Animación:** Todas las animaciones implementadas deben ser generadas exclusivamente por código. No se permite el uso de animaciones pre-renderizadas o importadas desde Blender.
4. **Escalabilidad:** El proyecto se limitó y ajusto a los requerimientos establecidos en la evaluación, por lo que añadir o implementar más conceptos provocaban fallo o errores en la programación y mucha exigencia gráfica para el computador.

Metodología de software aplicada

Para la gestión de este proyecto se optó por el uso de la metodología de desarrollo incremental, involucrando planificación inicial de un modelo en cascada y flexibilidad de construcción por etapas. Entre las etapas tenemos las siguientes:

- **Fase 1: Investigación y Planeación (Semanas 1-2)** Esta fase inicial definió el alcance y los cimientos conceptuales del proyecto. Las tareas incluyeron la búsqueda de referencias visuales, la definición de la fachada de la Torre de los Teen Titans y la selección de los dos cuartos a recrear.

- **Fase 2: Desarrollo de Modelos (Semanas 2-4)** Corresponde al primer incremento tangible del proyecto: la creación de los assets (activos) 3D. Esta etapa se centró en el aprendizaje de Blender y la realización y texturizado de todos los modelos requeridos para la escena.
- **Fase 3: Codificación (Semanas 4-6)** En esta fase se tomó el incremento de "Modelos" y se le dio vida en el entorno gráfico. Comprendió el desarrollo sobre el código base, la implementación de la lógica para la carga de los modelos en OpenGL y la creación de las animaciones programadas.
- **Fase 4: Pruebas (Semanas 6-7)** Esta fase se dedicó al refinamiento y depuración del incremento de "Codificación". Se enfocó en la corrección de errores generales de la aplicación, así como en los ajustes visuales y funcionales de los modelos y las animaciones.
- **Fase 5: Documentación y Entrega (Semanas 7-8)** El incremento final consistió en empaquetar el producto para su entrega. Incluyó la creación de los manuales técnico y de usuario, y la preparación final del repositorio de GitHub para la entrega del proyecto.

Este modelo incremental aseguró un flujo de trabajo ordenado, donde cada etapa (Modelado, Código, Pruebas) se construyó sobre el éxito de la anterior, facilitando la detección de problemas y garantizando que todos los componentes estuvieran listos para la fecha de entrega.

Diccionario de funciones y variables

Funciones principales

int main()

Parámetros: Ninguno.

Descripción: Punto de entrada del programa. Inicializa GLFW/GLEW, crea la ventana, configura la cámara y las luces, carga modelos y texturas, define animaciones y ejecuta el bucle principal de render hasta que se cierra la ventana.

void MouseCallback(GLFWwindow* window, double xPos, double yPos)

Parámetros:

- **window (GLFWwindow*)**: ventana asociada al evento.
- **xPos (double)**: posición X del mouse.

- yPos (double): posición Y del mouse.

Descripción:

Actualiza la orientación de la cámara usando el movimiento del ratón (yaw/pitch), usando lastX, lastY y firstMouse.

void KeyCallback(GLFWwindow* window, int key, int scancode, int action, int mode)

Parámetros:

- window (GLFWwindow*): ventana que recibe el evento.
- key (int): código de la tecla.
- scancode (int): scancode de la tecla.
- action (int): acción (GLFW_PRESS, GLFW_RELEASE, etc.).
- mode (int): modificadores (Shift, Ctrl, etc.).

Descripción: Maneja la entrada de teclado. Actualiza el arreglo global keys[], activa/desactiva animaciones (reloj, escoba, disco, perro, lámpara) y controla flags como active, gClockAnimating, AnimDog, etc.

void DoMovement()

Parámetros: Ninguno.

Descripción: Lee el arreglo global keys[] y mueve la cámara (camera.ProcessKeyboard(...)) según WASD/flechas, usando deltaTime para hacer el movimiento independiente del framerate.

void AnimationDog()

Parámetros: Ninguno.

Descripción: Controla el estado de animación del perro usando variables globales (dogPos, dogRot, FLegs, RLeds, head, tail, earL, earR, dogAnim, step, AnimDog). Cambia la posición y rotaciones del perro para simular caminar, mover cola, cabeza, orejas, etc.

void DrawMueble(Shader& shader, GLint modelLoc, GLuint VAO, const glm::vec3& pos, const glm::vec3& rot)

Parámetros:

- shader (Shader&): shader de iluminación a usar.
- modelLoc (GLint): location del uniform model en el shader.
- VAO (GLuint): VAO del cubo base.
- pos (const glm::vec3&): posición del mueble.
- rot (const glm::vec3&): rotación (en grados) del mueble.

