PROYECTO
CASA VACACIONAL CUAUTLA

Profesor: Ing. Carlos Aldair Román Balbuena

317032392 Martínez Martínez Francisco David

Grupo: 5
Clave: 1590

DESCRIPCIÓN DEL PROYECTO Y MANUAL TÉCNICO

Índice

Manual Técnico (español)

URL: https://github.com/FrankDavidMM/317032392_PROYECTOFINAL2023-2_GPO06.git

1. Introducción

1.1. Descripción del Proyecto

El proyecto consiste en desarrollar un recorrido virtual en OpenGL 3. Donde se deberá crear dos cuartos y una fachada, siguiendo imágenes de referencia. Incluye una cámara sintética y 4 animaciones relacionadas con el recorrido. Documenta el proyecto con diagramas, manuales y un análisis de costos. La entrega debe ser en formato digital con un archivo ejecutable. Asegúrate de que el espacio sea realista. Utiliza un repositorio de GitHub. El proyecto es individual y se penalizará la falta de originalidad.

1.2 Objetivo

El objetivo del proyecto es crear un recorrido interactivo y realista, donde el usuario pueda explorar dos cuartos y una fachada diseñados con un alto nivel de detalle donde pueda tener una experiencia inmersiva y de calidad al usuario. El objetivo incluye la integración de una cámara sintética, animaciones contextualizadas, documentación completa y un análisis de costos del proyecto.

1.3 Alcance del Proyecto

El objetivo del proyecto "Recorrido Virtual en OpenGL 3" es crear una experiencia inmersiva y realista mediante el desarrollo de un recorrido virtual interactivo. Los elementos clave del objetivo son los siguientes:

1. Diseño de ambientes realistas: El objetivo es recrear dos cuartos y una fachada basados en imágenes de referencia, utilizando un estilo artístico acorde con el espacio representado. Se busca lograr un alto nivel de realismo y detalle en la ambientación de los espacios.
2. Interacción fluida y exploración: Se pretende integrar una cámara sintética que permita al usuario explorar y navegar de manera fluida por los diferentes ambientes. El objetivo es brindar una experiencia de recorrido virtual inmersiva y cómoda.
3. Animaciones contextualizadas: Se busca enriquecer la experiencia del recorrido virtual mediante la inclusión de cuatro animaciones relacionadas con el contexto del espacio representado. Estas animaciones agregarán dinamismo y vida a los ambientes, generando un mayor impacto visual.
4. Documentación completa y de calidad: El objetivo es elaborar una documentación exhaustiva que incluya un diagrama de flujo del recorrido, un diagrama de Gantt para la planificación, una metodología de desarrollo utilizada, conclusiones, un manual de usuario y un manual técnico. La

documentación se presentará en español e inglés, y se prestará especial atención a la redacción y ortografía.

5. Análisis de costos y precios: Se llevará a cabo un análisis de costos del proyecto, considerando los gastos incurridos durante su desarrollo. Se buscará justificar el precio de venta propuesto mediante una argumentación sólida basada en los costos y el valor agregado del recorrido virtual.

El objetivo general del proyecto es lograr un recorrido virtual en OpenGL 3 que combine un diseño visualmente atractivo y realista con una experiencia interactiva y envolvente para el usuario. La documentación completa y el análisis de costos garantizarán una entrega integral del proyecto.

1.4 Metodología Scrum

1.4.1 Principios

La metodología Scrum se basa en los siguientes principios:

1. Transparencia: Toda la información relevante sobre el proyecto y el proceso de desarrollo debe ser compartida y accesible para todos los miembros del equipo y los stakeholders. Esto promueve la confianza y la colaboración.
2. Inspección: Se deben realizar inspecciones frecuentes y regulares de los artefactos, el progreso y el proceso de trabajo. Esto permite detectar posibles problemas o desviaciones y tomar medidas correctivas de manera oportuna.
3. Adaptación: Si se detectan desviaciones o se requieren cambios, se deben realizar ajustes en el plan y en el proceso de trabajo. Scrum fomenta la adaptabilidad para responder a los cambios y mantener el enfoque en los objetivos del proyecto.
4. Empoderamiento del equipo: El equipo de desarrollo tiene la autonomía y la responsabilidad de tomar decisiones relacionadas con el trabajo. Esto incluye la estimación de esfuerzo, la planificación del trabajo y la autoorganización para lograr los objetivos establecidos.
5. Entrega incremental: El producto se desarrolla en incrementos funcionales y entregables en cada sprint. Esto permite obtener retroalimentación temprana de los stakeholders y garantiza que el producto sea evolutivo y se ajuste a las necesidades cambiantes.
6. Colaboración: Scrum fomenta la colaboración estrecha y constante entre todos los miembros del equipo y los stakeholders. La comunicación efectiva y el trabajo conjunto son fundamentales para el éxito del proyecto.
7. Enfoque en el valor del negocio: El equipo se enfoca en entregar valor tangible y real al cliente o al usuario final. Se priorizan las funcionalidades y características que brinden el máximo beneficio y se minimiza el trabajo que no agrega valor.
8. Tiempo de entrega rápido: Los sprints son períodos cortos y definidos en los que se completan tareas y se entregan incrementos de trabajo. Esto permite obtener resultados rápidos y obtener retroalimentación temprana, lo que ayuda a guiar el desarrollo.

1.4.2 Implementación para el Proyecto

En base a los principios expuestos en el apartado 1.4.1, se puede afirmar que la metodología Scrum es altamente adecuada para el desarrollo de este proyecto, debido a su notable flexibilidad y a su enfoque en el ciclo de vida del proyecto. En este contexto, se puede subdividir el ciclo de vida del proyecto en sprints, los cuales representan etapas que permiten abordar las funcionalidades y necesidades establecidas por el Product Owner para satisfacer al cliente.

Es importante resaltar que los sprints se caracterizan por su capacidad de adaptación, lo que significa que el equipo de desarrollo tiene la flexibilidad necesaria para ajustar su enfoque y responder a las especificaciones definidas por el Product Owner. Asimismo, el objetivo fundamental de cada sprint es satisfacer las necesidades y requisitos establecidos por el cliente, a través de la orientación y dirección proporcionadas por el Product Owner.

En conclusión, la metodología Scrum se posiciona como una elección óptima para este proyecto, debido a su capacidad de respuesta a los cambios, su enfoque en la entrega de valor en incrementos y su alineación con las expectativas y requerimientos del cliente, a través del papel fundamental desempeñado por el Product Owner.

1.5 Diagrama de Gantt

Fecha de Inicio:	27/04/2023							
Fecha de Entrega	08/06/2023							
Sprints	Semanas	1	2	3	4	5	6	7
	Horas							
PLANIFICACIÓN	6 hrs							
Eleccion de Fachada	1 hrs							
Eleccion de Obj´s	1 hrs							
Herramientas	4 hrs							
MODELADO	44 hrs							
Objetos	26 hrs							
Texturizacion	8 hrs							
Fachada	10 hrs							
IMPLEMENTACIÓN	10 hrs							
Programación	15 hrs							
RELESE	2 hrs							
TOTAL	67 hrs							

1.6 Tabla de costos

Tipo	Cantidad	Costo Unitario	Descripción	Subtotal
*	1	$10,000.00	Equipo de Trab	$10,000.00
*	1	$2,450.00	Licencia 1/Año	$2,450.00
Objetos	12	$216.67	Diseño de Obje	$2,600.04
Texturass	12	$67.66	Diseño de Text	$811.92
Fachadas	1	$1,000.00	Diseño de Fach	$1,000.00
Horas	15	$150.00	Programación	$2,250.00
			Costo Total:	$19,111.96

2. Desarrollo

2.1 Primer Sprint (Planificación)

2.1.1 Sprint BackLog

1. Definir las herramientas y el entorno de desarrollo:
   - Investigar y seleccionar las herramientas adecuadas para el desarrollo en OpenGL, incluyendo librerías, frameworks o IDEs necesarios.
   - Configurar el entorno de desarrollo en base a las herramientas elegidas.
   - Establecer el uso del lenguaje de programación C++ como base para la implementación del proyecto.
2. Establecer el ambiente a representar:
   - Analizar y comprender los requisitos del cliente en relación al ambiente a representar, incluyendo la fachada y los cuartos.
   - Realizar una investigación sobre elementos arquitectónicos y de diseño necesarios para recrear el ambiente deseado.
   - Definir el alcance y los elementos específicos a incluir en la representación, considerando las necesidades del cliente.
3. Integrar OpenGL y aplicar transformaciones básicas:
   - Familiarizarse con el uso de OpenGL y comprender su funcionamiento básico.
   - Implementar las transformaciones básicas, como traslación, rotación y escalado, utilizando las capacidades de OpenGL.
   - Verificar que las transformaciones se apliquen correctamente a los elementos del entorno, permitiendo una representación precisa.
4. Trabajar con modelos .obj:
   - Investigar y seleccionar una librería o método adecuado para cargar y renderizar modelos en formato .obj.
   - Implementar la carga de modelos .obj en el entorno de desarrollo y asegurarse de que se puedan visualizar correctamente en la escena.
   - Realizar pruebas y ajustes para garantizar una carga y renderizado adecuados de los modelos .obj en el proyecto.
5. Implementar iluminación:
   - Estudiar y aplicar técnicas de iluminación en OpenGL, como iluminación ambiente, difusa y especular.
   - Integrar la iluminación en el entorno de desarrollo y asegurarse de que los elementos del ambiente se vean correctamente iluminados.

- Realizar ajustes y mejoras en la iluminación para lograr una apariencia realista y coherente en la representación del ambiente.

2.1.2. Elección de Fachada y Cuarto a recrear:

Durante el proceso de selección de la fachada para este proyecto, se siguieron criterios específicos con el objetivo de crear un ambiente acorde al entorno vacacional deseado. Afortunadamente, el cliente brindó imágenes de referencia que resultaron de gran utilidad para continuar con el desarrollo del proyecto.

La fachada en sí misma será una construcción de dos pisos, con cuatro paredes que conformarán una estructura sólida y estéticamente agradable. El enfoque principal estará en el primer piso, donde se ubicarán espacios clave como la sala, la cocina y una alacena.

Imagen 2.1.2.1

Imagen 2.1.2.2

Imagen 2.1.2.3

Imagen 2.1.2.4

La sala es un espacio donde se espera que los visitantes puedan disfrutar de momentos de relajación y confort. Se buscará recrear un ambiente acogedor, con elementos decorativos y muebles que transmitan calidez y tranquilidad. El objetivo es que los usuarios de la aplicación puedan experimentar la sensación

de estar en una sala espaciosa y acogedora, ideal para compartir momentos con amigos y familiares.

Imagen 2.1.2.5

Por otro lado, la cocina es un espacio fundamental en cualquier fachada. Se buscará representar una cocina funcional y moderna, con muebles y electrodomésticos que reflejen un estilo contemporáneo. La idea es transmitir la sensación de un espacio organizado y bien equipado, donde los usuarios puedan imaginarse preparando deliciosas comidas y compartiendo momentos culinarios.

Imagen 2.1.2.6

Cabe destacar que la selección de estos dos cuartos se realizó con base en su relevancia y protagonismo en la fachada. La sala y la cocina son espacios que captan la atención y definen la imagen general del ambiente. Al recrear estos

espacios de manera detallada y realista, se busca lograr una representación visual impactante y atractiva para los usuarios.

2.1.3. Objetos Para Modelar

Primer Cuarto (Sala):

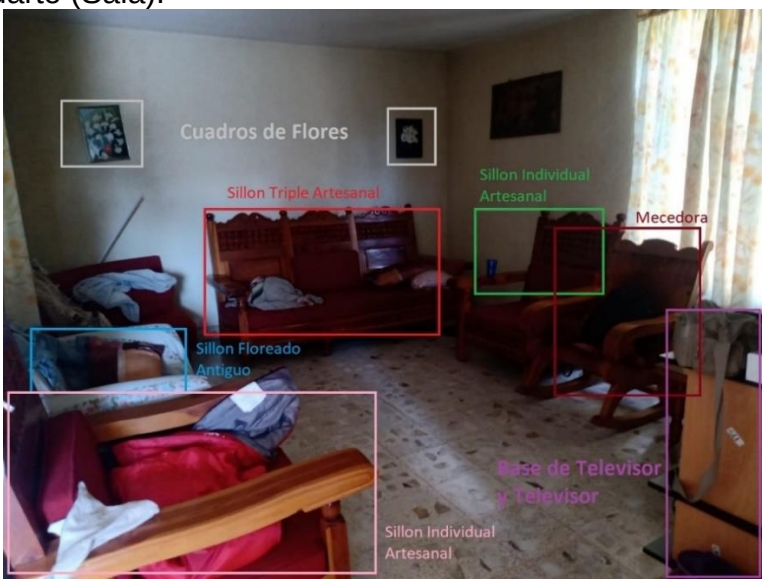


Imagen 2.1.3.1

- Sillón Floreado Antiguo
- 2 sillones Artesanales Individuales
- 1 sillón Grupal Artesanal
- 1 mecedora Artesanal
- Mueble para Televisor
- Televisor
- 2 cuadros de Flores

Segundo Cuarto (Cocina):

- Refrigerador Grande
- Microondas
- Estufa
- Mesa
- Alacena

2.1.4. Herramientas para utilizar

2.1.4.1 Maya

Para el diseño de los objetos se decidió utilizar Maya, ya que esta ofrece una amplia gama de herramientas y técnicas para el modelado 3D, lo que te permite crear una variedad de formas y objetos con precisión y detalle. Puedes utilizar herramientas de modelado poligonal para construir y editar geometrías mediante la manipulación de vértices, aristas y caras. Además, Maya también cuenta con herramientas de modelado basado en subdivisiones, que permiten crear superficies suaves y orgánicas mediante la subdivisión de polígonos.

El modelado procedimental es otra poderosa característica de Maya, que te permite crear geometrías complejas mediante la aplicación de algoritmos y reglas específicas. Esto ofrece un enfoque no destructivo y altamente editable para el modelado, lo que te brinda mayor flexibilidad y control sobre tus diseños.

Mapas UV en Maya:

Los mapas UV son fundamentales para aplicar texturas y materiales a modelos 3D de manera precisa. Maya proporciona un conjunto robusto de herramientas para crear y editar mapas UV. Puedes desplegar las superficies de tus modelos en un espacio 2D, donde puedes ajustar y manipular los puntos UV para asegurarte de que las texturas se apliquen correctamente y sin distorsiones.

Maya ofrece diferentes métodos para generar mapas UV, como proyecciones automáticas, desplegados por pelt y herramientas de corte y costura. Además, puedes utilizar herramientas de edición de UV para suavizar transiciones, ajustar la escala de los mapas UV y realizar otras modificaciones precisas.

2.1.4.2 GIMP

GIMP, un software de edición de imágenes gratuito y de código abierto, puede ser una herramienta valiosa para trabajar con las texturas y los mapas UV en el contexto de Maya. GIMP ofrece una amplia gama de funciones para ajustar y editar texturas, como recortar, redimensionar, corregir colores, aplicar filtros y mucho más. Esto permite adaptar las texturas a las necesidades específicas de cada modelo y optimizar su apariencia.

Además, GIMP puede utilizarse para retocar y corregir los mapas UV generados en Maya. Por ejemplo, si hay áreas con distorsiones o solapamientos en el mapeo UV, se pueden utilizar las herramientas de GIMP para realizar ajustes y

optimizaciones manuales. También se pueden utilizar técnicas de pintura y retoque en GIMP para agregar detalles o ajustar los mapas UV según sea necesario.

2.1.4.3 OpenGL

OpenGL es una API gráfica de bajo nivel que proporciona una interfaz para interactuar con el hardware de gráficos. Proporciona funciones básicas para la renderización de gráficos en 2D y 3D, pero no incluye las extensiones y características más nuevas y específicas del hardware que se desarrollan continuamente.

Aquí es donde entran en juego bibliotecas como GLEW y GLFW. Estas bibliotecas se crearon para facilitar el acceso a las extensiones de OpenGL y proporcionar funcionalidades adicionales que no están presentes en la API básica de OpenGL.

GLEW se encarga de la carga y gestión de las extensiones de OpenGL. Cuando se utilizan las funciones estándar de OpenGL, se tiene acceso a un conjunto básico de características. Sin embargo, los fabricantes de hardware y desarrolladores pueden agregar extensiones a OpenGL para ofrecer funcionalidades adicionales y optimizaciones específicas del hardware. GLEW permite cargar y utilizar estas extensiones de manera sencilla, sin tener que preocuparse por la gestión y compatibilidad de las diferentes extensiones en diferentes plataformas.

GLFW, por su parte, se enfoca en la creación de ventanas y la gestión de eventos relacionados con la entrada del usuario. Aunque OpenGL ofrece cierta funcionalidad para la creación de ventanas, es limitada y varía según la plataforma. GLFW proporciona una API más consistente y fácil de usar para la creación y gestión de ventanas en múltiples plataformas. También se encarga de manejar eventos de entrada, como el teclado y el mouse, de manera eficiente y consistente, facilitando así el desarrollo de aplicaciones gráficas interactivas.

2.1.4.4 Visual Studio

La elección de utilizar Visual Studio como la IDE para este proyecto se basó en varias razones. En primer lugar, Visual Studio es ampliamente reconocido y utilizado en la industria del desarrollo de software. Es una herramienta confiable y estable que ha demostrado su eficacia en numerosos proyectos.

Visual Studio ofrece un conjunto completo de características y herramientas que facilitan el desarrollo de aplicaciones. Su editor de código es potente y altamente personalizable, lo que permite a los desarrolladores ajustar el entorno según sus preferencias y necesidades específicas. Además, proporciona capacidades de depuración integradas, lo que simplifica el proceso de identificación y solución de problemas en el código.

2.1.4.5 Lenguaje C++

para el desarrollo del proyecto se decidió por el lenguaje c++ esto por varias razones. En primer lugar, el lenguaje C++ es conocido por su eficiencia y rendimiento, lo que resulta especialmente importante en proyectos que requieren un procesamiento rápido y una optimización del uso de recursos. Dado que nuestro proyecto involucra el modelado y renderizado de objetos 3D, era esencial contar con un lenguaje que nos permitiera trabajar de manera eficiente con grandes cantidades de datos y realizar cálculos complejos.

Además, C++ es un lenguaje ampliamente utilizado en la industria del desarrollo de software, especialmente en el ámbito de los gráficos y la computación visual. Esto significa que hay una gran cantidad de recursos, bibliotecas y herramientas disponibles que facilitan el desarrollo de aplicaciones 3D. Al optar por C++, pudimos aprovechar estas ventajas y acceder a una amplia gama de funciones y capacidades para impulsar nuestro proyecto.

2.1.4.6 Complementarias

NormalMap Online es una herramienta en línea que se utiliza para generar mapas de normales a partir de texturas existentes. Los mapas de normales son texturas especiales que se utilizan para agregar detalles de relieve a modelos 3D sin aumentar significativamente la complejidad de la geometría. Al utilizar NormalMap Online, pudimos crear mapas de normales para nuestros objetos, lo que mejoró su apariencia visual al agregar pequeños detalles y texturas en relieve. Estos mapas de normales fueron luego aplicados a los modelos en nuestro proyecto, lo que resultó en una mayor calidad y realismo en la representación de los objetos.

