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: 320229594

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 05

SEMESTRE: 2026-1

FECHA DE ENTREGA LÍMITE: 25/11/2025

CALIFICACIÓN: _____

INDICE

Contenido

Manual técnico - Español

Planteamiento del proyecto:

Objetivo:

El objetivo general será crear un recorrido virtual e interactivo en OpenGL 3 que presente dos cuartos y una fachada ficticia que se apegue al material de referencia que en este caso será la caricatura: “Un show más”. Añadiendo elementos decorativos y funcionales por medio de código para que el entorno esté contextualmente correcto.

Puntos clave:

- Recrear dos habitaciones y una fachada ficticia.
- Apegarse al estilo artístico del material original.
- Realizar cuatro animaciones con contexto adecuado.
- Integrar una cámara sintética.
- Otorgar documentación al usuario para el uso del software
- Desarrollar un análisis técnico de la realización del proyecto.

Especificaciones mínimas para correr el programa:

Tomando en referencia las especificaciones de Visual Studio 2022 para correr un código en su propia aplicación necesitamos de:

Componente	Requisito mínimo
Sistema Operativo	Versiones de 64 bits de Windows 10 o Windows 11. No es compatible con sistemas operativos de 32 bits.
Procesador	ARM 64 o x64 con cuatro núcleos o superior. No se admiten procesadores ARM 32.
Memoria RAM	4 GB de memoria como mínimo, aunque se recomienda tener de 8 a 16 GB de memoria RAM
Espacio en Disco Duro	En el caso específico de este programa, el archivo .exe tiene un tamaño final de 375 KB, mientras que la carpeta general contiene 75.4 MB
Tarjeta de Video (o Tarjeta Gráfica)	Debe admitir una resolución mínima de WXGA (1366 x 768), aunque se

	recomienda una resolución estable de 1920 x 1080 o superior. Soporte de controladores compatibles para OpenGL

Justificación de herramientas a utilizar:

- Software de modelado y texturizado (Blender):

Los elementos 3d utilizados para el proyecto fueron optimizados tanto en su geometría general, como en su texturizado en el software Blender, esto debido a que es un software de libre y gratuito para uso personal, comercial y educativo, ahorrando así el costo de licencias. Además, por su simplicidad y la gran comunidad detrás del software es fácil encontrar soluciones a problemas que se presenten al usar dicho programa.

- Programa de edición de imágenes (Gimp):

Gimp es de uso libre y multiplataforma disponible para diversos sistemas operativos (GNU/Linux, Windows, macOS, entre otros), permitiendo que todo el entorno cooperativo pueda apoyar en la edición de imágenes necesaria. Es un software intuitivo y de nuevo, con una gran comunidad que se mantiene apoyando en los problemas que llegan a surgir.

- Sistema de control de versiones (Github):

Github permite la creación de repositorios que almacenan todos los avances realizados en un proyecto, resaltando los cambios con total transparencia, teniendo una gestión sin igual y lo más importante, permite el trabajo cooperativo de forma segura y además gratuita hasta cierto limite de almacenamiento. Sin embargo, en el caso especifico de nuestro proyecto, el tamaño de los archivos no sobrepasa este límite. Permitiendo tener así a todo el equipo desarrollador conectado y actualizado.

- Entorno de desarrollo (Visual Studio 2022):

Visual Studio 2022 ofrece un entorno intuitivo, rápido, confiable y bien integrado que facilita el desarrollo del proyecto con OpenGL, tanto en la programación, manejo de librerías, control de versiones y simplicidad al importar los elementos con ayuda de OpenGL 3.

Lineamientos del proyecto

Alcances:

- Realizar la documentación adecuada para mejorar la utilización del proyecto tanto en español como en inglés.
- Replicar dos habitaciones y una fachada con el mismo estilo artístico.
- Controles simples y movimiento libre en el recorrido virtual.
- Optimización de modelos y texturas por medio del software de modelado Blender.
- Ejecutable final que integre completamente los requisitos y objetivos del proyecto.

Metas:

- Recrear un espacio virtual 3D en OpenGL usando como referencia “la casa del parque” de la caricatura “Un show más”.
- Ambientación inmersiva correspondiente al universo del material original.
- Poner a disposición del público una versión funcional en GitHub.
- Garantizar que los usuarios puedan entender el uso del proyecto.
- Una aplicación ejecutable e intuitiva.

Hitos del proyecto:

- Elección de elementos a recrear
- Establecimiento de las herramientas a usar
- Optimización de modelos y re texturizado
- Creación del proyecto y elementos fundamentales en OpenGL
- Implementación de modelos y ajustes necesarios en OpenGL
- Animación de elementos
- Iluminación de espacios
- Optimización general
- Documentación técnica y creación del manual de uso
- Publicación del proyecto en GitHub

Metodología de software:

Se planteó utilizar la metodología ágil Scrum basada en sprints para desarrollar el proyecto, aunque en el planteamiento del proyecto no se utilizó al cien por ciento un sprint, se dividió el resto del proyecto en 3 sprints que permitieron dividir de forma adecuada y tranquila el desarrollo del proyecto. Consistiendo en:

Sprint 1. Planificación y modelado inicial: En 5 días se desarrolló el entregable de este sprint, que consistió en tener todos los modelos ya optimizados y re texturizados, para que al final del plazo, se pudiera configurar el entorno de desarrollo en Visual Studio, de esta forma preparándonos para empezar a programar en el sprint 2 con las clases base (como la formación de ventanas, shaders, lógica, carga de modelos y texturas, etc) ya implementadas.

Sprint 2. Desarrollo del entorno y funcionalidad principal: Con una duración de 7 días se buscó completar los elementos fundamentales para el entorno y para la interactividad, cargando los modelos y posicionándolos en su debido espacio según las fotos de referencia, además, colocando iluminación y realizando animaciones.

El resultado al terminar este sprint consistió en una versión funcional y con mínimos detalles a pulir del entorno virtual, con inmersión e interactividad incluidas.

Sprint 3. Pulido final y documentación: Con una duración de 6 días, el objetivo era mayoritariamente recabar lo realizado durante los anteriores plazos para así finalmente hacer el reporte final del proyecto. Sin embargo, hubo detalles menores en iluminación y modelos que se volvieron a trabajar, para brindar un mejor resultado. Siendo así que al final de este plazo se obtuvo una versión final, lo más pulida a criterio del equipo para después reunir todos los detalles del proyecto para así generar la documentación e informar sobre el aspecto técnico, interactivo y a su vez dándole al usuario una guía de uso funcional.

Diagrama de Gantt

Fase	Actividad	Días transcurridos
1	**Planificación:** * Plantear el proyecto a recrear * Buscar imágenes de referencia	2
2	**Modelado:** * Buscar modelos en Sketchfab * Re texturizado	6
3	**Inicio del código:** Adición de clases fundamentales:	1
4	**Desarrollo del entorno:** * Carga de modelos * Realización de animaciones * Iluminación * Asignación de posiciones	5
5	**Detalles finales:** * Re posicionamiento de modelos * Re texturizado	2
6	**Documentación:** * Desarrollo de manual de usuario * Realización del manual técnico	5
	Total de días:	21

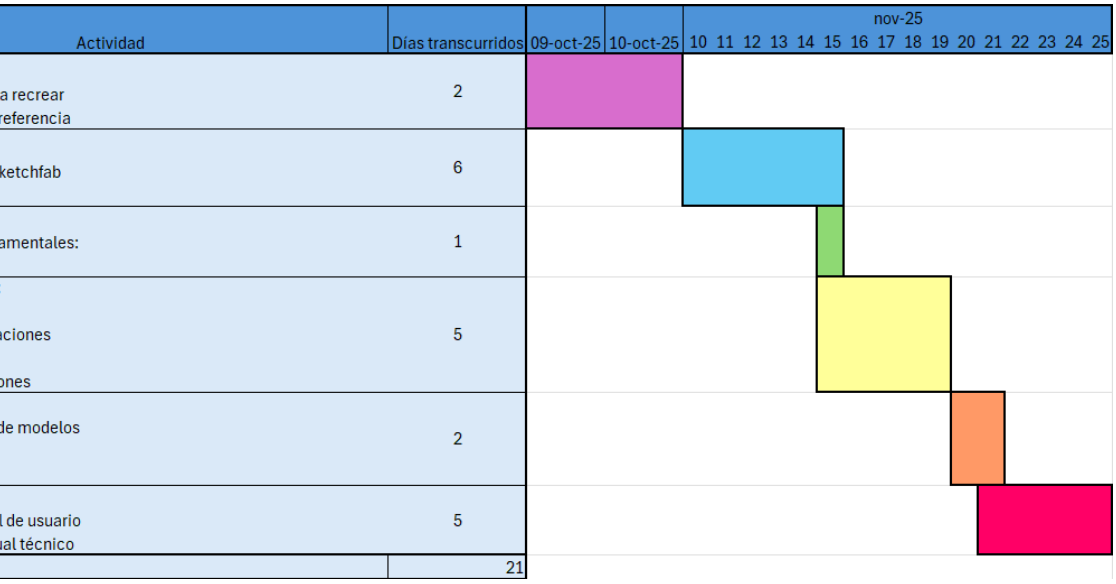
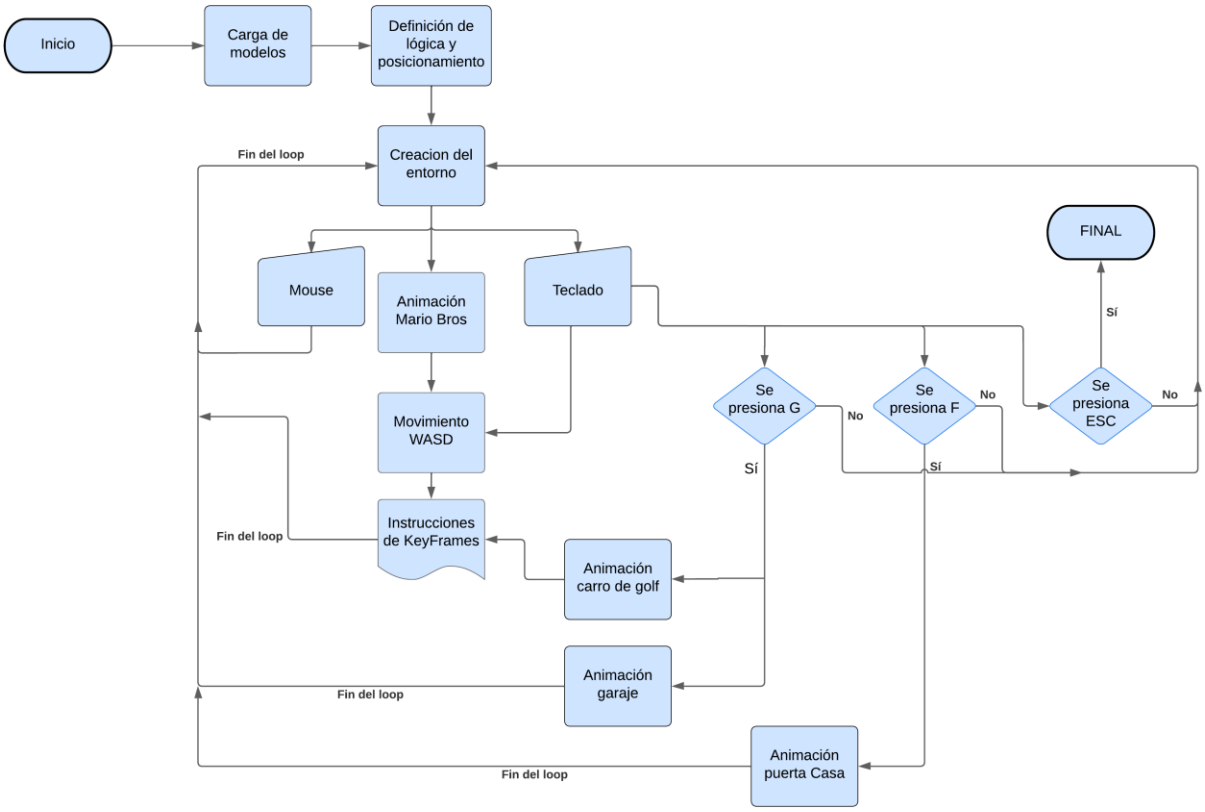