Descripción: Construye y dibuja el mueble de la TV usando varias cajas escaladas/trasladadas con la textura MuebleTV.

```
void DrawMesa(Shader& shader, GLint modelLoc, GLuint VAO, const
glm::vec3& pos, const glm::vec3& rot)
```

Parámetros:

- shader (Shader&): shader de iluminación a usar.
- modelLoc (GLint): location del uniform model en el shader.
- VAO (GLuint): VAO del cubo base.
- pos (const glm::vec3&): posición del mueble.
- rot (const glm::vec3&): rotación (en grados) del mueble.

Descripción: Dibuja la mesa del comedor/sala usando el VAO de cubo y textura Mesa para tablero y patas.

```
void DrawReloj(Shader& shader, GLint modelLoc, GLuint VAO,const
glm::vec3& pos, const glm::vec3& rot,float animTime,float minutePeriodSec,
float hourPeriodSec);
```

Parámetros:

- shader (Shader&): shader de iluminación.
- modelLoc (GLint): uniform model.
- VAO (GLuint): VAO del cubo.
- pos (const glm::vec3&): posición base del reloj.
- rot (const glm::vec3&): rotación base.
- animTime (float): tiempo acumulado de animación (por ej. gClockTime).
- minutePeriodSec (float): periodo en segundos de la manecilla de minutos.
- hourPeriodSec (float): periodo en segundos de la manecilla de horas.

Descripción: Dibuja el reloj (base y manecillas) usando VAO de cubo, texturas BaseReloj y Manecilla, calculando la rotación de las manecillas en función de animTime.

```
void DrawEscoba(Shader& shader, GLint modelLoc, GLuint VAO,const  
glm::vec3& pos, const glm::vec3& rot,float anguloCaida);
```

Parámetros:

- shader (Shader&): shader de iluminación.
- modelLoc (GLint): uniform model.
- VAO (GLuint): VAO del cubo.pos (const glm::vec3&): posición base del reloj.
- rot (const glm::vec3&): rotación base.
- anguloCaida (float): ángulo de inclinación/caída de la escoba.

Descripción: Dibuja la escoba (mango + cerdas) usando texturas MuebleTV/CerdaEscoba y aplica una rotación adicional para simular la animación de caerse.

```
void DrawSillaSimple(Shader& shader, GLint modelLoc, GLuint VAO,const  
glm::vec3& pos, const glm::vec3& rot);
```

Parámetros:

- shader (Shader&): shader de iluminación a usar.
- modelLoc (GLint): location del uniform model en el shader.
- VAO (GLuint): VAO del cubo base.
- pos (const glm::vec3&): posición del mueble.
- rot (const glm::vec3&): rotación (en grados) del mueble.

Descripción: Dibuja una silla sencilla (asiento, patas, respaldo) con el VAO de cubo y texturas de mueble.

```
void DrawLampara(Shader& shader, Shader& lampShader,GLint modelLoc,  
GLuint VAO,const glm::vec3& pos, const glm::vec3& rot,const glm::mat4&  
view, const glm::mat4& projection);
```

Parámetros:

- shader (Shader&): shader de iluminación general (base + tubo).
- lampShader (Shader&): shader simple para las pantallas/focos.
- modelLoc (GLint): uniform model (para el shader activo en cada momento).
- VAO (GLuint): VAO del cubo.
- pos (const glm::vec3&): posición de la lámpara.
- rot (const glm::vec3&): rotación base.
- view (const glm::mat4&): matriz de vista actual (para lampShader).

- projection (const glm::mat4&): matriz de proyección actual (para lampShader).

Descripción: Dibuja la lámpara de pie: base, tubo (con shader) y las pantallas/focos (con lampShader), cambiando de shader a mitad de la función.

void DrawDoorHinged(Shader& shader, GLint modelLoc, Model& model, const Animation& d);

Parámetros:

- shader (Shader&): shader de iluminación.
- modelLoc (GLint): uniform model.model (Model&): modelo 3D de la puerta.
- d (const Animation&): datos de animación de la puerta (ángulo, pivote, etc.).

Descripción: Aplica una transformación de puerta con bisagra (pivot en hingeLocal) al Model y lo dibuja según el estado de Animation (usado para puerta del comedor).

void DrawDiscoBall(Shader& shader, GLint modelLoc, Model& esfera, Model& tubo, const Animation& anim);

Parámetros:

- shader (Shader&): shader de iluminación.
- modelLoc (GLint): uniform model.esfera (Model&): modelo de la esfera del disco.
- tubo (Model&): modelo del tubo/cadena.
- anim (const Animation&): animación (posición, rotación, currentLength, etc.).

