



MANUAL TECNICO PROYECTO FINAL

**Laboratorio De Computación Grafica E
Interacción Humano Computadora**

**Universidad Nacional Autónoma
de México**

Facultad de Ingeniería

- 31803532-7
- 17/Mayo/2024

Tabla de contenido

Objetivos	4
Alcance del Proyecto	4
Limitantes	4
Restricciones	4
Restricciones de Entrega	4
Restricciones Creativas.	5
Consideraciones de animación	5
Recursos	5
OpenGL	5
Maya 2023	6
Visual Studio Community 2022	6
Paint 3D	6
GIMP	6
Luma AI Genie	7
Ganttpro	7
Trello	7
Genshin Impact	7
GitHub	8
Git	8
Metodología de Software	8
Lista de Actividades primera fase.	9
Lista de Actividades segunda fase.	9
Diagrama de Gantt primera fase	22
Diagrama de Gantt segunda fase	24
Diagrama de Flujo del Software	25
Animación Sencilla 01	25
Animación Sencilla 02	26
Animación Sencilla 03	27
Animación Sencilla 04	28
Animación Sencilla 05	29
Animación Compleja 01	30
Animación Compleja 02	31
Documentación del Código	32
Bibliotecas Utilizadas	32
Funciones Destacadas	34

Pivotes y transformaciones básicas.....	35
Animación Sencilla 01.....	36
Animación Sencilla 02.....	39
Animación Sencilla 03.....	42
Animación Sencilla 04.....	43
Animación Sencilla 05.....	45
Animación Compleja 01.....	47
Animación Compleja 02.....	49
SkyBox.....	52
Audio con irrklang	54
Análisis de Costos	55
Conclusiones	55
Anexos.....	56

Objetivos

- **Aplicación de Conocimientos:** Demostrar la aplicación práctica de los conocimientos adquiridos durante el curso.
- **Recreación 3D:** Seleccionar una fachada y un espacio, ya sean reales o ficticios, y recrearlos en 3D utilizando OpenGL, asegurando que los objetos virtuales sean lo más parecidos posible a sus imágenes de referencia.

Alcance del Proyecto

- **Selección de Espacios:** El proyecto incluirá la selección y recreación de una fachada y un espacio con al menos 7 objetos distintos.
- **Ambientación:** La recreación deberá capturar la esencia del ambiente de las imágenes de referencia.
- **Entrega Bilingüe:** La documentación del proyecto se entregará en español e inglés, evitando la traducción automática completa.

Limitantes

- **Individualidad:** El proyecto debe ser realizado y entregado de forma individual.
- **Evaluación:** Los proyectos con evaluaciones menores a 5 se considerarán deficientes y afectarán la calificación final del curso.
- **Repetición de Objetos:** Los objetos recreados repetidamente se contarán como un solo objeto en la evaluación.
- **Elementos Obligatorios:** Puertas, ventanas, chimeneas y escaleras son elementos obligatorios y no se cuentan como objetos para la evaluación.
- **Código Base:** Se debe utilizar el código base proporcionado durante el curso.

Restricciones

Restricciones de Entrega

- **Repositorio Oficial:** La entrega del proyecto debe realizarse exclusivamente a través de un repositorio en GitHub. Cualquier entrega fuera de esta plataforma resultará en la anulación del proyecto.

Restricciones Creativas.

- **Temáticas Prohibidas:** No se permite recrear espacios pertenecientes a la UNAM ni temáticas específicas como los Simpson, Rick and Morty, Bob Esponja, Kamehouse de Dragon Ball, la casa de Coraje el Perro Cobarde, contenido de Minecraft y la casa de las Chicas Superpoderosas.
- **Calidad Gráfica:** Se prohíbe la creación de ambientes 3D con características gráficas bajas, similares a las de juegos desarrollados para consolas como PS1, PS2, Xbox, Nintendo 64, etc.

Consideraciones de animación

- **Contexto de Animaciones:** Las animaciones deben tener un propósito y estar en armonía con el contexto del espacio recreado. No es suficiente con rotar un objeto sin motivo.
- **Complejidad de Animaciones:** Una animación compleja no debe ser meramente lineal o consistir solo en transformaciones básicas del objeto.

Recursos

Para desarrollar este proyecto de manera eficiente y efectiva, he empleado diversas aplicaciones que me han permitido gestionar todas las tareas y aspectos necesarios. A continuación, presento una lista detallada de las herramientas que he utilizado y cómo cada una ha contribuido al éxito de este proyecto.

OpenGL



OpenGL (Open Graphics Library) es una API (Interfaz de Programación de Aplicaciones) multiplataforma y de código abierto para renderizar gráficos 2D y 3D. Es ampliamente utilizada en el desarrollo de aplicaciones gráficas, como videojuegos, simuladores, aplicaciones de visualización científica, entre otros. Herramienta que nos permitirá

renderizar nuestro proyecto.

Maya 2023



Autodesk Maya es un software de modelado, animación, renderizado y simulación 3D ampliamente utilizado en la industria del entretenimiento,

especialmente en la producción de películas, animaciones, videojuegos y efectos visuales. Ofrece una amplia gama de herramientas y características para crear contenido digital complejo y de alta calidad. Fue una de las herramientas más usadas a lo largo del desarrollo de los modelos generados.

Visual Studio Community 2022



El IDE de Visual Studio es un panel de inicio creativo que se puede usar para editar, depurar y compilar código y, después, publicar una aplicación. Fue la herramienta más utilizada en torno al desarrollo del proyecto. A partir del lenguaje C++ se crearon y modificaron bloques de código dependiendo de los requerimientos necesarios solicitados.

Paint 3D



Paint 3D es una aplicación de Microsoft que viene preinstalada en Windows 10 y versiones posteriores. Aunque se basa en el clásico programa Paint, Paint 3D es una versión modernizada que permite a los usuarios crear imágenes en dos y tres dimensiones de manera sencilla y accesible. Se utilizó para desarrollar

algunas texturas que eran necesario hacer manualmente

GIMP



GIMP (GNU Image Manipulation Program) es un potente software de edición de imágenes de código abierto y gratuito, disponible para varias plataformas, incluyendo Windows, macOS y Linux. Permitió el ajustar correctamente el tamaño de nuestras texturas y realizar modificaciones para jugar con transparencias

Luma AI Genie



Es una herramienta revolucionaria que transforma texto en modelos 3D. Este modelo generativo 3D de Luma AI tiene la capacidad de crear modelos tridimensionales interactivos a partir de instrucciones dadas en lenguaje natural. Se uso para crear algunos objetos que no eran sencillos de recrear.

Ganttpro



Es una herramienta de gestión de proyectos en línea basada en diagramas de Gantt. Es utilizada por más de 350,000 gerentes de proyectos, líderes de equipo, directores ejecutivos y otros gerentes en diversos ámbitos. Se uso para el desarrollo de diagrama de gant.

Trello



Es una herramienta de gestión de proyectos que utiliza el sistema Kanban. Es conocida por su interfaz visual e intuitiva que permite a los equipos organizar y supervisar tareas y proyectos de manera eficiente. Ocupada para implementar la metodología de software.

Genshin Impact



juego. Se ocupó como fuente creativa para el proyecto.

La Relajatetera es una característica especial dentro del juego Genshin Impact que permite a los jugadores crear y personalizar su propio espacio de vivienda, similar a un sistema de "housing". Fue introducida en la versión 1.5 del

GitHub



objetivo de entregar el repositorio oficial.

GitHub es una plataforma de desarrollo colaborativo para almacenar, compartir y trabajar en código. Es ampliamente utilizada por programadores y empresas para la gestión de proyectos y control de versiones de software; utiliza Git para almacenar repositorios de código en la nube. Se utilizó para el

Git



Es un sistema de control de versiones distribuido que permite a los desarrolladores rastrear y gestionar cambios en el código fuente a lo largo del tiempo. Funciona localmente en la computadora del usuario, permitiendo trabajar en proyectos incluso sin conexión a internet

Metodología de Software

“Kanban” es una combinación de dos palabras japonesas: 看 (Kàn), que significa “signo” o “señal visual”, y 板 (Bǎn), que significa “tablero”. En Toyota, las tarjetas Kanban eran tarjetas de papel que indicaban que se necesitaba un producto nuevo, una pieza nueva o un inventario, y que disparaban el proceso de producción de dicho artículo.

En Kanban implementado para el desarrollo de software, los equipos comienzan con una lista de tareas pendientes. El trabajo se “extrae” de las tareas pendientes, según la carga laboral y capacidad de cada miembro del equipo. Luego, los miembros pueden hacer un seguimiento visual del trabajo a medida que avanza a través del ciclo de vida de las tareas, representado por etapas en un tablero Kanban, hasta su finalización. En su configuración actual, Kanban funciona como un método visual de gestión de proyectos que permite a los equipos encontrar un equilibrio entre la demanda de trabajo y la disponibilidad de los recursos del equipo.

Un tablero Kanban es una herramienta de gestión que ayuda a visualizar el trabajo, limitar el trabajo en curso (WIP) y maximizar la eficiencia. Los equipos utilizan el tablero para estructurar su carga de trabajo y gestionar el progreso en tiempo real.

El concepto de un tablero Kanban es sencillo. Cada columna representa una etapa diferente de su flujo de trabajo o proyecto. A medida que la tarea avanza, se desplaza a la columna correspondiente.

Lista de Actividades primera fase.

Número de Et	Nombre de tarea / Título	Asignado a	Fecha de inici	Fecha de final	Fecha límite
1	Primera Fase Proyecto		22/02/2024	10/05/2024	10/05/2024
2	Búsqueda de Imagen de Referencia	Jimenez Cerv	22/02/2024	29/02/2024	29/02/2024
3	Búsqueda de Texturas	Jimenez Cerv	01/03/2024	07/03/2024	
4	Desarrollo de la Fachada	Jimenez Cerv	07/03/2024	22/04/2024	
5	Desarrollo de Objeto 01	Jimenez Cerv	07/03/2024	25/03/2024	
6	Desarrollo Objeto 02	Jimenez Cerv	14/03/2024	19/04/2024	
7	Desarrollo Objeto 03	Jimenez Cerv	21/03/2024	19/04/2024	
8	Desarrollo Objeto 04	Jimenez Cerv	28/03/2024	19/04/2024	
9	Desarrollo Objeto 05	Jimenez Cerv	28/03/2024	19/04/2024	
10	Desarrollo Objeto 06	Jimenez Cerv	28/03/2024	19/04/2024	
11	Desarrollo Objeto 07	Jimenez Cerv	28/03/2024	19/04/2024	
12	Desarrollo Objetos Extra 01	Jimenez Cerv	10/04/2024	22/04/2024	
13	Desarrollo Objeto Extra 02	Jimenez Cerv	10/04/2024	22/04/2024	
14	Desarrollo Objeto Extra 03	Jimenez Cerv	10/04/2024	22/04/2024	
15	Desarrollo Objeto Extra 04	Jimenez Cerv	10/04/2024	22/04/2024	
16	Desarrollo Objeto Extra 05	Jimenez Cerv	10/04/2024	22/04/2024	
17	Desarrollo Objeto Extra 06	Jimenez Cerv	10/04/2024	22/04/2024	
18	Animación Sencilla 01	Jimenez Cerv	22/04/2024	23/04/2024	
19	Animación Sencilla 02	Jimenez Cerv	22/04/2024	23/04/2024	
20	Animación Sencilla 03	Jimenez Cerv	23/04/2024	24/04/2024	
21	Animaciones sencillas extras	Jimenez Cerv	24/04/2024	29/04/2024	
22	Animación Compleja 01	Jimenez Cerv	29/04/2024	02/05/2024	
23	Animación Compleja 02	Jimenez Cerv	29/04/2024	02/05/2024	
24	Manual Técnico ES	Jimenez Cerv	29/04/2024	06/05/2024	
25	Manual Técnico EN	Jimenez Cerv	07/05/2024	07/05/2024	
26	Manual Usuario ES	Jimenez Cerv	29/04/2024	06/05/2024	
27	Manual Usuario EN	Jimenez Cerv	07/05/2024	07/05/2024	
28	Skybox	Jimenez Cerv	02/05/2024	06/05/2024	
29	Agregando Audio	Jimenez Cerv	06/05/2024	06/05/2024	
30	Ejecutable Final	Jimenez Cerv	06/05/2024	06/05/2024	08/05/2024

Lista de Actividades segunda fase.

Cod	Número de Et	Nombre de tarea / Título	Asignado a	Fecha de inici	Fecha de final
	1	Tiempo Límite de Entrega	Jimenez Cerv	07/05/2024	20/05/2024
	2	Modificación Fachada Segundo Piso	Jimenez Cerv	07/05/2024	09/05/2024
	3	Modelo 01	Jimenez Cerv	08/05/2024	10/05/2024
	4	Modelo 02	Jimenez Cerv	08/05/2024	10/05/2024
	5	Modelo 03	Jimenez Cerv	08/05/2024	10/05/2024
	6	Modelo 04	Jimenez Cerv	08/05/2024	10/05/2024
	7	Modelo 05	Jimenez Cerv	08/05/2024	10/05/2024
	8	Modelo Extra 01	Jimenez Cerv	08/05/2024	10/05/2024
	9	Modelo Extra 02	Jimenez Cerv	08/05/2024	13/05/2024
	10	Modelo Extra 03	Jimenez Cerv	08/05/2024	13/05/2024
	11	Modelo Extra 04	Jimenez Cerv	08/05/2024	13/05/2024
	12	Modelo Extra 05	Jimenez Cerv	08/05/2024	15/05/2024
	13	Modelo Extra 06	Jimenez Cerv	08/05/2024	15/05/2024
	14	Animación Sencilla Extra	Jimenez Cerv	09/05/2024	10/05/2024
	15	Animación Sencilla Extra e iluminació	Jimenez Cerv	13/05/2024	15/05/2024
	16	Manual de Usuario ES-EN	Jimenez Cerv	13/05/2024	15/05/2024
	17	Manual Técnico ES-EN	Jimenez Cerv	15/05/2024	16/05/2024
	18	Ejecutable Final	Jimenez Cerv	15/05/2024	15/05/2024

Avance Tablero Kanban

Para evitar estancarse en un estado ambiguo de “trabajo en progreso” (WIP). Se definió el jueves de cada semana como día de entrega de la actividad conforme sea la demanda solicitada. Por lo que continuación se mostrará el avance del tablero Kanban conforme cada jueves desde la publicación del documento de referencia.

Jueves 22-febrero-2024

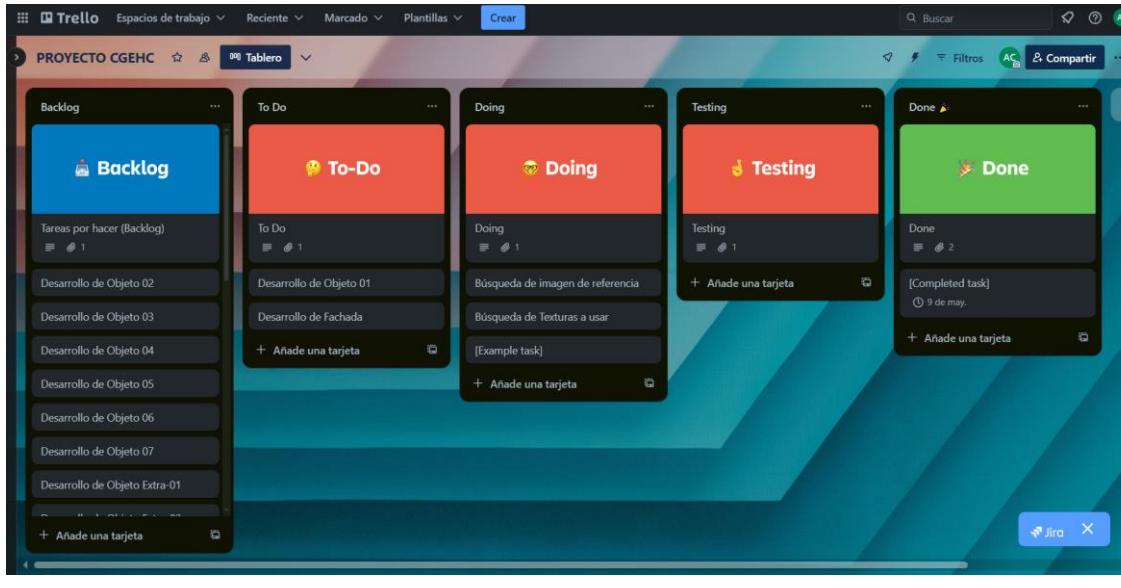
Se comienza a definir las tareas a realizar a lo largo del proyecto.

Jueves 29-febrero-2024

Se marco como completado la primera actividad y se empezó con la búsqueda de texturas.

