UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

PROYECTO 2 – Manual técnico

Nº de Cuenta: 320152841

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 05

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 25/11/2025

CALIFICACIÓN: _____

Índice

Contenido

Manual técnico – español

Planteamiento del proyecto

Objetivo.

El objetivo de este proyecto es crear un entorno virtual e interactivo basado en OpenGL 3.3 y GLSL, en donde se pueda tener un recorrido virtual en un ambiente recreado de una fachada y 2 cuartos apegándose a la temática de la caricatura "**Phineas and Ferb**", de manera que se añadirán objetos y elementos que referencien a este contexto declarado.

Puntos clave.

- Recreación de dos cuartos y fachada.
- Respetar el estilo artístico de la caricatura.
- Creación de 4 animaciones dentro del contexto.
- Integración de una cámara sintética.
- Creación de manual de usuarios y técnico.

Especificaciones de Hardware y Software.

Sistema operativo: Windows 11 Pro Versión 24H2

Procesador: AMD Ryzen 5 5600X 6-Core Processor (3.70 GHz).

Tarjeta Gráfica: Nvidia GeForce RTX 3060.

Memoria RAM: 32 GB a 3200 MHz.

Software de modelado: Blender 4.5.

Software de desarrollo: Visual Studio 2022.

Software editor de imagen: Gimp 3.

Herramientas utilizadas para el desarrollo.

Blender: software utilizado para la creación, mejora u optimización de modelos 3D así como para la texturización de los mismos, se optó por este software por su amigabilidad con el usuario y por ser de licencia gratuita, sin decir que su documentación ante errores es muy amplia ya activa.

GIMP 3: Gimp es un software de edición y creación de imágenes multiplataforma y de licencia gratuita, de manera que se fue de gran utilidad para la optimización y edición de imágenes utilizadas para texturizar objetos.

Visual Studio 2022: Visual studio es un entorno de desarrollo creado por Microsoft el cual sirvió para desarrollar de manera ordenada e intuitiva el proyecto, ya que tiene una excelente compatibilidad con OpenGL y GLSL a su vez que tiene integrado el control de versiones de GitHub que fue de gran ayuda.

GitHub y Git: Se opto por utilizar este sistema de control de versiones ya que es con el que tengo más familiaridad y entiendo mejor, de manera que puedo hacer los cambios rápidamente y de forma organizada sin complicaciones, a su vez es uno de los que tiene más documentación y comunidad activa por posibles errores.

Lineamientos del proyecto

Alcances

- Replicar lo más fiel posible la fachada y cuartos solicitados.
- Creación de animaciones dentro del contexto de la caricatura.
- Control simple e intuitivo del entorno virtual.
- Optimización de modelos y texturas a utilizar.
- Creación de documentación para usuario y manual técnico.

Metas

- Creación de un entorno virtual con temática de la caricatura “Phineas anda Ferb”.
- Fiel seguimiento a la texturización de acuerdo al contexto.
- Proporcionar un proyecto funcional y publico mediante Github.
- Archivos de ejecución al alcance del usuario.

Hitos

- Selección de elementos para recreación por fachada y cuartos.
- Selección de software y compatibilidad con hardware.
- Búsqueda de modelos 3D de utilidad.
- Creación de modelos 3D no encontrados y optimización de modelos.

- Implementación de texturizado a modelos.
- Creación de código base para OpenGL.
- Implementación de modelos a OpenGL.
- Implementación de animaciones en OpenGL.
- Optimización y corrección de código en OpenGL.
- Creación de documentación necesaria.
- Publicación del proyecto final mediante GitHub.

Metodología de Software:

Se opto por utilizar SCRUM como metodología de seguimiento, ya que es un modelo que permite ser flexible con sus pasos a seguir y pueden ser modificados dependiendo las necesidades de cada proyecto, en este caso el proyecto se llevo a cabo en los siguientes sprints generales:

Planificación y elección.

Durante 3 días que duro el sprint se realizo la planificación del proyecto y la elección de software a utilizar, así como la búsqueda de modelos en internet que pudiera servir para el proyecto y listar los faltantes, así mismo se buscaron texturas que fuera de acuerdo al contexto de la serie.

Modelado y creación del entorno de desarrollo.

Durante 6 días se llevo acabo el segundo sprint el cual consistió en la creación de los modelos restantes en Blender, y la optimización de los modelos obtenidos de internet, así como su texturizado mediante GIMP y Blender, de forma que al terminar con ello se pasó a la creación del proyecto en Visual Studio 2022, dejando todo listo con un código base. Se tuvo una holgura de 2 días.

Desarrollo del entorno gráfico y animaciones.

Durante 5 días se realizo el tercer sprint, en donde se llevó a cabo la importación de los modelos 3D a OpenGL ya texturizados, y se empezó la construcción de todo el escenario virtual, una vez concluido se empezó a

implementar las animaciones pertinentes con los objetos puestos. Se tuvo una holgura de 3 días.