Diagrama de flujo:



El diagrama de flujo es bastante sencillo, debido a que todo se basa en un bucle principal en el que constantemente se esta dibujando el programa y sus elementos, por lo que realmente las entradas, procesos y comprobaciones se realizan una y otra vez hasta que el usuario decide terminar el programa con la tecla **esc** del teclado, sin embargo, es obvio que en cada proceso mostrado en este diagrama de flujo se esconden muchos subprocesos. Es por eso que más adelante se presentan las variables y funciones que permiten que el programa se desarrolle correctamente

Costeo del proyecto:

Personal

Rol	Días de trabajo	Ingreso diario	Subtotal
Scrum Master	19	$ 2,239.053	$ 42,542.00
Modelador 3D	6	$ 3,318. 28	$ 18,709.66
Programador C++	7	$ 2,257.15	$ 15,800.00
Artista de iluminación	3	$ 3,500.00	$ 10,500.00
Animador 3D	4	$ 3,343.20	$ 13,372.79
Tester	4	$ 2,226.25	$ 8,905.00
Documentador	5	$ 2,034.00	$ 10,170.55
TOTAL			$123,000

Softwares

Programa	Costo
Blender	$ 0
Visual Studio 2022	$ 0
Librerias de interfaz (OpenGL / GLEW / GLFW)	$ 0
Edición de imágenes (Gimp)	$ 0
Control de versiones (Git / Github)	$ 0
TOTAL SOFTWARE	**$ 0**

Hardware

Computadora personal Dell inspiron 5593 i7 10gen. Comprada en 2020 a aproximadamente $16,000.00 pesos mxn. en un solo pago con una depreciación anual de $2880.00 pesos mxn. Resultando en un equipo con valor actual de $1600.00 pesos mexicanos, por cada integrante del equipo, contemplando un integrante en cada rol, tenemos un costo total en hardware de $11,200.00 pesos mexicanos con la devaluación ajustada.

Gastos operativos

Concepto	Costo
Electricidad	$ 2,000.00
Internet	$ 4,000.00
Renta de oficina (1 mes)	$ 25,000.00
Servicios y mantenimiento	$ 3,500.00
TOTAL GASTOS OPERATIVOS	**$ 34,500.00**

Capacitación

Concepto	Costo
Capacitación en entorno de desarrollo (OpenGL).	$3,000.00
Capacitación en Git y GitHub	$ 750.00
Material de referencia	$ 0.00 Por consultas en foros e IA's
TOTAL CAPACITACIÓN	$ 3,750.00

Cotización final

Rubro	Total
Personal	$123,000.00
Software	$ 0.00
Hardware	$ 11,200.00
Gastos operativos	$ 34,500.00
Capacitación	$ 3,750.00
SUBTOTAL	**$ 172,450.00**
Fondo de emergencia (15% de los gastos)	$ 25,867.50
SUBTOTAL ANTES DE IMPUESTOS	**$ 198,317.50**
IVA (16%)	$ 31,730.80
TOTAL DEL PROYECTO	**$ 230,048.30**

En este ejercicio de costos se contempla la renta de una oficina que permita el coworking y el avance tanto individual como en equipo además de el uso de software y hardware accesible y funcional para el desarrollo adecuado entre todos los colaboradores.

Glosario de variables:

Las siguientes variables enlistadas sirven en el código main y dan la estructura necesaria para que el código tenga contexto adecuado y permita que el recorrido cobre vida.

Tipo de variable	Nombre	Descripción
Animaciones		
float	rotacion	Sirve en el ámbito de animaciones para rotar de forma realista la puerta de la Casa.
float	rotacion2	Esta asignada a la puerta del garaje del modelo de la casa y permite que se abra la puerta del garaje en 90 grados
float	deslizamiento	Ayuda a hacer un movimiento realista en la puerta del garaje, deslizando a la vez que rota, haciendo una apertura con sentido.
bool	derrape	Bandera que permite iniciar la animación de derrape en el carrito de golf. Esta bandera inicia dicha animación cuando se abre la puerta del garaje con la tecla G.
float	movOffset	Establece la velocidad base de las animaciones al multiplicar este valor por la variable deltaTime para generar sensación de suavidad.
float	toffsetnumerou	Marca la velocidad de desplazamiento en la coordenada u.
float	ftoffsetnumerov	Establece la posición de la textura en la coordenada v y ayuda a desplazar en dicha dirección a la textura.

float	toffsetnumerocambiau	Establece la posición de la textura en su coordenada u y ayuda a desplazar en dicha dirección a la textura.
float	reproduciranimacion	Permite iniciar el proceso de la animación por keyframes sólo cuando el mismo proceso empieza desde cero.
float	habilitaranimacion	Sirve como una bandera para identificar cuando se reprodujo adecuadamente la animación por keyframes y permite reiniciar el arreglo para volver a correr la animación.
GLfloat	deltaTime	Hace visible y posible la diferencia del tiempo entre el frame actual y el anterior. Permite las animaciones y hace que estas sean independientes del framerate.
GLfloat	lastTime	Guarda el tiempo del frame anterior.
static double	limitFPS	Establece el tiempo mínimo entre frames. (1/60).
float	posXcarro	Posición base del carro en el entorno virtual en x,y,z.
float	posYcarro	
float	posZcarro	
float	movcarro_x	Es la definición inicial de las variables para el movimiento que después se hará con la animación
float	movcarro_y	
float	movcarro_z	
float	movcarro_xInc	Son el desplazamiento que hará y se le sumará al valor base al carro en base a los keyFrames para así generar la animación.
float	movcarro_yInc	
float	movcarro_zInc	

float	girocarro	Representa el giro deseado que hace el carro durante la animación.
float	giroCarroInc	Permite incrementar y cambiar el valor del giro original.
Macro int	MAX_FRAMES	Número de cuadros máximos.
Int	i_max_steps	Número de pasos, entre cuadros para interpolación, a mayor número, más lento será el movimiento.
int	I_curr_steps	Indica cuantos pasos se han ejecutado entre el frame actual y el siguiente.
FRAME	KeyFrame[MAX_FRAMES]	Es un arreglo de estructuras que guarda la posición y rotación del carro junto a sus movimientos posteriores.
int	FrameIndex	El número de cuadros guardados actualmente desde 0 para no sobreescribir.
bool	play	Bandera que inicia la animación por keyframes cuando está en true.
int	playIndex	Indica que keyframe está interpolando y usando actualmente.
Modelos		
Tipo de variable	**Nombre**	**Descripción**
glm::mat4	model	Es una matriz 4x4 que permite asignarle a un objeto 3D para después ajustar sus transformaciones.
glm::mat4	modelaux	Una matriz auxiliar que ayuda a guardar temporalmente las

		transformaciones de un objeto.
glm::vec3	color	Un vector que guarda 3 valores, con lo que asigna en un formato RGB un color para un elemento.
glm::vec2	toffset	Sirve para controlar el avance de una animación en función del tiempo.
Model	Cualquier variable de tipo Model	Estas variables sirven para cargar modelos 3D desde una carpeta para utilizarlos en el entorno virtual.
Texture	Cualquier variable de tipo Texture	Estas variables cargan imágenes, o texturas que permiten agregar texturas para utilizarlas en objetos de geometría primitiva creados en OpenGL.
General		
Tipo de variable	**Nombre**	**Descripción**
Camera	camera	Define el comportamiento de la cámara obtenido de la clase Camera.
Skybox	skybox	Es un vector que carga imágenes para generar el skybox en otra clase.
Material	Material_brillante	Guarda las características de iluminación que afectan a un material para hacerlo brillante.
Material	Material_opaco	Guarda las características de iluminación que afectan a un material para hacerlo opaco.

DirectionalLight	mainLight	Define la luz direccional que simula al “sol” en el entorno virtual.
PointLight	pointLight[]	Arreglo que contiene todas las luces de tipo PointLight del entorno.
SpotLight	spoLight[]	Arreglo que contiene todas las luces de tipo spotLight del entorno.
Static const char	vShader	Carga el vertex shader para utilizar elementos de iluminación y creación de objetos.
Static const char	fShader	Carga de fragment shader para añadir texturas y colores a los objetos.

Glosario de funciones:

A continuación se enlistan las funciones más importantes para el desarrollo del código y que interactúan exclusivamente en el archivo main del proyecto, exceptuando las mismas ya incorporadas en OpenGL, pues estas tienen su propia documentación realizada por sus autores.

Tipo de función	**Nombre de la función**	**Parámetros recibidos**	**Descripción**
Global y void	calcAverageNormals	Índices, indiceCount, vértices, verticeCount, vLenght, normalOffset	En base a las propiedades de un objeto realizado en OpenGL se calculan las normales promedio por vértice para modelos hechos de triángulos. Sumando primero las normales de cada cara que toca un vértice y luego normaliza el resultado.

Funcion global, void	CreateObjects()	Ninguno	Crea varias mallas Mesh, usando arreglos de vértices e índices. Como triángulos, el piso, o el recorrido de la textura de videojuego Mario Bros en la pantalla (uso sin fines de lucro).
Función global, void	CreateShaders()	Ninguno	Crea un objeto Shader, cargando los shader .vert y .frag y lo añade al vector shaderList.
Global, le sirve a los keyframes, void	resetElements()	Ninguno	Restablece la posición y rotación del carro al valor del keyFrame 0 antes de reproducir la animación.
Global, void	interpolation()	Ninguno	Calcula los incrementos y pasos entre un keyframe y el siguiente.
Global, void	animate()	Ninguno	Una máquina de estados que permite animación por keyframes cuando la bandera play es true. Avanzando los pasos de animación.
Global, void	inputKeyFrames()	bool getKeys	Espera entradas del teclado para iniciar animación, reiniciar la animación y bloquear la reproducción múltiple para los KeyFrames.
Función principal del programa (INT)	main()	Ninguno	Inicializa la ventana con GLFW y GLEW, carga los shaders, carga modelos, mallas y texturas, inicializa luces, configura parámetros para animaciones y es el loop inicial que calcula la posición y carga de elementos.