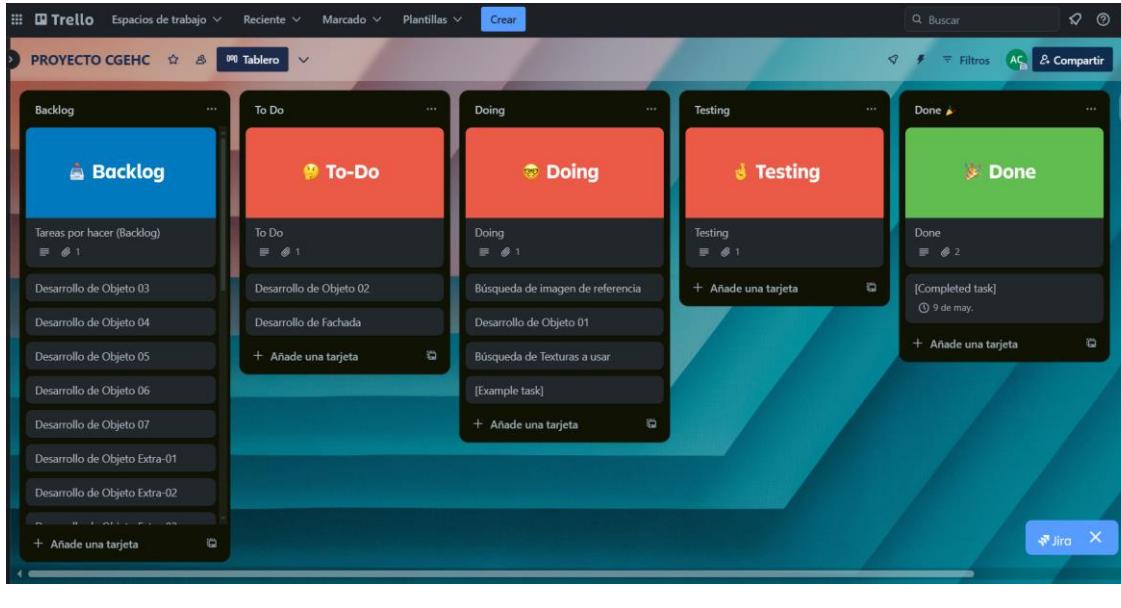
Jueves 07-marzo-2024

Se marcó como completado la búsqueda de texturas y se comenzó el avance de la fachada y primer objeto.



Jueves 14-marzo-2024

Se marcó como completado el avance del primer objeto, se agregó el segundo objeto y se sigue trabajando sobre la fachada.



Jueves 21-marzo-2024

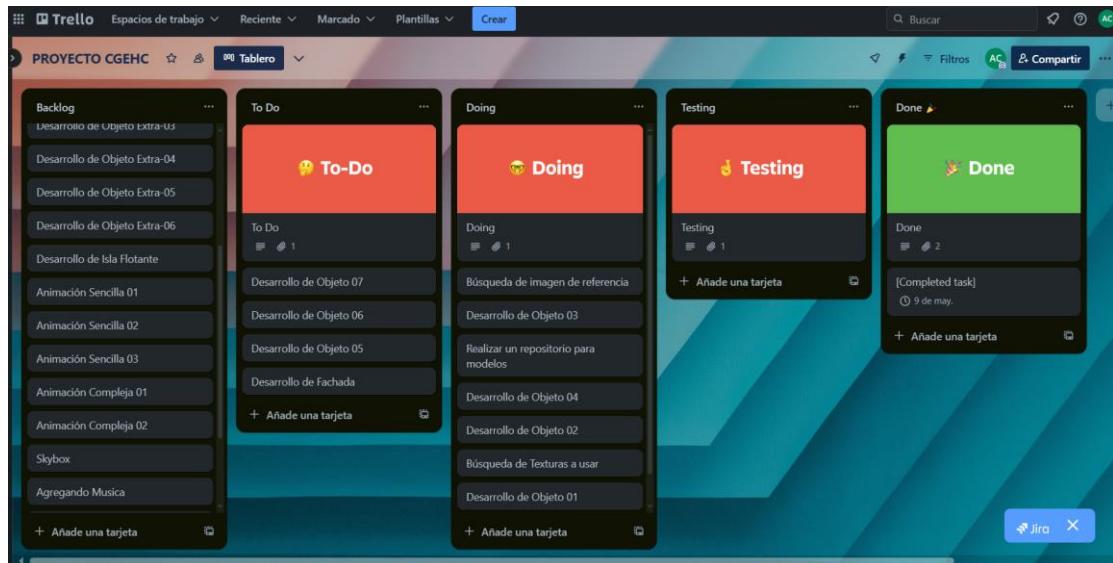
Se marco como completado el avance del segundo objeto, se agregó el tercer objeto y se sigue trabajando sobre la fachada.

Jueves 28-marzo-2024

Se marco como completado el avance del tercer objeto, se trabaja en el modelo 01 y en los siguientes modelos y se sigue trabajando sobre la fachada. Se crea un repositorio para el avance de los modelos

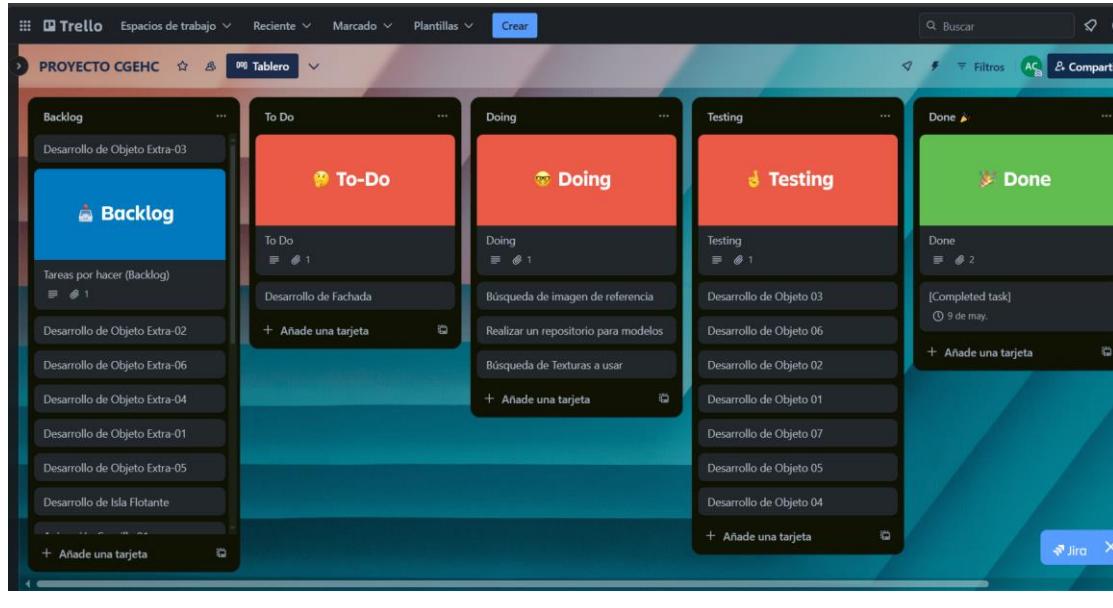
Jueves 04-abril -2024

Se marco como completado el avance de los primeros modelos y se sigue trabajando sobre la fachada.



Jueves 11-abril -2024

Se marco como completado el avance del modelo 05 y se sigue trabajando sobre la fachada. Se manda a OpenGL los modelos terminados



Jueves 18-abril -2024

Se marco como completado los modelos obligatorios, se sigue trabajando sobre la fachada y en los modelos extras. Se trabaja en la primera animación sencilla.

Backlog	To Do	Doing	Testing	Done
Desarrollo de Objeto Extra-03 Backlog Tareas por hacer (Backlog) Desarrollo de Isla Flotante Animación Sencilla 02 Animación Sencilla 03 Animación Compleja 01 Animación Compleja 02 Skybox + Añade una tarjeta	ToDo To Do Desarrollo de Fachada Desarrollo de Objeto Extra-06 Desarrollo de Objeto Extra-05 Desarrollo de Objeto Extra-04 Desarrollo de Objeto Extra-01 Desarrollo de Objeto Extra-02 Desarrollo de Objeto Extra-03 + Añade una tarjeta	Doing Doing Búsqueda de imagen de referencia Realizar un repositorio para modelos Búsqueda de Texturas a usar + Añade una tarjeta	Testing Testing Desarrollo de Objeto 03 Desarrollo de Objeto 06 Desarrollo de Objeto 02 Desarrollo de Objeto 01 Desarrollo de Objeto 07 Desarrollo de Objeto 05 Desarrollo de Objeto 04 + Añade una tarjeta	Done Done (Completed task) 9 de may. + Añade una tarjeta

A partir de este punto ya se contaba con un repositorio final para la entrega de código.

Domingo 21-abril -2024

Se marco como completado los todos los modelos, finalmente se marca como completado la fachada y se trabaja sobre la primera y segunda animación sencilla.

Backlog	To Do	Doing	Testing	Done
Desarrollo de Objeto Extra-03 Backlog Tareas por hacer (Backlog) Desarrollo de Isla Flotante Animación Sencilla 03 Animación Compleja 01 Animación Compleja 02 Skybox Agregando Musica + Añade una tarjeta	ToDo To Do Animación Sencilla 01 Animación Sencilla 02 Desarrollo de Fachada + Añade una tarjeta	Doing Doing Búsqueda de imagen de referencia Realizar un repositorio para modelos Búsqueda de Texturas a usar + Añade una tarjeta	Testing Testing Desarrollo de Objeto 03 Desarrollo de Objeto Extra-02 Desarrollo de Objeto Extra-01 Desarrollo de Objeto Extra-03 Desarrollo de Objeto Extra-05 Desarrollo de Objeto Extra-04 Desarrollo de Objeto Extra-06 + Añade una tarjeta	Done Done (Completed task) 9 de may. + Añade una tarjeta

Lunes 22-abril -2024

Se finaliza la primera y segunda animación sencilla. Y se trabaja sobre la tercera animación

Backlog	To Do	Doing	Testing	Done
Tareas por hacer (Backlog)	To Do	Búsqueda de imágenes de referencia	Realizar un repositorio para modelos	[Completed task] + Añade una tarjeta
Desarrollo de Isla Flotante				
Animación Compleja 01				
Animación Compleja 02				
Skybox				
Agregando Musica				
Manual Técnico ES				
Manual Técnico EN				
Manual Usuario				
Ejecutable Final				
+ Añade una tarjeta	+ Añade una tarjeta	+ Añade una tarjeta	+ Añade una tarjeta	+ Añade una tarjeta

Martes 23-abril -2024

Se finaliza la tercera animación y se trabaja sobre una animación sencilla extra.

Backlog	To Do	Doing	Testing	Done
Tareas por hacer (Backlog)	To Do	Búsqueda de imágenes de referencia	Realizar un repositorio para modelos	[Completed task] + Añade una tarjeta
Desarrollo de Isla Flotante				
Animación Compleja 01				
Animación Compleja 02				
Skybox				
Agregando Musica				
Manual Técnico ES				
Manual Técnico EN				
Manual Usuario				
Ejecutable Final				
+ Añade una tarjeta	+ Añade una tarjeta	+ Añade una tarjeta	+ Añade una tarjeta	+ Añade una tarjeta

Domingo 28-abril -2024

Se finaliza la animación sencilla extra, se trabaja sobre las animaciones complejas. Se trabaja sobre el manual técnico y de usuario

PROYECTO CGEHC

- Backlog**
 - Tareas por hacer (Backlog)
 - Desarrollo de Isla Flotante
 - Skybox
 - Agregando Musica
 - Ejecutable Final
- To Do**
 - To Do
 - Animación Compleja 01
 - Animación Compleja 02
 - Manual Técnico ES
 - Manual Técnico EN
 - Manual Usuario
- Doing**
 - Doing
 - Búsqueda de imagen de referencia
 - Realizar un repositorio para modelos
 - Búsqueda de Texturas a usar
 - + Añade una tarjeta
- Testing**
 - Testing
 - Desarrollo de Fachada
 - Animación Sencilla 03
 - Animaciones Sencillas Extras
 - Desarrollo de Objeto 03
 - Desarrollo de Objeto Extra-02
 - Animación Sencilla 01
 - Animación Sencilla 02
 - + Añade una tarjeta
- Done**
 - Done
 - [Completed task]
 - + Añade una tarjeta

Jueves 02-mayo -2024

Se finaliza la animación sencilla extra, las animaciones complejas. Se trabaja sobre el skybox

PROYECTO CGEHC

- Backlog**
 - Tareas por hacer (Backlog)
 - Agregando Musica
 - Ejecutable Final
 - + Añade una tarjeta
- To Do**
 - To Do
 - Skybox
 - Desarrollo de Isla Flotante
 - Manual Técnico ES
 - Manual Técnico EN
 - Manual Usuario
- Doing**
 - Doing
 - Búsqueda de imagen de referencia
 - Realizar un repositorio para modelos
 - Búsqueda de Texturas a usar
 - + Añade una tarjeta
- Testing**
 - Testing
 - Animación Compleja 02
 - Animación Compleja 01
 - Desarrollo de Fachada
 - Animación Sencilla 03
 - Animaciones Sencillas Extras
 - Desarrollo de Objeto 03
 - Desarrollo de Objeto Extra-02
 - + Añade una tarjeta
- Done**
 - Done
 - [Completed task]
 - + Añade una tarjeta

Domingo 05-mayo -2024

Se finalizan el skybox y se trabaja sobre el audio del proyecto

Backlog	To Do	Doing	Testing	Done
Desarrollo de Objeto Extra-03				
Backlog	To-Do	Doing	Testing	Done
Tareas por hacer (Backlog)	To Do	Doing	Testing	Done
+ Añade una tarjeta	1	1	1	2
Ejecutable Final	Agregando Musica	Búsqueda de imágenes de referencia	Animación Compleja 02.	
+ Añade una tarjeta	Manual Técnico ES	Realizar un repositorio para modelos	Desarrollo de Isla Flotante	
	Manual Técnico EN	Búsqueda de Texturas a usar	Skybox	
	Manual Usuario	+ Añade una tarjeta	Animación Compleja 01	
	+ Añade una tarjeta		Desarrollo de Fachada	
			Animación Sencilla 03	
			Animaciones Sencillas Extras	
			+ Añade una tarjeta	

Lunes 06-mayo -2024

Se finalizan el insertar audio, obtener ejecutable y manual de usuario en español

Backlog	To Do	Doing	Testing	Done
Desarrollo de Objeto Extra-03				
Backlog	To-Do	Doing	Testing	Done
Tareas por hacer (Backlog)	To Do	Doing	Testing	Done
+ Añade una tarjeta	1	1	1	2
Ejecutable Final	Manual Técnico ES	Búsqueda de imágenes de referencia	Testing	
+ Añade una tarjeta	Manual Técnico EN	Manual Usuario ES	+ Añade una tarjeta	
	Manual Usuario EN	Realizar un repositorio para modelos		
	+ Añade una tarjeta	Búsqueda de Texturas a usar		
		+ Añade una tarjeta		

Martes 07-mayo -2024

Se finalizan el manual técnico en español e inglés y el manual de usuario en inglés.

The screenshot shows a Trello board with the following columns and their contents:

- Backlog:** Desarrollo de Objeto Extra-03, Backlog, Tareas por hacer (Backlog), + Añade una tarjeta.
- To Do:** To Do, + Añade una tarjeta.
- Doing:** Doing, Doing, Manual Usuario EN, Manual Técnico EN, Manual Técnico ES, Búsqueda de imagen de referencia, Manual Usuario ES, Realizar un repositorio para modelos, Búsqueda de Texturas a usar, + Añade una tarjeta.
- Testing:** Testing, Testing, + Añade una tarjeta.
- Done:** Done, Desarrollo de Objeto 04, Desarrollo de Objeto 05, Desarrollo de Objeto 07, Desarrollo de Objeto 01, Desarrollo de Objeto 02, Desarrollo de Objeto 06, Desarrollo de Objeto Extra-06, + Añade una tarjeta.

Martes 07-mayo -2024

Se definen actividades de segunda fase y se trabaja sobre la modificación de la fachada

The screenshot shows a Trello board with the following columns and their contents:

- Backlog:** Backlog, Objeto 01, Objeto 02, Objeto 03, Objeto 04, Objeto 05, Objeto Extra 01, Objeto Extra 02, + Añade una tarjeta.
- To Do:** To Do, Modificación Fachada para Segundo Piso, + Añade una tarjeta.
- Doing:** Doing, Doing, [Example task], + Añade una tarjeta.
- Testing:** Testing, + Añade una tarjeta.
- Done:** Done, [Completed task] (23 de ene. de 2020), + Añade una tarjeta.

Miércoles 08-mayo -2024

Se sigue trabajando en la fachada y se empiezan los demás objetos a modelar

The screenshot shows a Trello board with the following columns and their contents:

- Backlog:** Contains items like "Backlog", "Animación Sencilla 01", "Animación Sencilla 02", "Iluminación", "Manual Usuario ES", "Manual Usuario EN", "Manual Técnico ES", "Manual Técnico EN", and a "+ Añade una tarjeta" button.
- To Do:** Contains items like "Objeto Extra 06", "Objeto Extra 05", "Objeto Extra 04", "Objeto Extra 03", "Objeto Extra 02", "Objeto Extra 01", "Objeto 05", "Objeto 04", "Objeto 03", "Objeto 02", and "Objeto 01".
- Doing:** Contains items like "Doing" (with 1 task), "Testing" (with 1 task), and "[Example task]".
- Testing:** Contains items like "Testing" (with 1 task) and "+ Añade una tarjeta".
- Done:** Contains items like "Done" (with 2 tasks), "[Completed task] (23 de ene. de 2020)", and "+ Añade una tarjeta".

Jueves 09-mayo -2024

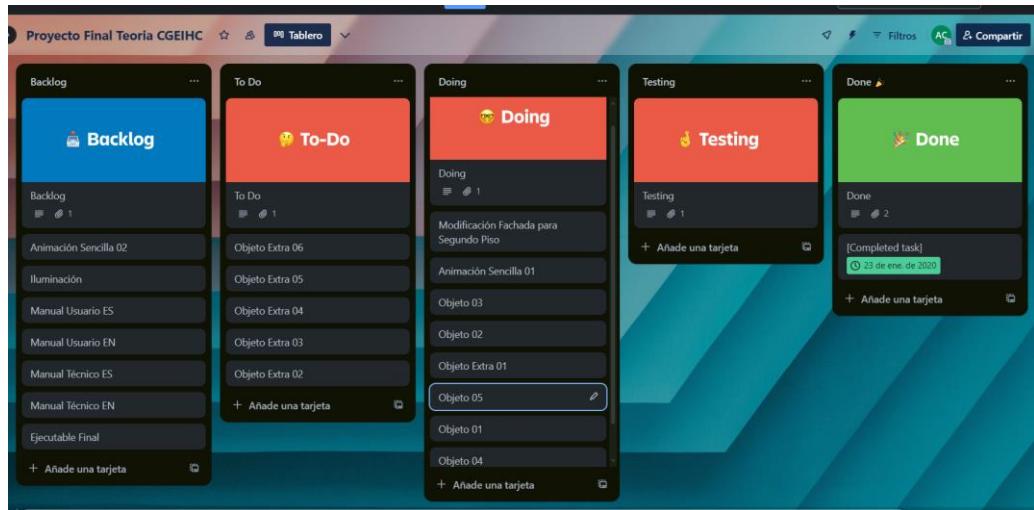
Se trabaja sobre la animación sencilla extra-01 y se termina la fachada modificada.