Pruebas y documentación.

Para este ultimo sprit que se llevo acabo en 3 días el cual consistió en la prueba del mundo virtual y las interacciones que este contiene para comprobar que se cumpliera con lo que se pide, a su vez se fue creando la documentación oficial del proyecto, tanto técnica como manual, y la publicación del proyecto en GitHub.

Diagrama de Gantt

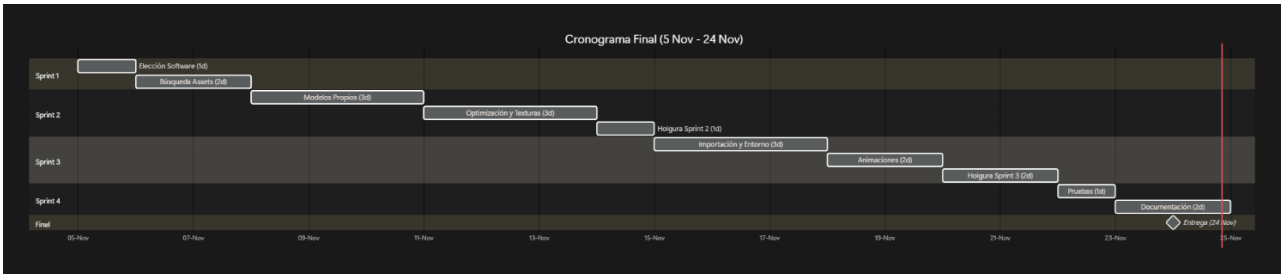


Ilustración 1. Cronograma de Gantt

Diagrama de Flujo

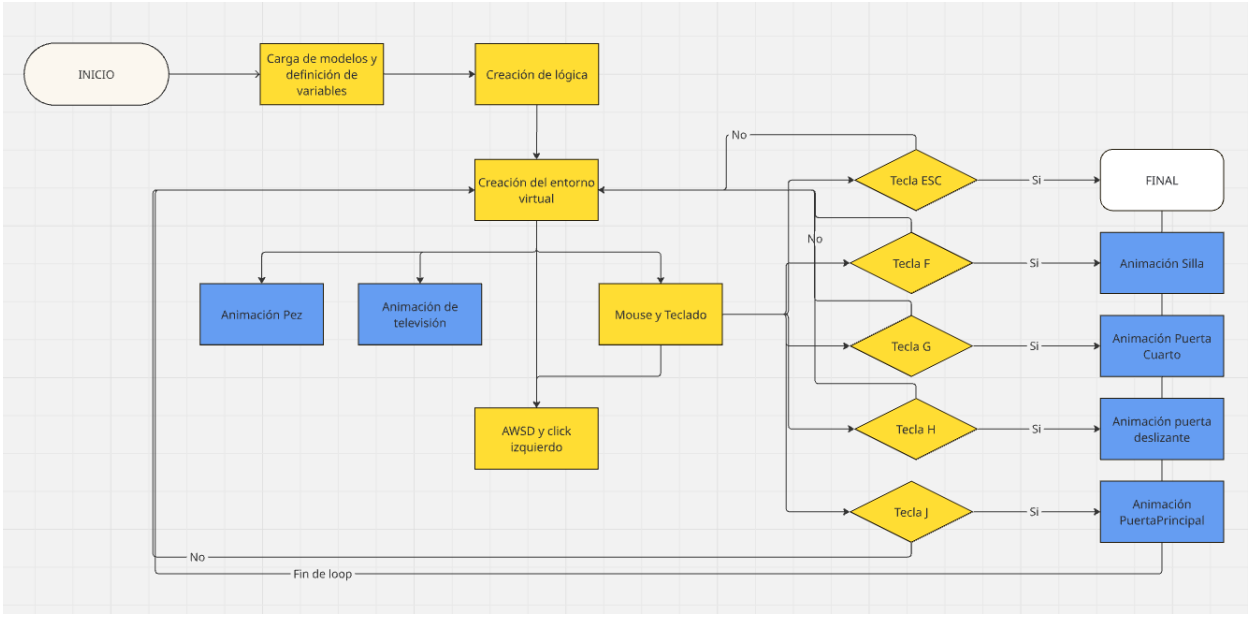


Ilustración 2. Diagrama de Flujo simplificado

Análisis financiero del proyecto.

Se crea un análisis financiero de lo invertido en el proyecto y el costo que tiene el mismo desglosado en sectores.

Personal involucrado en el proyecto:

Puesto	***Días trabajados***	***Ingreso por día***	***Subtotal***
Scrum Master	19	$ 2,400	$ 45,600
Modelador de Blender	8	$ 1,800	$ 14,400
Programador C++ y OpenGL	7	$ 3,000	$ 21, 000
Tester y documentador	3	$ 1,400	$ 4,200
Total			**$ 85,200**

Software utilizado:

Para el caso de los softwares utilizados el costo de uso entre licencias y demás es de $ 0 MXN, por lo tanto, no se muestra una tabla del desglose en este apartado.

