FACULTAD DE INGENIERÍA - UNAM

Computación Gráfica e Interacción Humano-Computadora

Proyecto Final:
Manual Técnico

Alumna: 116007203

Profesor: Ing. Carlos Aldair Román Balbuena
Grupo: 5

Fecha de entrega: 24/11/25

Semestre 2026-1

CONTENIDO

Manual Técnico

Introducción

Este proyecto se centra en el desarrollo de una aplicación para renderizado 3D utilizando OpenGL 3.3, una API gráfica estándar que permite crear entornos tridimensionales en tiempo real. La aplicación genera una escena interactiva con una estética caricaturesca, integrando iluminación dinámica, texturizado UV y animaciones matemáticas para brindar una experiencia inmersiva.

Para optimizar el desarrollo, se integraron las siguientes bibliotecas:

- **GLFW:** Encargada de crear la ventana, el contexto de OpenGL y gestionar las entradas de usuario (teclado y mouse).
- **GLEW:** Gestiona los punteros a las funciones modernas de OpenGL.
- **GLM:** Biblioteca matemática para operaciones de álgebra lineal (vectores, matrices y transformaciones) requeridas en el espacio 3D.
- **stb_image / SOIL2:** Utilizadas para la carga y decodificación de imágenes (texturas) en formatos PNG y JPG.

Los modelos 3D fueron procesados y exportados en formato OBJ para su lectura mediante la clase Model basada en Assimp, permitiendo aplicarles sombreadores (shaders) personalizados.

Objetivo

El objetivo principal es desarrollar un recorrido virtual en 3D bajo el estándar OpenGL 3, permitiendo explorar interactivamente una vivienda de dos habitaciones amuebladas con estilo cartoon. El entorno integra una cámara sintética tipo FPS (First Person Shooter) para la navegación libre.

El proyecto busca cumplir con requerimientos técnicos específicos: implementación de iluminación Blinn-Phong (Direccional, Puntual y Spotlight), carga de modelos complejos, y la ejecución de al menos cuatro animaciones interactivas (puertas, cajones, mecedora y radio) controladas por el usuario.

Pseudocódigo del funcionamiento del software

```
INICIO

    Inicializar GLFW y configurar ventana
    Inicializar GLEW

    Si error en inicialización:
        TERMINAR PROGRAMA

    Configurar Viewport y opciones de OpenGL (Depth Test, Blend)

    Cargar Shaders:
        - lightingShader (Objetos opacos)
        - animShader (Humo/Transparencias)
        - SkyBoxshader (Fondo)
        - lampShader (Fuentes de luz)

    Cargar Modelos 3D (Casa, Muebles, Decoración)
    Cargar texturas del Skybox (Cubemap)

    Inicializar variables de animación (rotaciones, tiempos)

    MIENTRAS la ventana no esté cerrada HACER:

        Calcular deltaTime (tiempo entre frames)
        Procesar entradas (Teclado/Mouse) -> DoMovement()

        Limpiar buffers de color y profundidad

        // Renderizado de Escena Opaca
        Usar lightingShader
        Actualizar matrices: View, Projection
        Configurar Luces (Direccional, 3 Puntuales, 1 Spotlight)

        PARA CADA objeto en la escena (Casa, Sillon, Piano...):
            Calcular matriz Model (Traslación, Rotación, Escala)
            Si es animado (Puerta, Cajón):
```

Aplicar transformación basada en variable de estado
Enviar uniforms al shader
Dibujar modelo

// Renderizado de Transparencias (Humo)
Usar animShader
Activar GL_BLEND
Calcular rotación Billboard (mirar hacia la cámara)
Dibujar Humo
Desactivar GL_BLEND

// Renderizado de Fuentes de Luz
Usar lampShader
Dibujar cubos en posiciones de luces

// Renderizado del Fondo
Usar SkyBoxshader
Dibujar cubo de entorno

Intercambiar buffers (SwapBuffers)
Procesar eventos (PollEvents)