The screenshot shows a Trello board with the following columns and their contents:

- Backlog:** Contains items like "Backlog", "Animación Sencilla 02", "Iluminación", "Manual Usuario ES", "Manual Usuario EN", "Manual Técnico ES", "Manual Técnico EN", "Ejecutable Final", and a "+ Añade una tarjeta" button.
- To Do:** Contains items like "To Do" (with 1 task).
- Doing:** Contains items like "Doing" (with 1 task), "Modificación Fachada para Segundo Piso", and "[Example task]".
- Testing:** Contains items like "Testing" (with 1 task) and "+ Añade una tarjeta".
- Done:** Contains items like "Done" (with 2 tasks), "[Completed task] (23 de ene. de 2020)", and "+ Añade una tarjeta".

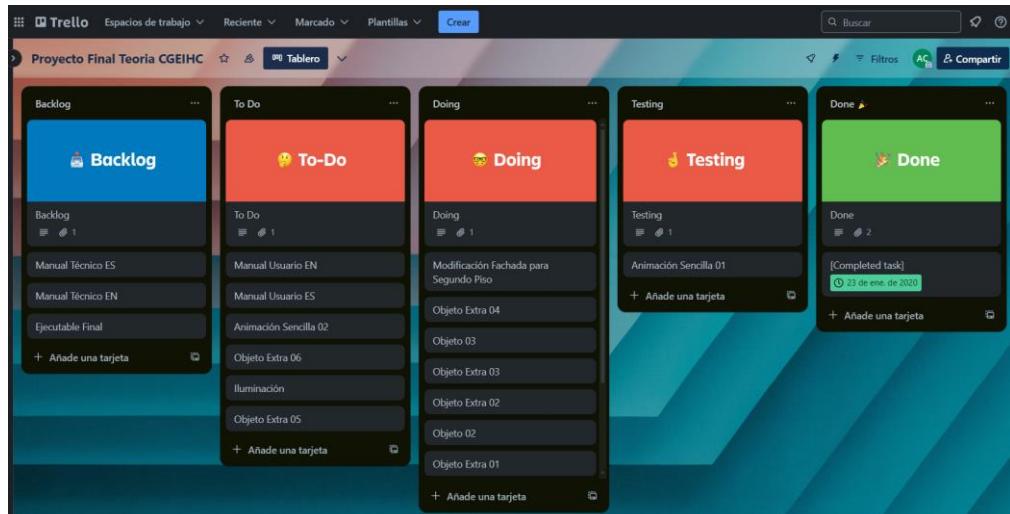
Viernes 10-mayo -2024

Se termina la animación sencilla 01 y los primeros modelos.



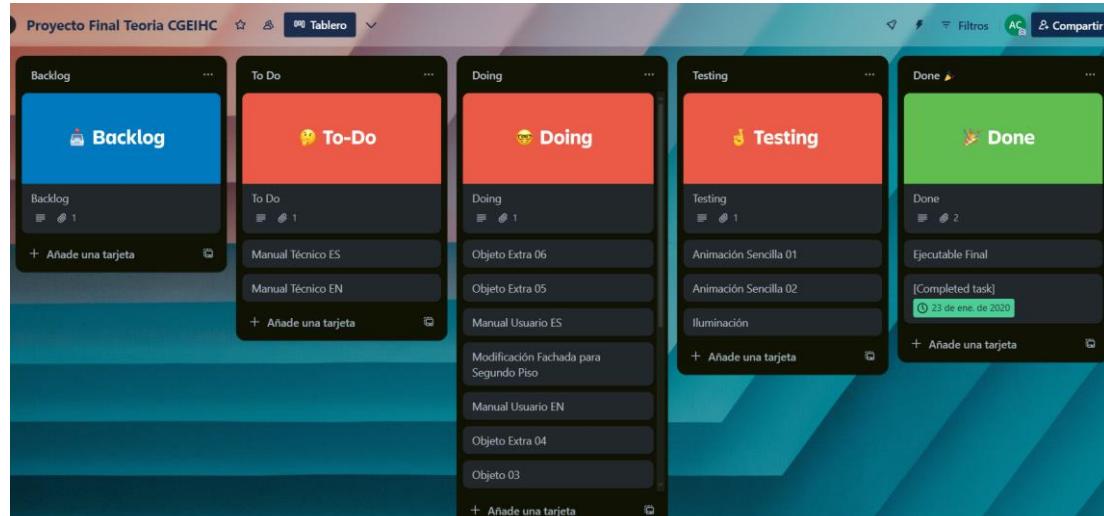
Lunes 13-mayo -2024

Se acaban 3 modelos, se trabaja sobre el manual de usuario, se realiza otra animación sencilla e iluminación.



Miercoles15-mayo -2024

Se termina los modelos restantes, se termina iluminación y animación extra; se trabaja sobre el manual técnico y se obtiene el ejecutable.



Jueves 16-mayo -2024

Se termina el manual técnico y el proyecto

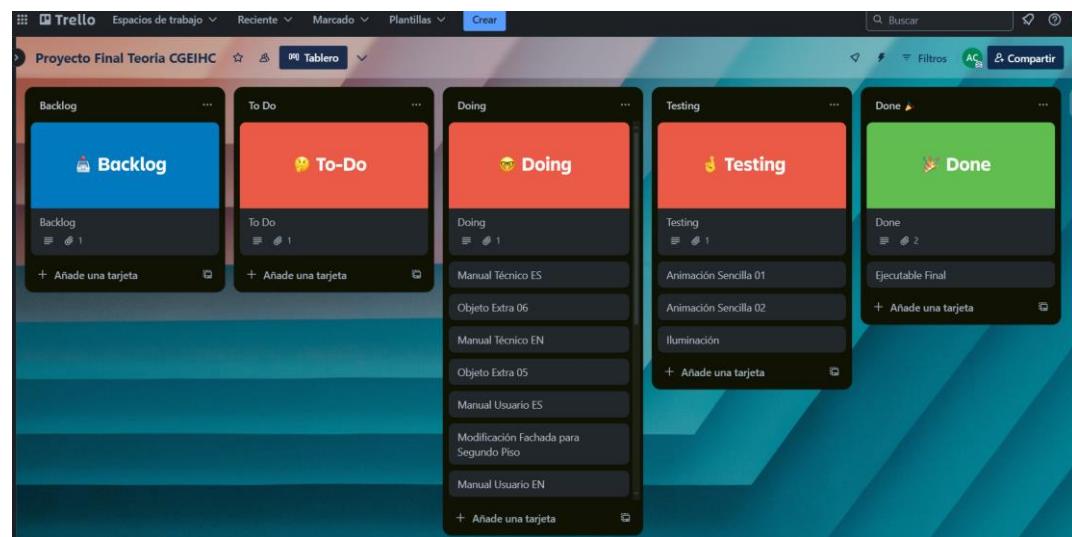


Diagrama de Gantt primera fase

Diagrama de Gantt por días.

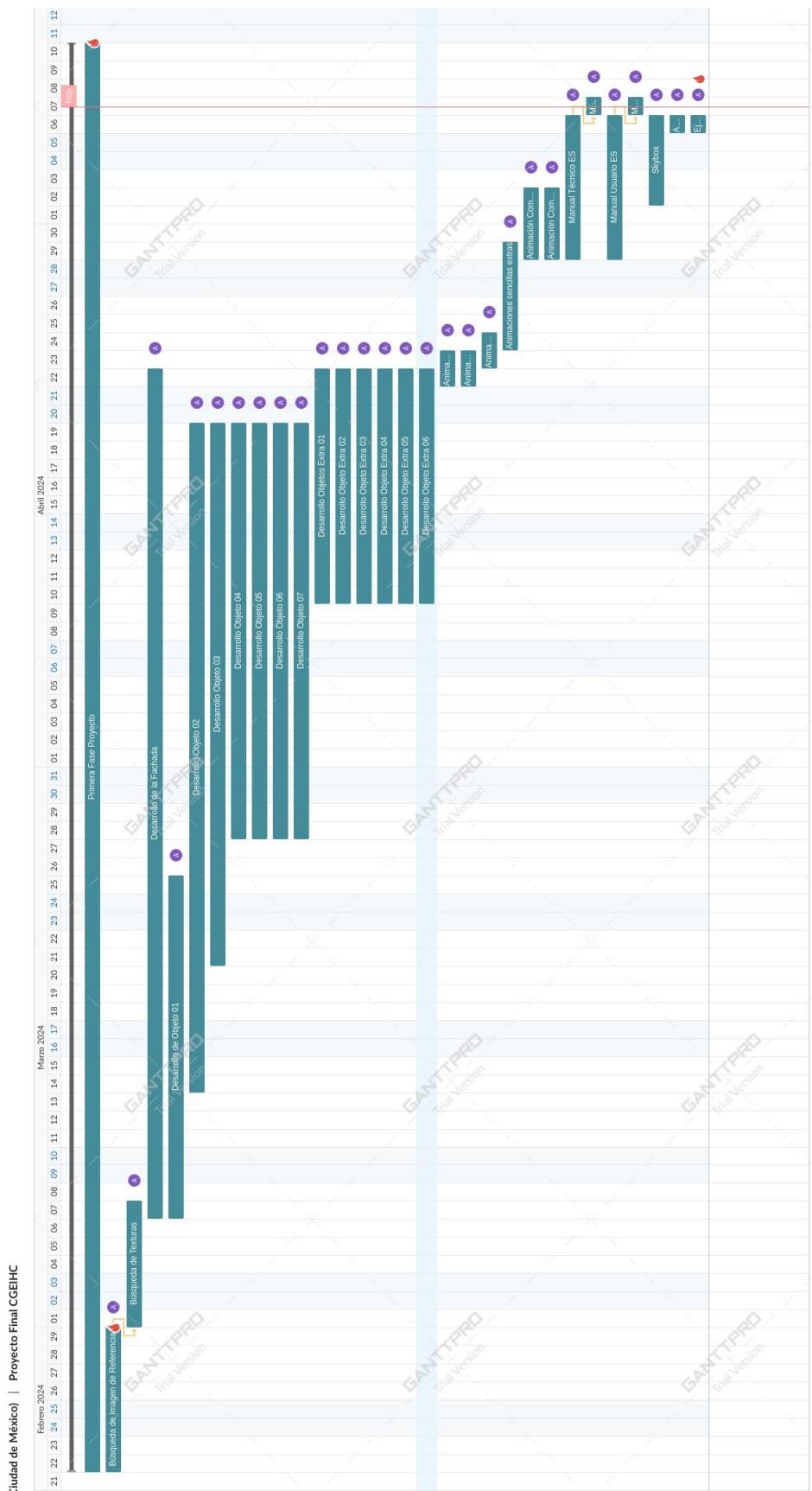


Diagrama de Gantt por semanas.