MODEL			
Tipo de función	**Nombre de la función**	**Parámetros recibidos**	**Descripción**
Void, clase Model	LoadModel()	String	Ruta relativa del archivo de modelo.
Void, clase model	RenderModel ()	Ninguno	Se encarga de cargar el modelo y mostrarlo en el entorno virtual creandole una malla y asignándole sus texturas.
CAMERA			
Tipo de función	**Nombre de la función**	**Parámetros recibidos**	**Descripción**
Camera, void	camera()	Vec 3 Posición inicial Vec3 arriba inicial GLfloat rotación GLfloat paso inicial GLfloat velocidadInicial GLfloat velocidadGiroInicial	Coloca la cámara en el entorno y la inicializa apuntando a donde se desee.
Camera, void	keycontrol()	bool key, GLfloat deltaTime	Verifica si se presionaron las teclas para el movimiento de la cámara y actúa de acuerdo a los inputs del teclado con w,a,s,d.
Camera, void	mouseControl ()	GLfloat xChange, GLfloat yChange	Ajusta la posición de la cámara de acuerdo con el movimiento que hace el usuario con el mouse.
Camera, glm::vec3	getCameraDir ection()	Ninguno	Normaliza el apuntado de la cámara.
Camera, mat4	calculateView Matrix()	Ninguno	Utilizando lookAt ajusta el punto visible de la cámara desde su posición actual.
Skybox			

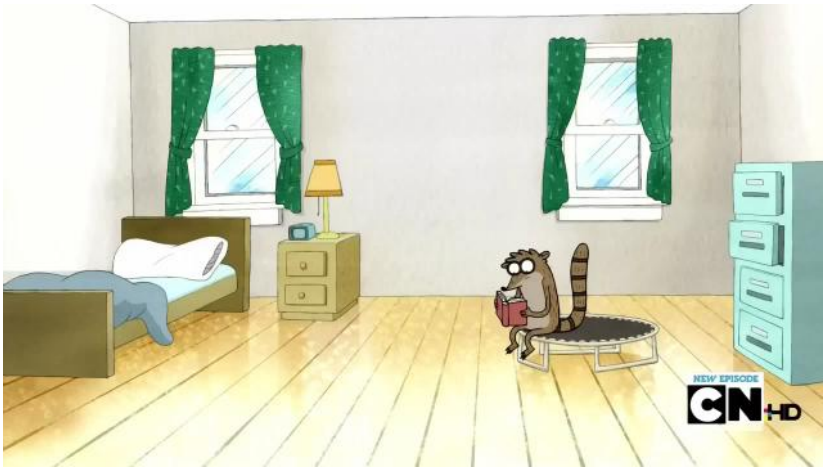
Tipo de función	Nombre de la función	Parámetros recibidos	Descripción
Skybox, void	DrawSkybox()	viewMatrix, projectionMatrix	En base a la matriz de visualización y de proyección coloca las texturas en un cubo gigante, creando así el skybox de tal forma que el usuario se mueva con ellas y nunca sean alcanzables.
Window			
Tipo de función	**Nombre de la función**	**Parámetros recibidos**	**Descripción**
GLboolean	getAbrirCerrarPuerta()	Ninguno	Inicia con una bandera el proceso de animación de abrir o cerrar la puerta de la casa.
GLboolean	getAbrirCerrarGaraje()	Ninguno	Inicia con una bandera el proceso de animación de abrir o cerrar la puerta del garaje.

Elementos y referencias del proyecto:

Fachada: referencia perteneciente al programa de televisión “un show más”, recreada en el proyecto final

Para los cuartos se hará una recreación el cuarto de los personajes principales (Mordecai y Rigby), la imagen de referencia para tal habitación es la siguiente:

Cuarto 1:



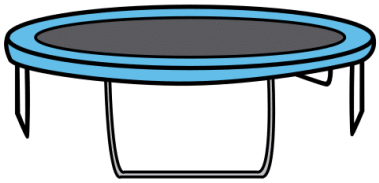
Los elementos para recrear en el cuarto son los mismos que aparecen en la imagen de referencia y para tenerlos mucho más claros, las imágenes de referencia son:

1. Cama:

2. Buró:

3. Lámpara de mesa:

4. Cama elástica miniatura:



5. **Cajonera:**

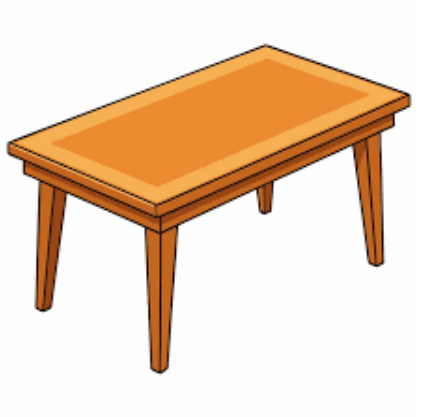
2do Cuarto. Sala de la casa:

Se buscará recrear los mismos elementos que aparecen en las imágenes de referencia, los cuales son:

1. Sofá:

2. Televisión:

3. Mesa de centro:



4. Armario:

5. Cuadro:

6. Atari (extra):

Resultados:

Después de presentar la organización del proyecto, la estrategia de desarrollo, los costos, el material original y las referencias, sólo queda visualizar el entorno 3D finalizado y sus diferentes elementos interactivos, pero primero, desglosaremos la parte técnica para comprender el resultado obtenido y con ello explorar el entorno virtual, sabiendo que encontraremos con precisión en cada uno de sus espacios.

Modelado 3D: En el entorno virtual podemos visualizar modelados 3D que tienen texturas que se apegan lo más posible al material original, además, usando mobiliario suficiente para decorar adecuadamente los espacios, desde sillones, cuadros, una lampara de mesa y las camas de los personajes de la serie, logrando una integración lo más fiel posible.

Animaciones: Encontramos aquí la interactividad necesaria para el usuario, permitiéndole usar 4 animaciones que hacen sentir viva la casa, desde la bienvenida al entorno podemos abrir o cerrar la puerta que nos da una entrada directa a la sala, para ahí mismo ver un juego de Mario Bros corriendo en la televisión. Mientras que afuera de la casa, podemos ver que la puerta del garaje puede ser abierta y cerrada con deslizamiento y apertura simultáneamente, y a su vez, al abrir la puerta de garaje, se activa la animación por keyframes de un carrito de golf que sale a derrapar, para después volver a guardarse.

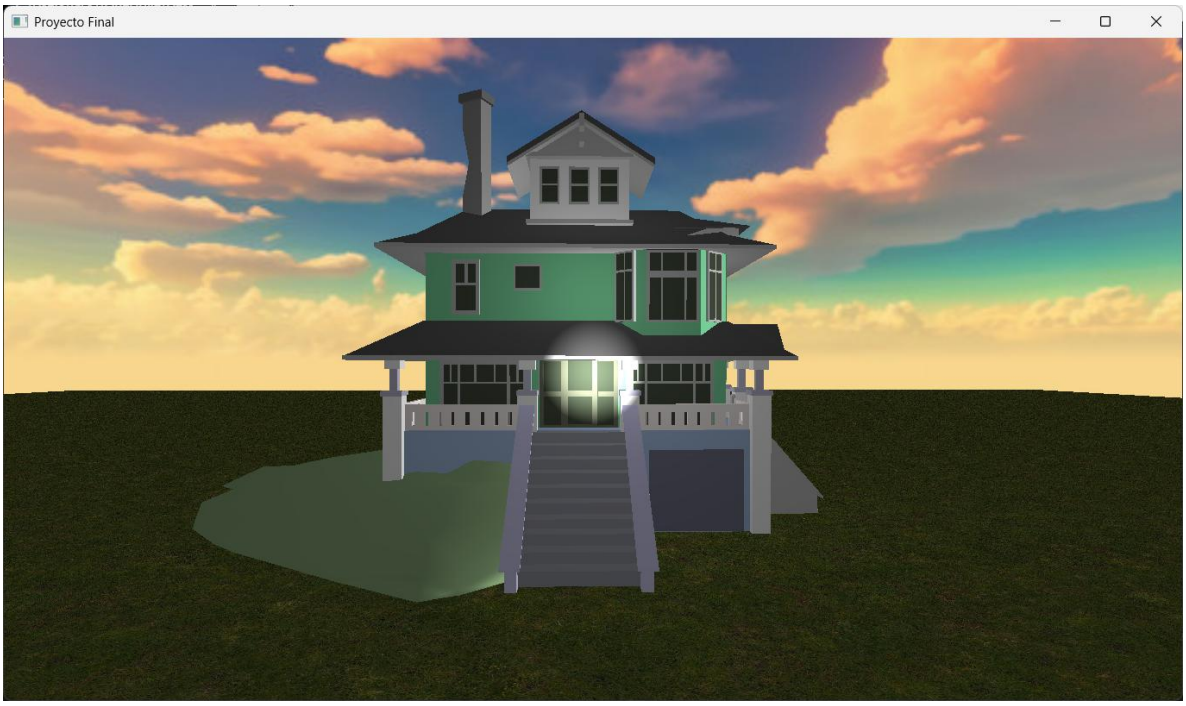
Recorrido: El usuario puede usar la cámara sintética para moverse de manera libre en el exterior de la casa y adentro de la misma, para así observar con detalle cada espacio.

Integración total: Se agrupa completamente el uso de librerías de generación de ventanas, de dibujo gráfico (OpenGL), de interfaz (GLFW), de texturizado (SOIL, stb_image), de carga de modelos (ASSIMP), clases que se encargan de la lógica por separado de procesos para hacer un conjunto de todas y con ello facilitar la programación y rendimiento estable, así como legible de la programación. Facilitando la creación del recorrido con cada línea bien documentada en su función.

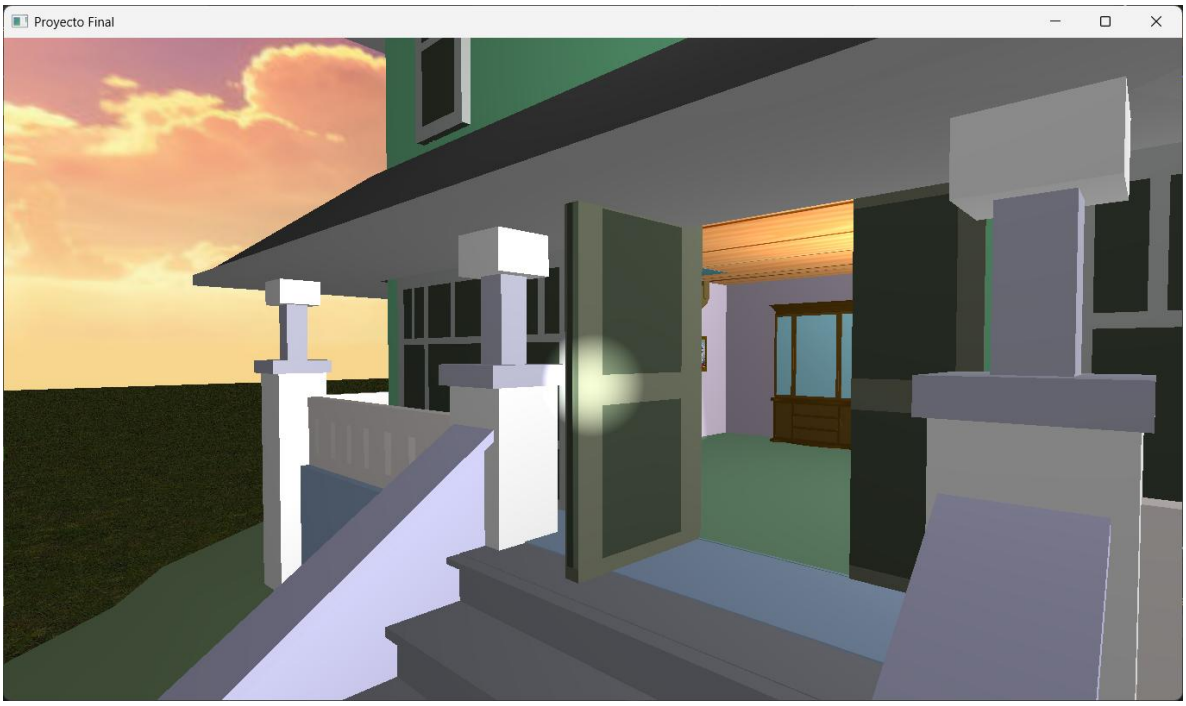
Iluminación realista: Contemplando que es un entorno de interiores, se aplico una iluminación en la cual, el usuario se puede sentir realmente como en una casa, con fuentes de luz realistas y contextualmente correctas.