Descripción: Dibuja la disco ball del techo usando dos modelos (tubo + esfera) y usa los campos de Animation para la extensión del tubo y rotación de la esfera.

VARIABLES

Variable	Descripción	Valores o variables asociadas
WIDTH, HEIGHT	Dimensiones iniciales de la ventana OpenGL	1920, 1080
SCREEN_WIDTH, SCREEN_HEIGHT	Tamaño real del framebuffer	Establecido por glfwGetFramebufferSize()
camera	Objeto cámara que controla posición y orientación	Inicializado en glm::vec3(2.0f, 47.0f, 40.0f)

lastX, lastY	Última posición registrada del mouse en X/Y	WIDTH / 2, HEIGHT / 2
firstMouse	Bandera para detectar el primer movimiento del mouse	true / false
keys[1024]	Arreglo de estados de teclas para entrada	true (presionada), false (liberada)
deltaTime	Tiempo entre el frame actual y el anterior	currentFrame - lastFrame
lastFrame	Tiempo del último frame renderizado	Actualizado cada iteración
gClockTime	Tiempo acumulado para animar el reloj	Incrementado usando deltaTime
gClockAnimating	Indica si el reloj está animando sus manecillas	true / false
gDoorComedor	Estructura Animation que controla la puerta del comedor	Contiene: angle, target, speed, hingeLocal, etc.
gEscoba	Estructura Animation para la escoba	Controla caída/rotación
gDisco	Estructura Animation para la disco ball	Contiene currentLength, maxLength, extendSpeed, etc.
active	Bandera que activa/desactiva la lámpara (pointLight[0])	true / false
pointLightPositions[4]	Posiciones de las point lights en la escena	[0] es la lámpara; las demás (0,0,0)
PantallaTV, MarcoTV, MuebleTV, Mesa, Manecilla, BaseReloj, CerdaEscoba	IDs de texturas cargadas	Generadas por loadTexture()
dogPos	Posición actual del perro	glm::vec3
dogRot	Rotación Y del perro	float
FLegs, RLeds	Rotación de patas delanteras y traseras	float
head, tail, earL, earR	Variables de animación de cabeza, cola y orejas	float controlado por AnimationDog()
dogAnim	Estado de animación del perro	int
step	Alternador para el ciclo de caminata del perro	true / false
AnimDog	Flag general para activar la animación del perro	true / false

rotDog	Rotación adicional del perro	float
VAO, VBO	Buffers del cubo base usado en varios modelos	Generados al inicio
lightPos	Posición base de una luz general (a veces no usada)	glm::vec3
dirLight (<i>uniform</i>)	Datos de la luz direccional (ambient, diffuse, specular)	Enviados con lightingShader.Use()
pointLights[4] (<i>uniform</i>)	Datos de las point lights de la escena	ambient, diffuse, specular, atenuación

Análisis de costos del proyecto

En esta sección se presenta una evaluación detallada de los recursos invertidos en el proyecto, incluyendo tiempo de desarrollo, materiales y gastos operativos, el objetivo es determinar el costo real de producción y estimar un precio de venta adecuado y este análisis permite visualizar el proyecto como un producto profesional, justificando tanto su valor como la inversión realizada.

Desglose de costos del proyecto

Concepto	Descripción	Horas Estimadas	Costo (MXN)
1. Programación OpenGL	Implementación de renderizado, shaders, cámara, iluminación, inputs, animaciones base.	40 h	\$5,600
2. Modelado 3D Modular	Creación de muebles, sala, objetos y estructuras usando cubos + transformaciones.	20 h	\$2,800
3. Integración de Animaciones	Perro, escoba, puerta, reloj, lámpara, disco ball.	20 h	\$2,800
4. Optimización	Ajustes de rendimiento, reducción de cálculos, mejoras visuales.	10 h	\$1,400
5. Pruebas y Depuración	Correcciones, errores por iluminación, clipping, cámara, texturas rotas.	15 h	\$2,100
6. Documentación Técnica	Diccionario de funciones, variables, diagramas, análisis, justificación.	10 h	\$1,400
7. Texturización básica	Integración de texturas de TV, muebles, mesa y escoba.	6 h	\$840
8. Ajustes finales de iluminación	Revisión de brillos, difusas, point lights, shading cartoon.	6 h	\$840

Total de horas directas: 127 hrs.

Costo por hora estimado: \$140 MXN/h.