Hardware utilizado:

En este caso no se hizo ninguna compra de equipo de hardware, pero se hace la cotización del uso del equipo personal y el uso de servicios personales que se utilizaron en el proyecto, en este caso el uso del equipo de cómputo al ser personal y armado por piezas no existe un precio comercial, pero conociendo los componentes y el año de compra hecho en 2021, el precio es aproximadamente de $ 46,000 MXN, teniendo en cuenta la devaluación y el contexto mexicano, hoy en día estaría rondando los $ 36,800 MXN.

Gastos operativos:

Concepto	***Costo MXN***
Electricidad	$ 1,800
Internet	$ 3,600
Servicios y mantenimiento	$ 1,250
Total	***$ 6,650***

Cotización Final:

Apartado	***Total***
Personal	$ 85,200
Software	$ 0
Hardware	$ 36,800
Gastos Operativos	$ 6,650
SUBTOTAL	**$ 128,650**
Fondo de emergencia (15%)	$ 19,294.5
SUBTOTAL ANTES DE IMPUESTOS	**$ 147,944.5**
IVA – 16%	$ 23,671.12
TOTAL, DESPUES DE IMPUESTOS	***$ 171,615.62***

Para este proyecto la evaluación financiera resulto en un cobro de ***$ 171,615.62 MXN,*** ya con todos los desgloses previstos y contemplados para cada apartado en que se divide en proyecto.

Glosario de variables.

A continuación, se enlistan las variables que se usaron en el proyecto, del código PFinal.cpp (main).

Variables de animación y lógica

Tipo de Variable	*Nombre de la Variable*	*Descripción*
glm::vec3	centroPecera	Coordenadas (x, y, z) que definen el punto central sobre el cual orbita el pez.
float	radioNadar	Define la distancia del radio del círculo que describe el pez al nadar.
float	velocidadNadar	Determina la rapidez del movimiento del pez.
float	anguloNado	Almacena el ángulo actual de rotación del pez para orientarlo correctamente mientras avanza.
bool	animacionSillaActiva	Bandera lógica (true/false) activada con la tecla 'F' para iniciar la secuencia de la silla.
glm::vec3	posSilla	Vector de posición actual de la silla, se actualiza en cada frame si la animación está activa.
float	anguloGiroSilla	Controla la rotación sobre el propio eje de la silla mientras se desplaza.
float	anguloCaidaSilla	Controla la inclinación de la silla una vez que choca con la pared (simula caída).
bool	sillaChoco	Bandera de estado que indica si la silla ya alcanzó el límite de la pared (limitePared).
float	limitePared	Coordenada en X que marca el punto máximo de desplazamiento de la silla.
float	rotation	Ángulo de apertura de la Puerta Principal. Se modifica con la tecla 'J'.
float	rotationCuarto	Ángulo de apertura de la Puerta del Cuarto. Se modifica con la tecla 'G'.
float	movOffset	Velocidad a la que se abren/cierran las puertas giratorias.
float	deslizamiento	Posición en X actual de la puerta corrediza.
float	maxDeslizamiento	Límite máximo de desplazamiento para la puerta corrediza.

float	offsetPantalla	Variable para desplazar las coordenadas de textura (UV) en la TV animada.

Estructuras principales para la plantilla del proyecto

Tipo de Variable	***Nombre de la Variable***	***Descripción***
Window	mainWindow	Instancia de la clase Window que gestiona el contexto de GLFW, dimensiones y eventos de entrada.
std::vector<Mesh*>	meshList	Contenedor dinámico que almacena punteros a las mallas geométricas básicas generadas por código (ej. el piso).
std::vector<Shader>	shaderList	Almacena los programas de shaders compilados.
Camera	camera	Instancia de la clase Camera que gestiona la matriz de vista, posición del observador y movimiento (WASD + Mouse).
Skybox	skybox	Objeto que renderiza el cubo de entorno (mapa de cubo) para simular el cielo/fondo.
GLfloat	deltaTime	Tiempo transcurrido entre el frame actual y el anterior; vital para sincronizar animaciones independientemente de los FPS.
glm::mat4	projection	Matriz de proyección en perspectiva calculada según el aspecto de la ventana.
glm::mat4	model	Matriz acumulativa que almacena traslación, rotación y escalado del objeto actual a renderizar.
glm::mat4	modelaux	Matriz auxiliar para guardar el estado de una transformación padre (jerarquías) y recuperarla después.