FIN MIENTRAS

Liberar memoria (Buffers, Arrays)
Terminar GLFW

FIN

Diagrama de Gantt

Cronograma de proyecto

ID	Fase / Actividad	Inicio	Fin	Sem 1 (06 Oct)	Sem 2 (13 Oct)	Sem 3 (20 Oct)	Sem 4 (27 Oct)	Sem 5 (03 Nov)	Sem 6 (10 Nov)	Sem 7 (17 Nov)	Sem 8 (24 Nov)
1	PLANEACION	06-Oct	12-Oct								
1.1	Búsqueda de referencias	06-Oct	08-Oct								
1.2	Definicion de objetos	09-Oct	12-Oct								
2	DESARROLLO	13-Oct	26-Oct								
2.1	Config. Entorno	13-Oct	17-Oct								
2.2	Implementación Shaders	18-Oct	22-Oct								
2.3	Carga de Modelos	23-Oct	26-Oct								
3	DISEÑO	27-Oct	09-Nov								
3.1	Amueblado de sala	27-Oct	31-Oct								
3.2	Recámara e Iluminación	01-Nov	05-Nov								
3.3	Texturizado y Ajustes	06-Nov	09-Nov								
4	LOGICA Y ANIMACION	10-Nov	19-Nov								
4.1	Programación Inputs	10-Nov	12-Nov								
4.2	Lógica Animaciones	13-Nov	17-Oct								
5	CIERRE Y ENTREGA	20-Nov	24-Oct								
5.1	Debugging final	20-Nov	21-Nov								
5.2	Manual y Costos	22-Nov	23-Nov								
5.3	Empaquetado Final	24-Nov	24-Nov								

	Completo
	Incompleto
	No iniciado

Alcance del proyecto

Funcionalidades

- Renderizado 3D: Visualización de modelos complejos con texturas difusas.
- Iluminación Híbrida: Simulación de luz solar (direccional), lámparas (puntuales) y linternas/focos (spotlight).
- Animaciones Interactivas: Control manual de apertura de y cajones.
- Animaciones Procedurales: Movimiento oscilatorio (mecedora) y vibración (radio) basados en funciones seno/coseno.

Efectos Visuales: Transparencias para cristales y humo, además de un Skybox para el entorno.

Exclusiones

- El proyecto no incluye motor de física para colisiones (el usuario puede atravesar paredes).
- No se implementa sistema de audio.
- No se utilizan técnicas de renderizado avanzado como Ray Tracing o PBR (Physically Based Rendering).

Futuras Extensiones

- Implementación de mapas de sombras (Shadow Mapping) para mayor realismo.
- Agregar sistema de colisiones (Bounding Boxes) para restringir el movimiento.
- Interfaz gráfica (GUI) para modificar el color de las luces en tiempo real.

Metodología de Software

- Se utilizó un enfoque ágil simplificado, apoyado por un Diagrama de Gantt para la planificación temporal y un tablero Kanban para el seguimiento de tareas.

Visualización y priorización

El diagrama de Gantt permitió visualizar las dependencias. Por ejemplo, no se podía programar la animación de la puerta sin antes tener el sistema de carga de modelos funcionando. Se priorizó la funcionalidad "Core" (Render loop y Cámara) antes que los detalles estéticos.

Principales Funcionalidades y Flujo del Programa

1. Librerías y Configuración: Se incluyen glew.h, glfw3.h y glm para manejar la base gráfica. Se define una ventana de 1280x720.
2. Cámara (FPS): La clase Camera gestiona la matriz de vista (GetViewMatrix). La función MouseCallback actualiza los ángulos Yaw y Pitch para mirar alrededor, mientras que DoMovement actualiza la posición (WASD).
3. Sistema de Shaders:
    - LightingShader: Procesa la luz sobre los materiales. Recibe estructuras de Material, DirLight, PointLight y SpotLight.
    - AnimShader: Se usa específicamente para el objeto "Humo", aplicando una matriz de transformación que anula la rotación de la cámara
4. Animaciones: Se utilizan banderas booleanas activadas en KeyCallback.
    - Lineales: Incrementan/decrementan una variable de rotación o posición hasta un límite.

- Cíclicas: Usan glfwGetTime() dentro de funciones sin() para movimientos continuos (Mecedora).