Finalmente, los elementos incorporados en el programa se ven así:

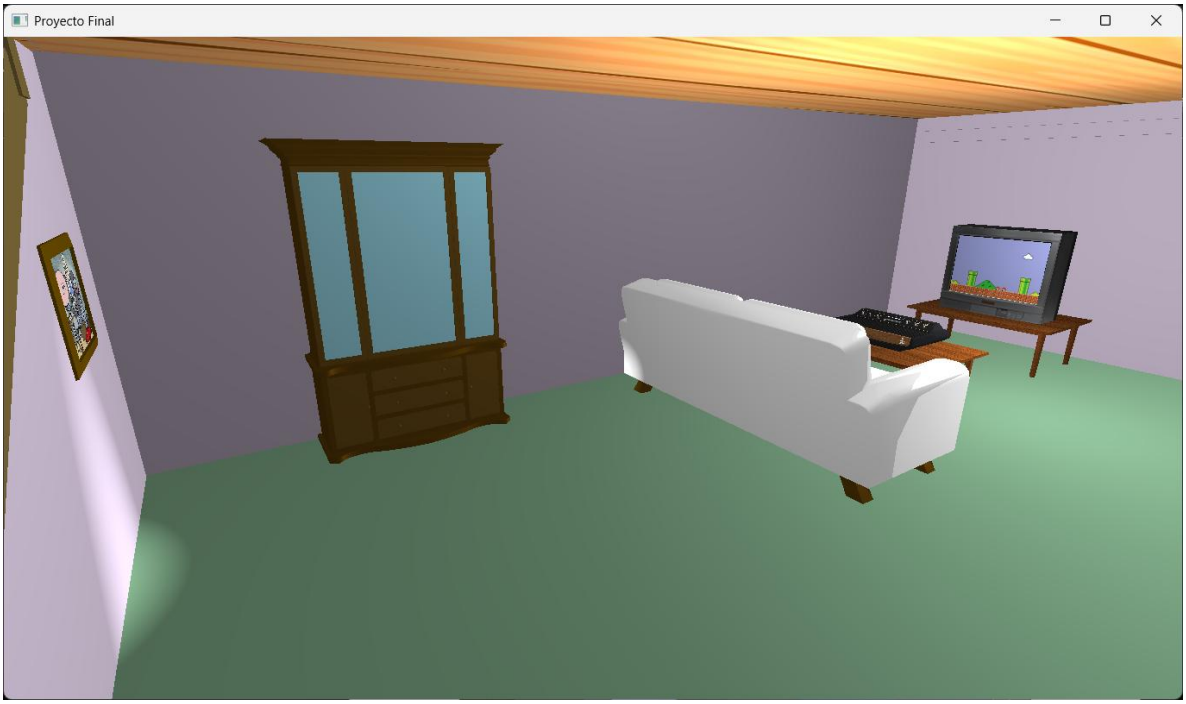
Casa desde perspectiva frontal.



Primera animación, abrir y cerrar la puerta con la tecla G.

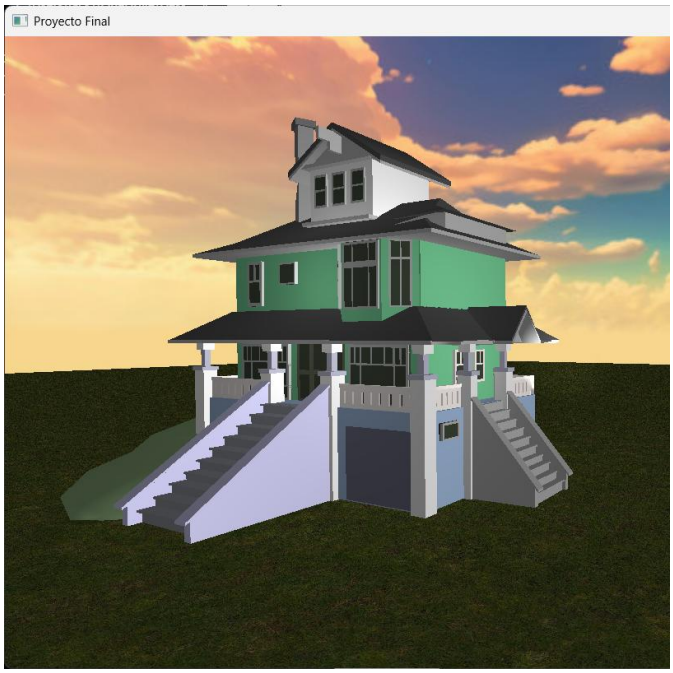


Decoración y 5 elementos de la sala: Tv y (su animación con el primer nivel de Mario corriendo), Mesa, sillón, gabinete y cuadro de personajes de Un show más.

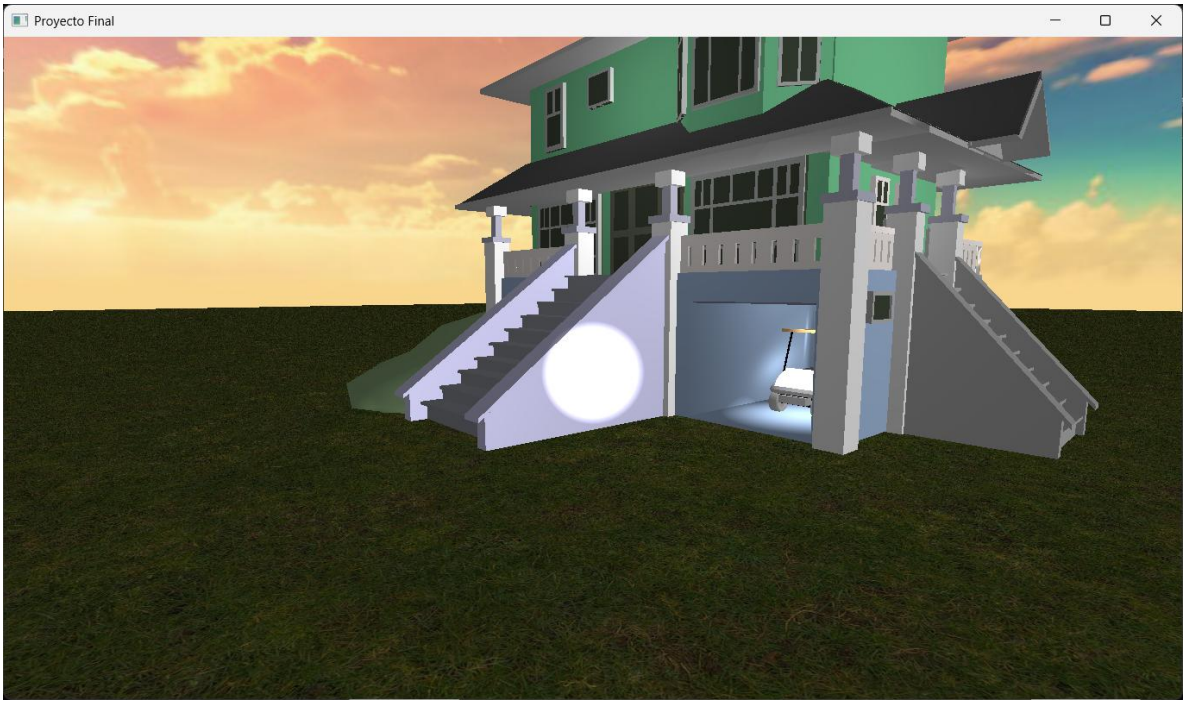


2do cuarto: Cama, buró, lámpara de mesa, trampolín y cajonera.

Vista de 45 grados de la entrada y una pared

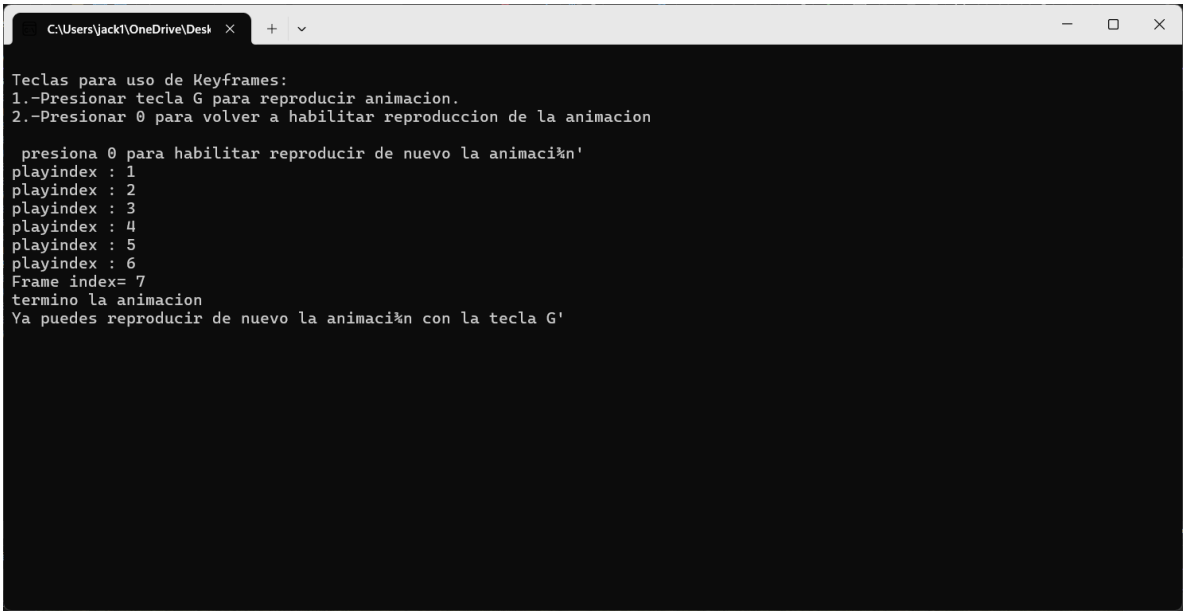


Apertura del garaje como tercera animación, revelando el carrito de golf y su animación por keyframes, aunque en la siguiente imagen aparece la animación terminada.



Vista de 45 grados desde atrás y un lateral.

Vista de la terminal para activar la animación de keyframes y el proceso de reactivarla para volver a utilizarla.



Conclusiones:

En la realización del proyecto se aplicaron cada uno de los conocimientos adquiridos en el curso, tanto en su parte teórica, como en su parte práctica en laboratorio, permitiendo hacer un proyecto dentro de los parámetros pedidos, permitiendo echar a volar la imaginación para crear un entorno interactivo y con un recorrido adecuado para el usuario. Al mismo tiempo mejorando las habilidades de programación para emplear lógica en animaciones y transformaciones de los espacios tridimensionales. Reafirmando y cerrando el curso con un recopilatorio satisfactorio de los temas aprendidos al ponerlos en práctica.