Iluminación y Shaders

Tipo de Variable	**Nombre de la Variable**	**Descripción**
DirectionalLight	mainLight	Objeto que simula una fuente de luz lejana, con rayos paralelos.
Material	Material_brillante	Define propiedades especulares altas para objetos brillantes.
Material	Material_opaco	Define propiedades especulares bajas para objetos mate.
GLuint	uniformModel	ID de la ubicación en el shader para la Matriz de Modelo.
GLuint	uniformProjection	ID para la Matriz de Proyección.
GLuint	uniformView	ID para la Matriz de Vista (cámara).
GLuint	uniformColor	ID para enviar cambios de tinte al fragment shader.

GLuint	uniformTextu-reOffset	ID para enviar el desplazamiento UV.

Modelos y Texturas

Tipo de Variable	Nombre de la Variable	Descripción
Texture	Todas las de textures	Permiten cargar imágenes que serán texturas para los objetos de geometría primitiva importados o creados en OpenGL.
Model	Todas los Models	Se encargan de cargar los modelos 3D importados de Blender en OpenGL.

Glosario de funciones.

En este apartado se da una pequeña descripción de las funciones que no pertenecen a las funciones originales a OpenGL, que son usadas para el proyecto, en este caso se dividieron en los siguientes apartados para una mejor visualización y entendimiento de las mismas.

Funciones Globales.

Nombre de la Función	*Retorno*	*Descripción Técnica*
calcAve-rageNor-mals	void	Cálculo de normales suavizadas. Recorre los vértices y los índices de una malla para calcular el vector normal promedio de cada vértice basándose en los triángulos adyacentes.
CreateObje-cts	void	Generación de geometría procedural. Se encarga de definir manualmente los arreglos de vértices e índices para objetos básicos que no se cargan desde un archivo .obj. Aquí se configuran sus coordenadas (x,y,z), coordenadas de textura (u,v) y normales.
CreateSha-ders	void	Compilación de programas GLSL. Instancia la clase Shader y llama a la carga de los archivos de texto (shader_light.vert, shader_light.frag) para compilarlos y enlazarlos en un programa de shaders utilizable por la GPU.

Métodos de clases complementarias (Windows, model, camera).

Método	*Descripción*
Initialise()	Configura el contexto de GLFW y GLEW, define el tamaño de la ventana y los buffers de profundidad.
getAbrirCe-rrarPuerta()	Detecta si la tecla asignada fue presionada en este frame para activar la bandera de animación de la puerta principal.

getAbrirCerrarCorrediza()	Detecta el input para la lógica de la puerta corrediza.
getXChange() / getYChange()	Devuelve el delta de movimiento del mouse en X y Y para rotar la cámara.
LoadModel(path)	Utiliza una librería externa para leer un archivo .obj, extraer sus mallas, materiales y texturas, y guardarlos en memoria listo para renderizar.
RenderModel()	Recorre todas las sub-mallas del modelo cargado y ejecuta las llamadas de dibujo (glDrawElements) para pintarlo en pantalla.
CreateMesh(vertices, indices...)	Toma los arreglos de datos crudos y los carga en la memoria de la tarjeta gráfica (VBOs e IBOs).
RenderMesh()	Activa el VAO (Vertex Array Object) de una malla simple y la dibuja.
LoadTextureA()	Carga una imagen (jpg/png) usando stb_image, genera un ID de textura en OpenGL y configura los parámetros de repetición y filtrado.
UseTexture()	Activa (glBindTexture) la textura actual en la unidad de textura correspondiente para que el shader la aplique al siguiente objeto dibujado.
UseMaterial(...)	Envía al shader los valores de intensidad especular y brillo (shininess) definidos para ese objeto, controlando cómo reacciona a la luz.
calculateViewMatrix()	Utiliza la posición y dirección de la cámara para generar la matriz View de lookAt.
keyControl(...)	Actualiza la posición (x,y,z) de la cámara basándose en las teclas presionadas (WASD) y el tiempo delta.
DrawSkybox(...)	Renderiza un cubo gigante alrededor de la cámara con texturas de entorno (cielo), desactivando la escritura en el buffer de profundidad para que siempre parezca estar "al fondo".

Elementos y referencias.

Se utilizará como referencia la fachada de la casa de Phineas and Ferb, teniendo como referencia las siguientes imágenes.

Casa de Phineas and Ferb

Para los dos cuartos elegidos dentro de la casa serán los siguientes:

Living Room (Sala):

Cuarto de Phineas y Ferb, para este cuarto no se pondrán todos los objetos y muebles que se tienen en la serie.

Para los objetos recreados se usarán los siguientes para cada cuarto, teniendo como referencia el estilo artístico de la caricatura animada:

Living Room (Sala):

Televisión antigua

Mesa de centro

Sillón

Maceta con planta

Bowl de palomitas

Para el cuarto de Phineas y Ferb:

Pecera

Silla

Cama

Estación de teléfono

Moai

Resultados

Finalmente, después de presentar todos los apartados desde la estrategia de desarrollo que se siguió, así como el equipo usado para el desarrollo del proyecto, incluyendo los precios y costos que este proyecto llevaría a cabo para su finalización, de manera que ahora solo queda mostrar los resultados conseguidos tanto de la fachada y de los dos cuartos dentro del entorno virtual.