TurboSquid es una plataforma en línea que ofrece una amplia biblioteca de modelos 3D listos para usar. Esta plataforma nos permitió acceder a una variedad de modelos gratuitos que se ajustaban a nuestras necesidades en el proyecto. Al utilizar los modelos descargados de TurboSquid, pudimos ahorrar tiempo y esfuerzo en la creación de modelos desde cero, especialmente para objetos más simples y comunes, como muebles o electrodomésticos. Estos modelos descargados fueron luego incorporados en nuestro proyecto, lo que nos permitió enfocarnos en aspectos más específicos y personalizados del desarrollo.

2.2 Segundo Sprint (Modelado)

2.2.1. Sprint BackLog

1. Modelar "Sillón Floreado Antiguo" basado en imagen de referencia, asegurando el mayor nivel de detalle y similitud con la realidad. Trabajar en las texturas para lograr una apariencia auténtica.
2. Modelar 2 "Sillones Artesanales Individuales" siguiendo las imágenes de referencia para capturar su diseño único y peculiaridades. Incorporar las texturas adecuadas para recrear la apariencia artesanal.

3. Modelar 1 "Sillón Grupal Artesanal" con atención a las imágenes de referencia para capturar su forma y estilo característicos. Añadir las texturas correspondientes para lograr un acabado auténtico y realista.
4. Modelar 1 "Mecedora Artesanal" basándose en imágenes de referencia para reproducir su forma y detalles específicos. Trabajar en las texturas para recrear la madera y otros materiales de manera convincente.
5. Modelar "Mueble para Televisor" siguiendo las imágenes de referencia para obtener las dimensiones y detalles precisos. Trabajar en las texturas para lograr un acabado realista y coherente con el material del mueble.
6. Modelar "Televisor" basado en imágenes de referencia para capturar su forma y proporciones exactas. Trabajar en las texturas para simular una pantalla y otros elementos realistas.
7. Modelar 2 "Cuadros de Flores" siguiendo las imágenes de referencia para obtener las formas y diseños florales correspondientes. Trabajar en las texturas para representar las flores y los marcos de manera auténtica.
8. Modelar "Refrigerador Grande" acorde a las imágenes de referencia para capturar su forma y detalles específicos. Trabajar en las texturas para recrear los materiales y los acabados característicos del refrigerador.
9. Modelar "Microondas" siguiendo las imágenes de referencia para obtener las dimensiones y características precisas. Trabajar en las texturas para lograr un aspecto metálico y otros detalles realistas.
10. Modelar "Estufa" basándose en imágenes de referencia para reproducir su diseño y elementos distintivos. Trabajar en las texturas para representar los materiales y las superficies de manera fiel.
11. Modelar "Mesa" siguiendo las imágenes de referencia para capturar su forma, tamaño y detalles específicos. Trabajar en las texturas para recrear la madera u otros materiales de manera auténtica.
12. Modelar "Alacena" basándose en las imágenes de referencia para obtener las dimensiones y detalles exactos. Trabajar en las texturas para lograr un acabado realista y coherente con el material de la alacena.
13. Modelar la fachada en 3D basada en imágenes de referencia de la casa. Asegurarse de capturar los detalles arquitectónicos y las texturas de los materiales de manera precisa y realista.

2.2.2. Modelado 3D.

Para los 7 objetos mencionados (Sillón Floreado Antiguo, 2 sillones Artesanales Individuales, 1 sillón Grupal Artesanal, 1 mecedora Artesanal, Mueble para Televisor, Televisor y 2 cuadros de Flores), se siguió un proceso similar utilizando la herramienta Maya y diversas funciones y herramientas disponibles en el software. A continuación, se detalla el proceso general:

1. Creación de la forma básica:
   - Se utilizó la herramienta Polygon Primitives para crear la forma general de cada objeto.

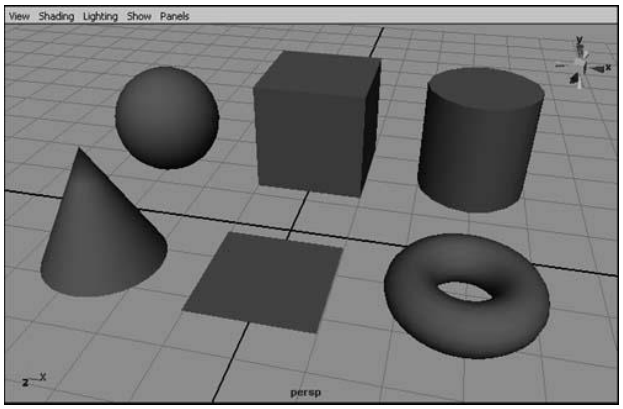


Imagen 2.2.2.1 (Polygon Primitives***)***

- Se ajustaron los vértices, aristas y caras para obtener la forma deseada.

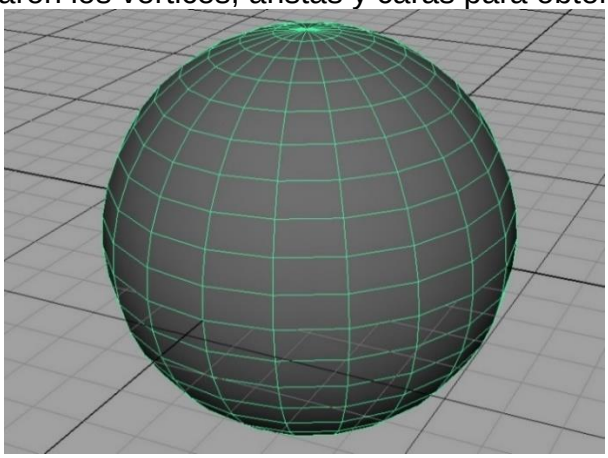


Imagen 2.2.2.2 (Vertices, edges, faces)

2. Añadir detalles y características:
   - Se utilizaron herramientas como Extrude, Bevel Component y Detach para agregar detalles y esculpir características específicas en cada objeto.

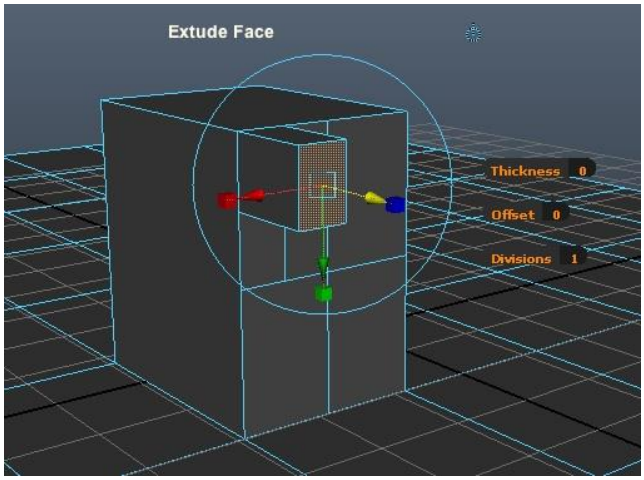


Imagen 2.2.2.3 (Extrude)

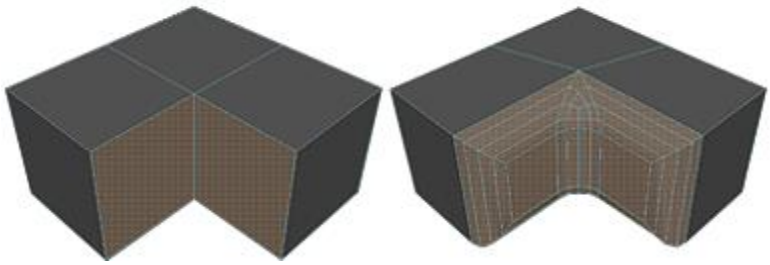


Imagen 2.2.2.3 (*Bevel Component*)

- Se ajustaron los parámetros de estas herramientas para obtener los resultados deseados.

3. Optimización del modelo:
   - Se utilizaron herramientas como Delete Conturtion History para mantener limpio el historial de modificaciones del objeto y reducir el tamaño del archivo.

4. Ajuste de transformaciones:
   - Se aplicó la herramienta Freeze Transformations para restablecer las transformaciones del objeto y asegurarse de que estuvieran en su estado inicial.

Para los últimos 5 objetos se realizaron consideraciones especiales (Refrigerador Grande, Microondas, Estufa, Mesa y Alacena) debido a su naturaleza más simple en comparación con los objetos previamente mencionados. Con el objetivo de optimizar el tiempo y los recursos, se decidió utilizar la plataforma TurboSquid para descargar modelos gratuitos que se ajustaran a las necesidades del proyecto.

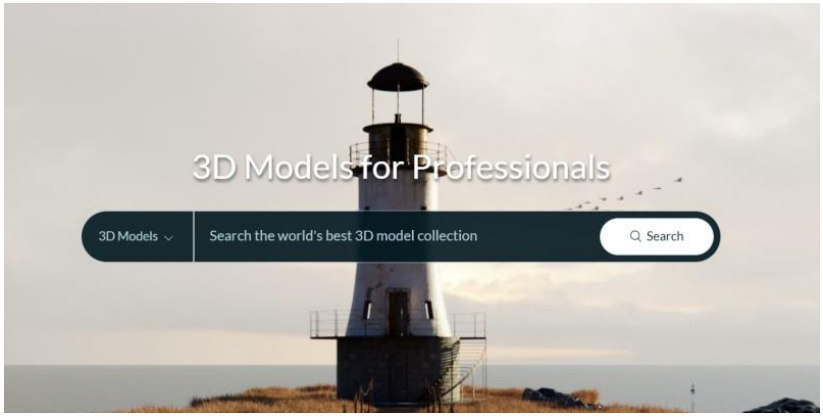


Imagen 2.2.2.4

TurboSquid es una reconocida plataforma en línea que ofrece una amplia variedad de modelos 3D de alta calidad creados por artistas profesionales de

todo el mundo. A través de esta plataforma, se pudo acceder a una extensa biblioteca de objetos 3D, incluidos los mencionados anteriormente.

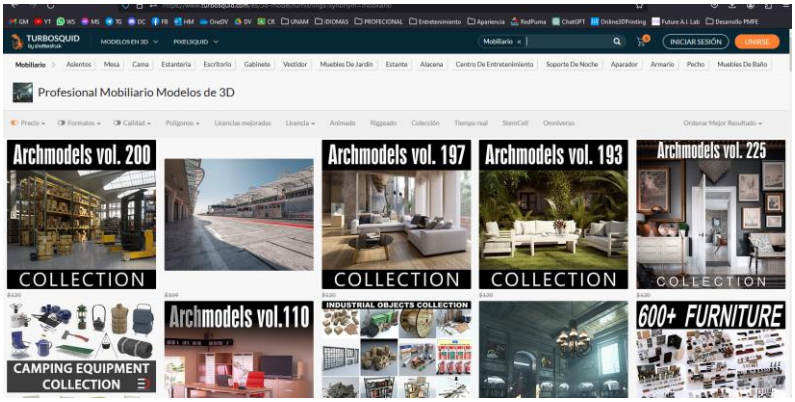


Imagen 2.2.2.5

La elección de utilizar modelos descargados de TurboSquid se basó en la simplicidad de los objetos requeridos y en la disponibilidad de modelos adecuados que cumplían con los requisitos de calidad y realismo visual para el proyecto. Esto permitió ahorrar tiempo en la etapa de modelado y enfocar los esfuerzos en otras áreas del proyecto que requerían mayor atención y personalización.

2.2.3 Materiales y Texturización

Para llevar a cabo la texturización de nuestros modelos, recurrimos a la herramienta GIMP. Utilizamos GIMP para realizar diversas tareas, como agregar canales alfa a las texturas y garantizar la transparencia adecuada en determinadas áreas de nuestros modelos. Esta función nos permitió lograr efectos visuales más realistas al aplicar las texturas en Maya.

Imagen 2.2.3.1 (GIMP)

Además, tuvimos que asegurarnos de que las imágenes de las texturas estuvieran escaladas en potencias de dos. Esto se debió a un requisito específico de Maya, ya que el software solo acepta texturas en formatos de tamaño potencia de dos, como 256x256, 512x512, 1024x1024, y así sucesivamente. Ajustamos y redimensionamos nuestras texturas utilizando GIMP para cumplir con este requisito y garantizar la compatibilidad adecuada con Maya.

En cuanto a los materiales en Maya, el material Phong desempeñó un papel crucial en la apariencia visual de nuestros objetos. El material Phong es conocido por su capacidad para simular interacciones realistas de luz y sombra en los objetos. Nos permitió controlar tanto los reflejos especulares como la reflectividad general de nuestros modelos, lo que resultó en un aspecto más auténtico y atractivo.

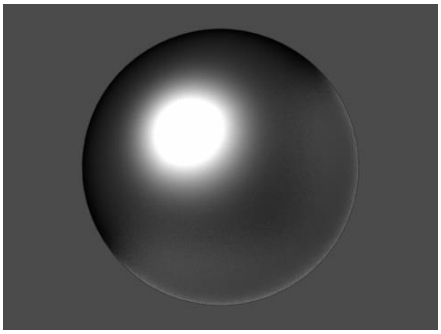


Imagen 2.2.3.2 (Ejemplo del Material Phong)

Además de la textura base, para lograr un acabado más completo, incorporamos un mapa especular al material Phong. Este mapa especular nos brindó mayor control sobre cómo la luz interactúa con nuestros modelos, permitiéndonos definir áreas de mayor o menor reflectividad. Sin embargo, para generar estos mapas especulares, recurrimos a una herramienta adicional llamada NormalMap Online. Esta herramienta en línea nos facilitó la creación de los mapas especulares necesarios para nuestros objetos, enriqueciendo así la apariencia visual y mejorando la forma en que interactúan con la iluminación.

Imagen 2.2.3.3 (Especular Online)

2.2.4 Exportación OBJ

La decisión de exportar los modelos a archivos .OBJ y luego llamarlos desde el lenguaje C++ se tomó por varias razones fundamentales en el contexto de nuestro proyecto.

En primer lugar, el formato de archivo .OBJ es ampliamente utilizado y reconocido en la industria del modelado 3D. Es un formato de archivo de texto simple que almacena la geometría y las propiedades de los modelos, como las coordenadas de los vértices, las normales y las coordenadas de las texturas. Esto significa que es compatible con una amplia gama de software y bibliotecas de renderizado, lo que facilita su integración en nuestro proyecto basado en C++.

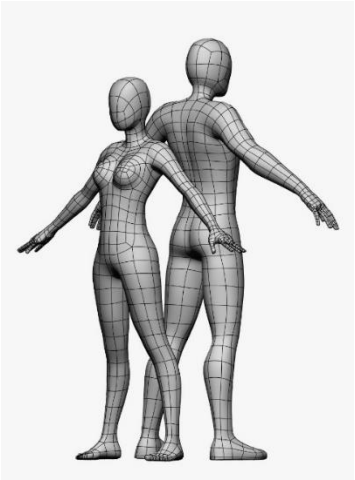


Imagen 2.2.4.1

Además, los archivos .OBJ son fáciles de leer y analizar desde el código C++. El formato utiliza una estructura simple y legible que permite extraer la información necesaria para representar los modelos en nuestro programa. Al exportar los modelos a archivos .OBJ, podemos cargar y procesar la información geométrica de manera eficiente en nuestro entorno de desarrollo de C++, lo que facilita la manipulación y renderizado de los objetos 3D.

Otra ventaja importante de los archivos .OBJ es su capacidad para almacenar datos adicionales, como los materiales y las texturas asociadas a los modelos. Estos datos son esenciales para lograr una representación visual realista de los objetos en nuestro proyecto. Al exportar los modelos junto con sus materiales y texturas a archivos .OBJ, podemos mantener la coherencia y la integridad de los datos, lo que nos permite aplicar adecuadamente las propiedades visuales a los modelos en nuestro programa C++.

2.3 Tercer Sprint (Implementación)

2.3.1 Sprint Backlog

Objetivo del Sprint: Implementar las funcionalidades básicas de gráficos 3D en el proyecto, incluyendo el uso de librerías, shaders, transformaciones básicas y desarrollo de código en C++ para la carga de modelos .obj.
Tareas:

1. Investigar y seleccionar las librerías adecuadas para el proyecto:
   - Evaluar opciones de librerías gráficas, como OpenGL o DirectX.
   - Realizar un análisis comparativo de las características, documentación y compatibilidad de las librerías.
   - Tomar una decisión informada sobre la librería a utilizar y documentarla en el informe del sprint.
2. Implementar shaders básicos:
   - Estudiar el funcionamiento de los shaders en gráficos 3D.
   - Crear shaders básicos, como vertex shaders y fragment shaders, para manipular la apariencia y los efectos visuales de los objetos en la escena.
   - Probar los shaders en un entorno de desarrollo para asegurar su correcto funcionamiento.
3. Integrar las transformaciones básicas 3D:
   - Investigar las transformaciones 3D, como traslación, rotación y escala.
   - Implementar las funciones y algoritmos necesarios para aplicar estas transformaciones a los objetos en la escena.
   - Verificar que las transformaciones se apliquen correctamente y que los objetos respondan adecuadamente a los cambios.
4. Desarrollo de código en C++ para carga de modelos .obj:
   - Estudiar el formato de archivo .obj y su estructura.
   - Implementar un código en C++ para cargar y renderizar modelos .obj en la escena.
   - Probar la carga de modelos .obj en el entorno de desarrollo y verificar que se visualicen correctamente.
5. Realizar pruebas y depuración:
   - Realizar pruebas exhaustivas para asegurar el correcto funcionamiento de las librerías, shaders, transformaciones y carga de modelos .obj.
   - Identificar y solucionar posibles errores o problemas en el código.
   - Realizar pruebas de rendimiento para evaluar el desempeño de la aplicación y optimizar el código si es necesario.
6. Documentar los resultados y avances del sprint:
   - Registrar todos los cambios, implementaciones y pruebas realizadas durante el sprint en un informe detallado.
   - Actualizar la documentación del proyecto con los nuevos elementos implementados.
   - Preparar una presentación para compartir los resultados y el progreso alcanzado hasta el momento.

2.3.2 Librerias a utilizar

1. iostream: Esta librería permite la entrada y salida estándar en el programa, lo que incluye la función de imprimir mensajes en la consola.
2. cmath: Esta librería proporciona funciones matemáticas comunes, como funciones trigonométricas, exponenciales y logarítmicas, que se pueden utilizar en cálculos y manipulaciones matemáticas.
3. GLEW: La librería GLEW (OpenGL Extension Wrangler Library) se utiliza para gestionar extensiones OpenGL de manera eficiente. Proporciona funciones y macros que facilitan la carga y el uso de extensiones de OpenGL en el proyecto.
4. GLFW: GLFW (Graphics Library Framework) es una librería de código abierto que se utiliza para crear y administrar ventanas, contextos OpenGL y manejo de entradas. Proporciona una interfaz de programación de aplicaciones (API) sencilla y portátil para interactuar con el sistema operativo y la ventana de la aplicación.
5. stb_image: La librería “stb_image” se utiliza para cargar imágenes en el proyecto. Permite cargar diferentes formatos de imagen y proporciona funciones para acceder a los píxeles y metadatos de la imagen.
6. GLM: La librería “GLM” (OpenGL Mathematics) se utiliza para realizar operaciones matemáticas en el proyecto. Proporciona estructuras y funciones matemáticas especialmente diseñadas para trabajar con gráficos en 3D, como transformaciones, cálculos de vectores y matrices.
7. SOIL2: La librería “SOIL2” se utiliza para cargar modelos en el proyecto. Permite cargar modelos en diferentes formatos y proporciona funciones para acceder a los datos del modelo, como vértices, normales y coordenadas de textura.