Bibliografía:

Cuarto Mordecai y Rigby:

- Donnichols. (2022, 8 de junio). *Old Bed*. Sketchfab https://sketchfab.com/3d-models/old-bed-210be84bcb5449f5a9f66a923c8ae307
- Modelo de buró creado por mí.
- uwurkowo. (2022, 14 de septiembre). *Night lamp*. Sketchfab https://sketchfab.com/3d-models/night-lamp-5a7532792fe04863af72e9cdbb7db65a

- Shahsavan M. (2024, 8 de junio). *Filing Cabinet – 6 MB*. Sketchfab. https://sketchfab.com/3d-models/filing-cabinet-6mb-d217a4bbdfa2426eb32ebf3b1007a8c3
- 1-3D.com. (2022, 25 de marzo). *Trampoline*. Sketchfab. https://sketchfab.com/3d-models/trampoline-052aaef6c5c34c5bac94607bcc12d2f9

Modelos de sala:

- Snomss. (2021, 3 de mayo). *Old Reliable Couch*. Sketchfab. https://sketchfab.com/3d-models/old-reliable-couch-c9a365e5ba9f456db4887dc1e5f4b5d5
- Tomitos. (2025, 8 de junio). *PSX CRT TV*. Sketchfab. https://sketchfab.com/3d-models/psx-crt-tv-14c4034011ff4faebd99de03c7ab1635
- amir.acc129am. (2023, 30 de marzo). *Living room table*. Sketchfab. https://sketchfab.com/3d-models/living-room-table-3c4337eb6f36476b9dada4b91bc6b4c6
- IvanovichRU. (2021, 4 de mayo). *Old China Cabinet*. Sketchfab. https://sketchfab.com/3d-models/old-china-cabinet-a6448e3833bf4ec68c0a919ece796ce2
- //A. (2024, 24 de junio). *Photo frame.* Sketchfab. https://sketchfab.com/3d-models/photo-frame-e9c61ed2e07940bebee2faa9920b8335
- (EXTRA) Dark_igorek. (2024, 20 de octubre). *Atari 2600.* Sketchfab. https://sketchfab.com/3d-models/atari-2600-2024d933f6214117a399ad4287ede64d

Modelos no implicitos en los cuartos (extra):

- Shahsavan. M. (2025, 8 de febrero). *Wooden Stairs_5_MB.* Sketchfab. https://sketchfab.com/3d-models/wooden-stairs-5-mb-56271a83e8ff487b84289e39a159d3c5
- apolodz. (2020, 21 de mayo). *Low Poly GolfCart*. Sketchfab. https://sketchfab.com/3d-models/low-poly-golfcart-5233072cb81145b99a06eaa94eafdd2e
- RMRO2744. (2025, 30 de junio). *Un_show_mas_casa.* Sketchfab. https://sketchfab.com/3d-models/un-show-mas-casa-f64cda605b9e40a0a1e52ef16450b90a

Texturas:

Obtenidas con gimp de colores planos.

Y algunas obtenidas de internet con los modelos.

TECHNICAL MANUAL - ENGLISH

Project proposal:

Objective:

The general objective will be to create a virtual interactive environment in OpenGL 3.3 that presents two rooms and a fictional facade that resembles the reference material, that in this case will be the cartoon: “regular show”. Adding decorative and functional ítems using code in a way the final environment is contextually right.

Key points:

- Recreate two romos and a fictional facade.
- Stick the original artistic style from the reference material.
- Make four animations with appropiate context.
- Add a synthetic camera.
- Give enough documentation to the user for them to use the software.
- Develop a technical analisys from the realization of the project.

Minimum specifications to run the program:

Taking into reference the specs that Visual Studio 2022 has to run a code in his own application, we need:

Componente	Requisito mínimo
Operative System	Versions of 64 bits of Windows 10 or Windows 11. It’s not compatible with OS that uses 32 bits.
Processor	ARM 64 or x64 with four cores or above. ARM 32 processors are not allowed.
RAM Memory	Minimum 4 GB of memory, though it’s highly recommended using between 8 and 16 GB of RAM memory.
Space in hard drive	On the specific case of this program, the .exe file has a final size of 375 KB, meanwhile, the general folder has a size of 75.4 MB.
Video card (or Graphic card)	It must admit a mínimum resolution of WXGA of (1366 x 768), although a

	minimum resolution of 1920 x 1080 or above is recommended as a stable resolution. It also has to have compatible drivers for OpenGL.

Justification of tools to be used:

- Modeling Software and texturing (Blender):

The 3D items used for the project were optimized both for general geometry and texturing on the blender software, this is because of the program being free for personal, commercial and educational purposes. In this way saving the cost of licenses from another software. Besides, for its simplicity and the great community behind this software it is easy to find solutions to problems that can appear when using said program.

- Image editor software (Gimp):

Gimp is a free to use and cross-platform available for different OS (GNU/Linux, Windows, macOS, amongst others), allowing everyone on the cooperative environment to help on the editing if necessary. It's an intuitive software with a great community behind that make support on every problem that may arise.

- Version control systems (Github):

Github allows the creation of repositories that store all the progress made on a project, highlighting the changes with transparency, having a clean management of every file uploaded, and the most important feature, allowing the cooperative work safely and free up to a certain amount of storage. However, in the specific case of our Project, the size of the files don't go above the limit, allowing in this way for all the developers in the team to be connected and updated.

- Development environment (Visual Studio 2022):

Visual Studio 2022 offers an intuitive, fast, reliable and well-integrated environment that makes the development much easier and the integration of libraries such as OpenGL, allowing all the possibilities to make great programming, using the version control and link it to our git and GitHub for simplicity on the development.

Project guidelines:

Scope:

- To realize the adequate documentation for the utilization of the Project both in Spanish and in English.
- Replicate two rooms and a facade with the same artistic style.
- Use simple controls and free movement on the virtual space.
- Optimize models and textures using the software, Blender.
- Final app capable of integrating all the requirements and objectives of the Project.

Goals:

- Recreate a virtual 3d space in OpenGL using as a reference the “Park house” from de cartoon “regular show”.
- Inmersive ambientation corresponding to the universe from the original material.
- Make available to the public a functional versión in GitHub.
- Guarantee that every user can understand the use of the Project.
- A running and intuitive application.

Project milestones:

- Choosing ítems to recreate.
- Establishing the tools to use.
- Optimize models and re texturing.
- Creation of the Project and fundamental classes in OpenGL.
- Implementation of models and necessary adjustments in OpenGL.
- Animations.
- Iluminating spaces.
- General Optimization.
- Technical documentation and creation of the user’s guide.
- Publishing the Project on GitHub.

Software Methodology:

This Project was made using the agile methodology Scrum, based on sprints to develop the project, though the approach of the project wasn't a part of any sprint, the total of the project was divided in 3 sprints that allowed the subdivision of tasks making the development calmed and easy. The sprints consisted of:

Sprint 1. Planification and initial modeling: In 5 days the result obtained from this sprint consisted on having every optimized model with their respective textures. Having a solid base to begin the configuration of the development environment in visual studio, in this way preparing everything for the making of the second sprint, with all of the basic classes (such as the window, shaders, logic, loading of models and textures, etc) implemented.

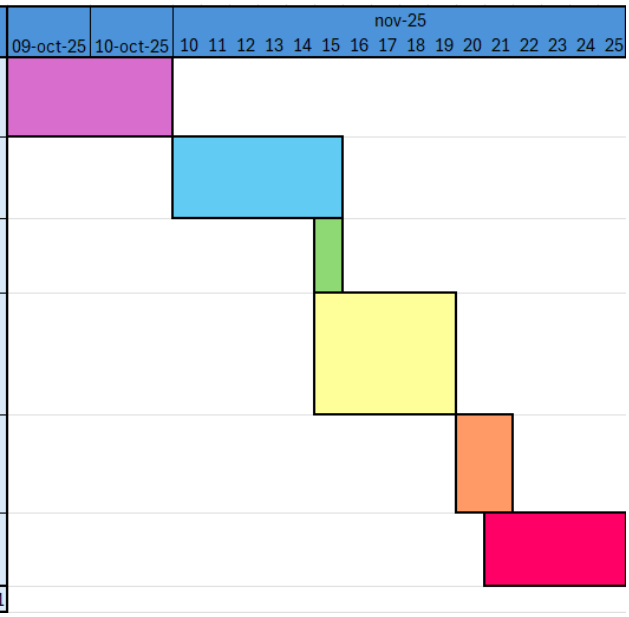
Sprint 2. Environment development and main functionalities: with a 7-day duration, in this sprint the goal was to complete every main item for the environment and the interactivity, loading the models and positioning them on their due space according to the reference pictures, also, setting de illumination and making the animations.

The result at the ending of this sprint was a functional and with minimum details to finish, having immersion and interactivity included.

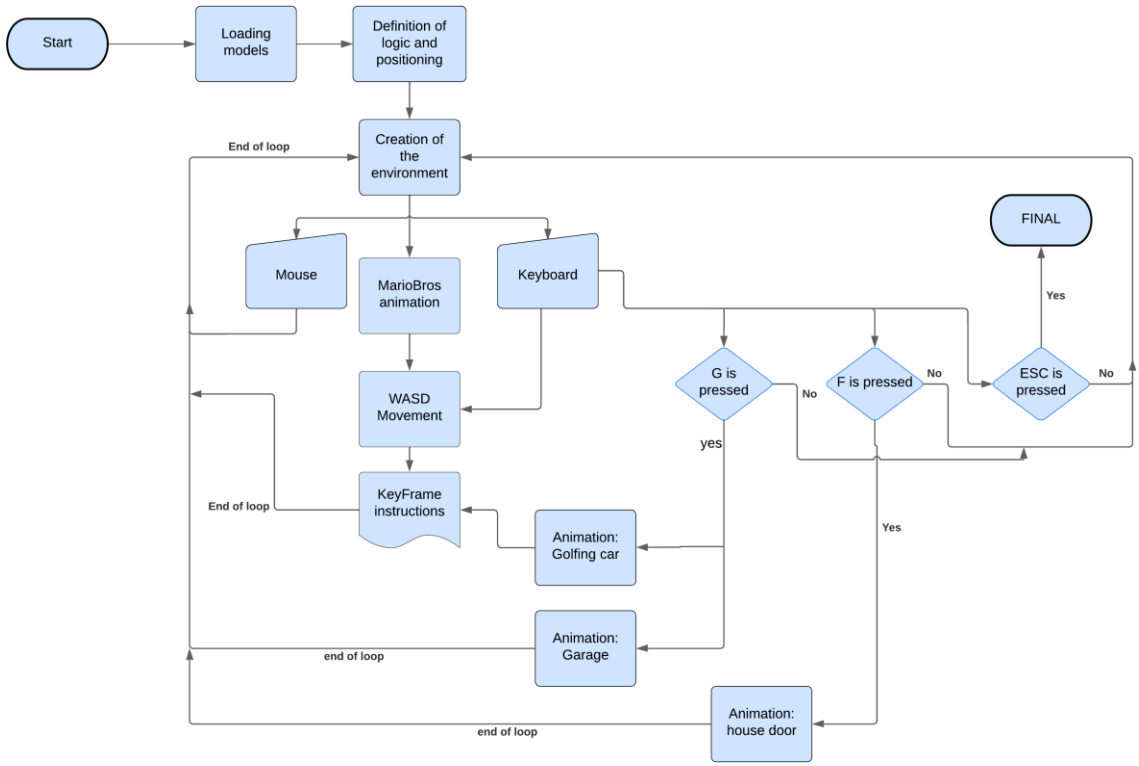
Sprint 3. Final details and documentation: with 6 days for this sprint, the objective was mostly to collect what was done in the last sprints for us to make the documentation. However, there were minor details in ilumination and modeling that needed to be worked again to obtain a better result. So, by the end of this sprint we had a final version of the code, really optimal for the team criteria, and once this version was finished the result were collected to make the documentation and inform about the technical aspects, the interactive items and also giving the user a guide for its use.