Costos fijos

Concepto	Descripción	Costo (MXN)
Electricidad	120 h de trabajo × 150 W × tarifa CFE \$2.3 kWh	\$41
Depreciación de PC	PC de \$20,000 MXN / 36 meses → prorrataeado	\$185
Internet	Uso proporcional (1 mes / 30 días de trabajo)	\$80
Periféricos	Mouse + teclado (1,200 MXN a 30 meses)	\$40
Software	Todo open source → \$0	\$0

Subtotal de costos fijos: \$346 MXN

Costos variables

Concepto	Costo
Mano de obra	\$17,780
Uso de texturas o assets	\$0 (recursos propios)
Software adicional	\$0

Contingencias (10%)

- Se recomienda cubrir:
- Cambios de diseño
- Ajustes visuales solicitados
- Bugs inesperados
- Tiempo extra por integración

$$(17,780+346) \times 0.10 = 2,212 \text{ MXN}$$

Costo total del proyecto y precio con ganancia

$$\text{Costo total} = 17,780 + 346 + 2,212 = 20,338 \text{ MXN}$$

Margen de Ganancia sugerido: 30%

$$20,338 \times 0.30 = 6,101$$

Precio base de venta:

$$20,338 + 6,101 = 26,439 \text{ MXN}$$

Impuestos y precio final al cliente

Impuesto	Cálculo	Monto
IVA (16%)	$26,439 \times 0.16$	4,230 MXN
ISR (10% utilidad)	$6,101 \times 0.10$	610 MXN

$$26,439 + 4,230 + 610 = 31,279 \text{ MXN}$$

Precio final al cliente: 31,280 MXN

Observaciones y justificación

El proyecto se desarrolla con herramientas open source como OpenGL, GLFW, GLEW, Blender y Visual Studio Code, reduciendo costos de licencias y permitiendo mayor inversión en programación y optimización.

Complejidad técnica

El sistema incluye:

- Programación a bajo nivel con C++ y OpenGL
- Uso de shaders (vertex + fragment)
- Animaciones manuales con matrices
- Sistema de iluminación completo (directional + point lights)
- Manejo de modelos, VAOs, VBOs
- Cámara FPS + callbacks + control de input
- Integración de texturas y materiales

Lo cual justifica un precio final superior a un proyecto estándar.

Contingencias (10%)

Permiten cubrir cambios inesperados, delays y requerimientos extras sin afectar la rentabilidad.

Ganancia proyectada

El margen del 30% es estándar en proyectos de desarrollo visual o simulación 3D, cubriendo soporte, mantenimiento y horas no facturadas.

Conclusiones

El desarrollo de este proyecto permitió aplicar de forma práctica los fundamentos del curso, culminando en una recreación 3D funcional de la Torre de los Teen Titans Go en OpenGL, se logró integrar exitosamente un flujo de trabajo completo, desde el modelado y texturizado de los siete objetos requeridos en Blender hasta su renderización y ambientación en la escena. La implementación de las cuatro animaciones generadas por código, incluyendo dos complejas, fue crucial para añadir dinamismo al entorno y demostrar el manejo de transformaciones.

El principal desafío técnico fue la correcta importación de los modelos de Blender al código base de OpenGL, asegurando la fidelidad visual de las texturas y escalas, asimismo, la programación de animaciones complejas que fueran coherentes con el contexto requirió una sólida comprensión de la lógica matemática. Este proyecto no solo cumple con los requisitos técnicos, sino que también consolida las habilidades en la creación de entornos gráficos interactivos, desde la conceptualización hasta la entrega documentada.

Referencias

- SH design. (2024, Enero 15). How to make a curved sofa with a boucle fabric in Blender | Blender Tutorial for Beginners [Video]. YouTube. <https://www.youtube.com/watch?v=5ZHDL1K6jq8>
- Blend nothing. (2023, Febrero 11). How to make stylized Gas Stove in blender [Video]. YouTube. <https://www.youtube.com/watch?v=WijNJMSq7iU>
- Nisakai. (2024, 15 de Abril). ¿Cómo usar Blender 3D? 🤪 | Tutorial para principiantes DESDE CERO en español 🌟 [Video]. YouTube. <https://www.youtube.com/watch?v=HliwF-iTTWo>
- Nisakai. (2024, 31 de mayo). GUÍA RÁPIDA: ¿Cómo modelar en Blender 3D? 🤯 | Tutorial de blender en español 🌟 [Video]. YouTube. https://www.youtube.com/watch?v=_f_6QN_G3bk
- davidgagul2005. (2018, Agosto 16). Blender Robot Dog [3D model]. Free3D. <https://free3d.com/3d-model/blender-robot-dog-319865.html>