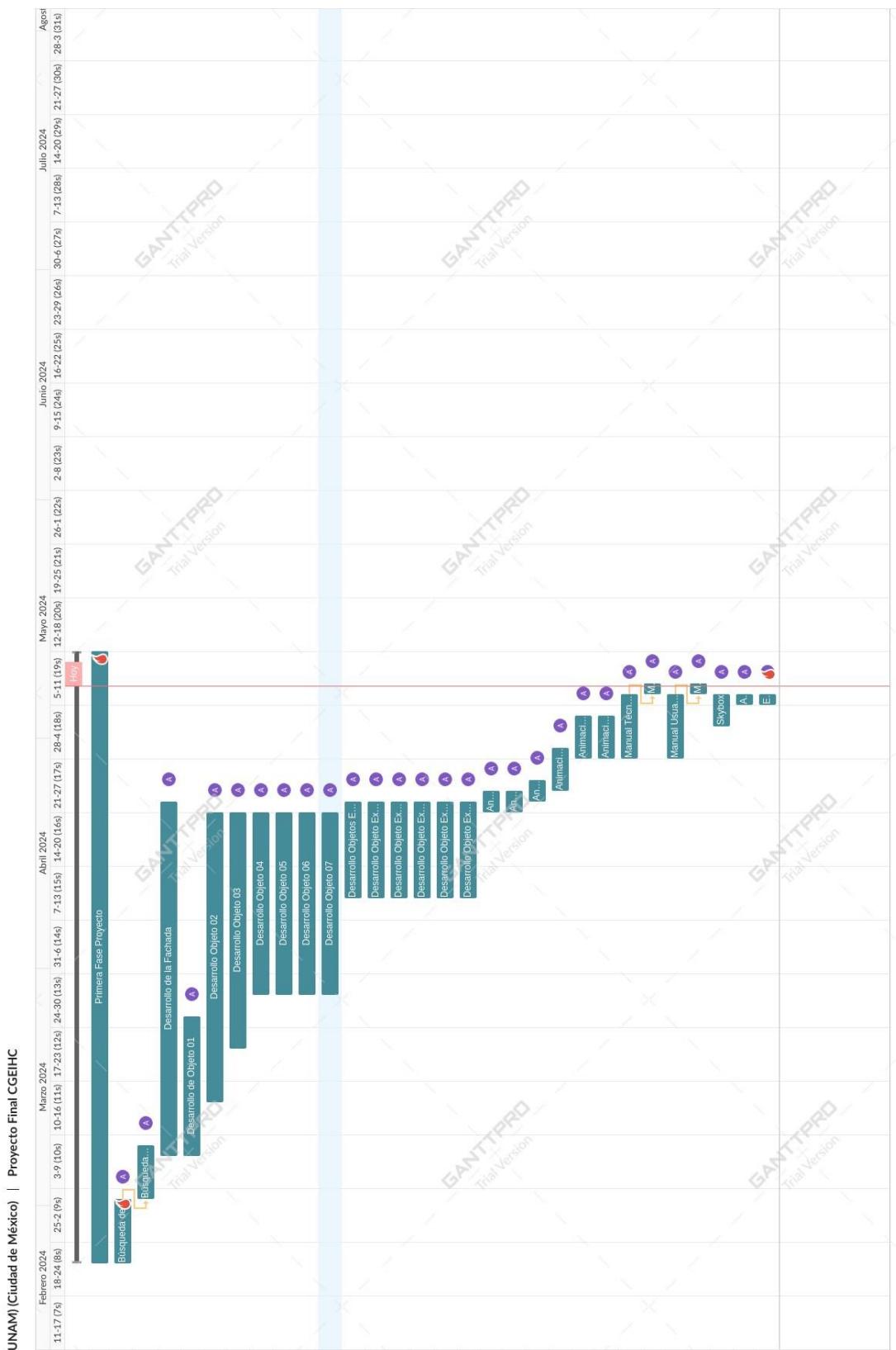


Diagrama de Gantt segunda fase

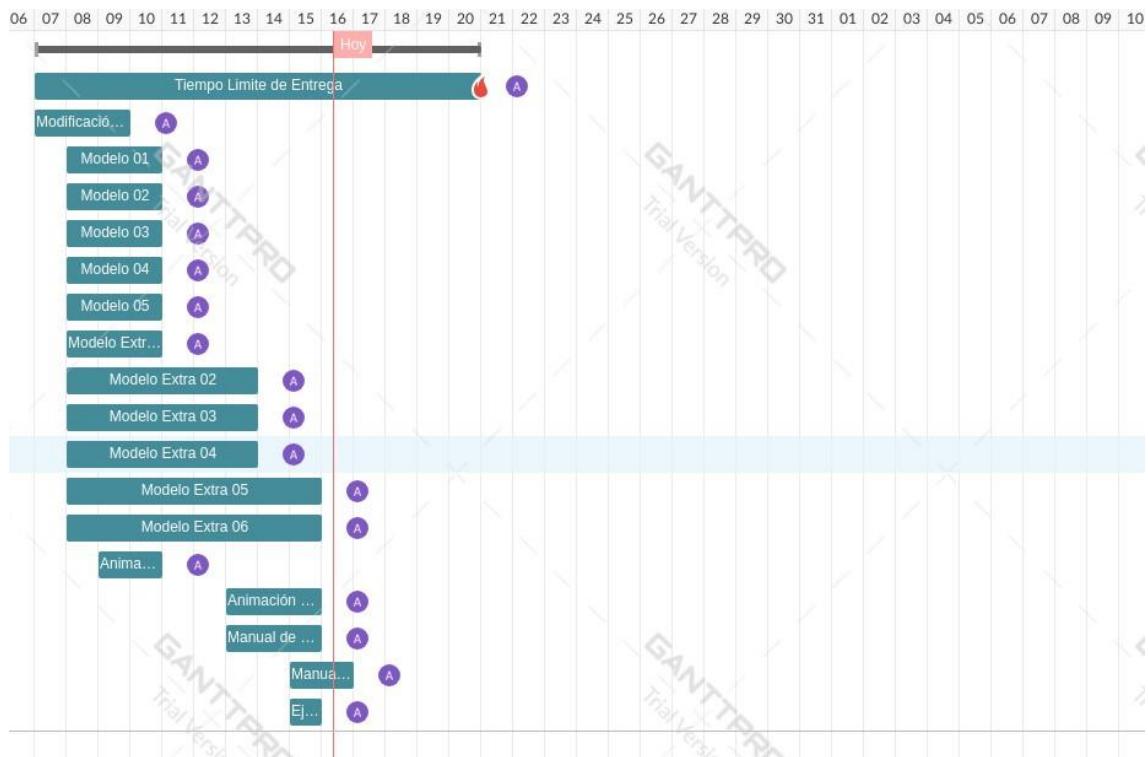
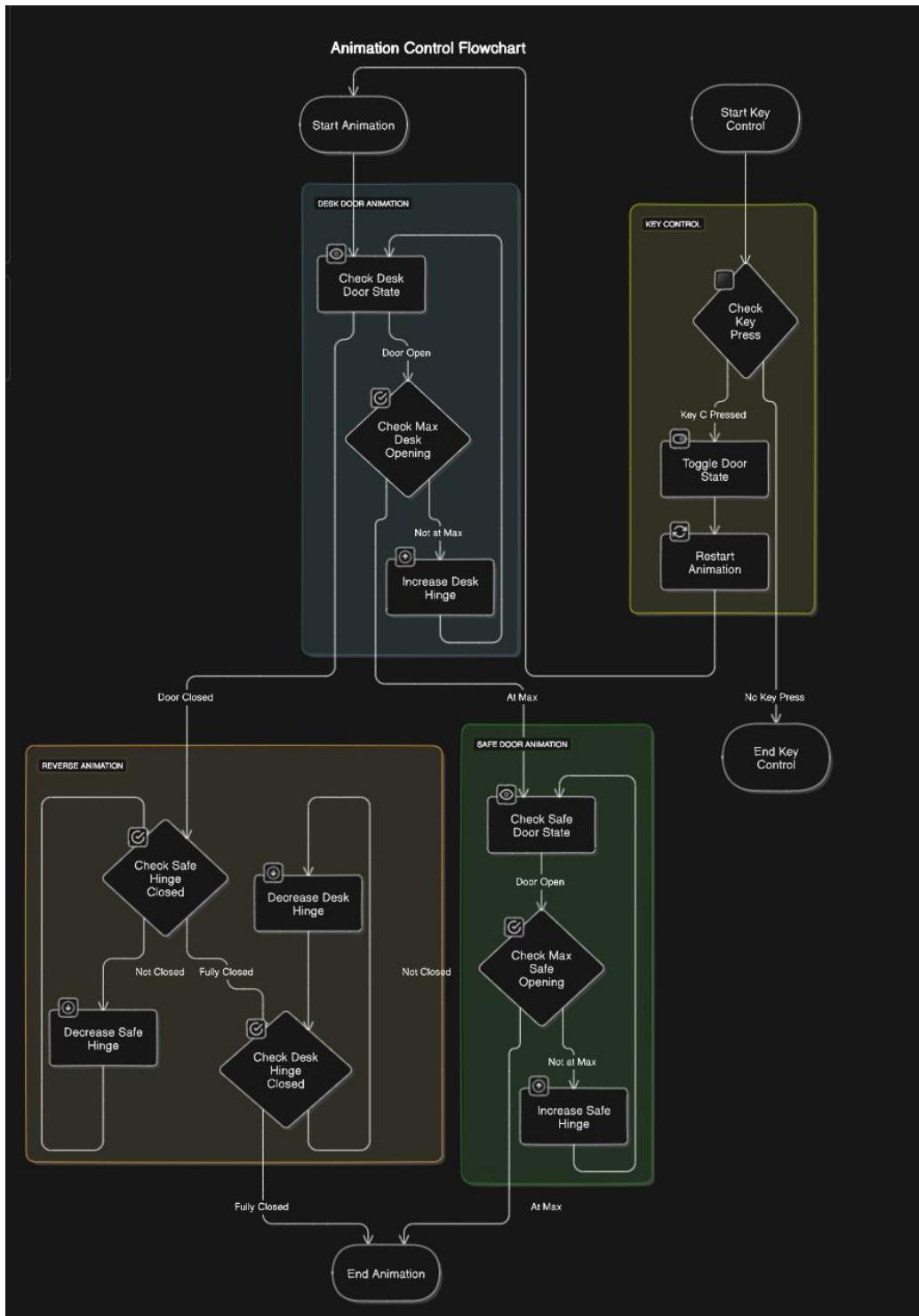


Diagrama de Flujo del Software

Animación Sencilla 01

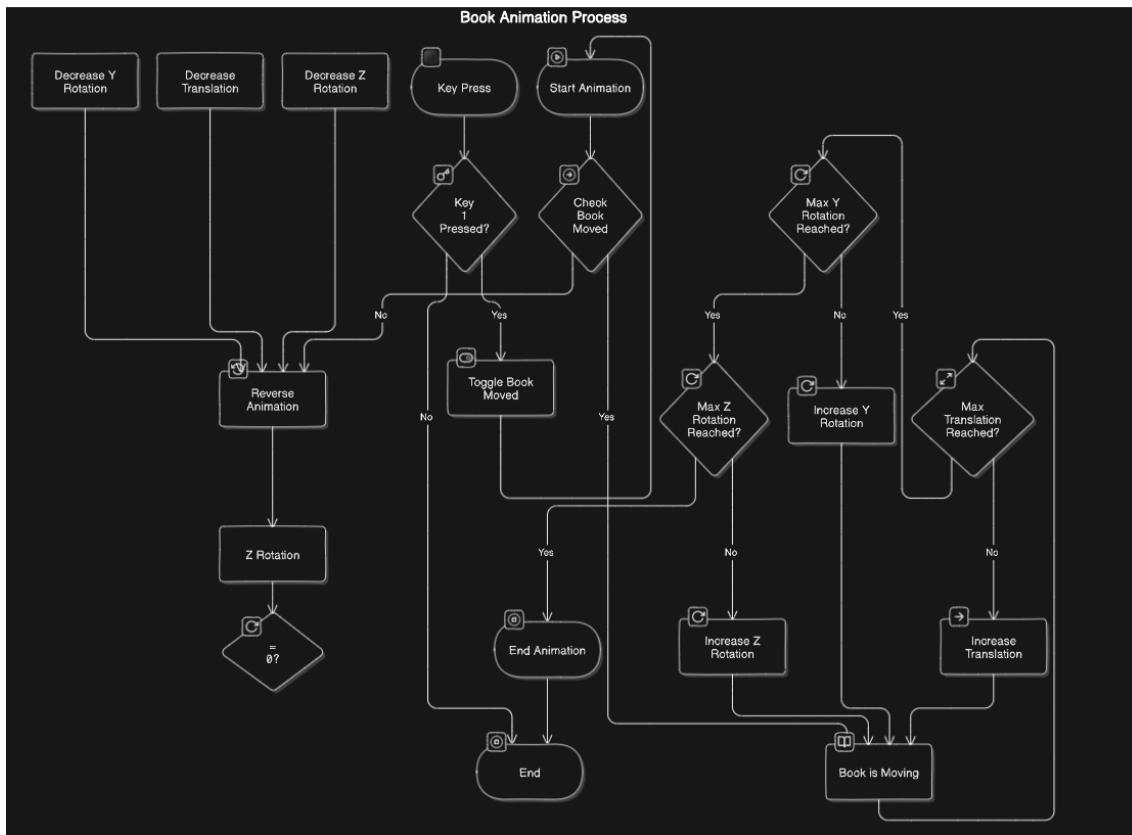
Aquí se agrupan 3 animaciones que se basan en la misma lógica la cuales consisten en:

- Abrir las puertas de la entrada a la casa
- Abrir las puertas inferiores del librero
- Abrir la puerta del escritorio y seguido de la puerta de la caja fuerte



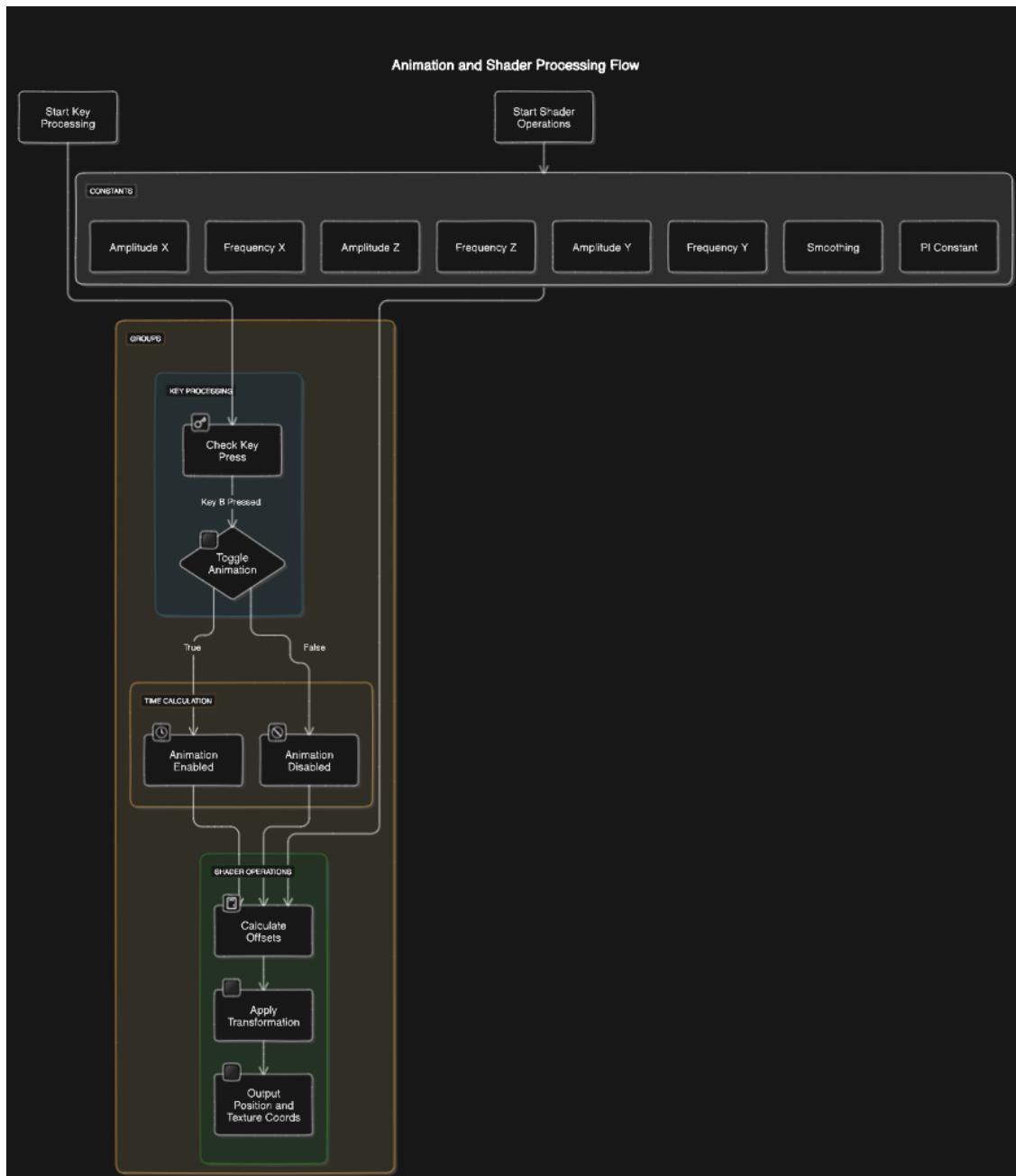
Animación Sencilla 02

Esta animación consiste en agarrar un libro, girarlo para ver la portada e inclinarlo para una mayor facilidad de observar



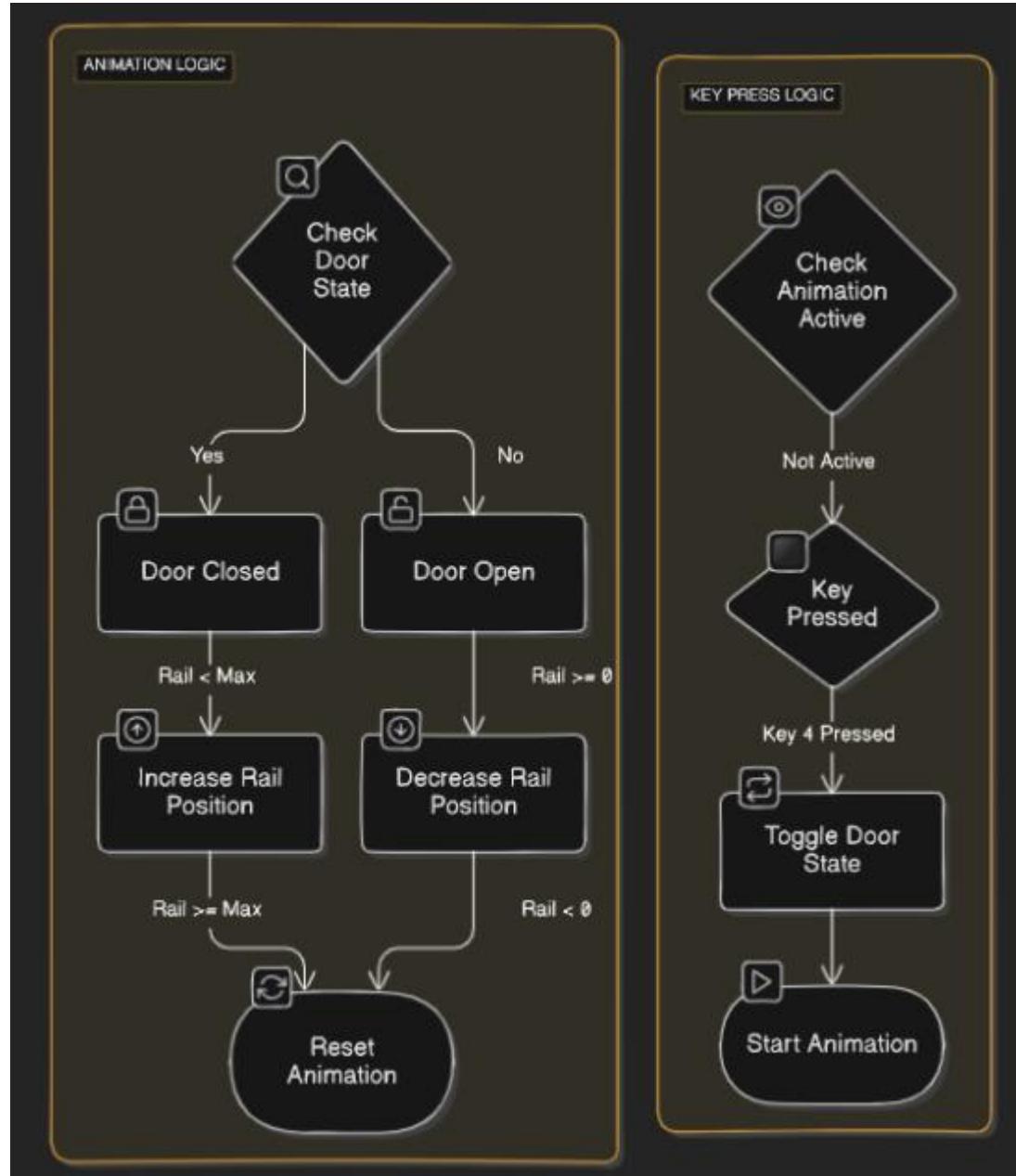
Animación Sencilla 03

Esta animación consiste en simular que la protogema (ítem especial en el contexto de Genshin Impact) está flotando.



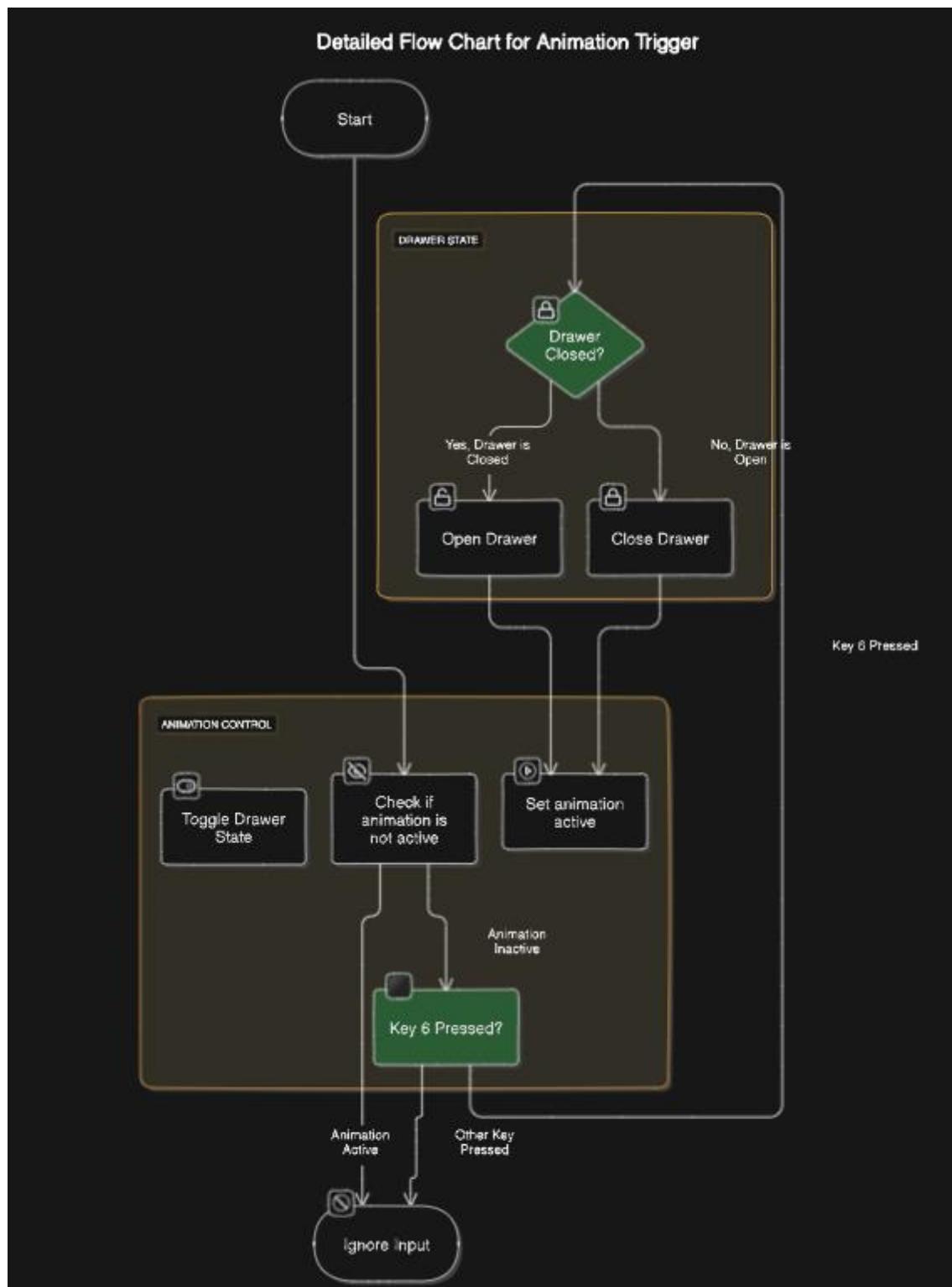
Animación Sencilla 04

Esta animación consiste en abrir la puerta del segundo cuarto la cual tiene un estilo oriental.