5. Ciclo de Renderizado: En cada iteración del bucle while:
    - Se calcula deltaTime.
    - Se pasan las matrices View y Projection a los shaders.
    - Se dibujan los objetos opacos primero.
    - Se habilita GL_BLEND para dibujar objetos transparentes (Humo, Lámpara).
    - Se dibuja el Skybox al final manipulando el buffer de profundidad.

Conclusiones

El desarrollo de esta fachada permitió consolidar el conocimiento sobre el pipeline gráfico de OpenGL. Se logró crear un entorno estable que gestiona múltiples fuentes de luz y modelos simultáneamente sin caídas graves de rendimiento.

La implementación de animaciones matemáticas (seno/coseno para la mecedora y radio) demostró ser una técnica eficiente para dar vida a la escena sin necesidad de rigging complejo. Asimismo, el uso de bibliotecas estándar como GLFW y GLM facilitó la abstracción de la complejidad matemática de bajo nivel, permitiendo enfocarse en la lógica de la escena. El proyecto cumple con los objetivos de interactividad y estética visual planteados.

Costo

El cálculo se basa en un desarrollo individual de aproximadamente 8 semanas (90 horas efectivas).

Tema	Actividad/Recurso	Cantidad	Costo Unitario	Costo Total
Laboral	Modelado e integración 3D	40 Hrs	$200 MXN	$8,000
	Programación C++ y Shaders	30 Hrs	$250 MXN	$7,500
	Pruebas y Documentación	20 Hrs	$150 MXN	$3,000
Software	Licencia Visual Studio (Community)	1 Hrs	0 MXN	$0
Hardware	Depreciación de Equipo	2 Meses	$700 MXN	$1,400
Servicios	Electricidad e Internet	180 Hrs	$10 MXN	$1,800
Subtotal				$21,700
Contingencia	Imprevistos (10%)			$2,170
Impuestos	IVA + ISR (Aprox 20% efectivo)			$4,340
TOTAL	Precio Final del Proyecto			$28,210 MXN

Referencias

- The Models Resource. (s.f.). Nuka-Cola Machine - Fallout 4 [Modelo 3D]. Recuperado el 24 de noviembre de 2025, de https://models.spriters-resource.com/pc_computer/fallout4/asset/312054

- The Models Resource. (s.f.). Faceless Bandit - Showdown Bandit [Modelo 3D]. Recuperado el 24 de noviembre de 2025, de https://models.spriters-resource.com/pc_computer/showdownbandit/asset/340836

- The Models Resource. (s.f.). Mia Winters - Resident Evil 7 [Modelo 3D]. Recuperado el 24 de noviembre de 2025, de https://models.spriters-resource.com/pc_computer/residentevil7/asset/327822

TECHNICAL MANUAL

Introduction

This project focuses on the development of a 3D rendering application using OpenGL 3.3, a standard graphics API that allows creating three-dimensional environments in real-time. The application generates an interactive scene with a cartoonish aesthetic, integrating dynamic lighting, UV texturing, and mathematical animations to provide an immersive experience.

To optimize development, the following libraries were integrated:

- **GLFW:** Responsible for creating the window, the OpenGL context, and managing user inputs (keyboard and mouse).
- **GLEW:** Manages pointers to modern OpenGL functions.
- **GLM:** Mathematics library for linear algebra operations (vectors, matrices, and transformations) required in 3D space.
- **stb_image / SOIL2:** Used for loading and decoding images (textures) in PNG and JPG formats.

The 3D models were processed and exported in OBJ format for reading via the Model class based on Assimp, allowing custom shaders to be applied to them.

Objective

The main objective is to develop a 3D virtual tour under the OpenGL 3 standard, allowing interactive exploration of a two-room dwelling furnished in a *cartoon* style. The environment integrates an FPS (First Person Shooter) type synthetic camera for free navigation.