Fachada

Ilustración 3. Vista frontal

Ilustración 4. Vista trasera

Ilustración 5. Living room

Ilustración 6

Ilustración 7. Cuarto Phineas and Ferb

Conclusión.

Como conclusión de este proyecto realmente siento que fue muy interesante desarrollarlo sobre todo porque aplique varias cosas que se vieron tanto en teoría como en laboratorio y sobre todo me permitió mejorar ciertos aspectos que en laboratorio no pude realizar con me hubiera gustado, siento que este tipo de proyectos en donde se puede realizar un poco de todas las áreas aplicadas en

computación gráfica son muy enriquecedores para nosotros como estudiantes y nos permite comprender que un proyecto como estos llevaba mucho tiempo y esfuerzo detrás. Además, la libertad de decidir que se puede agregar a tu entorno virtual siento que es un plus muy grande y permite que nosotros como alumnos hagamos el proyecto con más ganas y entusiasmo obteniendo un mejor resultado.

Technical Manual – English

Project proposal

Objetive

The goal of this project is to create a virtual and interactive environment based on OpenGL 3.3 and GLSL. It allows a virtual tour inside a recreated environment consisting of a facade and 2 rooms, following the theme of the cartoon "Phineas and Ferb", adding objects and elements that fit this context.

Key points

- Recreation of two rooms and a facade.
- Respecting the art style of the cartoon.
- Creation of 4 animations within the context.
- Integration of a synthetic camera.
- Creation of user and technical manuals.

Hardware and Software Specifications

- **Operating System:** Windows 11 Pro Version 24H2.
- **Processor:** AMD Ryzen 5 5600X 6-Core Processor (3.70 GHz).
- **Graphics Card:** Nvidia GeForce RTX 3060.
- **RAM:** 32 GB at 3200 MHz.
- **Modeling Software:** Blender 4.5.
- **Development Software:** Visual Studio 2022.
- **Image Editor:** Gimp 3.

Tools used for development

Blender: Software used to create, improve, or optimize 3D models, as well as for texturing them. I chose this software because it is user-friendly and free, plus it has a lot of documentation for errors.

GIMP 3: Gimp is a free, cross-platform image editing software. It was very useful for optimizing and editing images used to texture objects.

Visual Studio 2022: An IDE created by Microsoft which helped to develop the project in an organized way. It has excellent compatibility with OpenGL and GLSL, and the integrated GitHub version control was very helpful.

GitHub and Git: I chose this version control system because I am familiar with it. It allows me to make changes quickly and in an organized way without complications. It also has a large community for solving errors.

Project Guidelines

Scope

Goals

- Replicate the requested facade and rooms as closely as possible.
- Create animations within the cartoon's context.
- Simple and intuitive control of the virtual environment.
- Optimization of models and textures.
- Creation of documentation for the user and technical manual.

Milestones

- Creation of a virtual environment with the "Phineas and Ferb" theme.
- Faithful texturing according to the context.
- Provide a functional and public project via GitHub.
- Execution files available to the user.
- Selection of elements for the facade and rooms.
- Selection of software and hardware compatibility.
- Search for useful 3D models.
- Creation of missing 3D models and optimization of others.
- Implementation of textures on models.
- Creation of base code for OpenGL.
- Implementation of models into OpenGL.
- Implementation of animations in OpenGL.
- Optimization and correction of OpenGL code.
- Creation of necessary documentation.

- Publication of the final project on GitHub.

Software Methodology

I chose SCRUM as the tracking methodology because it is a flexible model where steps can be modified depending on the project's needs. In this case, the project was carried out in the following general sprints:

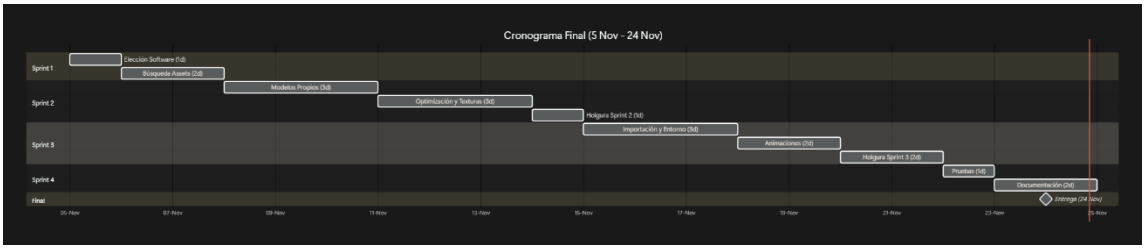
Planning and choice During the 3 days of this sprint, I planned the project and chose the software. I also searched for models online that could work for the project, listed the missing ones, and looked for textures that fit the series' context.