Animación Sencilla 05

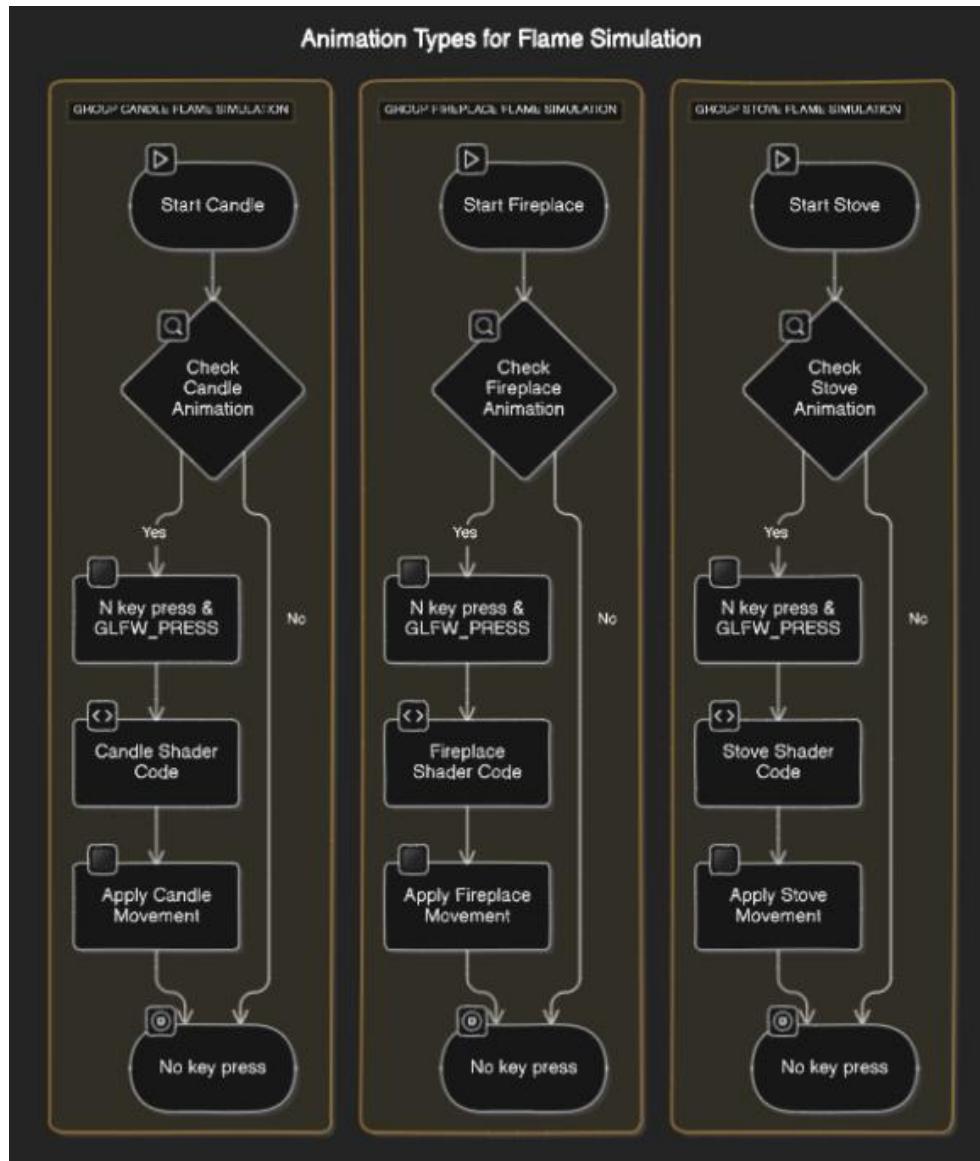
Esta animación se encarga de abrir los cajones del mueble que esta a lado de la cama. En total son 2 cajones.



Animación Compleja 01

Esta animación agrupa 3 tipos:

- Simular el movimiento de la flama de una vela.
- Simular las llamas de una chimenea
- Simular las llamas de una estufa



Animación Compleja 02

Esta animación consiste en 3 fases:

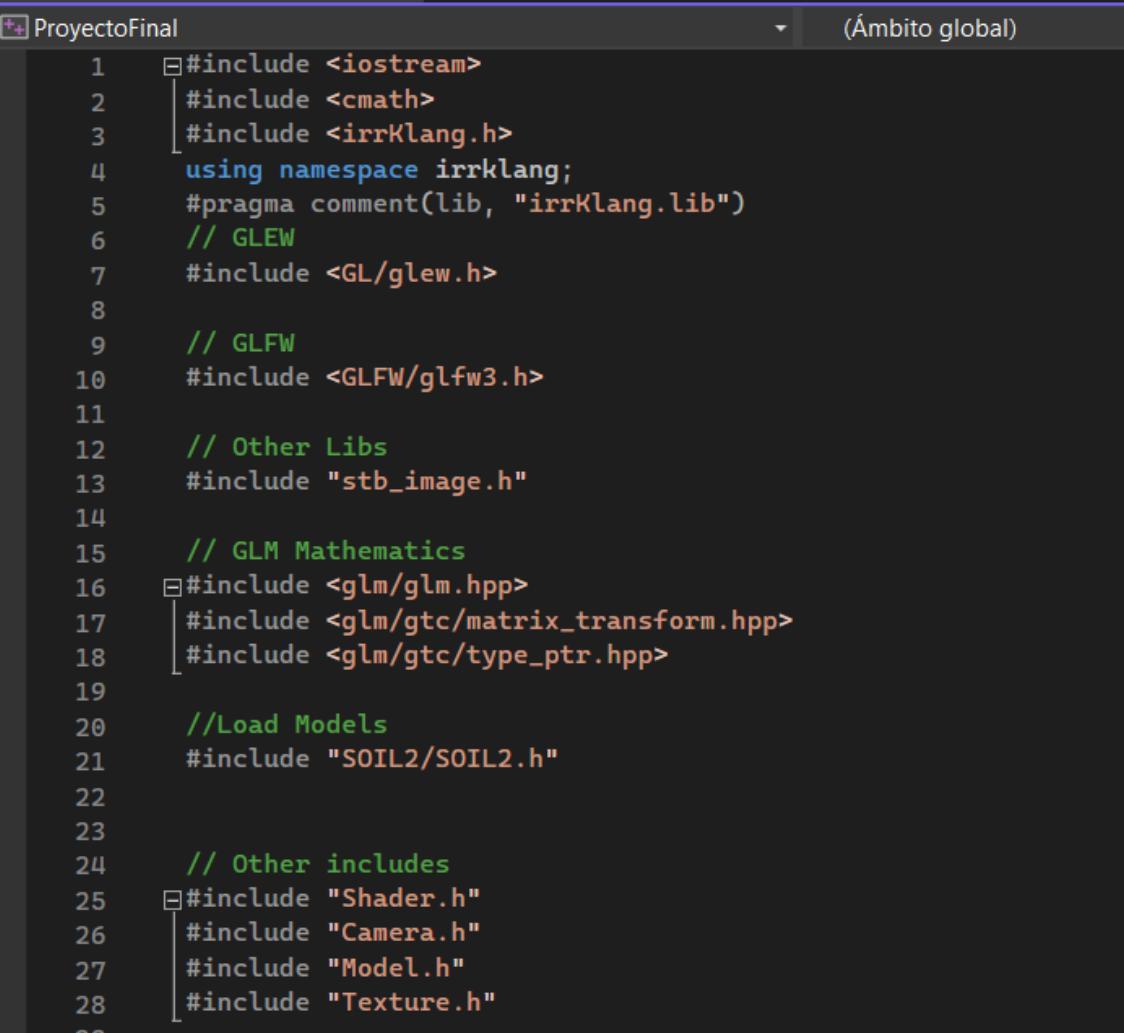
- Cuando se enciende la flama de la estufa
- Inicia el movimiento del aceite simulando que estáriendo unas piernas de pollo
- Aparece humo para mayor ambientalización



Documentación del Código

A continuación, se explicarán como se realizaron las animaciones (considerando el código principal y shaders si se ocuparon).

Bibliotecas Utilizadas.



```

1  #include <iostream>
2  #include <cmath>
3  #include <irrklang.h>
4  using namespace irrklang;
5  #pragma comment(lib, "irrklang.lib")
6  // GLEW
7  #include <GL/glew.h>
8
9  // GLFW
10 #include <GLFW/glfw3.h>
11
12 // Other Libs
13 #include "stb_image.h"
14
15 // GLM Mathematics
16 #include <glm/glm.hpp>
17 #include <glm/gtc/matrix_transform.hpp>
18 #include <glm/gtc/type_ptr.hpp>
19
20 // Load Models
21 #include "SOIL2/SOIL2.h"
22
23
24 // Other includes
25 #include "Shader.h"
26 #include "Camera.h"
27 #include "Model.h"
28 #include "Texture.h"
29

```

Biblioteca irrKlang.h

irrKlang es una biblioteca de sonido y motor de audio de alto nivel para C++, C# y todos los lenguajes .NET. Es conocida por su facilidad de uso y su capacidad para reproducir una variedad de formatos de archivo de audio.

Las ventajas de usar irrKlang a diferencia de otras librerías del mismo tipo son:

- **API de alto nivel:** irrKlang proporciona una API simple y potente para la reproducción de sonido en aplicaciones 2D y 3D como juegos, visualizaciones científicas y aplicaciones multimedia2.
- **Soporte de sonido 3D:** Tiene soporte incorporado para sonido 3D en todas las plataformas y controladores de audio, diseñado para ser eficiente y no consumir mucho tiempo de CPU2.

- **Emulación de audio 3D para hardware de baja gama:** irrKlang incluye un emulador de búfer de sonido 3D de alto rendimiento, proporcionando una experiencia de sonido casi real incluso en hardware de gama baja

Biblioteca glm

La biblioteca GLM (OpenGL Mathematics) es una herramienta esencial para desarrolladores que trabajan con OpenGL. Aquí tienes una descripción detallada de lo que hace:

Tipos de datos y funciones de GLSL:

- GLM implementa los tipos de datos y las funciones de GLSL (OpenGL Shading Language) en C++. Esto permite a los programadores trabajar con matrices, vectores y otras estructuras de datos de manera similar a como lo harían en GLSL.
- Por ejemplo, define tipos como `glm::mat3` o `glm::vec4` para tratar con matrices o vectores en aplicaciones gráficas.

Funciones matemáticas y transformaciones:

- GLM proporciona una amplia gama de funciones matemáticas y operaciones de transformación. Estas incluyen:
- Transformaciones de matrices: Rotación, traslación, escalado, proyección, etc.
- Operaciones vectoriales: Producto escalar, producto cruzado, normalización, etc.
- Funciones trigonométricas y exponenciales: Seno, coseno, exponencial, logaritmo, etc.
- Funciones de álgebra lineal: Determinante, inversa, diagonalización, etc.

Biblioteca SOIL2

es una pequeña biblioteca en C utilizada principalmente para cargar texturas en OpenGL. Está basada en stb_image, un código de dominio público. Carga un archivo de imagen directamente en una textura 2D de OpenGL.

Biblioteca stb_image

Es una biblioteca de imágenes de dominio público (o con licencia MIT) para C/C++. Es una biblioteca de un solo archivo que proporciona funciones para cargar y decodificar imágenes desde archivos o memoria.

Funciones Destacadas

KeyCallback

Esta función se encarga de detectar que tecla presiona el usuario, lo cual servirá para poder activar correctamente las animaciones.

```
// Is called whenever a key is pressed/released via GLFW
void KeyCallback(GLFWwindow *window, int key, int scanCode, int action, int mode)
{
    if (GLFW_KEY_ESCAPE == key && GLFW_PRESS == action)
    {
        glfwSetWindowShouldClose(window, GL_TRUE);
    }

    if (key >= 0 && key < 1024)
    {
        if (action == GLFW_PRESS)
        {
            keys[key] = true;
        }
        else if (action == GLFW_RELEASE)
        {
            keys[key] = false;
        }
    }
}
```

DoMovement

Esta función se encarga una vez presionada la tecla, hacer las modificaciones necesarias para poder visualizarlo en pantalla en su defecto el que funcionen las animaciones.

```
// Moves/alters the camera positions based on user input
void DoMovement()
{
    // Camera controls
    if (keys[GLFW_KEY_W] || keys[GLFW_KEY_UP])
    {
        camera.ProcessKeyboard(FORWARD, deltaTime);
    }

    if (keys[GLFW_KEY_S] || keys[GLFW_KEY_DOWN])
    {
        camera.ProcessKeyboard(BACKWARD, deltaTime);
    }

    if (keys[GLFW_KEY_A] || keys[GLFW_KEY_LEFT])
    {
        camera.ProcessKeyboard(LEFT, deltaTime);
    }

    if (keys[GLFW_KEY_D] || keys[GLFW_KEY_RIGHT])
    {
        camera.ProcessKeyboard(RIGHT, deltaTime);
    }
}
```

MouseCallback

Esta función nos sirve para poder mover la vista con ayuda del ratón.

```
void MouseCallback(GLFWwindow *window, double xPos, double yPos)
{
    if (firstMouse)
    {
        lastX = xPos;
        lastY = yPos;
        firstMouse = false;
    }

    GLfloat xOffset = xPos - lastX;
    GLfloat yOffset = lastY - yPos; // Reversed since y-coordinates go from bottom to left

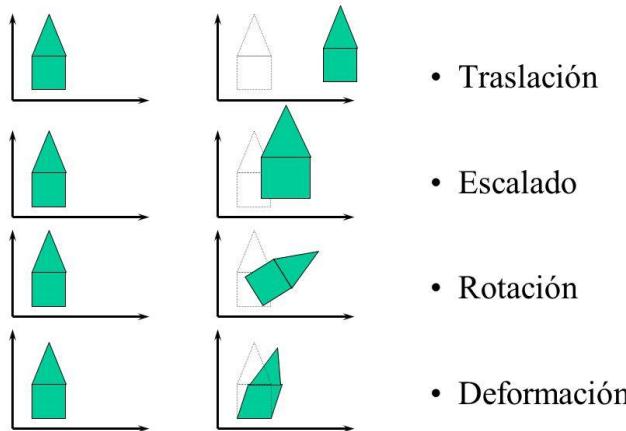
    lastX = xPos;
    lastY = yPos;

    camera.ProcessMouseMovement(xOffset, yOffset);
}
```

Pivotes y transformaciones básicas.

Un pivote es el punto central de cualquier primitiva básica o modelos según aplique. En el pivote se reflejará las transformaciones básicas que realicemos.

Transformaciones en 2D



Ahí se considera que el pivote está centrado, pero también podemos mover los pivotes a donde lo necesitemos, pero dejar la figura en su lugar. Estos son pivotes temporales los cuales sirven por ejemplo para animar cuando una puerta se abre.

En el código esto se refleja en la matriz de modelo y las matrices temporales se usa la biblioteca GLM y finalmente se vería como se muestra a continuación:

```
//=====
//Matrices Auxiliares
glm::mat4 model(1);
glm::mat4 model_Bisagra_izq(1); //Matriz Temporal para bisagras.
glm::mat4 model_Bisagra_der(1); //Matriz Temporal para bisagras.
glm::mat4 model_BisagraLibrero_der(1); //Matriz Temporal para bisagras.
glm::mat4 model_BisagraLibrero_izq(1); //Matriz Temporal para bisagras.
glm::mat4 model1_Libro01(1); //Matriz Temporal para libro 01
glm::mat4 model2_Libro01(1); //Matriz Temporal para libro 01
glm::mat4 model1_Libro02(1); //Matriz Temporal para libro 02
glm::mat4 model2_Libro02(1); //Matriz Temporal para libro 02
glm::mat4 model1_Libro03(1); //Matriz Temporal para libro 03
glm::mat4 model2_Libro03(1); //Matriz Temporal para libro 03

glm::mat4 model_Bisagra_escritorio(1); //Matriz Temporal para bisagras.
glm::mat4 model_Bisagra_caja(1); //Matriz Temporal para bisagras.
```

Animación Sencilla 01

Esta animación engloba 3 animaciones que tienen la misma esencia, pero describiremos la más completa que sería la apertura de la puerta del escritorio seguido de la puerta de la caja fuerte:

Se ocupan 2 variables de tipo bool:

- La primera llamada “animacion04” para que la animación no se vea interrumpida cuando se presione la misma tecla.
- La segunda variable llamada “puerta_escritorio” que nos indicara si las puertas están cerradas o abiertas.