2.3.3 Archivos de Encabezado:

1. “Shader.h”: Este archivo de encabezado contiene las declaraciones y definiciones relacionadas con los shaders. Los shaders son programas que se ejecutan en la tarjeta gráfica y controlan la apariencia y el comportamiento de los objetos renderizados. El archivo de encabezado “Shader.h” proporciona las funciones y estructuras necesarias para cargar, compilar y utilizar shaders en el proyecto.
2. “Camera.h”: Este archivo de encabezado se utiliza para definir la clase de la cámara. La cámara se encarga de controlar la vista y la posición del observador en la escena 3D. Proporciona funciones y métodos para configurar y manipular la posición, orientación y proyección de la cámara.
3. “Model.h”: Este archivo de encabezado define la clase Model, que se utiliza para cargar y representar modelos 3D en el proyecto. Proporciona funciones y métodos para cargar archivos de modelos en diferentes formatos, así como para manipular y renderizar los modelos en la escena.
4. “Texture.h”: Este archivo de encabezado se utiliza para definir la clase Texture, que se encarga de gestionar las texturas utilizadas en el proyecto.

Proporciona funciones y métodos para cargar, asignar y aplicar texturas a los objetos renderizados.

Cada uno de estos archivos de encabezado juega un papel fundamental en el proyecto al proporcionar las estructuras, funciones y métodos necesarios para el manejo de shaders, cámaras, modelos, texturas y animaciones. Estas clases y funciones son esenciales para la implementación y el funcionamiento adecuado del proyecto en términos de renderizado y visualización de objetos en un entorno 3D.

2.3.4 Shaders

1. **lightingShader:**
   - Descripción: Representa un objeto Shader utilizado en el proyecto de modelado.
   - Valor: Contiene el Shader utilizado para iluminación en el proyecto.
   - Tipo: Objeto Shader.
2. **lampShader:**
   - Descripción: Representa un objeto Shader utilizado en el proyecto de modelado.
   - Valor: Contiene el Shader utilizado para renderizar una lámpara en el proyecto.
   - Tipo: Objeto Shader.
3. **SkyBoxshader:**
   - Descripción: Representa un objeto Shader utilizado en el proyecto de modelado.
   - Valor: Contiene el Shader utilizado para renderizar el skybox (caja de cielo) en el proyecto.
   - Tipo: Objeto Shader.
4. **animShader:**
   - Descripción: Representa un objeto Shader utilizado en el proyecto de modelado.
   - Valor: Contiene el Shader utilizado para animaciones en el proyecto.
   - Tipo: Objeto Shader.
5. **Anim:**
   - Descripción: Representa un objeto Shader utilizado en el proyecto de modelado.
   - Valor: Contiene el Shader utilizado para animaciones específicas en el proyecto.
   - Tipo: Objeto Shader.

2.3.5 Funciones

1. **Función doMovement():**
   - Descripción: Esta función se encarga de procesar y alterar los valores utilizados en las transformaciones básicas de modelado 3D para realizar animaciones.

- Detalles:
  - Utiliza cálculos matemáticos para actualizar los valores de traslación, rotación y escala de los objetos en la escena 3D.
  - Los cambios realizados en esta función permiten lograr efectos de animación, como movimiento suave, rotación continua o cambios en la escala de los objetos.
  - Asegúrate de que los cálculos se realicen de manera eficiente para garantizar un rendimiento óptimo en la animación.

2. **Función keyCallback():**
   - Descripción: Esta función se utiliza para detectar cuando una tecla es accionada en el entorno virtual.
   - Detalles:
     - Captura eventos de teclado y realiza acciones correspondientes en función de las teclas presionadas.
     - Puede utilizarse para activar o desactivar ciertas animaciones, cambiar el modo de visualización, alternar entre diferentes cámaras o ejecutar cualquier otra acción relacionada con el teclado.
     - Asegúrate de que las acciones sean intuitivas y estén documentadas correctamente para la interacción del usuario.

3. **Función mouseCallback():**
   - Descripción: Esta función permite utilizar el mouse dentro del ambiente virtual para realizar interacciones o manipular objetos.
   - Detalles:
     - Captura eventos del mouse, como movimientos, clics o desplazamientos.
     - Utiliza los datos obtenidos del mouse para realizar acciones específicas en la escena 3D, como seleccionar objetos, rotar la cámara, interactuar con elementos interactivos o cualquier otra acción relacionada con el mouse.
     - Asegúrate de que las interacciones sean suaves y respondan de manera adecuada a los movimientos del mouse.

Estas funciones son elementos importantes para la interactividad y animación en el proyecto.

2.3.6 Iluminación

El proyecto se beneficia de la implementación de shaders de iluminación para lograr efectos realistas y mejorar la calidad visual de los modelos 3D. Gracias a la utilización de lighting shaders, se pueden simular tres tipos de iluminación: spotlight (foco), pointlight (punto de luz) y directionallight (luz direccional). Cada uno de estos tipos de luz tiene características específicas que permiten crear diferentes ambientes y efectos visuales en la escena.

En el caso de este proyecto en particular, se optó por utilizar en mayor medida la pointlight (punto de luz) para iluminar de manera más realista los modelos 3D. La pointlight es una fuente de luz omnidireccional que emite luz en todas las direcciones desde un punto específico en el espacio. Esto permite simular fuentes de luz como el sol o focos ubicados en posiciones estratégicas dentro de la escena.

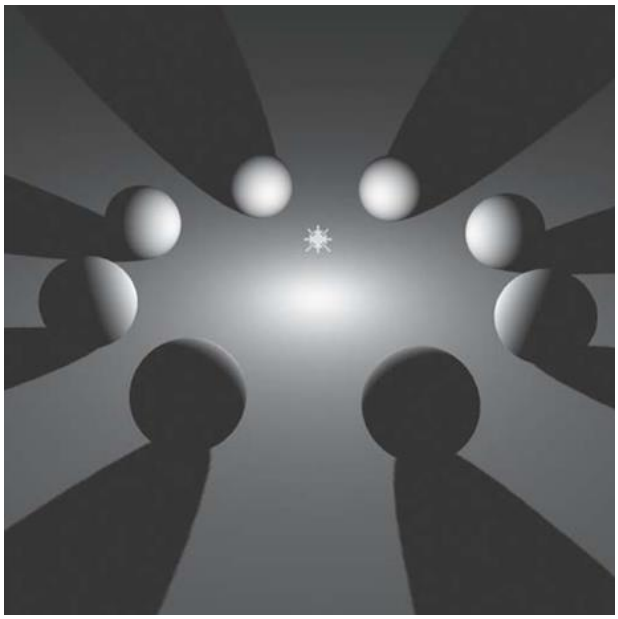


Imagen 2.3.6.1

La pointlight ofrece varios beneficios para el proyecto, como:

1. Realismo: Al posicionar las pointlights en lugares estratégicos, se puede simular la forma en que la luz se refleja y se proyecta sobre los objetos en la escena, creando sombras y resaltando detalles.

2. Flexibilidad: Las pointlights se pueden ajustar en términos de intensidad, color y posición para adaptarse a las necesidades de iluminación específicas de cada objeto o escena.

3. Interacción con materiales: La pointlight interactúa con los materiales aplicados a los modelos 3D, permitiendo resaltar características como el brillo, la reflexión y las texturas.

Al utilizar principalmente pointlights en el proyecto, se logra una iluminación más detallada y realista en comparación con un enfoque exclusivo en spotlight o directionallight. Esto ayuda a resaltar los detalles de los objetos y a crear una experiencia visualmente atractiva para los usuarios.

2.3.7 Animaciones

En el proyecto, se implementaron animaciones utilizando las transformaciones básicas de modelado 3D, que incluyen traslación, rotación y escalamiento. Estas transformaciones se aplicaron a los modelos 3D de la escena con el fin de lograr efectos de movimiento y cambio en su apariencia.

Para controlar las animaciones, se utilizaron variables que se modificaban matemáticamente a lo largo del tiempo de la animación. Estas variables eran ajustadas y actualizadas en función del estado de la animación que se deseaba recrear.

1. Traslación: Se utilizó la traslación para mover los modelos en diferentes direcciones dentro de la escena. Las variables que controlaban la posición de los modelos se actualizaron en cada cuadro de la animación, permitiendo que los objetos se movieran suavemente a lo largo del tiempo.

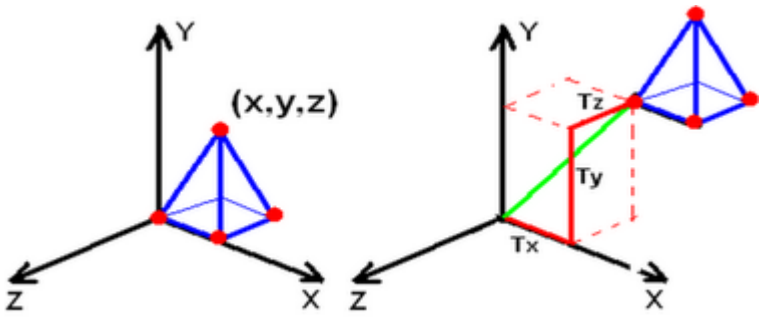


Imagen 2.3.7.1 (Traslación)

2. Rotación: Se aplicó la rotación para girar los modelos alrededor de un eje específico. Las variables que controlaban el ángulo de rotación se modificaron en cada cuadro de la animación, permitiendo que los objetos giraran de manera fluida y realista.

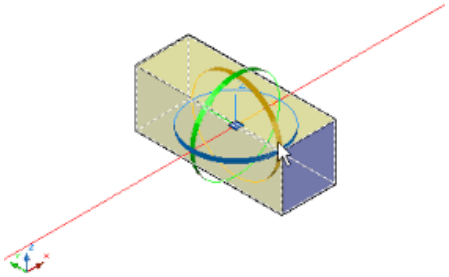


Imagen 2.3.7.1 (Rotación)

3. Escalamiento: Se utilizó el escalamiento para cambiar el tamaño de los modelos durante la animación. Las variables relacionadas con el factor de

escala se ajustaron en función del progreso de la animación, lo que permitió que los objetos se expandieran o contraigan de manera controlada.

2.3.8 Contexto

Dos hermanos decidieron pasar unas relajantes vacaciones en una casa vacacional ubicada en Cuautla. Durante su estancia, cada uno disfrutó de diferentes actividades en distintas áreas de la casa.

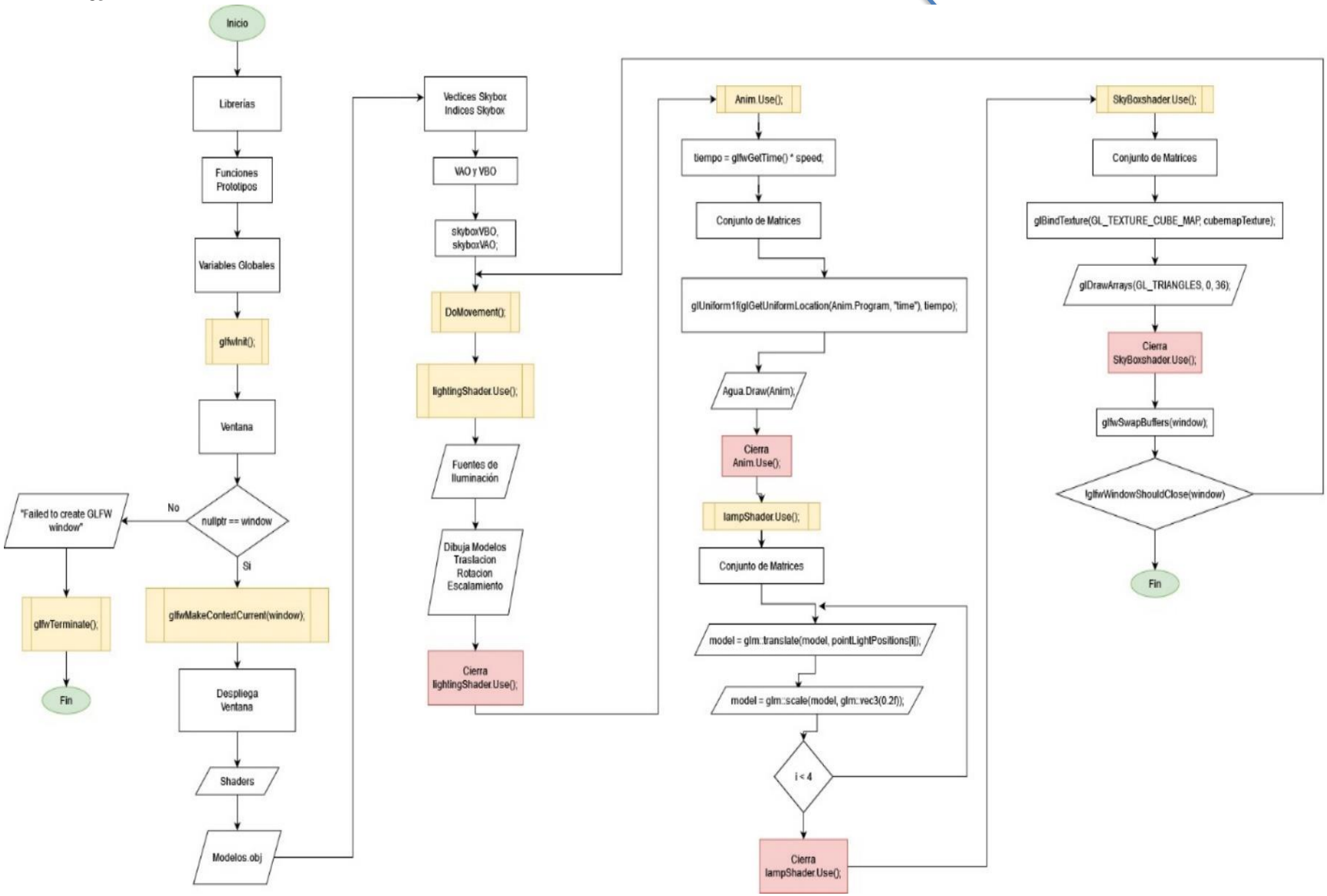
Hermano 1:
Uno de los hermanos decidió pasar un agradable rato en la sala de la casa vacacional. En este ambiente acogedor y confortable, pudo disfrutar de momentos de descanso, relajación y entretenimiento. Tal vez se sentó en un cómodo sillón floreado antiguo mientras leía un libro, escuchaba música o veía su programa de televisión favorito. La sala ofrecía un ambiente tranquilo y propicio para disfrutar de un tiempo de relajación y desconexión.

Hermano 2:
Por otro lado, el otro hermano decidió aprovechar el amplio espacio del patio de la casa para probar un dron. Con emocionantes momentos de diversión y entretenimiento, este hermano se sumergió en el mundo de la tecnología y la exploración aérea. Controlando el dron con habilidad, realizó vuelos suaves y emocionantes, observando desde las alturas los alrededores de la casa vacacional en Cuautla. El patio proporcionaba el espacio necesario para que el hermano pudiera disfrutar de su afición por los drones y experimentar con nuevas maniobras y perspectivas.

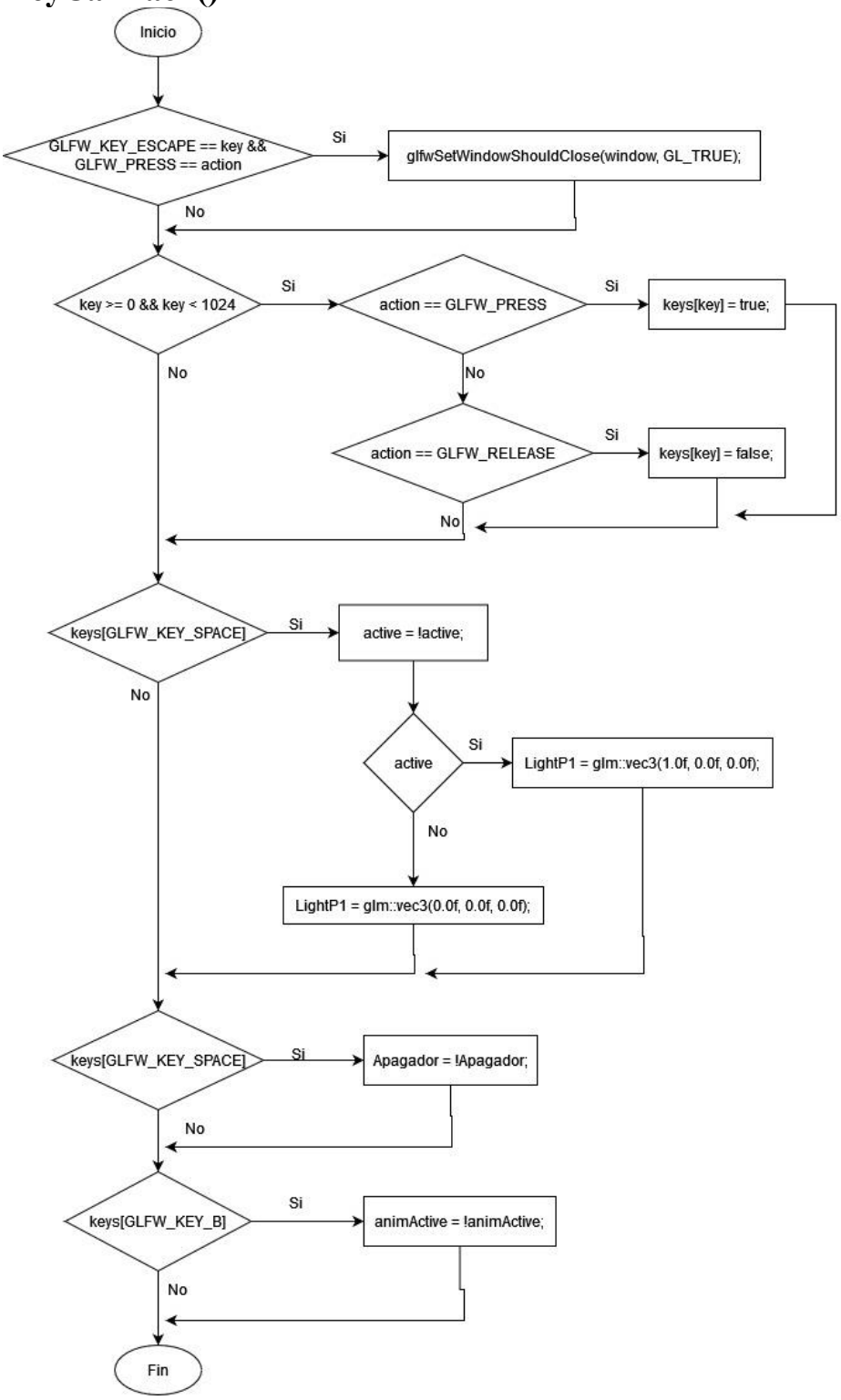
En conjunto, estos dos hermanos encontraron un equilibrio entre relajación y aventura durante sus vacaciones en la casa vacacional en Cuautla. Cada uno pudo disfrutar de actividades que les apasionaban en diferentes áreas de la propiedad, creando recuerdos inolvidables y aprovechando al máximo su tiempo juntos en un entorno tranquilo y agradable.