Modeling and creation of the development environment During 6 days, the second sprint took place. It consisted of creating the remaining models in Blender, optimizing the models found online, and texturing them using GIMP and Blender. After finishing this, I set up the project in Visual Studio 2022, leaving everything ready with a base code. *There was a slack (extra time) of 2 days.*

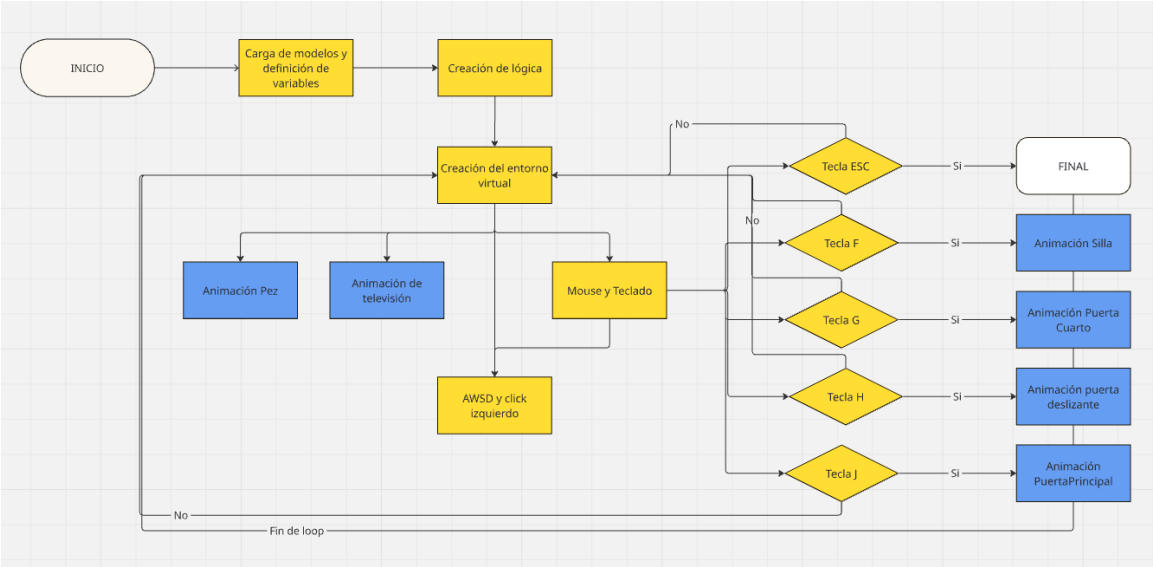
Development of the graphic environment and animations During 5 days, the third sprint was carried out. Here, I imported the textured 3D models into OpenGL and started building the entire virtual scenario. Once finished, I started implementing the animations for the objects. *There was a slack of 3 days.*

Testing and documentation For this last sprint, which took 3 days, I tested the virtual world and its interactions to check that requirements were met. At the same time, I created the official project documentation (technical and manual) and published the project on GitHub.

Gantt Chart



Flowchart



Financial Analysis of the Project

A financial analysis of the investment and costs of the project is created, broken down by sectors.

Personnel involved in the Project:

Position	***Days worked***	***Income per day***	***Subtotal***
Scrum Master	19	$2,400	$45,600
Blender Modeler	8	$1,800	$14,400
C++ and OpenGL Programmer	7	$3,000	$21,000
Tester and documenter	3	$1,400	$4,200
Total			**$85,200**

Software used: For the software used, the cost for licenses is $ 0 MXN, so no table is shown in this section.

Hardware used:

No new hardware was purchased, but I quoted the use of personal equipment and services. Since the computer was custom-built in 2021, there is no commercial price,

but knowing the components, the price was approximately $46,000 MXN. Considering devaluation and the Mexican context, today it would be around$ 36,800 MXN.

Operational expenses:

Concept	**Cost MXN**
Electricity	$1,800
Internet	$3,600
Services and maintenance	$1,250
Total	**$6,650**

Final Quote:

Section	**Total**
Personnel	$85,200
Software	$0
Hardware	$36,800
Operational Expenses	$6,650
SUBTOTAL	**$128,650**
Emergency Fund (15%)	$19,294.50
SUBTOTAL BEFORE TAXES	**$147,944.50**
VAT (IVA) - 16%	$23,671.12
TOTAL AFTER TAXES	***$171,615.62***

For this project, the financial evaluation resulted in a cost of **$ 171,615.62 MXN**, covering all the planned breakdowns.

Glossary of Variables

Below are the variables used in the project, from the PFinal.cpp (main) code.