2 variables (bisagra_escritorio y bisagra_caja) que servirán para que se aprecie la apertura de las puertas.

2 variables (MAXIMA_APERTURA_ESCRITORIO y MAXIMA_APERTURA_CAJA) que actuarán como valor máximo de apertura.

```
//=====
//Variables para la animación sencilla 04 -> Puertas del escritorio
bool animacion_04 = false;
bool puerta_escritorio = false;
float bisagra_escritorio = 0.0f;
float bisagra_caja = 0.0f;
const float MAXIMA_APERTURA_ESCRITORIO = 90;
const float MAXIMA_APERTURA_CAJA = 82;
```

Para poder activar la animación se presiona la tecla C, para lograr esto primero pasamos por una condicional para evitar interrumpir la animación si está en funcionamiento.

Si la variable puerta_escritorio es true, cuando pase por la tercera condicional, se le asignará el valor de false o viceversa y animación_04 tendrá el valor true.

```
//=====
// Tecla C para activar la animación del las puertas del escritorio
if (!animacion_04)
{
    if (key == GLFW_KEY_C && action == GLFW_PRESS)
    {
        if (puerta_escritorio) {
            puerta_escritorio = false;
        }
        else {
            puerta_escritorio = true;
        }
        animacion_04 = true;
    }
}
```

Se describe la tabla de casos que se consideró para realizar la animación.

Puerta_escritorio	Bisagra_escritorio	Bisagra_caja	Acción
Verdadero	Menor que MAXIMO	-----	Incrementar bisagra_escritorio
Verdadero	Mayor-igual que MAXIMO	Menor que MAXIMO	Incrementar bisagra_caja
Falso	-----	Mayor o igual a 0	Decrementar bisagra_caja
Falso	Mayor que 0	-----	Decrementar bisagra_escritorio

```
893 //=====
894 //Animación para realizar el movimiento del las puerta del librero y caja fuerte
895 if (puerta_escritorio) {
896     if (bisagra_escritorio < MAXIMA_APERTURA_ESCRITORIO) {
897         bisagra_escritorio += velocidad_rotacion * glfwGetTime();
898     } else {
899         if (bisagra_caja < MAXIMA_APERTURA_CAJA) {
900             bisagra_caja += velocidad_rotacion * glfwGetTime();
901         } else {
902             animacion_04 = false;
903         }
904     } else {
905         if (bisagra_caja >= 0) {
906             bisagra_caja -= velocidad_rotacion * glfwGetTime();
907         } else {
908             if (bisagra_escritorio >= 0) {
909                 bisagra_escritorio -= velocidad_rotacion * glfwGetTime();
910             } else {
911                 animacion_04 = false;
912             }
913         }
914     }
915 }
```

Nota: Normalizaremos la velocidad de las animaciones con ayuda de “glfwGetTime”

Para poder dibujar nuestras puertas necesitamos usar modelado jerárquico considerando lo siguiente:

- Todo objeto que deseamos animar debe centrarse en el mundo
- El orden de las transformaciones interfiere en el efecto de la animación deseada.

Considerando lo siguiente necesitamos usar 2 pivotes para realizar la animación. El primer pivote se encarga de ser el punto donde se realizará la rotación y el segundo pivote donde estará situada nuestra figura. Visualizando lo anterior en el código haremos uso de una matriz de modelo temporal para la correcta apertura de las puertas.

```
//=====
//Puerta del Librero
model = glm::mat4(1);
model_Bisagra_escritorio = model = glm::translate(model, glm::vec3(26.65f, 3.70f, -13.0f));//Matriz temporal para el pivote de apertura
model = glm::rotate(model_Bisagra_escritorio, glm::radians(-bisagra_escritorio), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::translate(model, glm::vec3(-1.16f, 0.0f, 0.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
objeto04_Puerta.Draw(lightingShader);

//=====
//Puerta de la Caja Fuerte
model = glm::mat4(1);
model_Bisagra_caja = model = glm::translate(model, glm::vec3(24.49f, 3.54f, -11.88f));//Matriz temporal para el pivote de apertura
model = glm::rotate(model_Bisagra_caja, glm::radians(bisagra_caja), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::translate(model, glm::vec3(0.89f, 0.0f, 0.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
objeto04_Puerta_Caja.Draw(lightingShader);
```

Animación Sencilla 02

Esta animación se aplica a 3 libros, los cuales son independientes entre sí. Tomando como ejemplo la realidad cuando tomamos un libro, lo giramos para ver la portada y al mismo tiempo lo inclinamos para poder apreciar mejor. Considerando lo anterior se realiza la animación correspondiente:

Se ocupan 2 variables de tipo bool:

- La primera llamada "animación_02_libro01" para que la animación no se vea interrumpida cuando se presione la misma tecla.
- La segunda variable llamada "libro_trasladado1" que nos indicara si el libro está en su cajón o está posicionado para una mejor lectura.

Se ocupan 3 variables encargadas de realizar la simulación:

- **Traslación_libro01:** Se encarga de sacar el libro del cajón o regresarlo según sea el caso
- **Rotación_ejey:** Variable encargada de rotar el libro en el eje Y para ver la portada
- **Rotación_ejez:** Variable encargada de inclinar el libro en el eje Z para una mayor comodidad

Se usan variables para limitar el movimiento deseado:

- **ROTACION_MAXIMA_Y:** Para apreciar la portada del libro, se establece como límite una rotación de 90°.
- **INCLINACION_MAXIMA:** Para mejorar la comodidad, no es necesario acostar todo el libro, así que se establece una inclinación de 15°.
- **TRASLACION_MAXIMA:** Para que el libro no se pase del punto donde queremos que se ubique, definimos este límite.
- **Velocidad_rotacion:** Encargada de hacer que la rotación del libro se vea lo más suave posible.
- **velocidad_traslacion:** Encargada de que la traslación del libro sea lo más suave posible.

```

75 //=====
76 //Variables para la animación sencilla 03 -> Libro 01
77     bool animacion_02_libro01 = false;
78     bool libro_trasladado1 = false;
79     float rotacion_ejey = 0.0f;
80     float rotacion_ejez = 0.0f;
81     const float ROTACION_MAXIMA_Y = 90;
82     const float INCLINACION_MAXIMA = 15;
83     const float velocidad_rotacion = 0.005;
84     const float velocidad_traslacion = 0.0006;
85     const float TRASLACION_MAXIMA = 2.0;
86     float traslacion_libro01 = 0.0f;
87

```

Para poder activar la animación debemos presionar las siguientes teclas (Nota: Los libros son independientes, puedes sacar solo 1 o los 3 al mismo tiempo):

Tecla 1	Libro 01
Tecla 2	Libro 02
Tecla 3	Libro 03

Primero se verifica que no esté en funcionamiento la animación, si es que está en funcionamiento esperamos a que acabe.

```
//=====
// Tecla 1 para activar la animación del libro
if (!animacion_02_libro01)
{
    if (key == GLFW_KEY_1 && action == GLFW_PRESS)
    {
        if (libro_trasladado1) {
            libro_trasladado1 = false;
        } else {
            libro_trasladado1 = true;
        }
        animacion_02_libro01 = true;
    }
}
```

Se describe la tabla de casos que se consideró para realizar la animación.

libro_trasladado1	traslacion_libro01	rotacion_ejey	rotacion_ejez	Acción
Verdadero	Menor que máximo	-----	-----	Incrementar traslacion_libro01
Verdadero	Igual o mayor a máximo y Menor que máximo en eje Y	Menor que máximo	-----	Incrementar rotacion_ejey
Verdadero	Igual o mayor a máximo en eje Y y Menor que máximo en eje Z	Igual o mayor a máximo	Menor que máximo	Incrementar rotacion_ejez
Verdadero	Igual o mayor a máximo en eje Y y Z	-----	-----	animacion_02_libro01 a falso
Falso	Mayor o igual a 0	Mayor o igual a 0	Mayor o igual a 0	Decrementar rotacion_ejez, rotacion_ejey, y traslacion_libro01 respectivamente
Falso	Menor que 0	-----	-----	animacion_02_libro01 a falso

```

798     //=====
799     //Animación para realizar la el movimiento del libro 01 del librero
800     if (libro_trasladado1) {
801         if (traslacion_libro01 < TRASLACION_MAXIMA) {
802             traslacion_libro01 += velocidad_traslacion * glfwGetTime();
803         } else {
804             if (rotacion_ejey < ROTACION_MAXIMA_Y) {
805                 rotacion_ejey += velocidad_rotacion * glfwGetTime();
806             } else {
807                 if (rotacion_ejez < INCLINACION_MAXIMA) {
808                     rotacion_ejez += velocidad_rotacion * glfwGetTime();
809                 } else {
810                     animacion_02_libro01 = false;
811                 }
812             }
813         }
814     } else {
815         if (rotacion_ejez >= 0) {
816             rotacion_ejez -= velocidad_rotacion * glfwGetTime();
817         } else {
818             if (rotacion_ejey >= 0) {
819                 rotacion_ejey -= velocidad_rotacion * glfwGetTime();
820             } else {
821                 if (traslacion_libro01 >= 0)
822                     traslacion_libro01 -= velocidad_traslacion * glfwGetTime();
823             } else {
824                 animacion_02_libro01 = false;
825             }
826         }
827     }
828 }
```

Al igual que en la animación anterior, haremos uso de 2 matrices temporales:

- Una para colocar el libro donde deseamos
- La otra matriz donde la animación se aplica.

```

//=====
//Libro 01 del librero
model = glm::mat4(1);
model1_Libro01 = model = glm::translate(model, glm::vec3(24.86f, 10.0f, -21.60f)); //Posición en el librero
model2_Libro01 = model = glm::translate(model1_Libro01, glm::vec3(0.0f, 0.0f, traslacion_libro01)); //Posición en el librero
model2_Libro01 = model = glm::rotate(model2_Libro01, glm::radians(-rotacion_ejey), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model2_Libro01, glm::radians(rotacion_ejez), glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
objeto03_Libro01.Draw(LightingShader);
//
```

Animación Sencilla 03

Para esta animación se usa un shader para mover la protogema(estrella) simulando que está flotando.

Se ocupan 2 variables:

- La primera llamada “animaciónSencilla_05” de tipo bool para que la animación inicie cuando se presione la misma tecla.
- La segunda variable llamada “proto” servirá para mandar al shader el reloj normalizado (glfwGetTime).

```
float velocidad_mano = -0.000001f;
//=====================================================================
//Variables para la animación sencilla 05 -> Movimiento de la protogema
bool animacionSencilla_05 = false;
float proto = 0.0f;
```

Aquí al activar con la tecla B, pasamos el valor del reloj a la variable proto

```
//=====================================================================
//Animación Sencilla 3 -> Movimiento de la protogema
if (animacionSencilla_05) {
    proto = glfwGetTime();
}
else {
    proto = 0.0f;
}
```

El shader que ocupamos para mover la protogema es el siguiente:

```
1 #version 330 core
2 layout (location = 0) in vec3 aPos;
3 layout (location = 1) in vec3 aNormal;
4 layout (location = 2) in vec2 aTexCoords;
5
6 const float amplitud_x = 0.05; // Amplitud del movimiento en el eje x
7 const float frequency_x = 7.0; // Frecuencia del movimiento en el eje x
8 const float amplitud_z = 0.09; // Amplitud del movimiento en el eje z
9 const float frequency_z = 1.5; // Frecuencia del movimiento en el eje z
10 const float amplitud_y = 0.09; // Amplitud del movimiento en el eje y (levantar)
11 const float frequency_y = 1.0; // Frecuencia del movimiento en el eje y (levantar)
12 const float suavisado = 110.0f;// Suavizar el movimiento del eje x
13 const float PI = 3.14159;
14
15 out vec2 TexCoords;
16
17 uniform mat4 model;
18 uniform mat4 view;
19 uniform mat4 projection;
20 uniform float time;
21
22 void main()
23 {
24     float x_offset = amplitud_x * cos(PI * frequency_x * time/suavisado);
25     float z_offset = amplitud_z * sin(PI * frequency_z * time);
26     float y_offset = amplitud_y * sin(PI * frequency_y * time);
27
28     vec3 offset = vec3(x_offset, y_offset, z_offset);
29
30     gl_Position = projection * view * model * vec4(aPos + offset, 1.0);
31     TexCoords = vec2(aTexCoords.x,aTexCoords.y);
32 }
```

Mediante una función senoidal modificamos la traslación en el eje z y eje y; una función coseno para el eje x.

```
//=====
animacionSencilla_03.Use();
modelLoc = glGetUniformLocation(animacionSencilla_03.Program, "model");
viewLoc = glGetUniformLocation(animacionSencilla_03.Program, "view");
projLoc = glGetUniformLocation(animacionSencilla_03.Program, "projection");
// Set matrices
glUniformMatrix4fv(viewLoc, 1, GL_FALSE, glm::value_ptr(view));
glUniformMatrix4fv(projLoc, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
model = glm::mat4(1);
glUniform1f(glGetUniformLocation(animacionSencilla_03.Program, "time"), proto);
model = glm::translate(model, glm::vec3(25.45f, 2.86f, -11.30f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
objeto04_protogema.Draw(animacionSencilla_03);
```

Finalmente, para dibujar nuestra protogema, la movemos a la posición deseada.

Animación Sencilla 04

Para esta animación se emplea para realizar la apertura de la puerta del segundo cuarto. Para realizar el movimiento se considera lo siguiente:

Se ocupan 2 variables de tipo bool para controlar la animación:

- La primera llamada “animacion01_Cuarto02” para que la animación no se vea interrumpida cuando se presione la misma tecla.
- La segunda variable llamada “puertaCerrada_cuarto02” que nos indicara si la puerta está cerrada o abierta.

Adicional se ocupan 3 variables extras:

- **APERTURA CUARTO:** La cual sirve como valor máximo de traslación para evitar que la puerta se desplace más de lo necesario.
- **Riel_puerta:** Sirve para realizar la traslación y simular el movimiento deseado.
- **Incremento_riel:** Sirve para hacer el incremento de riel_puerta pero debemos normalizar ese valor por lo que multiplicamos por GLFGETTIME () .

```
146 //=====
147 //Variables para la animación 01 del cuarto 2 -> Apertura de la puerta
148 bool animacion_01_cuarto02 = false;
149 bool puertaCerrada_cuarto02 = false;
150 const float APERTURA CUARTO = 7.4;
151 float riel_puerta= 0.0f;
152 float incremento_riel = 0.001f;
153
```

Para activar la animación debemos apretar la tecla 4 para abrir o cerrar la puerta, esto se realiza de la siguiente forma:

```

1300 //=====
1301 //Tecla 4 para activar la animación -> Apertura de Puerta del cuarto 02
1302 if (!animacion_01_cuarto02) {
1303     if (key == GLFW_KEY_4 && action == GLFW_PRESS)
1304     {
1305         // Si la puerta está cerrada, abrir
1306         if (puertaCerrada_cuarto02) {
1307             puertaCerrada_cuarto02 = false; // La puerta se cierra
1308         }
1309         else { // Si la puerta está abierta, cerrar
1310             puertaCerrada_cuarto02 = true; // La puerta está cerrada nuevamente
1311         }
1312         animacion_01_cuarto02 = true;
1313     }
1314 }

```

A partir de la siguiente tabla de casos se considera el movimiento de la puerta:

Caso	Condición	Estado de Variables	Acción	Resultado
1	cajonSup_cuarto02 == true y riel_CajonSup < APERTURA_CAJON	cajonSup_cuarto02 = true riel_CajonSup < APERTURA_CAJON	riel_CajonSup += incremento_cajon * glfwGetTime()	Incrementa riel_CajonSup hasta APERTURA_CAJON
2	cajonSup_cuarto02 == true y riel_CajonSup >= APERTURA_CAJON	cajonSup_cuarto02 = true riel_CajonSup >= APERTURA_CAJON	animacion_02_cuarto02 = false	Detiene la animación
3	cajonSup_cuarto02 == false y riel_CajonSup > 0	cajonSup_cuarto02 = false riel_CajonSup > 0	riel_CajonSup -= incremento_cajon * glfwGetTime()	Decrementa riel_CajonSup hasta 0
4	cajonSup_cuarto02 == false y riel_CajonSup <= 0	cajonSup_cuarto02 = false riel_CajonSup <= 0	animacion_02_cuarto02 = false	Detiene la animación

```

1120 //=====
1121 //Animación para realizar la apertura del cajón superior del cuarto 02
1122 if (cajonSup_cuarto02) {
1123     if (riel_CajonSup < APERTURA_CAJON) { //Si la apertura es menor al máximo
1124         riel_CajonSup += incremento_cajon * glfwGetTime(); //incrementamos las bisagras
1125     }
1126     else {
1127         // La animación ha terminado, restablecer la variable
1128         animacion_02_cuarto02 = false;
1129     }
1130 }
1131 else {
1132     if (riel_CajonSup >= 0) {
1133         riel_CajonSup -= incremento_cajon * glfwGetTime();
1134     }
1135     else {
1136         // La animación ha terminado, restablecer la variable
1137         animacion_02_cuarto02 = false;
1138     }
1139 }

```

Para dibujar nuestra puerta usamos una matriz temporal para posicionar nuestra puerta en el lugar deseado

```

objetoExterior_cuarto02.Draw(lightingShader);
//=====
//Puerta de la entrada del cuarto 02
model = glm::mat4(1);
model_puerta_cuarto02 = model = glm::translate(model, glm::vec3(-16.46f, 22.57f, -15.65f)); //Matriz temporal
model = glm::translate(model_puerta_cuarto02, glm::vec3(0.0f, 0.0f, riel_puerta));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Puerta_Cuarto02.Draw(lightingShader);