Gantt's Diagram:

Stage	Activity	Days elapsed
1	**Planification:** * Define the project to recreate * Search of reference images	2
2	**Modelating:** * Search for models in sketchfab * Re texturing	6
3	**Begining of the code:** * Adding fundamental classes:	1
4	**Development of the environment:** * Loading of models * Realization of animations * Illumination * Position asignments	5
5	**Final details:** * Re positioning of models * Re texturing	2
6	**Documentation:** * Realization of the user manual *Realization of the technical manual	5
	Total days:	21



Flowchart:



The flowchart it's pretty easy, this because all the code is based on a main loop with a while statement in which all of the 3D items are being drawn at the same time that all of the input checks are happening over and over again until the user decides to finish the program with the key **esc** in the keyboard**.** However, it's obvious that each process in this flowchart hides a lot of subprocesses, this can be seen later in the category glossary of variables and glossary of functions.

Project's costs:

Personal

Rol	Days of work	Daily salary	Subtotal
Scrum Master	19	$ 2,239.053	$ 42,542.00
3D modeler	6	$ 3,318. 28	$ 18,709.66
C++ programmer	7	$ 2,257.15	$ 15,800.00
Ilumination artist	3	$ 3,500.00	$ 10,500.00
3D Animator	4	$ 3,343.20	$ 13,372.79
Tester	4	$ 2,226.25	$ 8,905.00
Documenter	5	$ 2,034.00	$ 10,170.55
TOTAL			$123,000

Softwares

Software	Cost
Blender	$ 0
Visual Studio 2022	$ 0
Interface libraries (OpenGL / GLEW / GLFW)	$ 0
Image editor (Gimp)	$ 0
Version controller (Git / Github)	$ 0
TOTAL SOFTWARE	**$ 0**

Hardware

Personal computer Dell Inspiron 5593 i7 10th gen. purchased in 2020 at approximately $16,000.00 pesos mxn. On a single exhibition with an anual depreciation of $2880.00 pesos mxn. Resulting on a device valued on $1600.00 pesos mxn. For each team member, contemplating a member on each rol, we have a hardware cost of $11,200.00 pesos mxn with devaluation.

Operative expenses

Concept	Costs
Electricity	$ 2,000.00
Internet	$ 4,000.00
Office rental (1 month)	$ 25,000.00
Maintenance and services	$ 3,500.00
TOTAL OPERATING EXPENSES	**$ 34,500.00**

Staff training

Concept	Costs
Training in development environment (OpenGL).	$3,000.00
Git and GitHub training	$ 750.00
Reference material	$ 0.00 for consults and Usage of AI
TOTAL STUFF TRAINING	**$ 3,750.00**

FINAL

Rubro	Total
Personal	$123,000.00
Software	$ 0.00
Hardware	$ 11,200.00
Operative expenses	$ 34,500.00
Staff training	$ 3,750.00
SUBTOTAL	**$ 172,450.00**
Emergency fund (15% of expenses)	$ 25,867.50
SUBTOTAL BEFORE TAXES	**$ 198,317.50**
IVA (16%)	$ 31,730.80
PROJECT'S TOTAL	**$ 230,048.30**

En este ejercicio de costos se contempla la renta de una oficina que permita el coworking y el avance tanto individual como en equipo además de el uso de software y hardware accesible y funcional para el desarrollo adecuado entre todos los colaboradores.

In this cost exercise considers the rental of an office that allows coworking and progress both individually and as a team, as well as the use of accessible and functional software and hardware for the proper development among all collaborators.

Glosary of variables:

The following listed variables are used in the main code and provide the necessary structure for the code to have adequate context and allow the path to come to life.

Variable type	Name	Description
Animations		
float	rotacion	It is used in the animations to realistically rotate the door of the house.
float	rotacion2	It's asigned to the door of the garage and allows the 3d model to rotate 90 degrees to open or close this door.
float	deslizamiento	Ithelps to make a realistic movement in the garage door, sliding while rotating, making a meaningful opening of the garage.
bool	derrape	It's a flag that triggers the golfing cart's drift animation. The animation starts when the garage door is opened with the G key.
float	movOffset	It sets the base speed of the animations by multiplying this value by the deltaTime variable to generate a feeling of smoothness.
float	toffsetnumerou	Sets the speed of displacement on an image in the U coordinate.
float	ftoffsetnumerov	Set the position of the texture on the V coordinate and help in the movement of said direction in the texture
float	toffsetnumerocambiau	Set the position of the texture on the U coordinate and help in the movement of said direction in the texture

float	reproduciranimacion	It allows to start the keyframe animation process only when the process itself starts from scratch.
float	habilitaranimacion	It serves as a flag to identify when the animation was played correctly by keyframes and allows the array to be reset to run the animation again.
GLfloat	deltaTime	It makes the time difference between the actual and previous frames visible and possible. It allows animations and makes them independent of the frame rate.
GLfloat	lastTime	Saves the time from the last frame.
static double	limitFPS	Sets the mínimum time between frames. (1/60).
float	posXcarro	It's the initial position of the golfing car on the virtual environment on x, y and z coordinates.
float	posYcarro	
float	posZcarro	
float	movcarro_x	It's the initial definition of the variables for the movement that will later be made with the animation by keyframes.
float	movcarro_y	
float	movcarro_z	
float	movcarro_xInc	They are the displacement that will be made and will be added to the base value of the car based on the keyframes in order to generate the animation.
float	movcarro_yInc	
float	movcarro_zInc	
float	girocarro	It represents the desired turn that the car makes during the animation.

float	giroCarroInc	It allows to increment and change the value of the original turn.
Macro int	MAX_FRAMES	Number of max frames.
Int	i_max_steps	Number of steps between frames to interpolate, the higher the number, the slowest the movement will be.
int	I_curr_steps	It tells how many steps has been executed between the current frame and the next one.
FRAME	KeyFrame[MAX_FRAMES]	It's an array of structures that saves the position and rotation of the car along to their subsequent movements.
int	FrameIndex	The number of frames saved, starting from 0 to avoid overwriting.
bool	play	A flag that starts the animation with keyframes when it's true.
int	playIndex	Indicates which keyframe it's currently interpolating and using.
Models		
Variable type	**Name**	**Description**
glm::mat4	model	A 4x4 matrix that allows to assign an 3D object to later adjust with transformations.
glm::mat4	modelaux	An auxiliar matrix that helps save temporarily a 3D model and its transformations .
glm::vec3	color	A vector that saves 3 values, which will assign as a RGB format to give one item some specific color.

glm::vec2	toffset	Controls the progress of an animation in function of the time.
Model	Cualquier variable de tipo Model	These variables help to load 3D models from a selected folder and file to use them on a virtual environment.
Texture	Cualquier variable de tipo Texture	These variables load images, or textures that allows to use in objects made out of primitive geometry, created in OpenGL.
General		
Variable type	**Name**	**Description**
Camera	camera	Defines the behaviour of the camera, obtained from the camera class.
Skybox	skybox	It is a vector that loads images to generate the skybox on another class.
Material	Material_brillante	Saves the characteristics of illumination that affects a material to make it shiny.
Material	Material_opaco	Saves the characteristics of illumination that affects a material to make it obscure.
DirectionalLight	mainLight	Defines the directional light to simulate the “sun” on the virtual environment.
PointLight	pointLight[]	Array that contains all of the lights that are PointLight type from the environment.
SpotLight	spoLight[]	Array that contains all the spotlight-type lights in the environment.

Static const char	vShader	Loads the vertex shader to use items of iluminations and creations of objects.
Static const char	fShader	Loads the fragment shader to add textures and colors on the objects.

Glosario de funciones:

The following list includes the most important functions for code development that interact exclusively in the project's main file, excluding those already incorporated in OpenGL, as these have their own documentation created by their authors.

Function type	**Function name**	**Received parameters**	**Description**
Global y void	calcAverageNormals	Índices, indiceCount, vértices, verticeCount, vLenght, normalOffset	Based on the properties of an object created in OpenGL (vertices and indices), the average vertex normals are calculated for models made of triangles. This is done by first adding up the normals of each face that touches a vertex and then normalizing the result.
Funcion global, void	CreateObjects()	None	Creates various Meshes using arrays of vertices and indices. As triangles, the floor, or the image for the game of Mario Bros on the tv screen (using it with no-profit intentions).
Función global, void	CreateShaders()	None	Creates a Shader object, loading the vertex shader and the fragment shader adding them to the vector ShaderList.

Global, le sirve a los keyframes, void	resetElements()	None	Reset the golfing car's position and rotation to the keyFrame value 0 before playing the animation.
Global, void	interpolation()	None	Calculates the increments and steps between one keyframe and the next.
Global, void	animate()	None	A state machine that allows animation by keyframes when the play flag is true. Advancing the animation steps.
Global, void	inputKeyFrames()	bool getKeys	Wait for keyboard inputs to initiate animations, reboot the animation and block the multiple reproduction for the keyframes.
Función principal del programa (INT)	main()	None	Initializes the window with GLFW and GLEW, loads shaders, loads models, meshes and textures, initializes lights, configures parameters for animations and is the initial loop that calculates the position and loading of elements.
MODEL			
Function type	**Function name**	**Received parameters**	**Description**
Void, clase Model	LoadModel()	String	Relative route of the model file.
Void, clase model	RenderModel()	None	It is responsible for loading the model and displaying it in the virtual environment by creating a mesh and assigning its textures.
CAMERA			
Function type	**Function name**	**Received parameters**	**Description**

Camera, void	camera()	Vec 3 Posición inicial Vec3 arriba inicial GLfloat rotación GLfloat paso inicial GLfloat velocidadInicial GLfloat velocidadGiroInicial	Set the camera on the environment and initializes it pointing to where we desire.
Camera, void	keycontrol()	bool key, GLfloat deltaTime	Checks for input on the keys for the movement of the camera and acts according to the inputs of the keyboard with W, A, S, D.
Camera, void	mouseControl()	GLfloat xChange, GLfloat yChange	Adjusts the position of the camera according with the movement that the user makes with the mouse.
Camera, glm::vec3	getCameraDirection()	None	Normalizes the camera pointing.
Camera, mat4	calculateViewMatrix()	None	Using the function lookAt adjusts the visible point of the camera from its current position render what's at sight.
Skybox			
Function type	**Function name**	**Received parameters**	**Description**
Skybox, void	DrawSkybox()	viewMatrix, projectionMatrix	Based on the visualization and projection matrix, it places the textures in a giant cube, thus creating the skybox in such a way that the user moves with them and they are never reachable.
Window			
Function type	**Function name**	**Received parameters**	**Description**

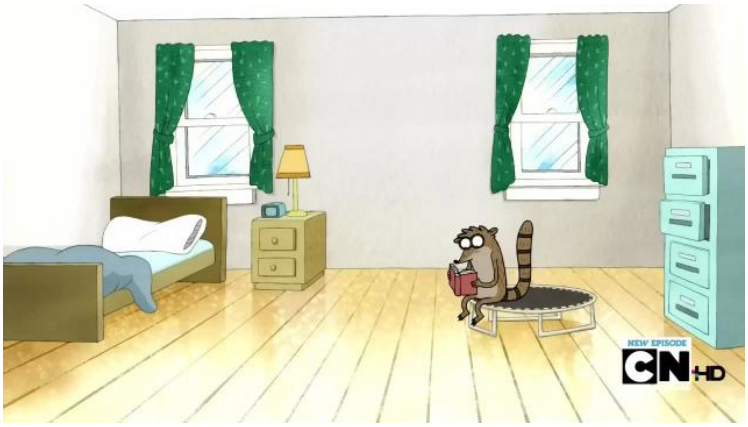
GLboolean	getAbrirCerrarPuerta()	None	The animation process of opening or closing the house door begins with a flag.
GLboolean	getAbrirCerrarGaraje()	None	The animation process of opening or closing the garage door begins with a flag.