Type	Name	Description
glm::vec3	centroPecera	Coordinates (x, y, z) defining the center point the fish orbits around.
float	radioNadar	Defines the radius distance of the circle the fish swims.
float	velocidadNadar	Determines the speed of the fish movement.
float	anguloNado	Stores the current rotation angle of the fish to orient it correctly.
bool	animacionSillaActiva	Logic flag (true/false) activated with the 'F' key to start the chair sequence.
glm::vec3	posSilla	Current position vector of the chair, updated every frame if animation is active.
float	anguloGiroSilla	Controls the rotation on the chair's own axis while moving.
float	anguloCaidaSilla	Controls the chair's tilt once it hits the wall (simulates falling).
bool	sillaChoco	State flag indicating if the chair reached the wall limit (limitePared).
float	limitePared	X coordinate marking the maximum displacement of the chair.
float	rotation	Opening angle of the Main Door. Modified with the 'J' key.
float	rotationCuarto	Opening angle of the Room Door. Modified with the 'G' key.
float	movOffset	Speed at which the rotating doors open/close.
float	deslizamiento	Current X position of the sliding door.
float	maxDeslizamiento	Maximum displacement limit for the sliding door.
float	offsetPantalla	Variable to shift texture coordinates (UV) on the animated TV.

Animation and logic variables

Type	Name	Description
glm::vec3	centroPecera	Coordinates (x, y, z) defining the center point the fish orbits around.
float	radioNadar	Defines the radius distance of the circle the fish swims.
float	velocidadNadar	Determines the speed of the fish movement.
float	anguloNado	Stores the current rotation angle of the fish to orient it correctly.
bool	animacionSillaActiva	Logic flag (true/false) activated with the 'F' key to start the chair sequence.
glm::vec3	posSilla	Current position vector of the chair, updated every frame if animation is active.
float	anguloGiroSilla	Controls the rotation on the chair's own axis while moving.
float	anguloCaidaSilla	Controls the chair's tilt once it hits the wall (simulates falling).
bool	sillaChoco	State flag indicating if the chair reached the wall limit (limitePared).
float	limitePared	X coordinate marking the maximum displacement of the chair.
float	rotation	Opening angle of the Main Door. Modified with the 'J' key.
float	rotationCuarto	Opening angle of the Room Door. Modified with the 'G' key.
float	movOffset	Speed at which the rotating doors open/close.
float	deslizamiento	Current X position of the sliding door.
float	maxDeslizamiento	Maximum displacement limit for the sliding door.
float	offsetPantalla	Variable to shift texture coordinates (UV) on the animated TV.

Main structures for the project template

Type	Name	Description
Window	mainWindow	Instance of the Window class managing GLFW context, dimensions, and inputs.
std::vector<Mesh*>	meshList	Dynamic container storing pointers to basic geometric meshes generated by code (e.g., floor).
std::vector<Shader>	shaderList	Stores the compiled shader programs.
Camera	camera	Instance of the Camera class managing view matrix, observer position, and movement (WASD + Mouse).
Skybox	skybox	Object rendering the environment cube (cube map) to simulate the sky/background.
GLfloat	deltaTime	Time elapsed between the current and previous frame; vital for syncing animations regardless of FPS.
glm::mat4	projection	Perspective projection matrix calculated based on window aspect ratio.
glm::mat4	model	Cumulative matrix storing translation, rotation, and scaling of the object to render.
glm::mat4	modelaux	Auxiliary matrix to save the state of a parent transformation (hierarchies) and recover it later.

Lighting and Shaders

Type	Name	Description
DirectionalLight	mainLight	Object simulating a distant light source with parallel rays.
Material	Material_brillante	Defines high specular properties for shiny objects.
Material	Material_opaco	Defines low specular properties for matte objects.
GLuint	uniformModel	Shader location ID for the Model Matrix.
GLuint	uniformProjection	ID for the Projection Matrix.
GLuint	uniformView	ID for the View Matrix (camera).
GLuint	uniformColor	ID to send tint changes to the fragment shader.
GLuint	uniformTextureOffset	ID to send the UV displacement.

Models and Textures

Type	Name	Description
Texture	All textures	Allow loading images to be used as textures for primitive geometry or imported OpenGL objects.
Model	All Models	Responsible for loading 3D models imported from Blender into OpenGL.

Glossary of Functions

Here is a brief description of the functions that are not original OpenGL functions but are used in the project. They are divided into sections for better understanding.

Global Functions

Function Name	Return	Technical Description
calcAverageNormals	void	Smoothed normals calculation. Iterates through vertices and indices of a mesh to calculate the average normal vector of each vertex based on adjacent triangles.
CreateObjects	void	Procedural geometry generation. Manually defines vertex and index arrays for basic objects not loaded from an .obj file. Configures coordinates (x,y,z), texture coordinates (u,v), and normals.
CreateShaders	void	GLSL program compilation. Instantiates the Shader class and calls for loading text files (shader_light.vert, shader_light.frag) to compile and link them into a shader program usable by the GPU.