2.3.9 Diagramas de Flujo

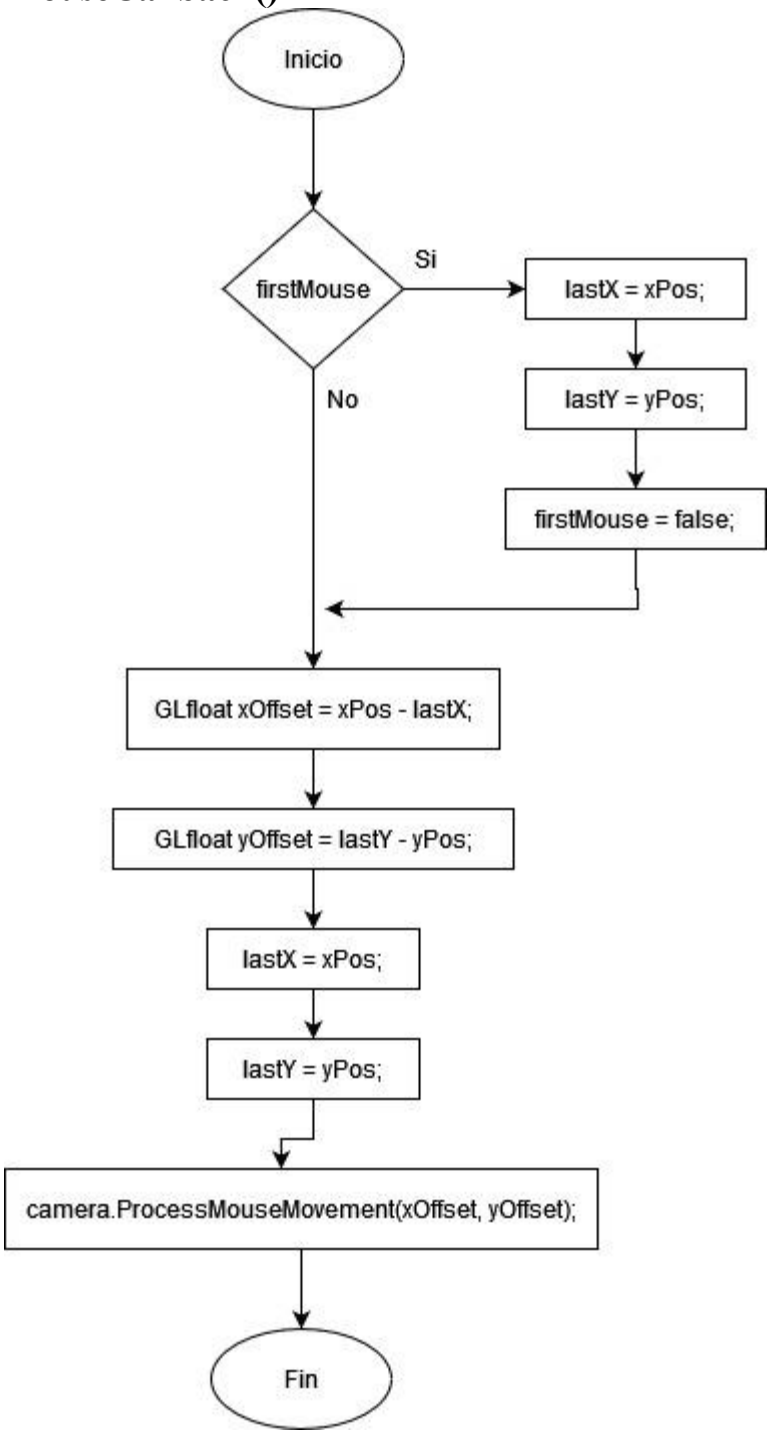
Main:



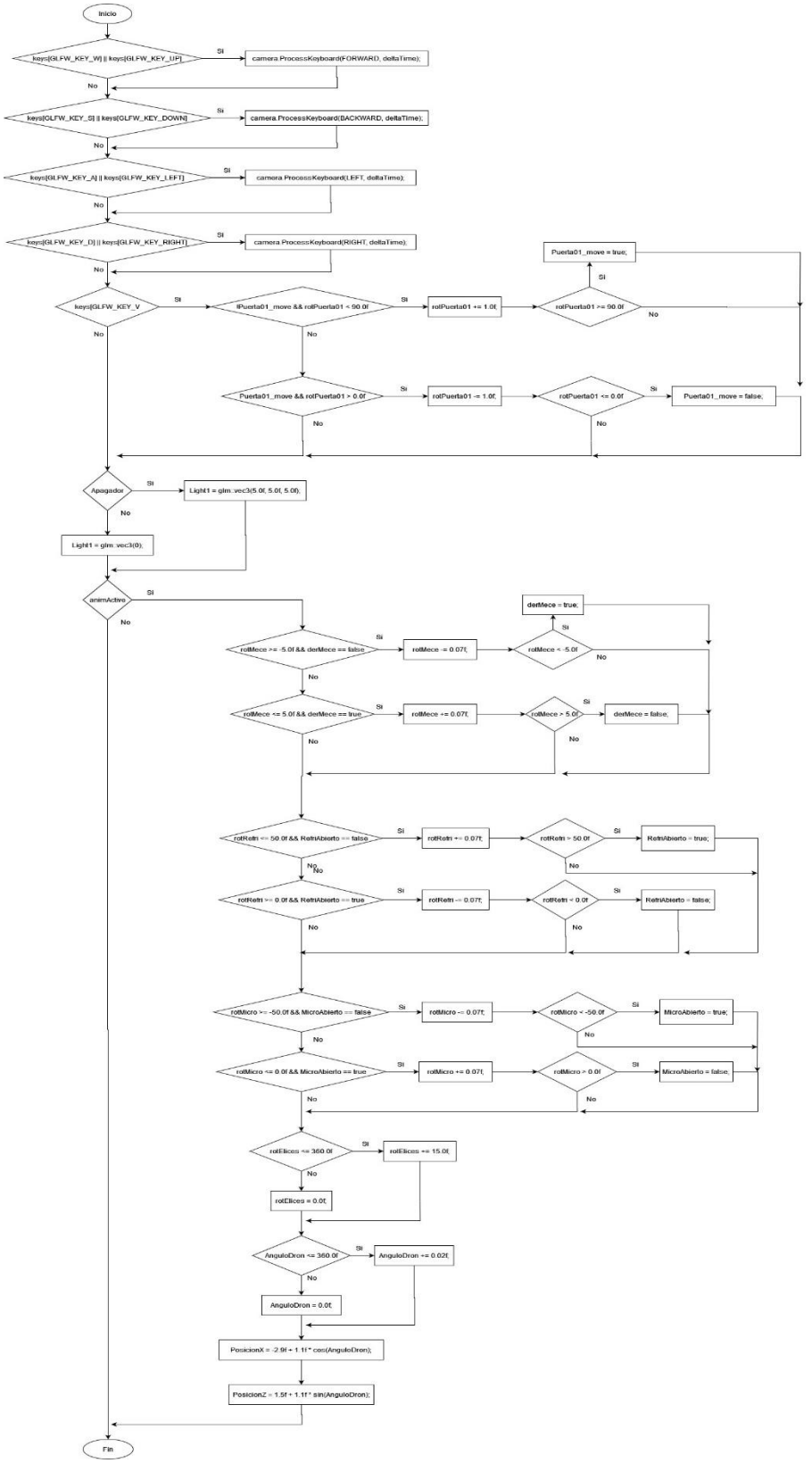
KeyCallBack()



MouseCallback()



DoMovement()


Inicio
keys[GLFW_KEY_W] || keys[GLFW_KEY_UP]
Si
camera.ProcessKeyboard(FORWARD, deltaTime);
No
keys[GLFW_KEY_S] || keys[GLFW_KEY_DOWN]
Si
camera.ProcessKeyboard(BACKWARD, deltaTime);
No
keys[GLFW_KEY_A] || keys[GLFW_KEY_LEFT]
Si
camera.ProcessKeyboard(LEFT, deltaTime);
No
keys[GLFW_KEY_D] || keys[GLFW_KEY_RIGHT]
Si
camera.ProcessKeyboard(RIGHT, deltaTime);
No
keys[GLFW_KEY_V
Si
!Puerta01_move && rotPuerta01 < 90.0f
Si
rotPuerta01 += 1.0f;
rotPuerta01 >= 90.0f
Si
Puerta01_move = true;
No
No
Puerta01_move && rotPuerta01 > 0.0f
Si
rotPuerta01 -= 1.0f;
rotPuerta01 <= 0.0f
Si
Puerta01_move = false;
No
No
No
Apagador
Si
Light1 = glm::vec3(5.0f, 5.0f, 5.0f);
No
Light1 = glm::vec3(0);
animActiva
Si
No
rotMece >= -5.0f && derMece == false
Si
rotMece -= 0.07f;
rotMece < -5.0f
Si
derMece = true;
No
No
rotMece <= 5.0f && derMece == true
Si
rotMece += 0.07f;
rotMece > 5.0f
Si
derMece = false;
No
No
rotRetri <= 50.0f && RetriAbierto == false
Si
rotRetri += 0.07f;
rotRetri > 50.0f
Si
RetriAbierto = true;
No
No
rotRetri >= 0.0f && RetriAbierto == true
Si
rotRetri -= 0.07f;
rotRetri < 0.0f
Si
RetriAbierto = false;
No
No
rotMicro >= -50.0f && MicroAbierto == false
Si
rotMicro -= 0.07f;
rotMicro < -50.0f
Si
MicroAbierto = true;
No
No
rotMicro <= 0.0f && MicroAbierto == true
Si
rotMicro += 0.07f;
rotMicro > 0.0f
Si
MicroAbierto = false;
No
No
rotElices <= 360.0f
Si
rotElices += 15.0f;
No
rotElices = 0.0f;
AnguloDron <= 360.0f
Si
AnguloDron += 0.02f;
No
AnguloDron = 0.0f;
PosicionX = -2.9f + 1.1f * cos(AnguloDron);
PosicionZ = 1.5f + 1.1f * sin(AnguloDron);
Fin

2.3.10 Diccionario de Variables

Variable	Tipo	Descripción
WIDTH	GLuint	Ancho de la ventana, con un valor constante de 800.
HEIGHT	GLuint	Altura de la ventana, con un valor constante de 600.
SCREEN_WIDTH	int	Ancho de la pantalla.
SCREEN_HEIGHT	int	Altura de la pantalla.
camera	Camera	Objeto de la clase Camera que representa la cámara y almacena sus coordenadas iniciales.
lastX	GLfloat	Coordenada X del último punto en el eje de la ventana dividido por 2.
lastY	GLfloat	Coordenada Y del último punto en el eje de la ventana dividido por 2.
animActive	bool	Indicador para activar o desactivar las animaciones.
rotMece	float	Ángulo de rotación de la mecedora.
derMece	bool	Indicador de si la mecedora ha llegado a su límite.
rotPuerta01	float	Ángulo de rotación para abrir y cerrar puertas.
Apagador	bool	Indicador para apagar o encender la luz.
rotElices	float	Ángulo de rotación de las hélices del dron.
PosicionX	float	Posición X del dron.
PosicionZ	float	Posición Z del dron.
AnguloDron	float	Ángulo de posición del dron para calcular X y Z.
rotRefri	float	Ángulo de rotación de la puerta del refrigerador.
RefriAbierto	bool	Indicador para saber si la puerta del refrigerador está abierta o cerrada.
rotMicro	float	Ángulo de rotación de la puerta del microondas.
MicroAbierto	bool	Indicador para saber si la puerta del microondas está abierta o cerrada.

3. Resultados Obtenidos

3.1 Fachada

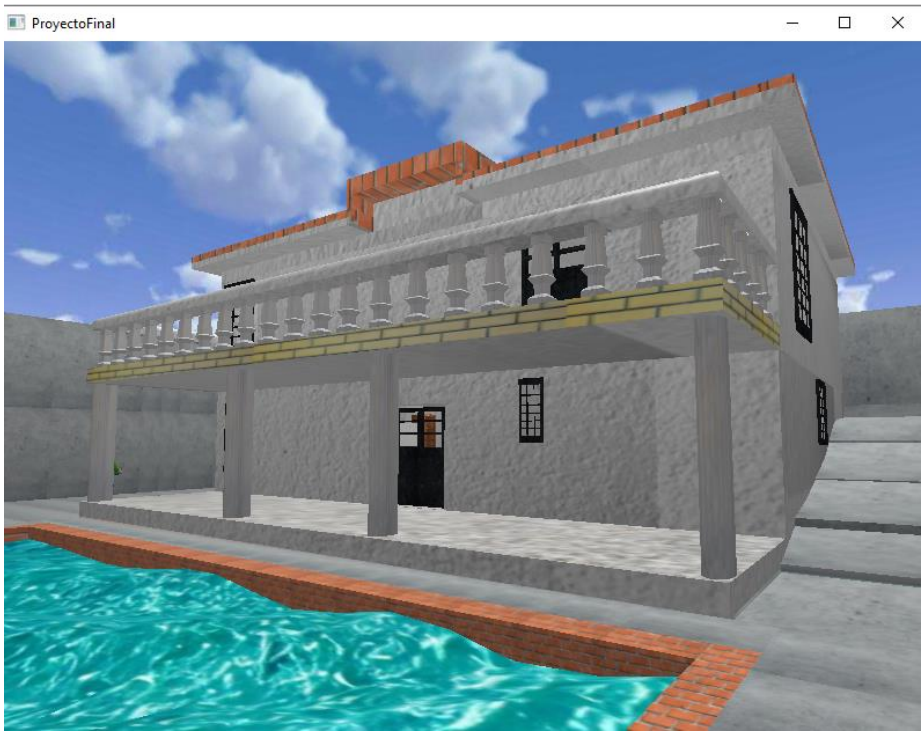


Imagen 3.1.1

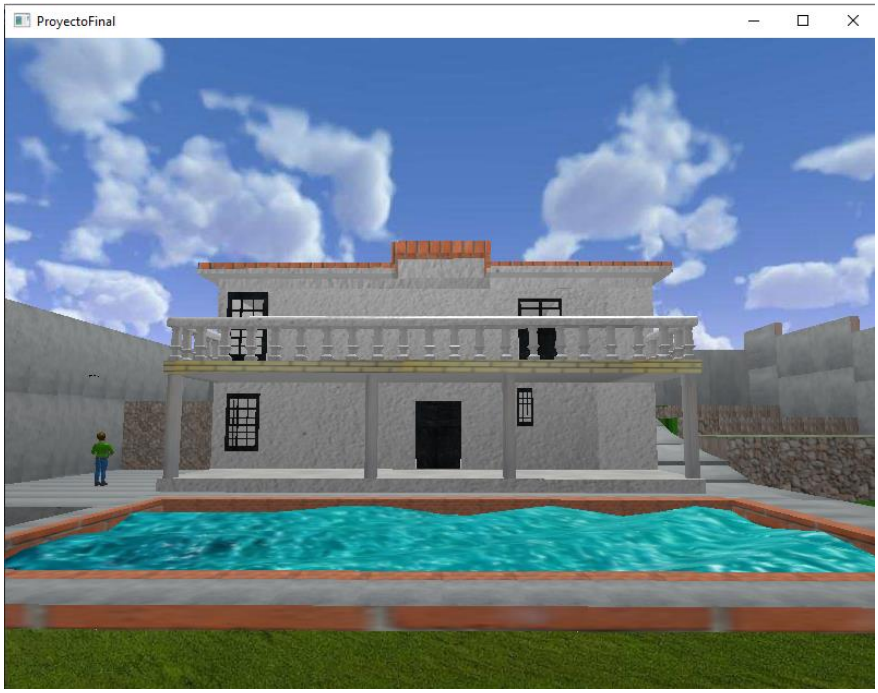


Imagen 3.1.2

3.2 Cuartos

3.2.1 Sala

Imagen 3.2.1.1

Imagen 3.2.1.2

3.2.2 Cocina

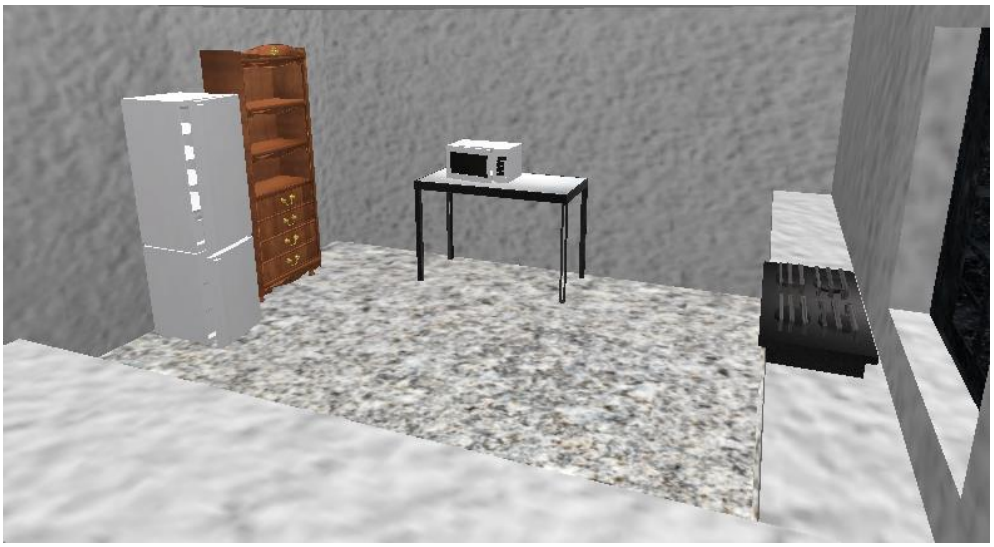


Imagen 3.2.2.1

3.3 Animaciones

Imagen 3.3.1

Imagen 3.3.2

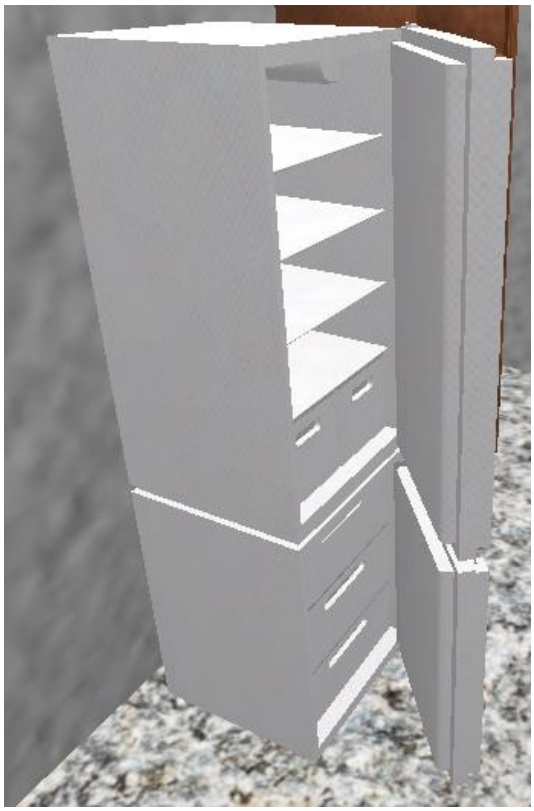


Imagen 3.3.3

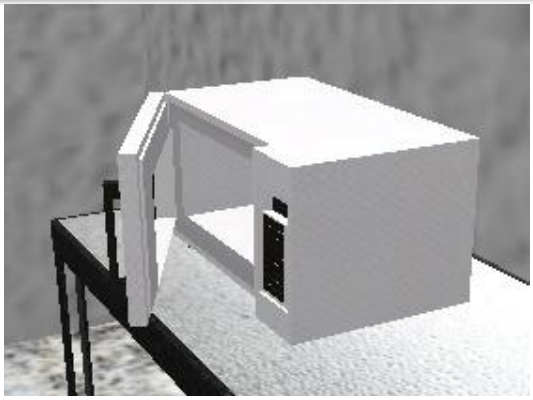


Imagen 3.3.3

4. Lecciones Aprendidas y Conclusión:

4.1 Aprendizaje

Durante el desarrollo del proyecto bajo la metodología SCRUM, se adquirieron diversos aprendizajes que contribuyeron al crecimiento y desarrollo del equipo. A continuación, se resumen los principales aprendizajes obtenidos:

1. Herramientas de diseño y edición de imágenes: Durante el proyecto, se tuvo la oportunidad de trabajar con diferentes herramientas de diseño y edición de imágenes, como GIMP. A través de su utilización, se aprendió a manipular y mejorar las texturas de los modelos, agregando canales alfa y escalando las imágenes en potencias de 2 para su correcta implementación en Maya. Esto permitió obtener resultados visuales de alta calidad y coherencia estética en el proyecto.
2. Uso de Maya: Se adquirieron conocimientos sobre el software Maya, una poderosa herramienta de modelado y animación en 3D. A través de su utilización, se aprendió a crear y manipular modelos tridimensionales, aplicar texturas, realizar transformaciones básicas (traslación, rotación, escalamiento) y generar animaciones utilizando las funciones y herramientas disponibles en Maya. Esta experiencia en Maya fue fundamental para lograr el aspecto visual deseado en el proyecto.
3. Implementación de OpenGL en lenguaje C: Se exploró la implementación de OpenGL en lenguaje C para lograr la interacción y renderizado en tiempo real de los modelos en el proyecto. A través de esta implementación, se aprendió a llamar y gestionar los modelos, junto con su información asociada, como texturas, coordenadas y componentes especulares. Esto permitió la visualización y manipulación de los modelos en un entorno virtual, brindando una experiencia inmersiva y realista.