Project ítems and references:

Facade: Reference belonging to the tv show “regular show”, recreated on the final project.

The recreation to make is the room of the main characters (Mordecai and Rigby). The reference image for such room is the following:

1st Room:



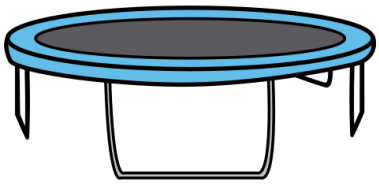
The 5 objects to recreate in the room are the same that appear in the reference image and to have them clearer, the reference image for each object are:

1. Bed:

2. Night table:

3. lamp:

4. mini trampoline:



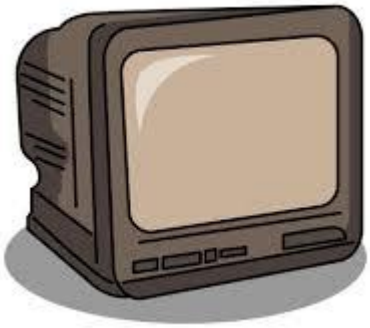
5. Drawers:

2nd Room. Living room:

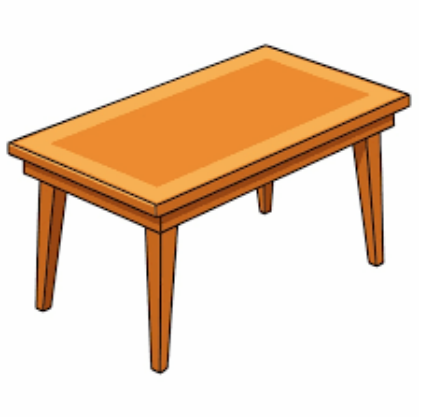
The objects to recreate are the same that appears in these two images. Which are:

1. Couch:

2. Tv:



3. Table:



4. Closet:

5. Photo frame:

6. Atari (extra):

Results:

After presenting the organization behind the project, the development strategy, the costs, the original material and the references, we only have to visualize the 3D environment finished and its different interactions, but first, we will break down the technical part to comprehend the obtained results and with that explore the environment knowing what we can find on each space.

3D Modeling: In this space we can visualize 3D modeling with textures that stick to the cartoon as much as possible, also using furniture enough to decorate the spaces, using a couch, a frame, the beds for each character of the series, achieving faithful integration.

Animations: Here we find the interactivity needed for the user, allowing him to use 4 animations that make the house feel alive, from the welcoming to the path we can open or close the main door , getting to the living room in which we can see a game of Mario Bros running in the Tv. While outside of the house we can see that the garage's door can be open and closed rotating and a glide simultaneously, and at the same time, by opening the garage door, the animation made out of jeyframes can be activated. In which the golfing car starts drifting and later goes back in to the garage.

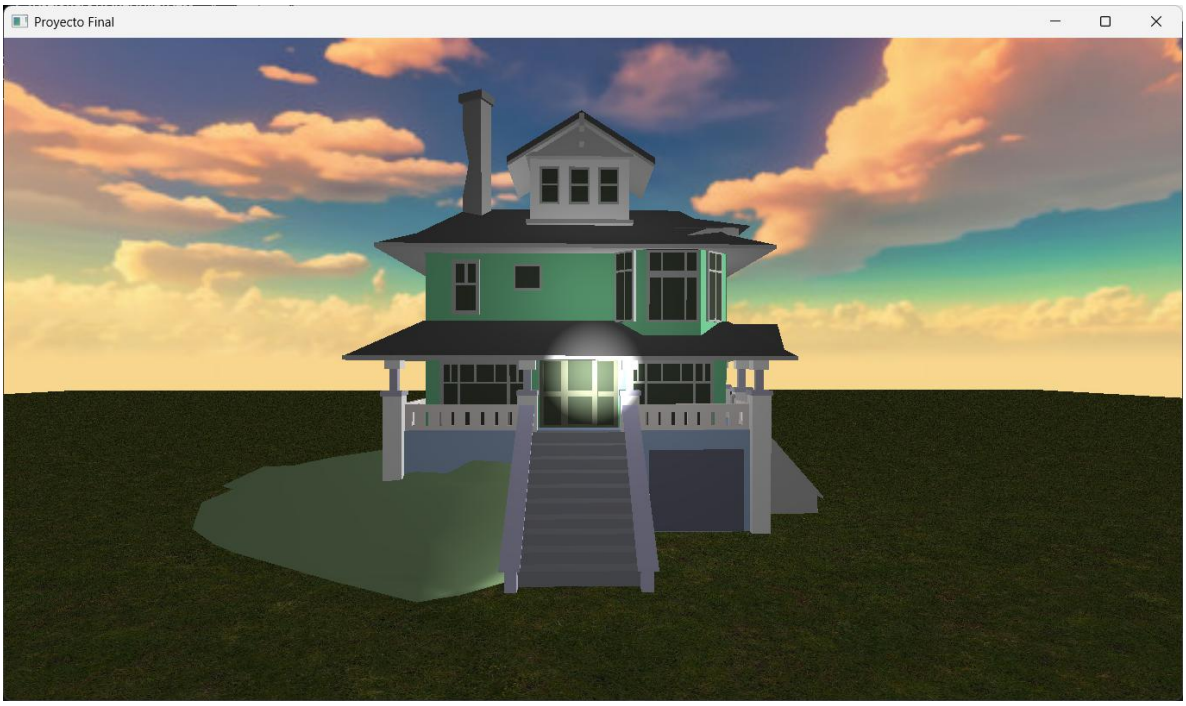
Path: The user can use the synthetic to move in a free way outside and inside of the house to watch every detail in every space.

Total integration: The use of libraries for window generation, graphics (OpenGL), interface (GLFW), texturing (SOIL, stb_image), and model loading (ASSIMP) is fully consolidated, along with classes that handle the logic of separate processes, into a single package. This facilitates programming, ensuring stable performance and readability. It also simplifies the creation of a flowchart with each line of code clearly documented for its function.

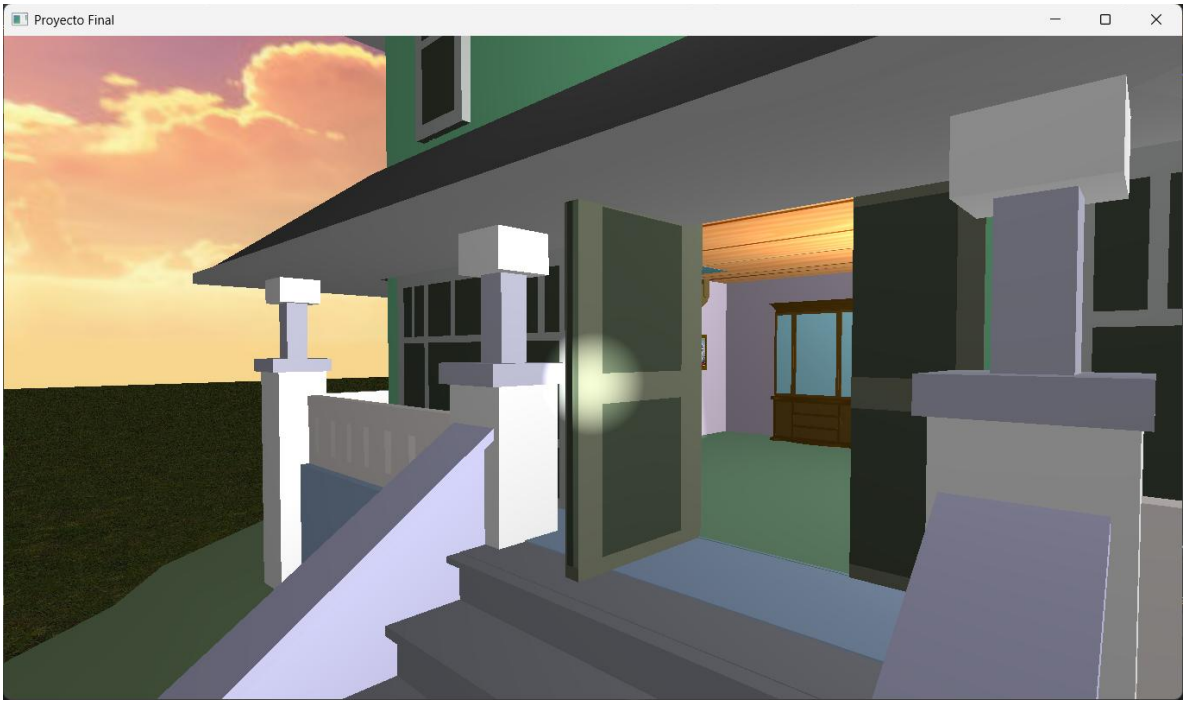
Realistic Illumination: Given that it is an indoor environment, lighting was applied in a way in which the user can truly feel at home, with realistic and contextually correct light sources.

Finally, the object incorporated on the software can be seen as:

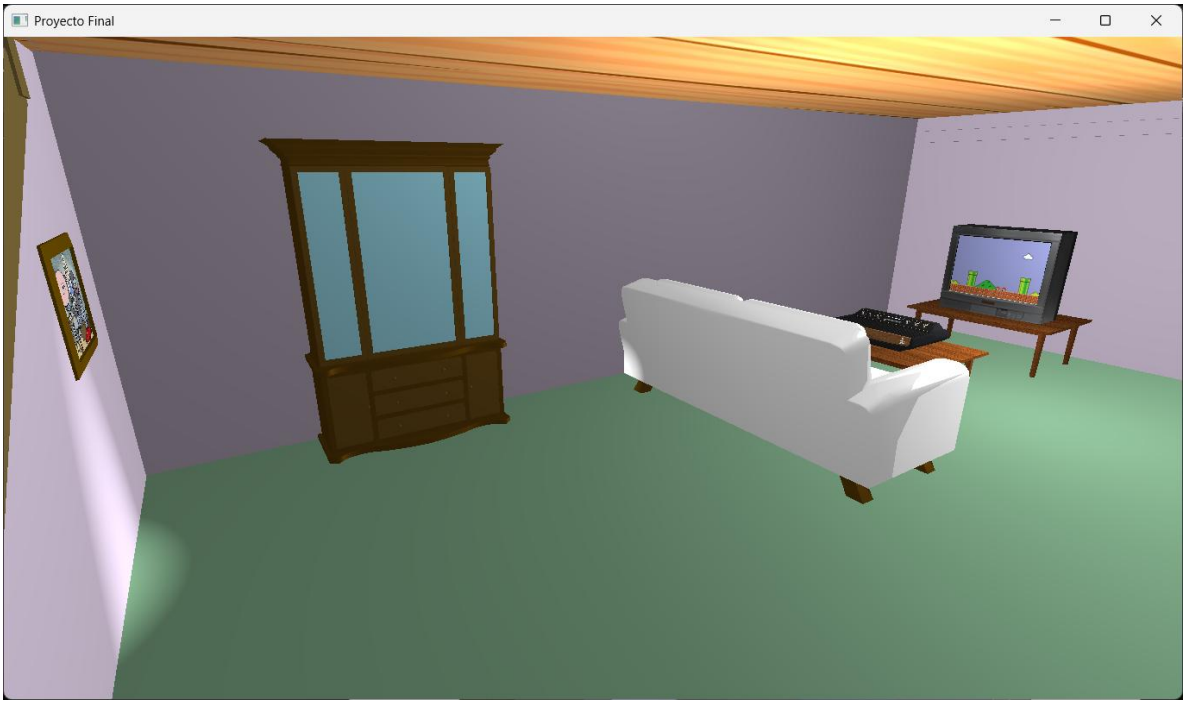
Facade from a front view:



First animation, opening or closing the door with the key ‘G’.