Complementary Class Methods (Window, model, camera)

Method	Description
Initialise()	Configures GLFW and GLEW context, defines window size and depth buffers.
getAbrirCerrarPuerta()	Detects if the assigned key was pressed in this frame to activate the main door animation flag.
getAbrirCerrarCorrediza()	Detects input for the sliding door logic.
getXChange() / getYChange()	Returns the mouse movement delta in X and Y to rotate the camera.

LoadModel(path)	Uses an external library to read an .obj file, extract meshes, materials, and textures, and store them in memory ready to render.
RenderModel()	Iterates through all sub-meshes of the loaded model and executes draw calls (glDrawElements) to paint it on screen.
CreateMesh(...)	Takes raw data arrays and loads them into graphics card memory (VBOs and IBOs).
RenderMesh()	Activates the VAO (Vertex Array Object) of a simple mesh and draws it.
LoadTextureA()	Loads an image (jpg/png) using stb_image, generates an OpenGL texture ID, and sets wrapping/filtering parameters.
UseTexture()	Activates (glBindTexture) the current texture in the corresponding texture unit so the shader applies it to the next object.
UseMaterial(...)	Sends specular intensity and shininess values to the shader, controlling how the object reacts to light.
calculateViewMatrix()	Uses camera position and direction to generate the lookAt View matrix.
keyControl(...)	Updates camera position (x,y,z) based on pressed keys (WASD) and delta time.
DrawSkybox(...)	Renders a giant cube around the camera with environment textures (sky), disabling depth buffer writing so it always appears "in the background".

Elements and References

The facade of Phineas and Ferb's house will be used as a reference, using the following images.

Phineas and Ferb's House

For the two rooms chosen inside the house:

Living Room:

Phineas and Ferb's Room: For this room, not all objects and furniture from the series will be included.

For the recreated objects, the following will be used for each room, referencing the cartoon's art style:

Living Room:

Old TV

Coffee table

Armchair

Pot with plant

Popcorn bowl

Phineas and Ferb's Room:

Fish tank

Chair

Bed

Phone booth

Moai

Results

Finally, after presenting the development strategy, the equipment used, and the costs, here are the results achieved for the facade and the two rooms in the virtual environment.

Facade

Illustration 3. Front view

Illustration 4. Back view

Illustration 5. Living Room

Illustration 6

Phineas and Ferb's Room

Conclusion

In conclusion, I feel that developing this project was very interesting, especially because I applied many things seen in both theory and lab. It allowed me to improve certain aspects I couldn't do in the lab as I wanted. I feel these types of projects, where you apply a bit of every area in computer graphics, are very enriching

for us students and help us understand that a project like this takes a lot of time and effort.

Also, the freedom to decide what to add to your virtual environment is a big plus and allows us to do the project with more enthusiasm, getting a better result.

Referencias / References

DansterDev. (2021, 14 de julio).Crea puertas y ventanas para tus edificios de la forma mas rápida y sencilla! Blender 2.93. https://www.youtube.com/watch?v=qCjGswuufxc

DansterDev. (2021, 26 de mayo).

Modeling buildings with easy interiors in Blender! Explained step by step!. https://www.youtube.com/watch?v=Q1IiIIIriAs

Papalegoas. (2019, 28 de enero). Plantas de interior modelo 3d. turbosquid. https://www.turbosquid.com/es/3d-models/indoor-plants-3d-model-1372214?dd_referrer=

CleoFinn. (2021, 22 de marzo). televisor modelo 3d. turbosquid https://www.turbosquid.com/es/3d-models/tv-3d-model-1709038

dimension Dazzle. (2025, 23 de junio). Paquete de palomitas de maíz modelo 3d. turbosquid. https://www.turbosquid.com/es/3d-models/3d-popcorn-pack-2424507

Vlad. (2025, 10 de septiembre). TinyLivingPack. Sketchfab. https://sketchfab.com/3d-models/tinylivingpack-e192841259554d1cabb08c7a8180be9f#download

AtheneaAtlas. (2025, 29 de julio). Moai. Sketchfab. https://sketchfab.com/3d-models/moai-ca950165931e42ea80bbcaa165147598#download

Darkay. (2021 , 10 de julio). Cabina Telefónica. Sketchfab. https://sketchfab.com/3d-models/cabina-telefonica-10d9ec53b8a140c7a7785ff54a7ecc22#download

printable_models. (2018, 24 de septiembre). acuario de agua salada v1. Free3D. https://free3d.com/es/modelo-3d/saltwater-aquarium-v1--172967.html?dd_referrer=

bananacake. (2017, 18 de febrero). Silla modelo 3d. Free3D. https://free3d.com/es/modelo-3d/chair-16205.html

printable_models. (2018, 06 de septiembre). Pescado v1 modelo 3d. Free3D. https://free3d.com/es/modelo-3d/fish-v1--996288.html