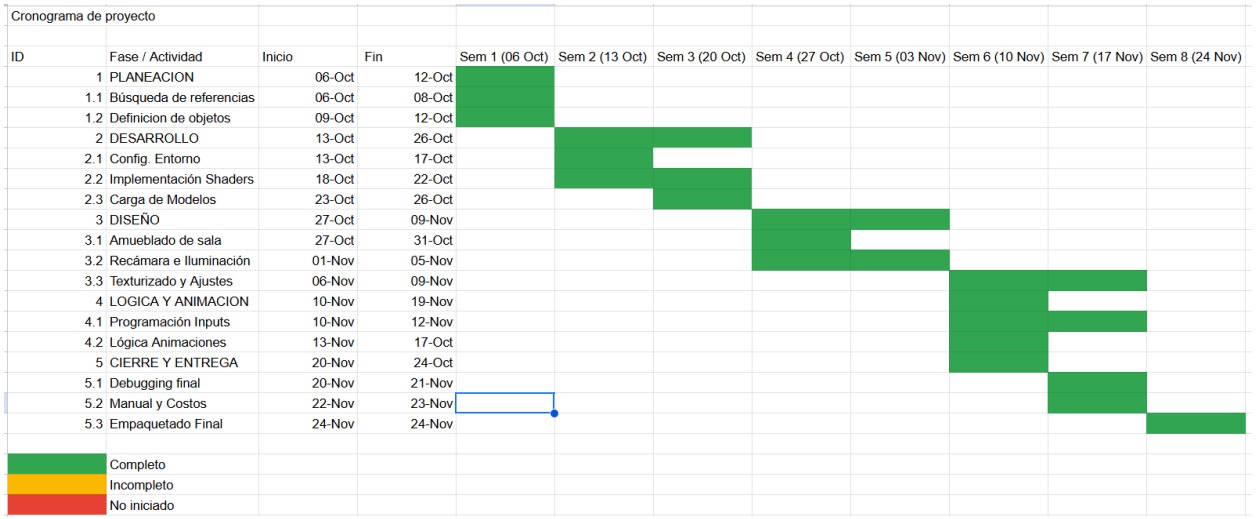
The project seeks to meet specific technical requirements: implementation of Blinn-Phong lighting (Directional, Point, and Spotlight), loading of complex models, and the execution of at least four interactive animations (doors, drawers, rocking chair, and radio) controlled by the user.

Software Functionality Pseudocode

```
START

    Initialize GLFW and configure window
    Initialize GLEW

    If initialization error:
        TERMINATE PROGRAM

    Configure Viewport and OpenGL options (Depth Test, Blend)

    Load Shaders:
        - lightingShader (Opaque objects)
        - animShader (Smoke/Transparencies)
        - SkyBoxshader (Background)
        - lampShader (Light sources)

    Load 3D Models (House, Furniture, Decoration)
    Load Skybox textures (Cubemap)

    Initialize animation variables (rotations, times)

    WHILE window is not closed DO:

        Calculate deltaTime (time between frames)
        Process inputs (Keyboard/Mouse) -> DoMovement()

        Clear color and depth buffers

        // Opaque Scene Rendering
        Use lightingShader
        Update matrices: View, Projection
        Configure Lights (Directional, 3 Point, 1 Spotlight)
```

```
        FOR EACH object in the scene (House, Chair, Piano...):
            Calculate Model matrix (Translation, Rotation, Scale)
            If animated (Door, Drawer):
                Apply transformation based on state variable
            Send uniforms to shader
            Draw model

        // Transparency Rendering (Smoke)
        Use animShader
        Enable GL_BLEND
        Calculate Billboard rotation (face towards camera)
        Draw Smoke
        Disable GL_BLEND

        // Light Source Rendering
        Use lampShader
        Draw cubes at light positions

        // Background Rendering
        Use SkyBoxshader
        Draw environment cube

        Swap buffers (SwapBuffers)
        Process events (PollEvents)

    END WHILE

    Free memory (Buffers, Arrays)
    Terminate GLFW

END
```

Gantt Chart



Project Scope

Functionalities

- **3D Rendering:** Visualization of complex models with diffuse textures.
- **Hybrid Lighting:** Simulation of sunlight (directional), lamps (point), and flashlights/spotlights.
- **Interactive Animations:** Manual control of door and drawer opening.
- **Procedural Animations:** Oscillating movement (rocking chair) and vibration (radio) based on sine/cosine functions.
- **Visual Effects:** Transparencies for glass and smoke, plus a Skybox for the environment.

