317097742 Grupo 3

Manual técnico
Centro comercial - Aura

índice

Objetivos

El objetivo principal de este proyecto es desarrollar un entorno virtual interactivo que reproduce una plaza comercial utilizando Maya como el software para el modelado y detalle de la estructura arquitectónica y objetos internos de la tienda. Así mismo, involucrando OpenGL para la implementación de funcionalidades interactivas, animaciones dinámicas y sistemas de iluminación.

El proyecto busca ofrecer una experiencia inmersiva que permita a los usuarios explorar de manera virtual el espacio. El presente manual ha sido diseñado para proporcionar una comprensión más explorada de cada interacción, elemento y proceso involucrado en la creación de esta animación.

Alcance del proyecto

El objetivo del proyecto es desarrollar un entorno virtual interactivo que simula la plaza comercial, proporcionando una experiencia visual, funcional, realista y detallada; modelada en Maya y animada en OpenGL.

Se presentan los siguientes puntos de alcance en el proyecto:

- **Modelado 3D:**
  - Modelado detallado de la fachada exterior de la tienda.
  - Modelado completo del interior de las tiendas, incluyendo muebles fijos como mesas de exposición y estanterías.
  - Creación de réplicas digitales de productos y espacios.

- **Texturizado y Materiales:**
  - Aplicación de texturas y materiales para todos los objetos y estructuras internas y externas, basados en fotografías de la tienda real.

- **Iluminación:**
  - Implementación de un sistema de iluminación que refleje tanto las condiciones de iluminación natural como artificial dentro del entorno.

- **Animaciones:**
  - Animaciones básicas de objetos interactivos, como la apertura y cierre de puertas, movimientos de productos y otros elementos animados. También contemplando animaciones complejas aplicadas en los objetos de la tienda.

- **Interactividad:**
  - Desarrollo de controles básicos para la navegación dentro del entorno virtual utilizando teclado y mouse.
  - Interactividad con ciertos objetos dentro del entorno, como operar dispositivos electrónicos y abrir puertas.

- **Optimización y Rendimiento:**
    - Asegura que el entorno virtual sea fluido y funcione eficientemente en computadoras con especificaciones medianas.

Limitantes

Durante el desarrollo nos enfrentamos a una serie de complicaciones técnicas que requerían soluciones creativas y adaptativas. Una de las principales dificultades surgió al intentar importar la animación del cristal directamente desde Maya. En este proceso, se encontró un problema recurrente donde al importar un objeto a OpenGL, se agregaban texturas que no corresponden, generando un bug en la visualización.

La solución a este inconveniente implicó una modificación del fragment shader, la activación del canal alfa y una nueva variable para permitir la transparencia. Una vez aplicados estos ajustes, se probó nuevamente importando un modelo sólido a OpenGL y aplicando la transparencia, lo que finalmente permitió visualizar los modelos transparentes correctamente.

Otra limitación significativa se relaciona con las mala optimización de los modelos ya que, esto retrasaba el arranque del proyecto, la computadora utilizada para el proyecto cuenta con un procesador Ryzen 5 de la serie 3500, 8 GB de RAM, 512 GB de SSD gráficos integrados, el equipo mostró dificultades en ocasiones al cargar las visualizaciones en Maya, especialmente en lo que respecta a la fachada del proyecto. Además, se experimentaron retrasos al cargar el proyecto en Visual Studio. Aunque una vez optimizados los modelos, texturas y mapas uv mejoró mucho el comportamiento.

Metodología de software

Para el desarrollo del proyecto Apple Store, se optó por emplear una metodología de Desarrollo Incremental. Esta decisión se basó en la necesidad de tener una buena gestión de la complejidad del modelado y la programación en OpenGL, permitiendo la flexibilidad necesaria para adaptarse a cambios y refinamientos a lo largo del proceso de desarrollo. Esta elección fue crucial para integrar la retroalimentación continua del profesor y ajustar el proyecto de acuerdo con las necesidades y expectativas de la propuesta inicial.