En resumen, el proyecto bajo la metodología SCRUM brindó la oportunidad de aprender y desarrollar habilidades en el manejo de diversas herramientas de diseño y edición de imágenes, así como en el uso de software especializado como Maya y la implementación de OpenGL en lenguaje C. Estos aprendizajes fueron fundamentales para alcanzar los objetivos del proyecto y sentaron las bases para futuros proyectos que requieran de un enfoque similar en el campo del diseño y desarrollo en 3D.

4.2 Conclusiones

En conclusión, el proyecto desarrollado bajo la metodología SCRUM ha sido un proceso enriquecedor que ha permitido alcanzar los objetivos establecidos y obtener resultados satisfactorios. A lo largo de este proyecto, se han empleado diversas herramientas y técnicas de diseño y desarrollo en 3D, lo que ha contribuido al crecimiento y aprendizaje del equipo.

La utilización de herramientas como GIMP para la manipulación de texturas, Maya para el modelado y animación en 3D, y OpenGL en lenguaje C para la implementación y renderizado de los modelos, ha demostrado ser un enfoque eficaz y sólido para lograr resultados visualmente atractivos y realistas. Estas herramientas han permitido una interacción fluida con los modelos, brindando una experiencia inmersiva y de alta calidad.

Además, el trabajo en equipo y la adopción de la metodología SCRUM han sido elementos clave para el éxito del proyecto. La colaboración constante, la comunicación efectiva y la flexibilidad para adaptarse a los cambios han permitido mantener un flujo de trabajo eficiente y una respuesta ágil a los requisitos y necesidades del proyecto.

En términos de aprendizaje, este proyecto ha proporcionado una amplia experiencia en el uso de herramientas de diseño y edición de imágenes, así como en el manejo de software especializado en modelado y animación en 3D. Estos conocimientos adquiridos son valiosos y se pueden aplicar en futuros proyectos relacionados con el diseño y desarrollo en 3D.

En resumen, este proyecto ha sido una oportunidad para adquirir nuevos conocimientos, fortalecer habilidades y obtener resultados exitosos en el campo del diseño y desarrollo en 3D. La combinación de metodologías ágiles, herramientas especializadas y trabajo en equipo ha sido la clave para alcanzar los objetivos planteados y crear un producto final de alta calidad.

5. Referencias

1. Departamento de Ingeniería. (2013). Gráficos por computadora con OpenGL. Recuperado de https://ingenieriayeducacion.files.wordpress.com/2013/12/graficosporcomputadorayopengl.pdf
2. Woo, M., Neider, J., Davis, T., & Shreiner, D. (1999). Guía de programación OpenGL: la guía oficial para aprender OpenGL, versión 1.2. Addison-Wesley Longman Publishing Co., Inc.

Technical Manual (English)

URL: https://github.com/FrankDavidMM/317032392_PROYECTOFINAL2023-2_GPO06.git

1. Introduction

1.1. Project Description

and a facade based on reference images. It includes a synthetic camera and 4 animations related to the tour. Document the project with diagrams, manuals, and a cost analysis. The deliverable should be in digital format with an executable file. Ensure that the space is realistic. Use a GitHub repository. The project is individual, and a lack of originality will be penalized.

1.2 Objective

The objective of the project is to create an interactive and realistic tour where the user can explore two rooms and a facade designed with a high level of detail, providing an immersive and high-quality user experience. The objective includes the integration of a synthetic camera, contextualized animations, comprehensive documentation, and a cost analysis of the project.

1.3 Project Scope

The objective of the "Virtual Tour in OpenGL 3" project is to create an immersive and realistic experience through the development of an interactive virtual tour. The key elements of the objective are as follows:

1. Design of realistic environments: The goal is to recreate two rooms and a facade based on reference images, using an artistic style that aligns with the depicted space. The aim is to achieve a high level of realism and detail in the ambiance of the environments.
2. Smooth interaction and exploration: The project aims to integrate a synthetic camera that allows users to explore and navigate seamlessly through the different environments. The objective is to provide an immersive and comfortable virtual tour experience.
3. Contextualized animations: The project aims to enrich the virtual tour experience by including four animations related to the context of the represented space. These animations will add dynamism and vitality to the environments, creating a greater visual impact.
4. Comprehensive and high-quality documentation: The objective is to develop thorough documentation, including a flowchart of the tour, a Gantt chart for planning, the development methodology used, conclusions, a user manual, and a technical manual. The documentation will be presented in both Spanish and English, with special attention given to writing style and spelling.
5. Cost and pricing analysis: A cost analysis of the project will be conducted, considering the expenses incurred during its development. The objective is to

justify the proposed selling price through a solid argumentation based on costs and the added value of the virtual tour.

The overall objective of the project is to achieve a virtual tour in OpenGL 3 that combines visually appealing and realistic design with an interactive and immersive user experience. The comprehensive documentation and cost analysis will ensure a comprehensive project delivery.

1.4 Scrum Methodology

1.4.1 Principles

The Scrum methodology is based on the following principles:

1. Transparency: All relevant information about the project and the development process should be shared and accessible to all team members and stakeholders. This promotes trust and collaboration.
2. Inspection: Frequent and regular inspections of artifacts, progress, and the work process should be carried out. This allows for the identification of potential issues or deviations and enables timely corrective actions.
3. Adaptation: If deviations are detected or changes are required, adjustments should be made to the plan and work process. Scrum encourages adaptability to respond to changes and maintain focus on project objectives.
4. Empowered Team: The development team has autonomy and responsibility to make decisions related to the work. This includes effort estimation, work planning, and self-organization to achieve established objectives.
5. Incremental Delivery: The product is developed in functional and deliverable increments in each sprint. This allows for early feedback from stakeholders and ensures that the product is evolutionary and adaptable to changing needs.
6. Collaboration: Scrum fosters close and constant collaboration among all team members and stakeholders. Effective communication and teamwork are essential for project success.
7. Business Value Focus: The team focuses on delivering tangible and real value to the customer or end-user. Prioritization is given to functionalities and features that provide maximum benefit, while minimizing work that does not add value.
8. Rapid Delivery Time: Sprints are short and defined periods in which tasks are completed and work increments are delivered. This allows for quick results and early feedback, which helps guide development..

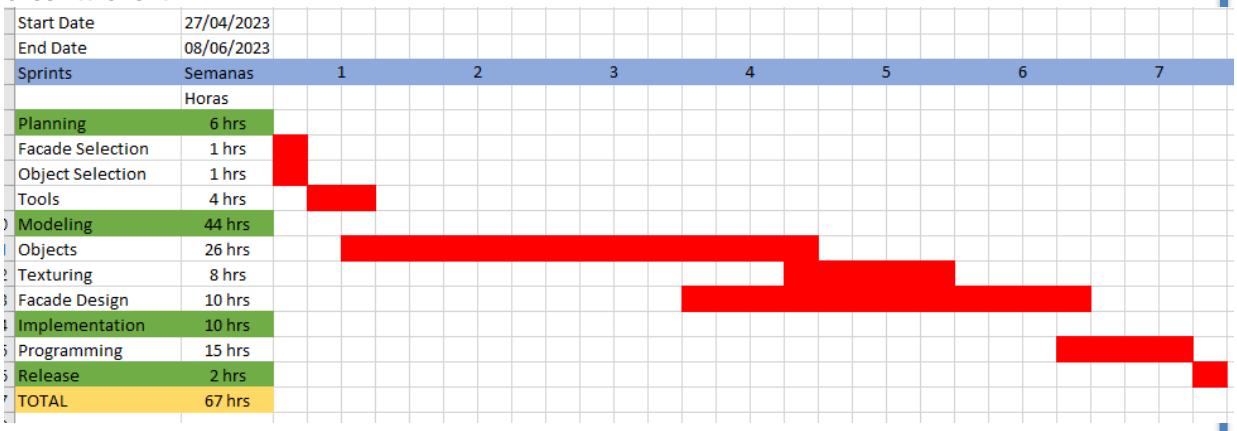
1.4.2 Implementation for the Project

Based on the principles discussed in section 1.4.1, it can be affirmed that the Scrum methodology is highly suitable for the development of this project, due to its remarkable flexibility and its focus on the project lifecycle. In this context, the project lifecycle can be divided into sprints, which represent stages that allow addressing the functionalities and needs established by the Product Owner to satisfy the customer.

It is important to highlight that sprints are characterized by their adaptability, which means that the development team has the necessary flexibility to adjust their approach and respond to the specifications defined by the Product Owner. Furthermore, the fundamental objective of each sprint is to meet the needs and requirements established by the customer, through the guidance and direction provided by the Product Owner.

In conclusion, the Scrum methodology stands as an optimal choice for this project, due to its ability to respond to changes, its focus on delivering value in increments, and its alignment with customer expectations and requirements, through the pivotal role played by the Product Owner.

1.5 Gantt Chart



1.6 Cost Table

Type	Quantity	Unit Cost	Description	Subtotal
*	1	$10,000.00	Work Equipment	$10,000.00
*	1	$2,450.00	1-Year Maya Autodesk License	$2,450.00
Objects	12	$216.67	Object Design	$2,600.04
Textures	12	$67.66	Texture Design	$811.92
Facades	1	$1,000.00	Facade Design	$1,000.00
Hours	15	$150.00	Programming	$2,250.00
			Total Cost:	$19,111.96

2. Development

2.1 First Sprint (Planning)

2.1.1 Sprint BackLog

1. Define tools and development environment:
   - Research and select appropriate tools for OpenGL development, including libraries, frameworks, or necessary IDEs.
   - Set up the development environment based on the chosen tools.

- Establish the use of the C++ programming language as the foundation for project implementation.

2. Establish the environment to be represented:
   - Analyze and understand the client's requirements regarding the environment to be represented, including the facade and the rooms.
   - Conduct research on necessary architectural and design elements to recreate the desired environment.
   - Define the scope and specific elements to include in the representation, considering the client's needs.
3. Integrate OpenGL and apply basic transformations:
   - Familiarize with the use of OpenGL and understand its basic functioning.
   - Implement basic transformations such as translation, rotation, and scaling using OpenGL capabilities.
   - Verify that the transformations are correctly applied to the environment elements, allowing for accurate representation.
4. Work with .obj models:
   - Research and select a suitable library or method for loading and rendering .obj format models.
   - Implement loading of .obj models into the development environment and ensure they can be properly visualized in the scene.
   - Perform testing and adjustments to ensure proper loading and rendering of .obj models in the project.
5. Implement lighting:
   - Study and apply lighting techniques in OpenGL, such as ambient, diffuse, and specular lighting.
   - Integrate lighting into the development environment and ensure that the environment elements are correctly illuminated.
   - Make adjustments and improvements in lighting to achieve a realistic and coherent appearance in the representation of the environment.

2.1.2. Facade and Room Selection:

During the facade selection process for this project, specific criteria were followed to create an environment suitable for the desired vacation setting. Fortunately, the client provided reference images that were extremely useful in the project's development.

The facade itself will be a two-story construction with four walls that form a solid and aesthetically pleasing structure. The main focus will be on the first floor, where key spaces such as the living room, kitchen, and pantry will be located.

Image 2.1.2.1

Image 2.1.2.2

Image 2.1.2.3

Image 2.1.2.4

The living room is a space where visitors are expected to enjoy moments of relaxation and comfort. The goal is to recreate a cozy environment with decorative elements and furniture that convey warmth and tranquility. The objective is to allow users of the application to experience the feeling of being in a spacious and welcoming living room, ideal for sharing moments with friends and family.

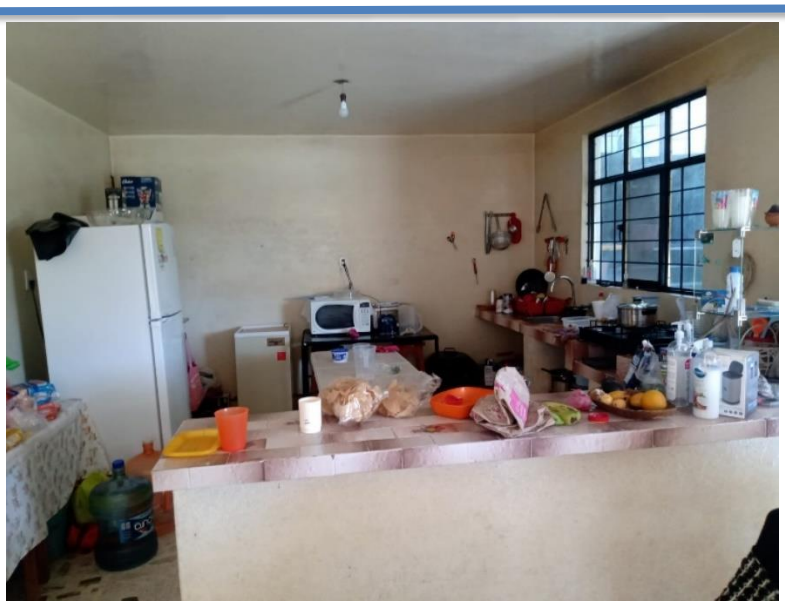


Image 2.1.2.5

On the other hand, the kitchen is a fundamental space in any facade. The aim is to represent a functional and modern kitchen with furniture and appliances that reflect a contemporary style. The idea is to convey the feeling of an organized and well-equipped space where users can imagine themselves preparing delicious meals and sharing culinary moments.

Image 2.1.2.6

It is worth noting that the selection of these two rooms was based on their relevance and prominence in the facade. The living room and kitchen are spaces that capture attention and define the overall image of the environment. By recreating these spaces in a detailed and realistic manner, the goal is to achieve a visually stunning and appealing visual representation for users.

2.1.3. Objects to Model:

First Room (Living Room):

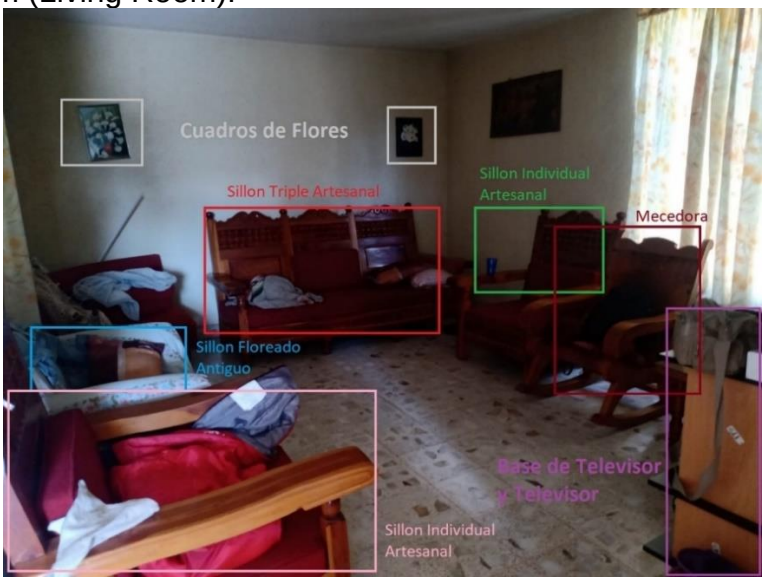


Imagen 2.1.3.1

- Antique Floral Armchair
- 2 Handcrafted Individual Armchairs
- 1 Handcrafted Group Armchair
- 1 Handcrafted Rocking Chair
- TV Stand
- Television
- 2 Flower Paintings

Second Room (Kitchen):

- Large Refrigerator

- Microwave
- Stove
- Table
- Pantry

2.1.4. Tools to Use

2.1.4.1 Maya

For object design, we have decided to use Maya as it offers a wide range of tools and techniques for 3D modeling, allowing you to create a variety of shapes and objects with precision and detail. You can use polygonal modeling tools to build and edit geometries by manipulating vertices, edges, and faces. Additionally, Maya also has subdivision modeling tools that enable the creation of smooth and organic surfaces by subdividing polygons.

Procedural modeling is another powerful feature of Maya, allowing you to create complex geometries by applying specific algorithms and rules. This provides a non-destructive and highly editable approach to modeling, giving you greater flexibility and control over your designs.

UV Mapping in Maya:

UV maps are essential for applying textures and materials to 3D models accurately. Maya provides a robust set of tools for creating and editing UV maps. You can unfold the surfaces of your models in a 2D space, where you can adjust and manipulate the UV points to ensure textures are applied correctly and without distortion.

Maya offers different methods for generating UV maps, such as automatic projections, pelt unwrapping, and cutting and stitching tools. Additionally, you can use UV editing tools to smooth transitions, adjust UV map scale, and make other precise modifications.

2.1.4.2 GIMP

GIMP, a free and open-source image editing software, can be a valuable tool for working with textures and UV maps in the context of Maya. GIMP offers a wide range of functions for adjusting and editing textures, such as cropping, resizing, color correction, applying filters, and more. This allows you to tailor textures to the specific needs of each model and optimize their appearance.