Exclusions

- The project does not include a physics engine for collisions (the user can pass through walls).
- No audio system is implemented.
- Advanced rendering techniques like Ray Tracing or PBR (Physically Based Rendering) are not used.

Future Extensions

- Implementation of Shadow Mapping for greater realism.
- Adding a collision system (Bounding Boxes) to restrict movement.
- Graphical User Interface (GUI) to modify light colors in real-time.

Software Methodology

A simplified agile approach was used, supported by a **Gantt Chart** for temporal planning and a **Kanban** board for task tracking.

Visualization and Prioritization

The Gantt chart allowed visualization of dependencies. For example, the door animation could not be programmed without first having the model loading system working. "Core" functionality (Render loop and Camera) was prioritized over aesthetic details.

Main Functionalities and Program Flow

1. Libraries and Configuration: glew.h, glfw3.h, and glm are included to handle the graphics base. A 1280x720 window is defined.

2. Camera (FPS): The Camera class manages the view matrix (GetViewMatrix). The MouseCallback function updates *Yaw* and *Pitch* angles to look around, while DoMovement updates the position (WASD).

3. Shader System:

- LightingShader: Processes light on materials. Receives structures for Material, DirLight, PointLight, and SpotLight.
- AnimShader: Used specifically for the "Smoke" object, applying a transformation matrix that cancels camera rotation (Billboarding).

4. Animations: Boolean flags activated in KeyCallback are used.

- **Linear:** Increment/decrement a rotation or position variable up to a limit.
- **Cyclic:** Use glfwGetTime() inside sin() functions for continuous movements (Rocking Chair).

5. Rendering Loop: In each iteration of the while loop:

- deltaTime is calculated.
- View and Projection matrices are passed to shaders.
- Opaque objects are drawn first.
- GL_BLEND is enabled to draw transparent objects (Smoke, Lamp).
- The Skybox is drawn at the end by manipulating the depth buffer.

Conclusions

The development of this facade allowed consolidating knowledge about the OpenGL graphics pipeline. A stable environment was created that manages multiple light sources and models simultaneously without severe performance drops.

The implementation of mathematical animations (sine/cosine for the rocking chair and radio) proved to be an efficient technique to bring the scene to life without the need for complex rigging. Likewise, the use of standard libraries such as GLFW and GLM facilitated the abstraction of low-level mathematical complexity, allowing focus on scene logic. The project meets the stated interactivity and visual aesthetic objectives.

Cost Analysis

The calculation is based on an individual development of approximately 8 weeks (90 effective hours).

Tema	Actividad/Recurso	Cantidad	Costo Unitario	Costo Total
Laboral	Modelado e integración 3D	40 Hrs	$200 MXN	$8,000
	Programación C++ y Shaders	30 Hrs	$250 MXN	$7,500
	Pruebas y Documentación	20 Hrs	$150 MXN	$3,000
Software	Licencia Visual Studio (Community)	1 Hrs	0 MXN	$0
Hardware	Depreciación de Equipo	2 Meses	$700 MXN	$1,400
Servicios	Electricidad e Internet	180 Hrs	$10 MXN	$1,800
Subtotal				$21,700
Contingencia	Imprevistos (10%)			$2,170
Impuestos	IVA + ISR (Aprox 20% efectivo)			$4,340
TOTAL	Precio Final del Proyecto			$28,210 MXN

References

- **The Models Resource.** (n.d.). *Nuka-Cola Machine - Fallout 4* [3D Model]. Retrieved November 24, 2025, from https://models.spriters-resource.com/pc_computer/fallout4/asset/312054
- **The Models Resource.** (n.d.). *Faceless Bandit - Showdown Bandit* [3D Model]. Retrieved November 24, 2025, from https://models.spriters-resource.com/pc_computer/showdownbandit/asset/340836
- **The Models Resource.** (n.d.). *Mia Winters - Resident Evil 7* [3D Model]. Retrieved November 24, 2025, from https://models.spriters-resource.com/pc_computer/residentevil7/asset/327822