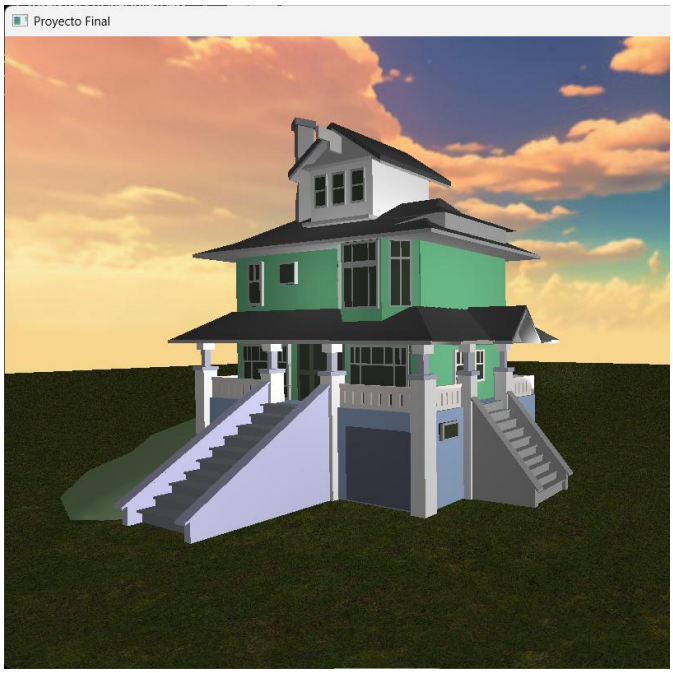


Decoration and 5 objects of the living room: Tv (and its animation with the first level of Mario running), Table, Couch, Closet, Photo frame of the characters of regular show and an atari 2600.

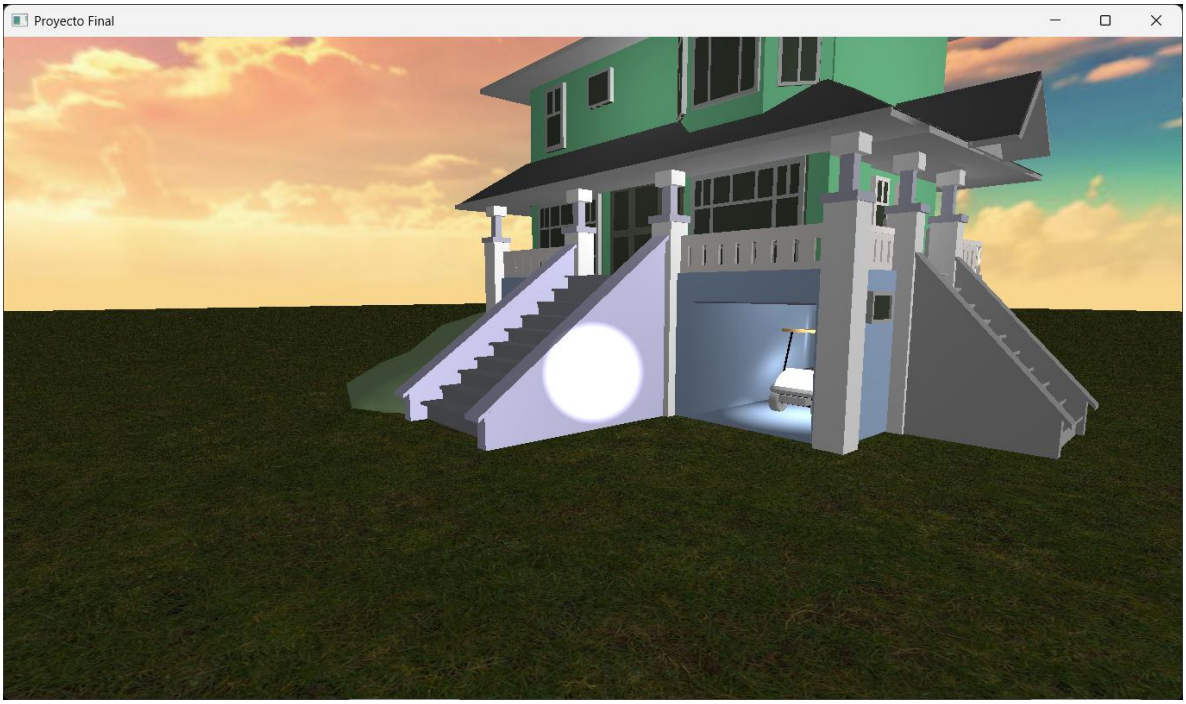


2nd room: Bed, night table, Lamp, trampoline and drawers.

45-degree view of the entrance and one lateral.

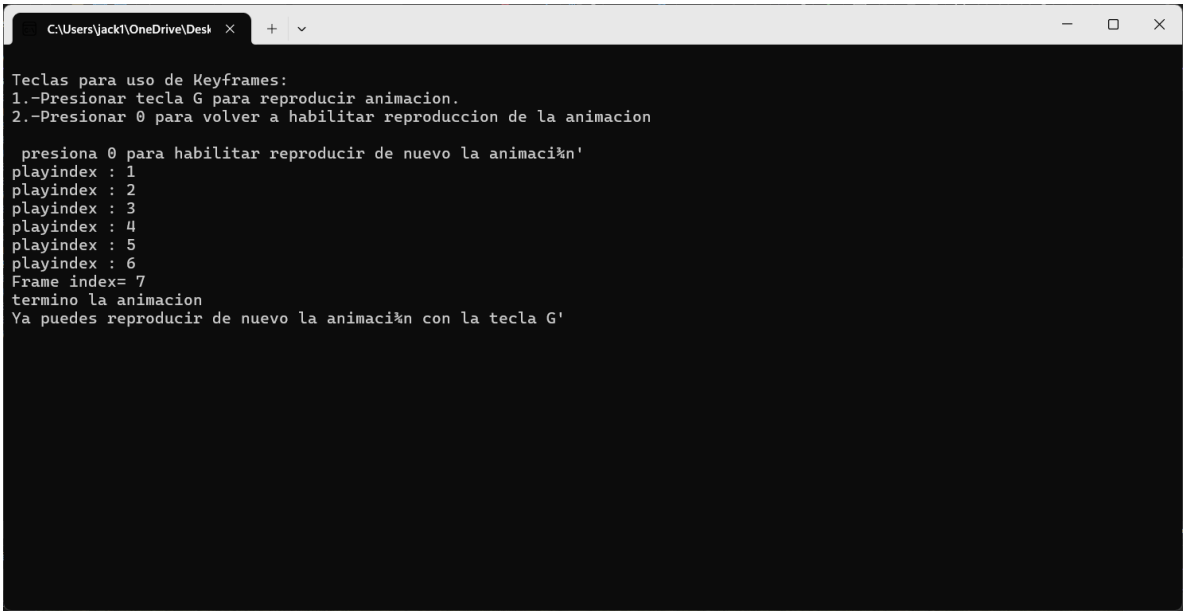


Opening of the garage as Third animation, revealing the golfing car and its animation made with keyframes, though on the next image the animation of the car is already finished.



45-degree angle view from behind and a lateral.

What the terminal show to activate the keyFrames animation and the process of reactivating it to use it again.



Conclusions:

In the realization of the project, all the knowledge acquired during the course was applied, both in its theoretical and practical laboratory components. This allowed for the creation of a project within the required parameters, unleashing the imagination to design an interactive environment with a user-friendly interface. At the same time, improving my programming skills to apply in the application some logic in animations and transformations of three-dimensional spaces. Concluding with a satisfying practical application of the learned topics.

Bibliography:

Mordecai & Rigby's room:

- Donnichols. (2022, 8 de junio). *Old Bed*. Sketchfab https://sketchfab.com/3d-models/old-bed-210be84bcb5449f5a9f66a923c8ae307
- Modelo de buró creado por mí.
- uwurkowo. (2022, 14 de septiembre). *Night lamp*. Sketchfab https://sketchfab.com/3d-models/night-lamp-5a7532792fe04863af72e9cdbb7db65a

- Shahsavan M. (2024, 8 de junio). *Filing Cabinet – 6 MB*. Sketchfab. https://sketchfab.com/3d-models/filing-cabinet-6mb-d217a4bbdfa2426eb32ebf3b1007a8c3
- 1-3D.com. (2022, 25 de marzo). *Trampoline*. Sketchfab. https://sketchfab.com/3d-models/trampoline-052aaef6c5c34c5bac94607bcc12d2f9

Living room models:

- Snomss. (2021, 3 de mayo). *Old Reliable Couch*. Sketchfab. https://sketchfab.com/3d-models/old-reliable-couch-c9a365e5ba9f456db4887dc1e5f4b5d5
- Tomitos. (2025, 8 de junio). *PSX CRT TV*. Sketchfab. https://sketchfab.com/3d-models/psx-crt-tv-14c4034011ff4faebd99de03c7ab1635
- amir.acc129am. (2023, 30 de marzo). *Living room table*. Sketchfab. https://sketchfab.com/3d-models/living-room-table-3c4337eb6f36476b9dada4b91bc6b4c6
- IvanovichRU. (2021, 4 de mayo). *Old China Cabinet*. Sketchfab. https://sketchfab.com/3d-models/old-china-cabinet-a6448e3833bf4ec68c0a919ece796ce2
- //A. (2024, 24 de junio). *Photo frame.* Sketchfab. https://sketchfab.com/3d-models/photo-frame-e9c61ed2e07940bebee2faa9920b8335
- (EXTRA) Dark_igorek. (2024, 20 de octubre). *Atari 2600.* Sketchfab. https://sketchfab.com/3d-models/atari-2600-2024d933f6214117a399ad4287ede64d

Models not implicit in the rooms (extra):

- Shahsavan. M. (2025, 8 de febrero). *Wooden Stairs_5_MB.* Sketchfab. https://sketchfab.com/3d-models/wooden-stairs-5-mb-56271a83e8ff487b84289e39a159d3c5
- apolodz. (2020, 21 de mayo). *Low Poly GolfCart*. Sketchfab. https://sketchfab.com/3d-models/low-poly-golfcart-5233072cb81145b99a06eaa94eafdd2e
- RMRO2744. (2025, 30 de junio). *Un_show_mas_casa.* Sketchfab. https://sketchfab.com/3d-models/un-show-mas-casa-f64cda605b9e40a0a1e52ef16450b90a

Textures:

Obtained with gimp from plain colors

And some obtained from internet with the models