Additionally, GIMP can be used to touch up and correct UV maps generated in Maya. For example, if there are areas with distortions or overlaps in the UV mapping, GIMP's tools can be used to make manual adjustments and optimizations. Painting and retouching techniques in GIMP can also be utilized to add details or fine-tune UV maps as needed.

2.1.4.3 OpenGL

OpenGL is a low-level graphics API that provides an interface to interact with graphics hardware. It provides basic functions for rendering graphics in 2D and 3D but does not include the latest hardware-specific extensions and features that are continuously developed.

This is where libraries like GLEW and GLFW come into play. These libraries were created to facilitate access to OpenGL extensions and provide additional functionalities not present in the basic OpenGL API.

GLEW takes care of loading and managing OpenGL extensions. When using standard OpenGL functions, you have access to a basic set of features. However, hardware manufacturers and developers can add extensions to OpenGL to offer additional functionalities and hardware-specific optimizations. GLEW allows for easy loading and utilization of these extensions, without worrying about managing and ensuring compatibility with different extensions across different platforms.

GLFW, on the other hand, focuses on window creation and handling user input events. While OpenGL provides some functionality for window creation, it is limited and varies across platforms. GLFW provides a more consistent and user-friendly API for creating and managing windows across multiple platforms. It also handles input events, such as keyboard and mouse, efficiently and consistently, making it easier to develop interactive graphics applications.

2.1.4.4 Visual Studio

The choice to use Visual Studio as the IDE for this project was based on several reasons. Firstly, Visual Studio is widely recognized and used in the software development industry. It is a reliable and stable tool that has proven its effectiveness in numerous projects.

Visual Studio offers a comprehensive set of features and tools that facilitate application development. Its code editor is powerful and highly customizable, allowing developers to tailor the environment to their preferences and specific needs. Additionally, it provides integrated debugging capabilities, simplifying the process of identifying and resolving issues in the code.

2.1.4.5 C++ Language

For the development of the project, the decision was made to use the C++ language for several reasons. Firstly, the C++ language is known for its efficiency and performance, which is particularly important in projects that require fast processing and optimization of resource usage. Since our project involves modeling and rendering 3D objects, it was essential to have a language that allowed us to work efficiently with large amounts of data and perform complex calculations.

Moreover, C++ is widely used in the software development industry, especially in the field of graphics and visual computing. This means that there is a wealth of

resources, libraries, and tools available that facilitate the development of 3D applications. By opting for C++, we were able to leverage these advantages and access a wide range of functions and capabilities to drive our project.

2.1.4.6 Additional Tools

NormalMap Online is an online tool used to generate normal maps from existing textures. Normal maps are special textures used to add surface details to 3D models without significantly increasing the complexity of the geometry. By using NormalMap Online, we were able to create normal maps for our objects, improving their visual appearance by adding subtle details and textured reliefs. These normal maps were then applied to the models in our project, resulting in higher quality and realism in the rendering of the objects.

TurboSquid is an online platform that offers a vast library of ready-to-use 3D models. This platform allowed us to access a variety of free models that fit our needs in the project. By using downloaded models from TurboSquid, we were able to save time and effort in creating models from scratch, especially for simpler and common objects such as furniture or appliances. These downloaded models were then incorporated into our project, allowing us to focus on more specific and customized aspects of development.

2.2 Second Sprint (Modeling)

2.2.1. Sprint BackLog

1. Model "Antique Floral Armchair" based on reference image, ensuring the highest level of detail and similarity to reality. Work on textures to achieve an authentic appearance.
2. Model 2 "Individual Artisanal Armchairs" following the reference images to capture their unique design and peculiarities. Incorporate appropriate textures to recreate the artisanal appearance.
3. Model 1 "Artisanal Group Armchair" with attention to reference images to capture its characteristic shape and style. Add corresponding textures to achieve an authentic and realistic finish.
4. Model 1 "Artisanal Rocking Chair" based on reference images to reproduce its specific shape and details. Work on textures to recreate wood and other materials convincingly.
5. Model "TV Stand" following the reference images to obtain precise dimensions and details. Work on textures to achieve a realistic finish consistent with the material of the furniture.
6. Model "Television" based on reference images to capture its exact shape and proportions. Work on textures to simulate a screen and other realistic elements.
7. Model 2 "Flower Paintings" following the reference images to obtain the corresponding floral shapes and designs. Work on textures to represent the flowers and frames authentically.

8. Model "Large Refrigerator" according to the reference images to capture its specific shape and details. Work on textures to recreate the characteristic materials and finishes of the refrigerator.
9. Model "Microwave" following the reference images to obtain precise dimensions and features. Work on textures to achieve a metallic look and other realistic details.
10. Model "Stove" based on reference images to reproduce its design and distinctive elements. Work on textures to represent the materials and surfaces faithfully.
11. Model "Table" following the reference images to capture its shape, size, and specific details. Work on textures to recreate the wood or other materials authentically.
12. Model "Cabinet" based on reference images to obtain exact dimensions and details. Work on textures to achieve a realistic finish consistent with the material of the cabinet.
13. Model the 3D facade based on reference images of the house. Ensure to capture the architectural details and textures of the materials accurately and realistically.

2.2.2. 3D Modeling

For the 7 mentioned objects (Antique Flowered Armchair, 2 Individual Artisan Armchairs, 1 Group Artisan Armchair, 1 Artisan Rocking Chair, TV Stand, Television, and 2 Flower Paintings), a similar process was followed using the Maya tool and various functions and tools available in the software. The general process is detailed below:

1. Creating the basic shape:
   - The Polygon Primitives tool was used to create the general shape of each object.

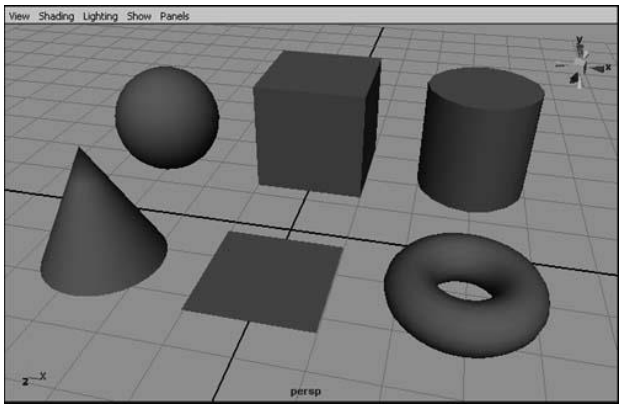


Image 2.2.2.1 (Polygon Primitives***)***

   - Vertex, edges, and faces were adjusted to achieve the desired shape.

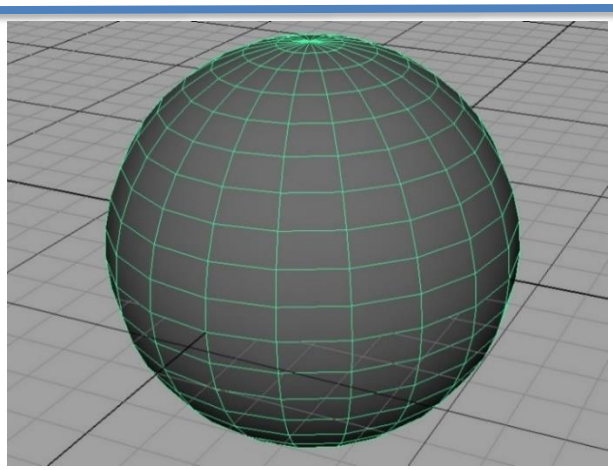


Imagen 2.2.2.2 (Vertex, edges, faces)

2. Adding details and features:
   - Tools such as Extrude, Bevel Component, and Detach were used to add details and sculpt specific features on each object.

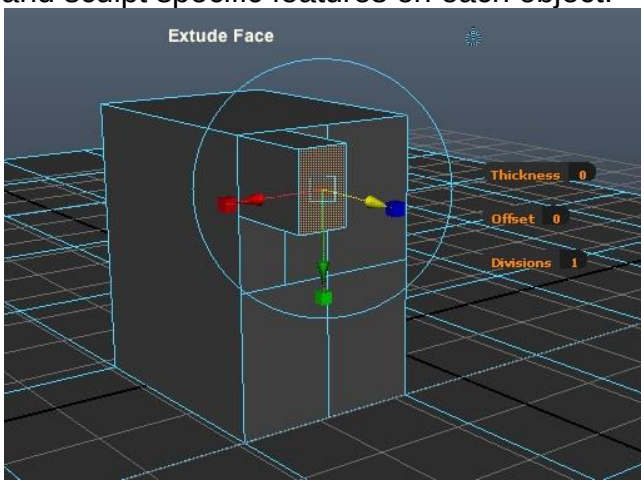


Image 2.2.2.3 (Extrude)

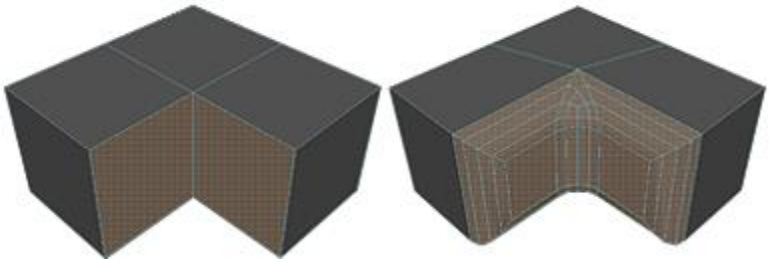


Image 2.2.2.3 (*Bevel Component*)

   - Parameters of these tools were adjusted to achieve the desired results.

3. Model optimization:

- Tools like Delete Contour History were used to maintain a clean modification history of the object and reduce file size.

4. Transformation adjustments:
   - The Freeze Transformations tool was applied to reset the object's transformations and ensure they were in their initial state.

For the last 5 objects, special considerations were made (Large Refrigerator, Microwave, Stove, Table, and Cupboard) due to their simpler nature compared to the previously mentioned objects. To optimize time and resources, it was decided to use the TurboSquid platform to download free models that met the project's needs.

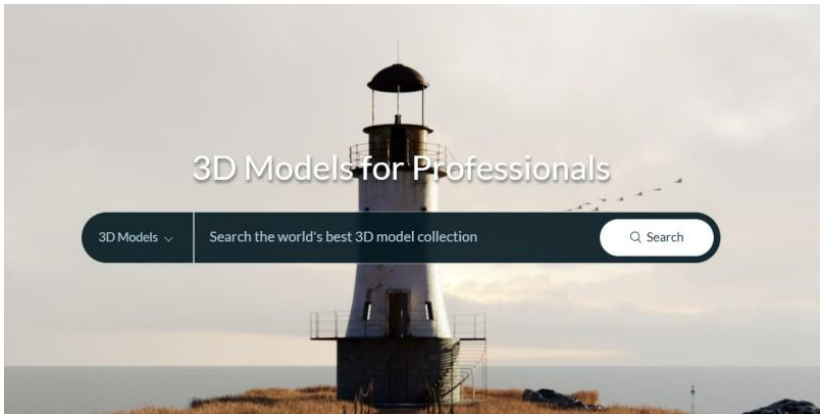


Image 2.2.2.4

TurboSquid is a renowned online platform that offers a wide variety of high-quality 3D models created by professional artists from around the world. Through this platform, an extensive library of 3D objects, including the ones mentioned above, could be accessed.

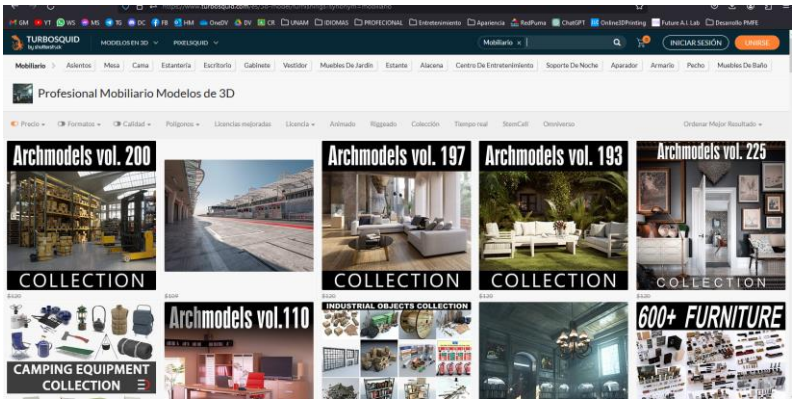


Image 2.2.2.5

The choice to use downloaded models from TurboSquid was based on the simplicity of the required objects and the availability of suitable models that met the quality and visual realism requirements for the project. This allowed saving

time in the modeling stage and focusing efforts on other areas of the project that required more attention and customization.

2.2.3 Materials and Texturing

To carry out the texturing of our models, we relied on the GIMP tool. We used GIMP to perform various tasks, such as adding alpha channels to the textures and ensuring proper transparency in certain areas of our models. This function allowed us to achieve more realistic visual effects when applying the textures in Maya.

Image 2.2.3.1 (GIMP)

Additionally, we had to make sure that the texture images were scaled to power of two sizes. This was a specific requirement of Maya, as the software only accepts textures in power of two size formats, such as 256x256, 512x512, 1024x1024, and so on. We adjusted and resized our textures using GIMP to meet this requirement and ensure proper compatibility with Maya.

Regarding materials in Maya, the Phong material played a crucial role in the visual appearance of our objects. The Phong material is known for its ability to simulate realistic light and shadow interactions on objects. It allowed us to control both specular reflections and the overall reflectivity of our models, resulting in a more authentic and appealing look.

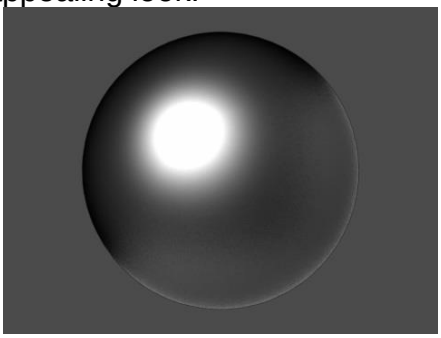


Image 2.2.3.2 (Example of Phong Material)

In addition to the base texture, to achieve a more comprehensive finish, we incorporated a specular map into the Phong material. This specular map gave us greater control over how light interacts with our models, allowing us to define areas of higher or lower reflectivity. However, to generate these specular maps, we relied on an additional tool called NormalMap Online. This online tool facilitated the creation of the necessary specular maps for our objects, enriching the visual appearance and improving how they interact with lighting.

Image 2.2.3.3 (Specular Online)

2.2.4 OBJ Export

The decision to export the models to .OBJ files and then load them from the C++ language was made for several fundamental reasons in the context of our project.

Firstly, the .OBJ file format is widely used and recognized in the 3D modeling industry. It is a simple text file format that stores the geometry and properties of the models, such as vertex coordinates, normals, and texture coordinates. This means that it is compatible with a wide range of software and rendering libraries, making it easy to integrate into our C++ based project.

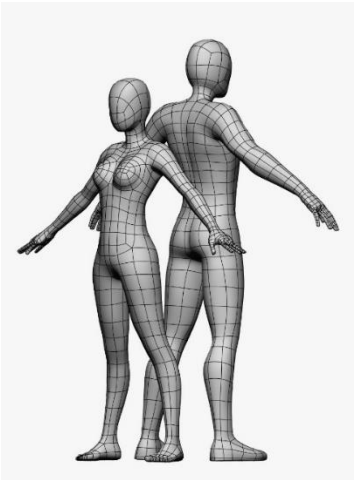


Imagen 2.2.4.1

Furthermore, .OBJ files are easy to read and analyze from C++ code. The format uses a simple and readable structure that allows us to extract the necessary information to represent the models in our program. By exporting the models to .OBJ files, we can efficiently load and process the geometric information in our C++ development environment, facilitating the manipulation and rendering of the 3D objects.

Another important advantage of .OBJ files is their ability to store additional data, such as the materials and textures associated with the models. This data is essential to achieve a realistic visual representation of the objects in our project. By exporting the models along with their materials and textures to .OBJ files, we can maintain the coherence and integrity of the data, allowing us to properly apply the visual properties to the models in our C++ program.

2.3 Third Sprint (Implementation)

2.3.1 Sprint Backlog

Sprint Objective: Implement the basic 3D graphics functionalities in the project, including the use of libraries, shaders, basic transformations, and C++ code development for loading .obj models.
Tasks:

1. Research and select the appropriate libraries for the project:
   - Evaluate options for graphics libraries, such as OpenGL or DirectX.
   - Conduct a comparative analysis of the features, documentation, and compatibility of the libraries.
   - Make an informed decision about the library to be used and document it in the sprint report.
2. Implement basic shaders:
   - Study the functioning of shaders in 3D graphics.
   - Create basic shaders, such as vertex shaders and fragment shaders, to manipulate the appearance and visual effects of objects in the scene.
   - Test the shaders in a development environment to ensure their correct functioning.
3. Integrate basic 3D transformations:
   - Research 3D transformations, such as translation, rotation, and scaling.
   - Implement the necessary functions and algorithms to apply these transformations to objects in the scene.
   - Verify that the transformations are applied correctly and that objects respond appropriately to the changes.
4. Develop C++ code for loading .obj models:
   - Study the .obj file format and its structure.
   - Implement C++ code to load and render .obj models in the scene.
   - Test the loading of .obj models in the development environment and verify that they are displayed correctly.
5. Perform testing and debugging:

- Conduct thorough testing to ensure the proper functioning of libraries, shaders, transformations, and .obj model loading.
- Identify and resolve any potential errors or issues in the code.
- Perform performance testing to evaluate the application's performance and optimize the code if necessary.

6. Document the results and progress of the sprint:
   - Record all changes, implementations, and tests conducted during the sprint in a detailed report.
   - Update the project documentation with the newly implemented elements.
   - Prepare a presentation to share the results and the progress achieved so far.

2.3.2 Libraries to be Used

1. **cmath:** This library provides common mathematical functions, such as trigonometric, exponential, and logarithmic functions, which can be used in mathematical calculations and manipulations.
2. **GLEW**: The GLEW (OpenGL Extension Wrangler Library) is used to efficiently manage OpenGL extensions. It provides functions and macros that facilitate the loading and use of OpenGL extensions in the project.
3. **GLFW:** GLFW (Graphics Library Framework) is an open-source library used for creating and managing windows, OpenGL contexts, and input handling. It provides a simple and portable application programming interface (API) for interacting with the operating system and the application's window.
4. **stb_image:** The "stb_image" library is used for loading images in the project. It allows for loading different image formats and provides functions to access the pixels and metadata of the image.
5. **GLM:** The "GLM" (OpenGL Mathematics) library is used for performing mathematical operations in the project. It provides data structures and mathematical functions specifically designed for working with 3D graphics, such as transformations, vector calculations, and matrix operations.
6. **SOIL2:** The "SOIL2" library is used for loading models in the project. It allows for loading models in different formats and provides functions to access model data, such as vertices, normals, and texture coordinates.

2.3.3 Header Files:

1. "Shader.h": This header file contains declarations and definitions related to shaders. Shaders are programs that run on the graphics card and control the appearance and behavior of rendered objects. The "Shader.h" header file provides the necessary functions and structures to load, compile, and use shaders in the project.
2. "Camera.h": This header file is used to define the camera class. The camera is responsible for controlling the view and position of the observer in the 3D scene. It provides functions and methods to configure and manipulate the camera's position, orientation, and projection.