```

Animación Sencilla 05

Esta animación se encarga de abrir los cajones del mueble que está a un costado de la cama. Para lograr esta animación se considera la siguiente:

- cajonSup_cuarto02: Variable booleana que indica si el cajón superior del cuarto 02 debe abrirse (true) o cerrarse (false).
- riel_CajonSup: Variable que representa la posición actual del riel del cajón superior. Esta variable se incrementa o decrementa para simular el movimiento de apertura o cierre del cajón.
- APERTURA_CAJON: Constante que representa el valor máximo de apertura del cajón. Cuando riel_CajonSup alcanza este valor, el cajón está completamente abierto.
- incremento_cajon: Valor que indica cuánto se debe incrementar o decrementar riel_CajonSup en cada actualización para simular el movimiento suave del cajón.
- glfwGetTime(): Función que devuelve el tiempo transcurrido desde la última llamada a esta función. Se usa para normalizar el incremento o decremento de riel_CajonSup según el tiempo, asegurando un movimiento suave.
- animacion_02_cuarto02: Variable booleana que indica si la animación de apertura o cierre del cajón está en curso (true) o ha terminado (false).

```
//=====
//Variables para la animación 02 y 03 del cuarto 2 -> Apertura de los cajones
bool animacion_02_cuarto02 = false;
bool animacion_03_cuarto02 = false;
bool cajonSup_cuarto02 = false;
bool cajonInf_cuarto02 = false;
const float APERTURA_CAJON = 1.35;
float riel_CajonSup = 0.0f;
float riel_CajonInf = 0.0f;
float incremento_cajon = 0.0001f;
```

Para activar la animación debemos apretar la tecla 5 o 6 para abrir los cajones (son independientes cada uno).

```

//=====
//Tecla 5 para activar la animación -> Apertura del Cajon Superior del cuarto 02
if (!animacion_02_cuarto02) {
    if (key == GLFW_KEY_5 && action == GLFW_PRESS)
    {
        // Si la puerta está cerrada, abrir
        if (cajonSup_cuarto02) {
            cajonSup_cuarto02 = false; // La puerta se cierra
        }
        else { // Si la puerta está abierta, cerrar
            cajonSup_cuarto02 = true; // La puerta está cerrada nuevamente
        }
        animacion_02_cuarto02 = true;
    }
}
//=====
//Tecla 6 para activar la animación -> Apertura del Cajon Inferior del cuarto 02
if (!animacion_03_cuarto02) {
    if (key == GLFW_KEY_6 && action == GLFW_PRESS)
    {
        // Si la puerta está cerrada, abrir
        if (cajonInf_cuarto02) {
            cajonInf_cuarto02 = false; // La puerta se cierra
        }
        else { // Si la puerta está abierta, cerrar
            cajonInf_cuarto02 = true; // La puerta está cerrada nuevamente
        }
        animacion_03_cuarto02 = true;
    }
}

```

Para lograr esta animación se considera la siguiente tabla de casos:

Caso	Condición	Estado de Variables	Acción	Resultado
1	cajonSup_cuarto02 == true y riel_CajonSup < APERTURA_CAJON	cajonSup_cuarto02 = true riel_CajonSup < APERTURA_CAJON	riel_CajonSup += incremento_cajon * glfwGetTime()	Incrementa riel_CajonSup hasta APERTURA_CAJON
2	cajonSup_cuarto02 == true y riel_CajonSup >= APERTURA_CAJON	cajonSup_cuarto02 = true riel_CajonSup >= APERTURA_CAJON	animacion_02_cuarto02 = false	Detiene la animación
3	cajonSup_cuarto02 == false y riel_CajonSup > 0	cajonSup_cuarto02 = false riel_CajonSup > 0	riel_CajonSup -= incremento_cajon * glfwGetTime()	Decrementa riel_CajonSup hasta 0
4	cajonSup_cuarto02 == false y riel_CajonSup <= 0	cajonSup_cuarto02 = false riel_CajonSup <= 0	animacion_02_cuarto02 = false	Detiene la animación

```

//=====
//Animación para realizar la apertura del cajon superior del cuarto 02
if (cajonSup_cuarto02) {
    if (riel_CajonSup < APERTURA_CAJON) { //Si la apertura es menor al máximo
        riel_CajonSup += incremento_cajon * glfwGetTime(); //incrementamos las bisagras
    }
    else {
        // La animación ha terminado, restablecer la variable
        animacion_02_cuarto02 = false;
    }
}
else {
    if (riel_CajonSup >= 0) {
        riel_CajonSup -= incremento_cajon * glfwGetTime();
    }
    else {
        // La animación ha terminado, restablecer la variable
        animacion_02_cuarto02 = false;
    }
}

```

Para dibujar nuestros cajones usamos una matriz temporal para posicionar nuestra puerta en el lugar deseado.

```
//=====
//Cajon superior del mueble del cuarto 02
model = glm::mat4(1);
model_puerta_cuarto02 = model = glm::translate(model, glm::vec3(30.9f, 18.61f, -16.99f)); //Matriz temporal para el pivote de apertura
model = glm::translate(model_puerta_cuarto02, glm::vec3(-rieL_CajonSup, 0.0f, 0.0f));
glmUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
objeto04_Cajon.Draw(lightingShader);

//=====
//Cajon inferior del mueble del cuarto 02
model = glm::mat4(1);
model_puerta_cuarto02 = model = glm::translate(model, glm::vec3(30.96f, 17.11f, -16.99f)); //Matriz temporal para el pivote de apertura
model = glm::translate(model_puerta_cuarto02, glm::vec3(-rieL_CajonInf, 0.0f, 0.0f));
glmUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
objeto04_Cajon.Draw(lightingShader);
glBindVertexArray(0);
```

Animación Compleja 01

Estas son las animaciones complejas, las que no involucran movimientos lineales. Como primera animación compleja se tiene la llama de una vela, la cual se usar para la chimenea y estufa.

```
//=====
//Variables para la animación compleja 01 -> Movimiento de la flama
bool Compleja_01 = false;
float tiempo = 0.0f;
float transparenciaFlama = 0.0f;
const float MAXIMA_TRANSPARENCIA_VELA = 0.6f;
float velocidad_aparicion = 0.0001f;
//=====
```

Se ocupa una variable de tipo Bool para activar la animación (mandar el reloj normalizado al shader para que comience)

Para aplicar más realismo se usa la variable transparenciaFlama para desaparecer la flama cuando no esté la animación activa y viceversa.

```
animacionCompleja_01.Use();
//tiempo = glfwGetTime(); //normalizar la velocidad
// Get location objects for the matrices on the lamp shader (these could be different on a different shader)
modelLoc = glGetUniformLocation(animacionCompleja_01.Program, "model");
viewLoc = glGetUniformLocation(animacionCompleja_01.Program, "view");
projLoc = glGetUniformLocation(animacionCompleja_01.Program, "projection");
// Set matrices
glUniformMatrix4fv(viewLoc, 1, GL_FALSE, glm::value_ptr(view));
glUniformMatrix4fv(projLoc, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(animacionCompleja_01.Program, "activaTransparencia"), 0.0);
//Vela individual
 glEnable(GL_BLEND); //Activa la funcionalidad para trabajar el canal alfa
 glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
 glUniform4f(glGetUniformLocation(animacionCompleja_01.Program, "colorAlpha"), 1.0f, 1.0f, 0.7f, transparenciaFlama);
model = glm::mat4(1);
glmUniform1f(glGetUniformLocation(animacionCompleja_01.Program, "time"), tiempo);
model = glm::translate(model, glm::vec3(18.96f, 6.05f, -10.89f));
glmUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
objeto04_flama.Draw(animacionCompleja_01);
```

Al usar un nuevo shaders debemos declarar de nuevo las matrices de vista, proyección y modelo. Agregamos también el jugar con la transparencia del objeto esto con ayuda de GL_BLEND que nos ayuda a activar el canal alpha de nuestra textura.

Pasando a la parte más importante que es el shader que ocupamos:

```

animacionCompleja_01.vs  X  animacionSencilla_03.vs  318035327_Personal_GPO13.cpp
1 //Animación Compleja para las velas
2 #version 330 core
3 layout (location = 0) in vec3 aPos;
4 layout (location = 1) in vec3 aNormal;
5 layout (location = 2) in vec2 aTexCoords;
6
7 out vec2 TexCoords;
8
9 uniform mat4 model;
10 uniform mat4 view;
11 uniform mat4 projection;
12 uniform float time;
13
14 const float amplitude_y = 0.01; // Amplitud del movimiento en el eje y
15 const float frequency_y = 10.0; // Frecuencia del movimiento en el eje y
16 const float amplitude_xz = 0.01; // Amplitud del movimiento en los ejes x y z
17 const float frequency_xz = 10.0; // Frecuencia del movimiento en los ejes x y z
18
19 void main()
20 {
21     // Obtener la distancia del vértice al centro del objeto
22     float distance = length(aPos);
23
24     // Deformación del vértice en los ejes x y z (simulando una leve oscilación)
25     float x_offset = amplitude_xz * sin(time * frequency_xz * distance);
26     float z_offset = amplitude_xz * cos(time * frequency_xz * distance);
27
28     // Solo aplicar el shader a los vértices de la mitad del objeto hacia arriba
29     if (aPos.y > 0.0) {
30         // Deformación del vértice en el eje y (simulando el movimiento de una llama)
31         float y_offset = amplitude_y * sin(time * frequency_y * distance);
32
33         // Aplicar la deformación a la posición del vértice
34         vec3 deformed_position = vec3(aPos.x + x_offset, aPos.y + y_offset, aPos.z + z_offset);
35
36         // Calcular la posición final del vértice en el espacio de vista y proyección
37         gl_Position = projection * view * model * vec4(deformed_position, 1.0);
38     } else {
39         // Si el vértice está debajo de la mitad del objeto, mantener su posición original
40         gl_Position = projection * view * model * vec4(aPos, 1.0);
41     }
42     TexCoords = aTexCoords;
43 }
44

```

Primero aplicamos una deformación en el eje x y eje z para simular que hay viento qué mueve la vela. Para evitar que se deforme toda la forma de la flama, pasamos por una condicional para verificar que los vértices que están por encima de la mitad del objeto se deformen, y los que están abajo no lo hagan.

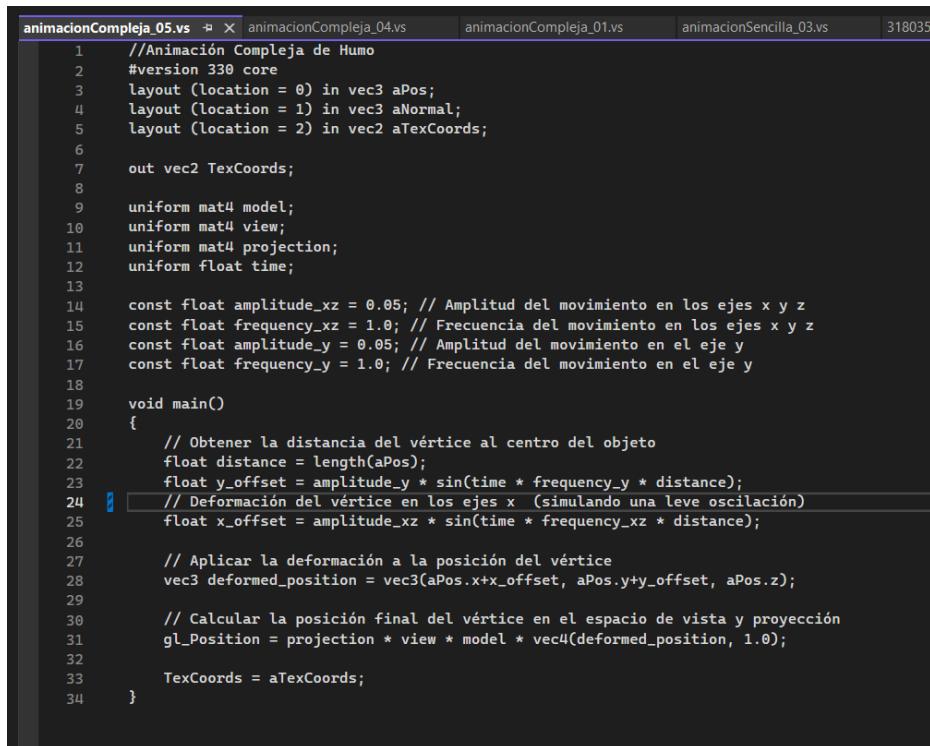
Para lograr esto necesitamos la distancia que hay entre los vértices, esto se logra con ayuda de la variable “distance” la cual se le asigna `length(aPos)`.

Las texturas no se le aplica la misma deformación para que simule el cambio de colores.

Animación Compleja 02

Esta animación involucra 3 animaciones complejas las cuales describiré el shader de cada una (Arriba se describió el shader de la llama el cual se ocupó para el fuego de la estufa y chimenea)

Shader del humo:



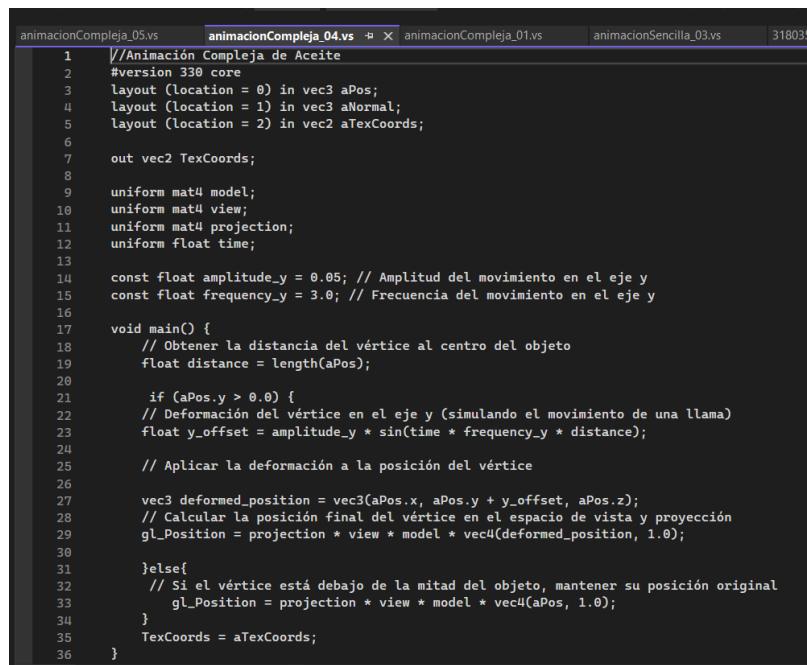
```

animacionCompleja_05.vs # X animacionCompleja_04.vs animacionCompleja_01.vs animacionSencilla_03.vs 318035
1 //Animación Compleja de Humo
2 #version 330 core
3 layout (location = 0) in vec3 aPos;
4 layout (location = 1) in vec3 aNormal;
5 layout (location = 2) in vec2 aTexCoords;
6
7 out vec2 TexCoords;
8
9 uniform mat4 model;
10 uniform mat4 view;
11 uniform mat4 projection;
12 uniform float time;
13
14 const float amplitude_xz = 0.05; // Amplitud del movimiento en los ejes x y z
15 const float frequency_xz = 1.0; // Frecuencia del movimiento en los ejes x y z
16 const float amplitude_y = 0.05; // Amplitud del movimiento en el eje y
17 const float frequency_y = 1.0; // Frecuencia del movimiento en el eje y
18
19 void main()
20 {
21     // Obtener la distancia del vértice al centro del objeto
22     float distance = length(aPos);
23     float y_offset = amplitude_y * sin(time * frequency_y * distance);
24     // Deformación del vértice en los ejes x (simulando una leve oscilación)
25     float x_offset = amplitude_xz * sin(time * frequency_xz * distance);
26
27     // Aplicar la deformación a la posición del vértice
28     vec3 deformed_position = vec3(aPos.x+x_offset, aPos.y+y_offset, aPos.z);
29
30     // Calcular la posición final del vértice en el espacio de vista y proyección
31     gl_Position = projection * view * model * vec4(deformed_position, 1.0);
32
33     TexCoords = aTexCoords;
34 }

```

En este shader solo hacemos una deformación en el eje x y el eje y, para simular el movimiento típico del humo. La textura no sufre la misma deformación

Shader de Aceite:



```

animacionCompleja_05.vs # X animacionCompleja_04.vs animacionCompleja_01.vs animacionSencilla_03.vs 318035
1 //Animación Compleja de Aceite
2 #version 330 core
3 layout (location = 0) in vec3 aPos;
4 layout (location = 1) in vec3 aNormal;
5 layout (location = 2) in vec2 aTexCoords;
6
7 out vec2 TexCoords;
8
9 uniform mat4 model;
10 uniform mat4 view;
11 uniform mat4 projection;
12 uniform float time;
13
14 const float amplitude_y = 0.05; // Amplitud del movimiento en el eje y
15 const float frequency_y = 3.0; // Frecuencia del movimiento en el eje y
16
17 void main()
18 {
19     // Obtener la distancia del vértice al centro del objeto
20     float distance = length(aPos);
21
22     if (aPos.y > 0.0) {
23         // Deformación del vértice en el eje y (simulando el movimiento de una llama)
24         float y_offset = amplitude_y * sin(time * frequency_y * distance);
25
26         // Aplicar la deformación a la posición del vértice
27
28         vec3 deformed_position = vec3(aPos.x, aPos.y + y_offset, aPos.z);
29         // Calcular la posición final del vértice en el espacio de vista y proyección
30         gl_Position = projection * view * model * vec4(deformed_position, 1.0);
31
32     } else{
33         // Si el vértice está debajo de la mitad del objeto, mantener su posición original
34         gl_Position = projection * view * model * vec4(aPos, 1.0);
35     }
36     TexCoords = aTexCoords;
37 }

```

Al igual que en el shader del fuego, esta solo se aplica para la mitad de los vértices que están por encima de la mitad. Esto para que no se deformé el objeto completamente.