3. "Model.h": This header file defines the Model class, which is used to load and render 3D models in the project. It provides functions and methods to load model files in different formats, as well as manipulate and render the models in the scene.
4. "Texture.h": This header file is used to define the Texture class, which manages the textures used in the project. It provides functions and methods to load, assign, and apply textures to rendered objects.

Each of these header files plays a fundamental role in the project by providing the necessary structures, functions, and methods for handling shaders, cameras, models, textures, and animations. These classes and functions are essential for the implementation and proper functioning of the project in terms of rendering and displaying objects in a 3D environment.

2.3.4 Shaders

1. **lightingShader:**
   - Description: Represents a Shader object used in the modeling project.
   - Value: Contains the Shader used for lighting in the project.
   - Type: Shader object.
2. **lampShader:**
   - Description: Represents a Shader object used in the modeling project.
   - Value: Contains the Shader used to render a lamp in the project.
   - Type: Shader object..
3. **SkyBoxshader:**
   - Description: Represents a Shader object used in the modeling project.
   - Value: Contains the Shader used to render the skybox in the project.
   - Type: Shader object.

4. **animShader:**
   - Description: Represents a Shader object used in the modeling project.
   - Value: Contains the Shader used for animations in the project.
   - Type: Shader object..
5. **Anim:**
   - Description: Represents a Shader object used in the modeling project.
   - Value: Contains the Shader used for specific animations in the project.
   - Type: Shader object.

2.3.5 Functions

1. **doMovement() Function:**
   - Description: This function is responsible for processing and altering the values used in basic 3D modeling transformations to perform animations.
   - Details:
     - It uses mathematical calculations to update the translation, rotation, and scale values of objects in the 3D scene.
     - The changes made in this function achieve animation effects such as smooth movement, continuous rotation, or changes in object scale.

- Ensure that the calculations are performed efficiently to ensure optimal performance in the animation.

2. **keyCallback() Function:**
   - Description: This function is used to detect when a key is pressed in the virtual environment.
   - Details:
     - It captures keyboard events and performs corresponding actions based on the pressed keys.
     - It can be used to activate or deactivate certain animations, change the display mode, toggle between different cameras, or perform any other keyboard-related action.
     - Ensure that the actions are intuitive and properly documented for user interaction.

3. **mouseCallback() Function:**
   - Description: This function allows the use of the mouse within the virtual environment to perform interactions or manipulate objects.
   - Details:
     - It captures mouse events such as movements, clicks, or scrolls.
     - It uses the mouse data obtained to perform specific actions in the 3D scene, such as selecting objects, rotating the camera, interacting with interactive elements, or any other mouse-related action.
     - Ensure that the interactions are smooth and respond appropriately to mouse movements.

These functions are important elements for interactivity and animation in the project.

2.3.6 Lighting

The project benefits from the implementation of lighting shaders to achieve realistic effects and enhance the visual quality of the 3D models. By using lighting shaders, three types of lighting can be simulated: spotlight, point light, and directional light. Each of these light types has specific characteristics that allow the creation of different environments and visual effects in the scene.

In the case of this particular project, the point light was chosen to be used more extensively to realistically illuminate the 3D models. The point light is an omnidirectional light source that emits light in all directions from a specific point in space. This allows simulating light sources such as the sun or spotlights located in strategic positions within the scene.

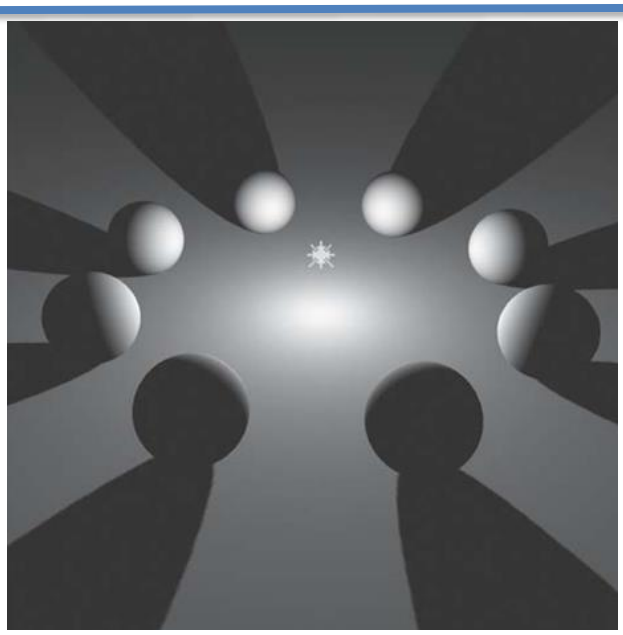


Image 2.3.6.1

The point light offers several benefits for the project, such as:

1. Realism: By positioning the point lights in strategic locations, it is possible to simulate how light reflects and projects onto objects in the scene, creating shadows and highlighting details.
2. Flexibility: Point lights can be adjusted in terms of intensity, color, and position to fit the specific lighting needs of each object or scene.
3. Interaction with materials: Point lights interact with the materials applied to the 3D models, allowing for highlighting features such as shine, reflection, and textures.

By predominantly using point lights in the project, more detailed and realistic lighting is achieved compared to an exclusive focus on spotlights or directional lights. This helps to highlight object details and create a visually appealing experience for users.

2.3.7 Animations

In the project, animations were implemented using basic 3D modeling transformations, including translation, rotation, and scaling. These transformations were applied to the 3D models in the scene to achieve movement effects and changes in their appearance.
To control the animations, variables were used that were mathematically modified over the course of the animation timeline. These variables were adjusted and updated based on the desired animation state to be recreated.

4. Translation: Translation was used to move the models in different directions within the scene. The variables controlling the position of the models were

updated in each frame of the animation, allowing the objects to move smoothly over time.

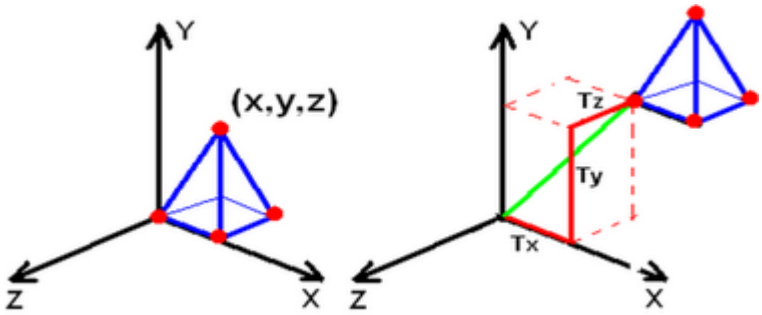


Image 2.3.7.1 (Translation)

5. Rotation: Rotation was applied to rotate the models around a specific axis. The variables controlling the rotation angle were modified in each frame of the animation, allowing the objects to rotate in a fluid and realistic manner.

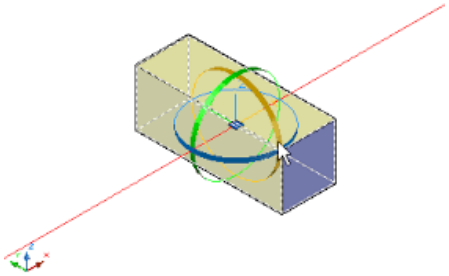


Imagen 2.3.7.1 (Rotation)

6. Scaling: Scaling was used to change the size of the models during the animation. The variables related to the scale factor were adjusted based on the progress of the animation, allowing the objects to expand or contract in a controlled manner.

2.3.8 Context

Two brothers decided to spend a relaxing vacation in a vacation home located in Cuautla. During their stay, each of them enjoyed different activities in different areas of the house.

Brother 1:

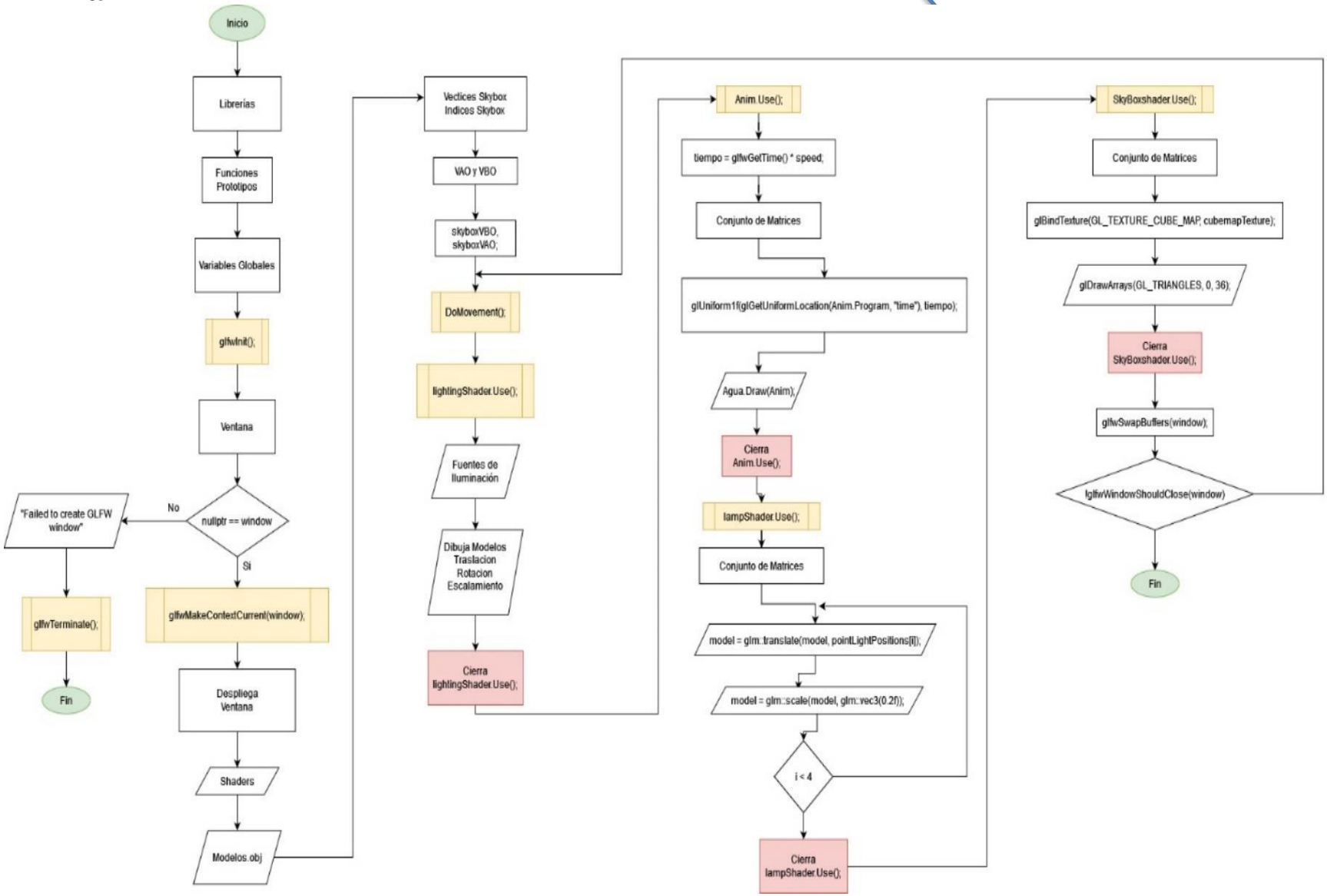
One of the brothers decided to have a pleasant time in the living room of the vacation home. In this cozy and comfortable environment, he was able to enjoy moments of rest, relaxation, and entertainment. Perhaps he sat on a comfortable old floral armchair while reading a book, listening to music, or watching his favorite TV show. The living room offered a calm and conducive atmosphere to enjoy a time of relaxation and disconnection.

Brother 2:
On the other hand, the other brother decided to take advantage of the spacious patio of the house to fly a drone. With exciting moments of fun and entertainment, this brother immersed himself in the world of technology and aerial exploration. Skillfully controlling the drone, he performed smooth and thrilling flights, observing the surroundings of the vacation home in Cuautla from above. The patio provided the necessary space for the brother to enjoy his passion for drones and experiment with new maneuvers and perspectives.

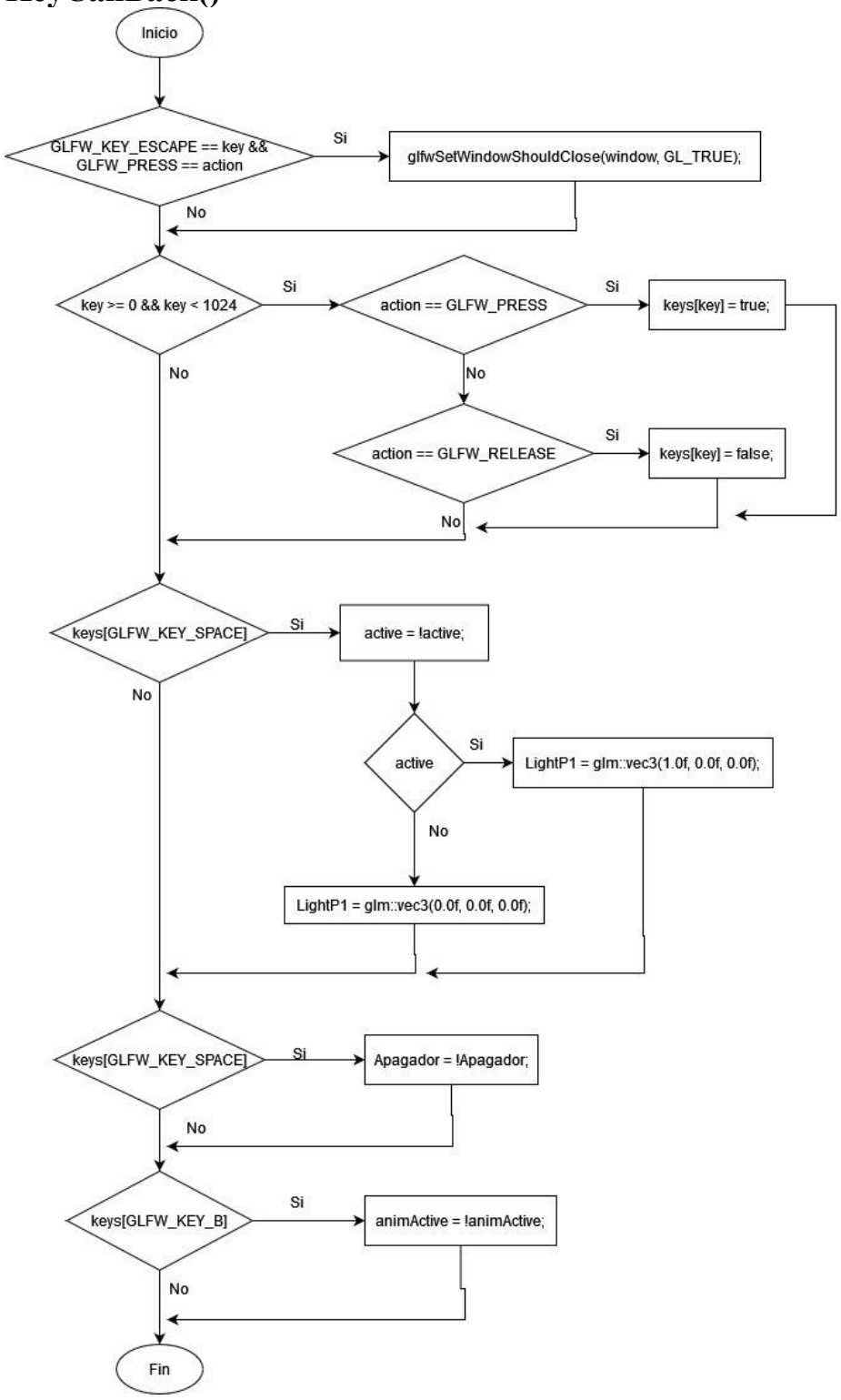
Together, these two brothers found a balance between relaxation and adventure during their vacation in the vacation home in Cuautla. Each of them was able to enjoy activities they were passionate about in different areas of the property, creating unforgettable memories and making the most of their time together in a peaceful and pleasant environment.

2.3.9 Flowcharts

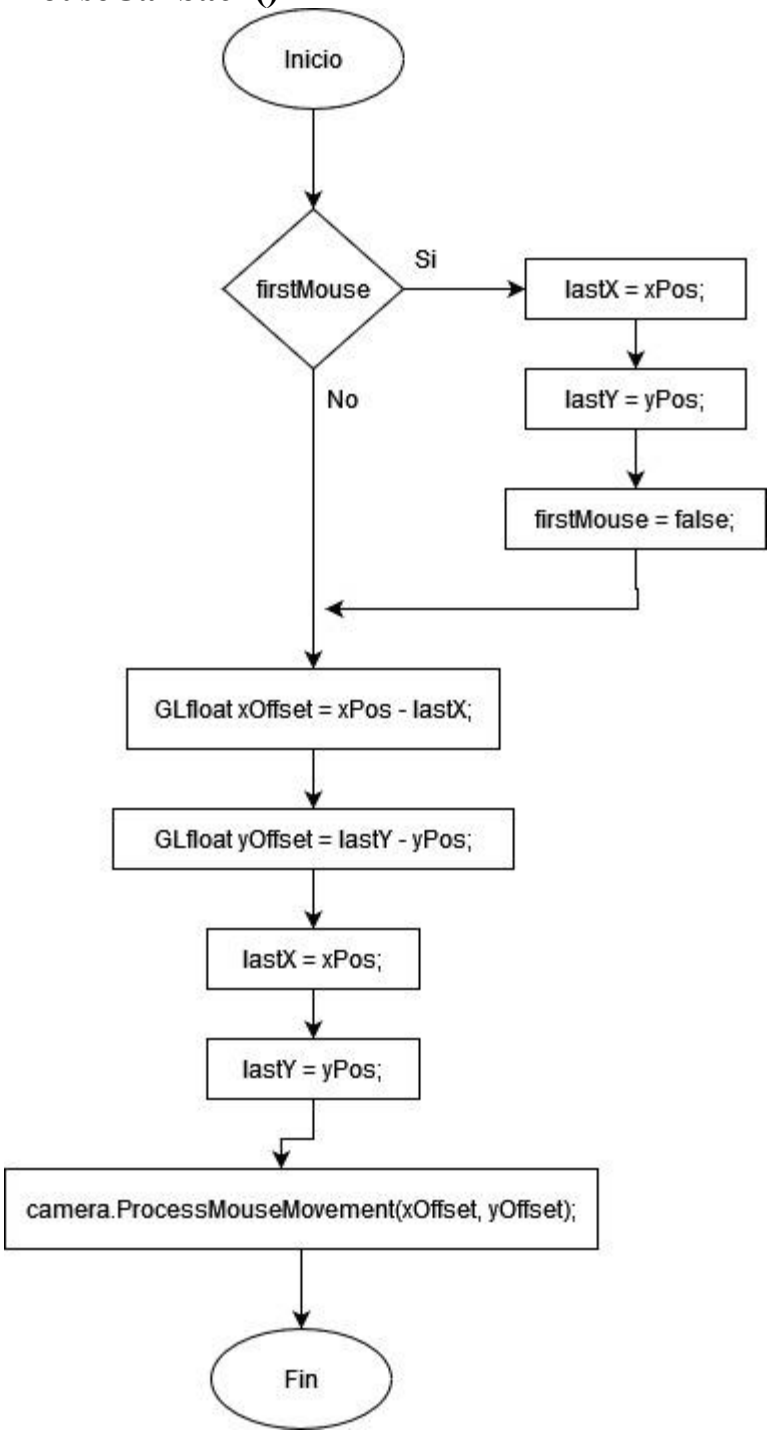
Main:



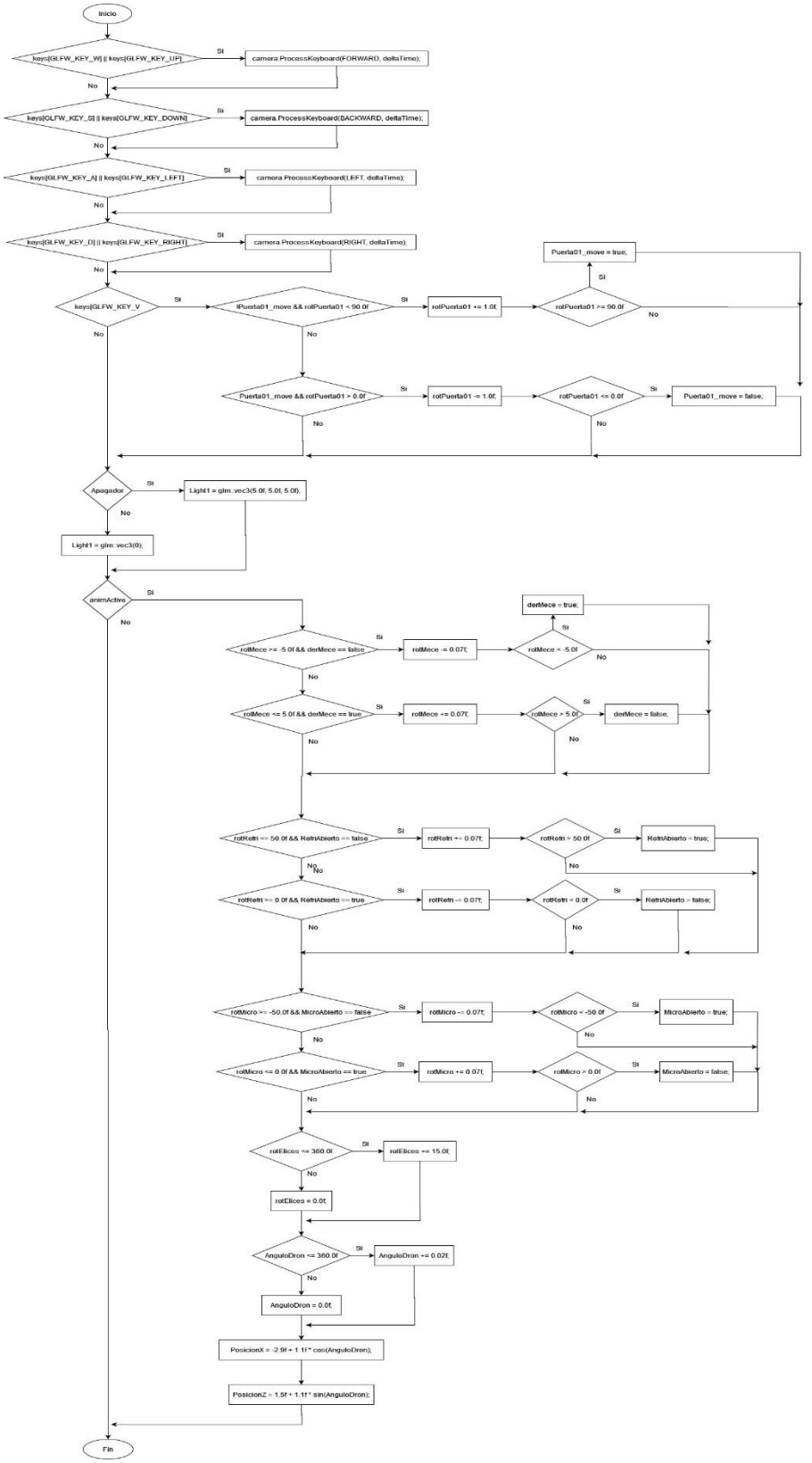
KeyCallBack()



MouseCallback()



DoMovement()


Inicio
keys[GLFW_KEY_W] || keys[GLFW_KEY_UP]
Si
camera.ProcessKeyboard(FORWARD, deltaTime);
No
keys[GLFW_KEY_S] || keys[GLFW_KEY_DOWN]
Si
camera.ProcessKeyboard(BACKWARD, deltaTime);
No
keys[GLFW_KEY_A] || keys[GLFW_KEY_LEFT]
Si
camera.ProcessKeyboard(LEFT, deltaTime);
No
keys[GLFW_KEY_D] || keys[GLFW_KEY_RIGHT]
Si
camera.ProcessKeyboard(RIGHT, deltaTime);
No
keys[GLFW_KEY_V
Si
!Puerta01_move && rotPuerta01 < 90.0f
Si
rotPuerta01 += 1.0f;
rotPuerta01 >= 90.0f
Si
Puerta01_move = true;
No
No
Puerta01_move && rotPuerta01 > 0.0f
Si
rotPuerta01 -= 1.0f;
rotPuerta01 <= 0.0f
Si
Puerta01_move = false;
No
No
No
Apagador
Si
Light1 = glm::vec3(5.0f, 5.0f, 5.0f);
No
Light1 = glm::vec3(0);
animActiva
Si
No
rotMece >= -5.0f && derMece == false
Si
rotMece -= 0.07f;
rotMece < -5.0f
Si
derMece = true;
No
No
rotMece <= 5.0f && derMece == true
Si
rotMece += 0.07f;
rotMece > 5.0f
Si
derMece = false;
No
No
rotRetri <= 50.0f && RetriAbierto == false
Si
rotRetri += 0.07f;
rotRetri > 50.0f
Si
RetriAbierto = true;
No
No
rotRetri >= 0.0f && RetriAbierto == true
Si
rotRetri -= 0.07f;
rotRetri < 0.0f
Si
RetriAbierto = false;
No
No
rotMicro >= -50.0f && MicroAbierto == false
Si
rotMicro -= 0.07f;
rotMicro < -50.0f
Si
MicroAbierto = true;
No
No
rotMicro <= 0.0f && MicroAbierto == true
Si
rotMicro += 0.07f;
rotMicro > 0.0f
Si
MicroAbierto = false;
No
No
rotElices <= 360.0f
Si
rotElices += 15.0f;
No
rotElices = 0.0f;
AnguloDron <= 360.0f
Si
AnguloDron += 0.02f;
No
AnguloDron = 0.0f;
PosicionX = -2.9f + 1.1f * cos(AnguloDron);
PosicionZ = 1.5f + 1.1f * sin(AnguloDron);
Fin

2.3.10 Variable Dictionary

Variable	Type	Description
WIDTH	GLuint	Width of the window, with a constant value of 800
HEIGHT	GLuint	Height of the window, with a constant value of 600
SCREEN_WIDTH	int	Screen width
SCREEN_HEIGHT	int	Screen height
camera	Camera	Object of the Camera class representing the camera and storing its initial coordinates
lastX	GLfloat	X coordinate of the last point on the window axis divided by 2
lastY	GLfloat	Y coordinate of the last point on the window axis divided by 2
animActive	bool	Indicator to activate or deactivate animations
rotMece	float	Rotation angle of the rocking chair
derMece	bool	Indicator if the rocking chair has reached its limit
rotPuerta01	float	Rotation angle to open and close doors
Apagador	bool	Indicator to turn the light on or off
rotElices	float	Rotation angle of the drone's propellers
PosicionX	float	X position of the drone
PosicionZ	float	Z position of the drone
AnguloDron	float	Angle of the drone's position to calculate X and Z
rotRefri	float	Rotation angle of the refrigerator door
RefriAbierto	bool	Indicator to know if the refrigerator door is open or closed
rotMicro	float	Rotation angle of the microwave door
MicroAbierto	bool	Indicator to know if the microwave door is open or closed

3. Results Obtained

3.1 Facade

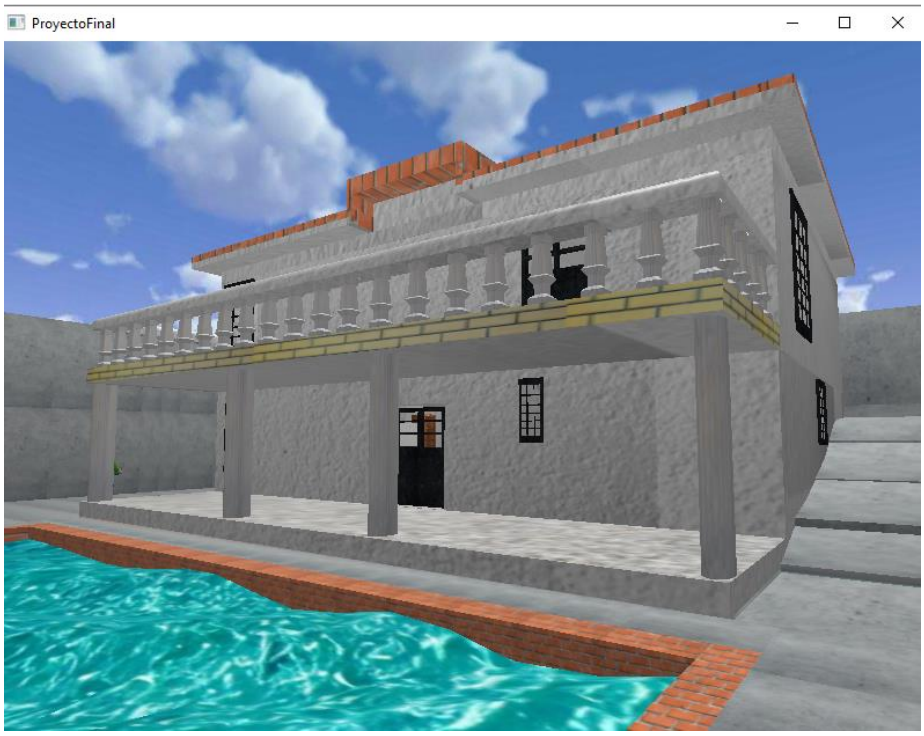


Image 3.1.1

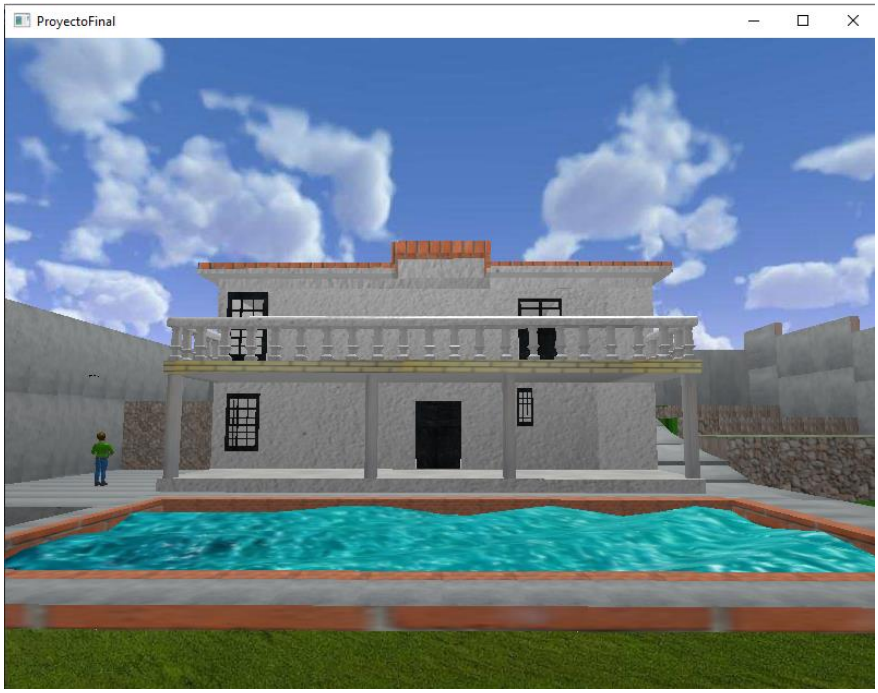


Image 3.1.2

3.2 Rooms

3.2.1 Living Room

Image 3.2.1.1

Image 3.2.1.2

3.2.2 Kitchen

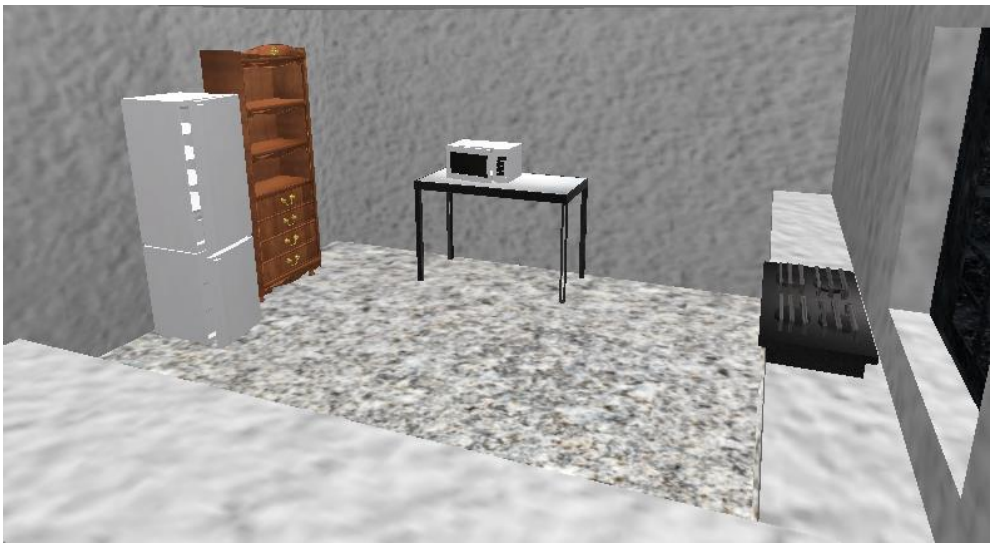


Image 3.2.2.1

3.3 Animations

Image 3.3.1

Image 3.3.2

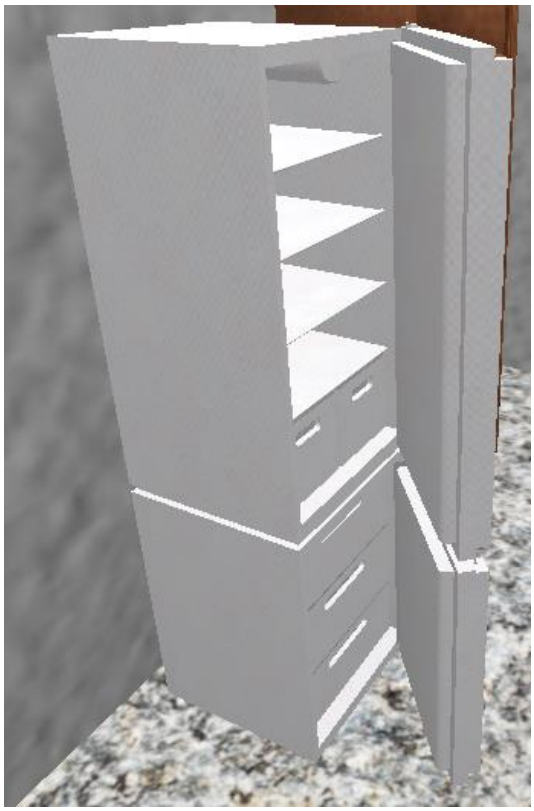


Image 3.3.3

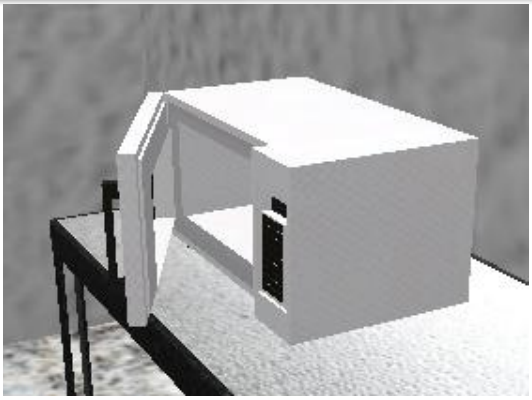


Image 3.3.3

4. Lessons Learned and Conclusion:

4.1 Learning

During the project development using the SCRUM methodology, various learnings were acquired that contributed to the growth and development of the team. The main learnings obtained are summarized below:

1. Design and image editing tools: During the project, there was an opportunity to work with different design and image editing tools such as GIMP. Through their use, manipulation and enhancement of model textures were learned, including adding alpha channels and scaling images to powers of 2 for proper implementation in Maya. This allowed achieving high-quality visual results and aesthetic coherence in the project.
2. Use of Maya: Knowledge about the Maya software, a powerful 3D modeling and animation tool, was acquired. Through its use, creating and manipulating three-dimensional models, applying textures, performing basic transformations (translation, rotation, scaling), and generating animations using the available functions and tools in Maya were learned. This experience in Maya was crucial to achieve the desired visual aspect in the project.
3. Implementation of OpenGL in the C language: The implementation of OpenGL in the C language was explored to achieve interaction and real-time rendering of the models in the project. Through this implementation, calling and managing models along with their associated information such as textures, coordinates, and specular components were learned. This allowed the visualization and manipulation of models in a virtual environment, providing an immersive and realistic experience. In summary, the project under the SCRUM methodology provided an opportunity to learn and develop skills in handling various design and image editing tools, as well as the use of specialized software like Maya and the implementation of OpenGL in the C language. These learnings were fundamental to achieve the project objectives and laid the foundation for future projects requiring a similar approach in the field of 3D design and development.

.

4.2 Conclusions

In conclusion, the project developed under the SCRUM methodology has been an enriching process that has allowed achieving the established objectives and obtaining satisfactory results. Throughout this project, various tools and techniques of 3D design and development have been employed, contributing to the growth and learning of the team.

The use of tools like GIMP for texture manipulation, Maya for 3D modeling and animation, and OpenGL in the C language for implementation and rendering of models has proven to be an effective and robust approach to achieve visually appealing and realistic results. These tools have allowed smooth interaction with the models, providing an immersive and high-quality experience.

Furthermore, teamwork and the adoption of the SCRUM methodology have been key elements for the success of the project. Constant collaboration, effective communication, and flexibility to adapt to changes have allowed maintaining an efficient workflow and an agile response to the project requirements and needs.

In terms of learning, this project has provided extensive experience in the use of design and image editing tools, as well as handling specialized software for 3D modeling and animation. These acquired knowledge are valuable and can be applied to future projects related to 3D design and development.

In summary, this project has been an opportunity to acquire new knowledge, strengthen skills, and achieve successful results in the field of 3D design and development. The combination of agile methodologies, specialized tools, and teamwork has been the key to achieving the set objectives and creating a final product of high quality.

5. References

3. Department of Engineering. (2013). Computer Graphics with OpenGL. Retrieved from https://ingenieriayeducacion.files.wordpress.com/2013/12/graficosporcomputadorayopengl.pdf
4. Woo, M., Neider, J., Davis, T., & Shreiner, D. (1999). OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2. Addison-Wesley Longman Publishing Co., Inc.