Pasamos a la parte más interesante porque al igual que el fuego, jugamos con la transparencia gracias a GL_BLEND. Cuando se aprieta la tecla M inicia la animación con ayuda de una variable de tipo bool:

```
//=====
//Tecla M para activar la animación Compleja_03
if (key == GLFW_KEY_M && action == GLFW_PRESS) {
    Compleja_03 = !Compleja_03;
}
```

Posteriormente cuando esa variable de tipo bool (Compleja_03) es true pasamos a la siguiente parte:

```
//=====
//Animación Compleja_03 -> Flama de la Estufa, aceite y humo
if (Compleja_03) {
    tiempo3 = glfwGetTime();
    if (transparenciaHumo <= MAXIMA_TRANSparencia) {
        transparenciaHumo += velocidad_humo * glfwGetTime();
    }

    if (transparenciaEstufa <= MAXIMA_TRANSparencia_ESTUFA) {
        transparenciaEstufa += velocidad_aparicion * glfwGetTime();
    }
    Light2 = glm::vec3(1.0f, 0.2715f, 0.0f);
}
else {
    if (transparenciaHumo >= 0.0f) {
        transparenciaHumo -= velocidad_humo * glfwGetTime();
    }

    if (transparenciaEstufa >= 0.0f) {
        transparenciaEstufa -= velocidad_aparicion * glfwGetTime();
    }
    tiempo3 = 0.0f;
    Light2 = glm::vec3(0); //Cuando es solo un valor en los 3 vectores pueden dejar solo una componente
}
```

Esta animación se hizo considerando la siguiente tabla de casos:

Compleja_03	transparenciaHumo	transparenciaEstufa	Acción	Light2
Verdadero	Menor o igual a máximo	Menor o igual a máximo	Incrementar transparenciaHumo y transparenciaEstufa	(1.0, 0.2715, 0.0)
Falso	Mayor o igual a 0	Mayor o igual a 0	Decrementar transparenciaHumo y transparenciaEstufa	(0)

Para poder dibujar nuestros objetos necesitamos posicionarlos en el lugar donde queremos por lo que queda de la siguiente forma:

```
//Aceite de la estufa
animacionCompleja_04.Use();
modelLoc = glGetUniformLocation(animacionCompleja_04.Program, "model");
viewLoc = glGetUniformLocation(animacionCompleja_04.Program, "view");
projLoc = glGetUniformLocation(animacionCompleja_04.Program, "projection");
// Set matrices
glUniformMatrix4fv(viewLoc, 1, GL_FALSE, glm::value_ptr(view));
glUniformMatrix4fv(projLoc, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(animacionCompleja_04.Program, "activaTransparencia"), 0.0);
glUniform4f(glGetUniformLocation(animacionCompleja_04.Program, "colorAlpha"), 1.0f, 1.0f, 0.0f, 0.10f);
model = glm::mat4(1);
glUniform1f(glGetUniformLocation(animacionCompleja_04.Program, "time"), tiempo3);

model = glm::translate(model, glm::vec3(25.15f, 3.58f, -1.88f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
objeto07_aceite.Draw(animacionCompleja_04);
 glBindVertexArray(0);

//=====================================================================
//Humo de la estufa
animacionCompleja_05.Use();
modelLoc = glGetUniformLocation(animacionCompleja_05.Program, "model");
viewLoc = glGetUniformLocation(animacionCompleja_05.Program, "view");
projLoc = glGetUniformLocation(animacionCompleja_05.Program, "projection");
// Set matrices
glUniformMatrix4fv(viewLoc, 1, GL_FALSE, glm::value_ptr(view));
glUniformMatrix4fv(projLoc, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(animacionCompleja_05.Program, "activaTransparencia"), 0.0);
glUniform4f(glGetUniformLocation(animacionCompleja_05.Program, "colorAlpha"), 1.0f, 1.0f, 1.0f, transparenciaHumo);
model = glm::mat4(1);
glUniform1f(glGetUniformLocation(animacionCompleja_05.Program, "time"), tiempo3);

model = glm::translate(model, glm::vec3(25.19f, 4.72f, -2.01f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
objeto07_humo.Draw(animacionCompleja_05);
```

SkyBox

Para hacer nuestro skybox, usaremos un cubo gigante el cual se hará desde OpenGL, o sea que no ocuparemos ningún obj para este.

Se define un array `skyboxVertices` que contiene las coordenadas (x, y, z) de los vértices del cubo que representa el skybox. Cada conjunto de tres valores representa las coordenadas de un vértice:

```
GLfloat skyboxVertices[] = {
    // Positions
    -1.0f,  1.0f,  -1.0f,
    -1.0f,  -1.0f,  -1.0f,
    1.0f,  -1.0f,  -1.0f,
    1.0f,  -1.0f,  -1.0f,
    1.0f,  1.0f,  -1.0f,
    -1.0f,  1.0f,  -1.0f,
    -1.0f,  -1.0f,  1.0f,
    -1.0f,  -1.0f,  -1.0f,
    -1.0f,  1.0f,  -1.0f,
    -1.0f,  1.0f,  1.0f,
    -1.0f,  -1.0f,  1.0f,
    -1.0f,  -1.0f,  1.0f,
    1.0f,  -1.0f,  1.0f,
    1.0f,  1.0f,  1.0f,
    1.0f,  1.0f,  -1.0f,
    1.0f,  -1.0f,  -1.0f,
    1.0f,  -1.0f,  -1.0f,
    1.0f,  -1.0f,  1.0f,
    1.0f,  1.0f,  -1.0f,
    1.0f,  1.0f,  1.0f,
};
```

Posteriormente:

Se generan los identificadores para el array de vértices y el buffer de vértices utilizando las funciones glGenVertexArrays y glGenBuffers. Estos objetos son necesarios para almacenar y administrar los datos de los vértices.

Se define un vector faces que contiene las rutas de las texturas que se usarán para el skybox en cada una de las seis caras (derecha, izquierda, arriba, abajo, atrás, adelante). Luego, se carga el cubemap (textura de mapeo de cubo) utilizando la función TextureLoading: LoadCubemap, que carga y configura las texturas según las rutas especificadas en el vector faces.

```

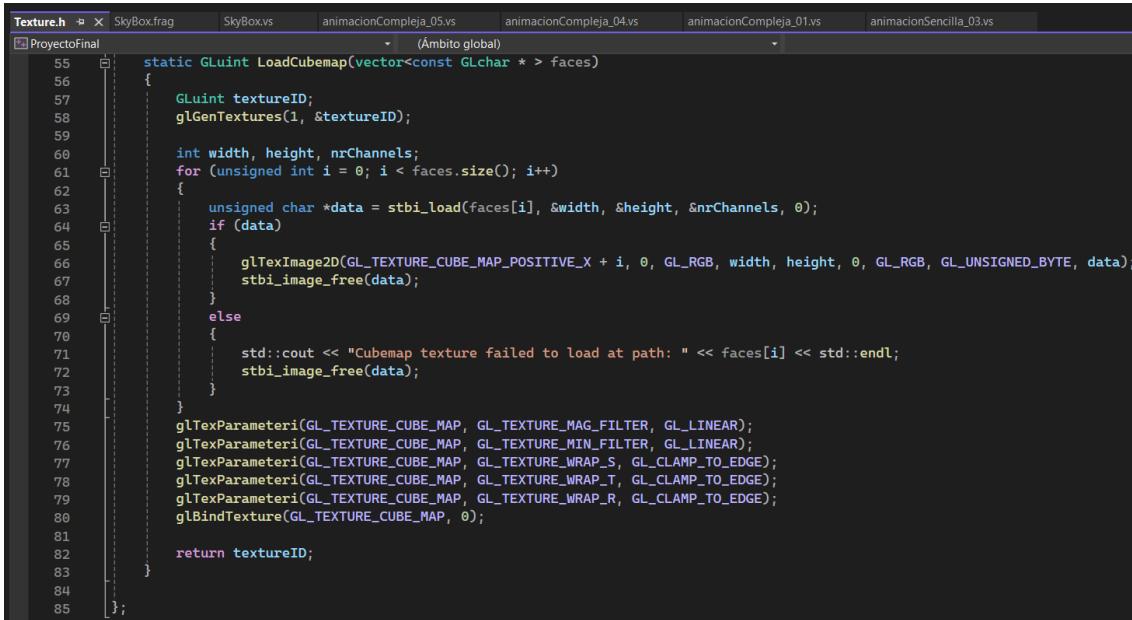
//SkyBox
GLuint skyboxVBO, skyboxVAO;
glGenVertexArrays(1, &skyboxVAO);
glGenBuffers(1, &skyboxVBO);
 glBindVertexArray(skyboxVAO);
 glBindBuffer(GL_ARRAY_BUFFER, skyboxVBO);
glBufferData(GL_ARRAY_BUFFER, sizeof(skyboxVertices), &skyboxVertices, GL_STATIC_DRAW);
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(GLfloat), (GLvoid*)0);

// Load textures del skybox
vector<const GLchar*> faces;
faces.push_back("Skybox/right.tga");
faces.push_back("Skybox/left.tga");
faces.push_back("Skybox/top.tga");
faces.push_back("Skybox/bottom.tga");
faces.push_back("Skybox/back.tga");
faces.push_back("Skybox/front.tga");

GLuint cubemapTexture = TextureLoading::LoadCubemap(faces);

```

Para que el skybox funcione correctamente se deben modificar las caras del cubo con las normales apuntando hacia adentro y se debe generar un efecto de parallax dentro de este cubo y ese efecto lo metemos en el uso de la textura. Aquí se ocupan texturas en formato TGA. Esta modificación se realiza en el archivo Texture.h



```

Texture.h
static GLuint LoadCubemap(vector<const GLchar * > faces)
{
    GLuint textureID;
    glGenTextures(1, &textureID);

    int width, height, nrChannels;
    for (unsigned int i = 0; i < faces.size(); i++)
    {
        unsigned char *data = stbi_load(faces[i], &width, &height, &nrChannels, 0);
        if (data)
        {
            glBindTexture(GL_TEXTURE_CUBE_MAP, i, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
            stbi_image_free(data);
        }
        else
        {
            std::cout << "Cubemap texture failed to load at path: " << faces[i] << std::endl;
            stbi_image_free(data);
        }
    }
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_R, GL_CLAMP_TO_EDGE);
    glBindTexture(GL_TEXTURE_CUBE_MAP, 0);
}

return textureID;
}

```

Finalmente, para mandar a dibujar nuestro cubo usamos glDrawArrays

```

//=====
// Draw skybox as last
glDepthFunc(GL_LEQUAL); // Change depth function so depth test passes when values are equal to depth buffer's content
SkyBoxshader.Use();
view = glm::mat4(glm::mat3(camera.GetViewMatrix())); // Remove any translation component of the view matrix
glm::uniformMatrix4fv(glGetUniformLocation(SkyBoxshader.Program, "view"), 1, GL_FALSE, glm::value_ptr(view));
glm::uniformMatrix4fv(glGetUniformLocation(SkyBoxshader.Program, "projection"), 1, GL_FALSE, glm::value_ptr(projection));

// skybox cube
 glBindVertexArray(skyboxVAO);
 glActiveTexture(GL_TEXTURE1);
 glBindTexture(GL_TEXTURE_CUBE_MAP, cubemapTexture);
 glDrawArrays(GL_TRIANGLES, 0, 36);
 glBindVertexArray(0);
glDepthFunc(GL_LESS); // Set depth function back to default

```

Audio con irrKlang

Para colocar audio el proyecto existe varias bibliotecas, pero por facilidad se usó la biblioteca irrKlang

Para hacer su uso correcto se realiza lo siguiente:

```

3     #include <irrklang.h>
4     using namespace irrklang;
5     #pragma comment(lib, "irrklang.lib")
6     // GLEW

```

Donde:

- **using namespace irrklang;**: Esta línea indica al compilador que el espacio de nombres irrKlang se utilizará en este archivo de código. Esto significa que no es necesario especificar irrKlang: antes de cada uso de una clase o función de irrKlang.
- **#pragma comment (lib, "irrklang.lib")**: Esta línea es específica de entornos de desarrollo basados en Windows. Le indica al compilador enlazador que debe incluir la biblioteca de enlace irrKlang.lib al compilar el código. Esta biblioteca contiene la implementación de irrKlang que el programa necesita para funcionar correctamente en tiempo de ejecución.

Dentro de nuestro Main:

```

146 int main()
147 {
148     // start the sound engine with default parameters
149     ISoundEngine* engine = createIrrKlangDevice();
150     if (!engine)
151         return 0; // error starting up the engine
152
153     // play some sound stream, looped
154     engine->play2D("Music/musicaAmbiental.mp3", true);
155
156

```

- **ISoundEngine* engine = createIrrKlangDevice ()**: Esta línea crea una instancia del motor de sonido irrKlang utilizando la función createIrrKlangDevice (). Esta función inicializa el motor de sonido con los parámetros predeterminados y devuelve un puntero a un objeto ISoundEngine, que es la interfaz principal para interactuar con el motor de sonido irrKlang.
- **if (! engine) return 0**: Después de crear el motor de sonido, esta línea comprueba si la inicialización fue exitosa. Si el puntero engine es nulo, significa que ocurrió un error al inicializar el motor de sonido, por lo que la aplicación sale del programa con un código de error (0 en este caso).
- **engine->play2D ("Music/musicaAmbiental.mp3", true)**: Una vez que el motor de sonido se ha inicializado correctamente, esta línea reproduce un archivo de audio en formato MP3

Análisis de Costos

El siguiente apartado tiene como finalidad un análisis de los costos previstos por el autor para la venta de este proyecto. Se plantean los siguientes puntos:

- Como autor original de la obra el número de horas empleadas para el proyecto se ven reflejadas en el diagrama de Gantt.
- El costo por día definido con base en el salario mínimo es de: \$248.93 MXN
- Costo por soporte incluido en el proyecto: \$450.00 MXN

En la siguiente tabla se hace un desglose del total de gastos del autor para su elaboración.

Autor	Horas Empleadas	Sueldo x día	Días de trabajo	Total
Jimenez Cervantes Angel Mauricio	435	248.93	18.12	\$ 4 510.61 MXN

Costos operativos

Durante el proyecto se requirió el uso de cómputo especializado, de la cual la renta del equipo es de \$4200 MXN pesos al mes para una empresa dedicada para este tipo de situaciones.

También se contempla el uso de luz al mes para una sola persona de alrededor de \$150 MXN y del uso del internet de alrededor de \$300 MXN.

Con ello la suma de lo previsto en la tabla anterior más lo contemplado en los costos operativos sería de:

$$\text{Total} = \$4\,510.61 + \$4\,650.00 = \$9,161.00 \text{ MXN redondeando.}$$

Si se realiza la compra del equipo de computo especializado, el precio es de \$9799.00 MXN por lo que el costo total sería de:

$$\text{Total} = \$4\,510.61 + \$10249.00 = \$14,760.00 \text{ MXN redondeando.}$$

Conclusiones

Este proyecto demuestra una sólida aplicación de los conocimientos adquiridos durante el curso, destacando la capacidad para recrear entornos tridimensionales utilizando OpenGL. Los objetivos específicos incluyen la selección y recreación de una fachada y un espacio, así como la captura de la esencia del ambiente de las imágenes de referencia. La entrega bilingüe y el uso de un repositorio oficial en GitHub cumplen con los requisitos establecidos.

Las limitaciones individuales y los criterios de evaluación subrayan la importancia de la calidad y la originalidad del proyecto, así como la necesidad de cumplir con los estándares establecidos. Las restricciones creativas, como la prohibición de

ciertas temáticas y la exigencia de calidad gráfica, garantizan la integridad del proyecto y fomentan la creatividad dentro de los límites establecidos.

El proyecto hace un uso efectivo de diversas herramientas y recursos, desde software de modelado como Maya y Paint 3D hasta bibliotecas como irrKlang para la gestión del sonido. La metodología de software, incluyendo el uso de Kanban para la gestión de proyectos, contribuye a una ejecución eficiente y organizada del proyecto.

La documentación detallada del código, incluyendo la explicación de las animaciones y las bibliotecas utilizadas, proporciona una comprensión clara de los procesos y decisiones tomadas durante el desarrollo del proyecto. En resumen, este proyecto exhibe un enfoque metódico, creativo y técnicamente competente para la recreación de entornos tridimensionales utilizando OpenGL.

Anexos

Para mayor detalle de las bibliotecas usadas puedes acceder a los siguientes enlaces:

- Documentación de OpenGL versión 3.30: [GLSLangSpec.3.30.pdf \(kronos.org\)](#)
- Documentación de GLM: [0.9.8: OpenGL Mathematics \(g-truc.net\)](#)
- Documentación SOIL2: [SpartanJ/SOIL2: SOIL2 is a tiny C library used primarily for uploading textures into OpenGL. \(github.com\)](#)
- Documentación de irrKlang: [irrKlang: irrKlang 1.6.0 API documentation \(ambiera.com\)](#)
- Documentación stb_image: [nothings/stb: stb single-file public domain libraries for C/C++ \(github.com\)](#)

- Documentación assimp: [The Asset-Importer-Lib Documentation — Asset-Importer-Lib March 2022 v5.2.3 documentation \(assimp-docs.readthedocs.io\)](https://assimp.readthedocs.io/en/v5.2.3/documentation.html)

Para acceder al código o los modelos puedes consultar los siguientes repositorios.

Repositorio Implementación	Repositorio Modelos