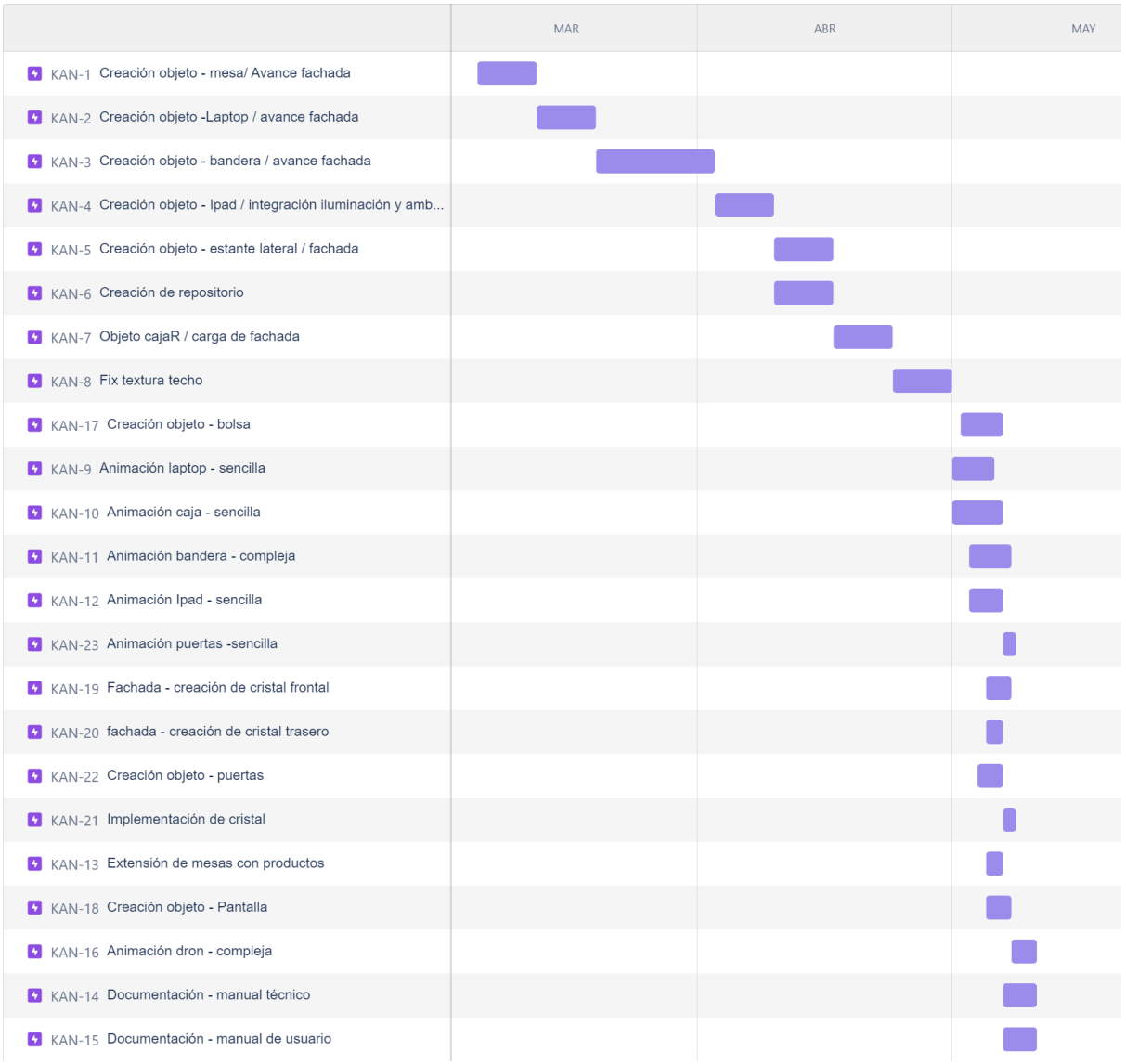
- **Organización en fases:**
  Se dividió en varias fases incrementales, cada una con objetivos específicos y entregables claros. Esta estructura permitió concentrarse en completar segmentos del entorno en etapas manejables, garantizando un progreso constante y visible.

- **Entregas semanales:**
  Se estableció un ciclo de entregas semanales, donde cada semana se presentaban avances significativos del proyecto. Estas presentaciones incluían desde nuevos modelos 3D hasta mejoras en la animación y la interactividad. Esta regularidad en

las entregas no solo ayudó a mantener el proyecto en curso bajo un escrutinio constante sino también permitió la implementación oportuna de cambios sugeridos por los revisores.

- **Mejora continua:**
  Con cada entrega semanal se evaluaban los resultados técnicos y el feedback recibido por el profesor. Esto permitió realizar ajustes técnicos que mejoraron la eficiencia y la calidad del desarrollo.

Diagrama de Gantt



Diagramas de flujo

Las animaciones sencillas fueron definidas dentro de la función DoMovement, ya que se encuentra ejecutada dentro del while y esto ayuda a tener siempre a la escucha del comportamiento de las variables para activar o desactivar x animación.

Diagrama de flujo - puertas

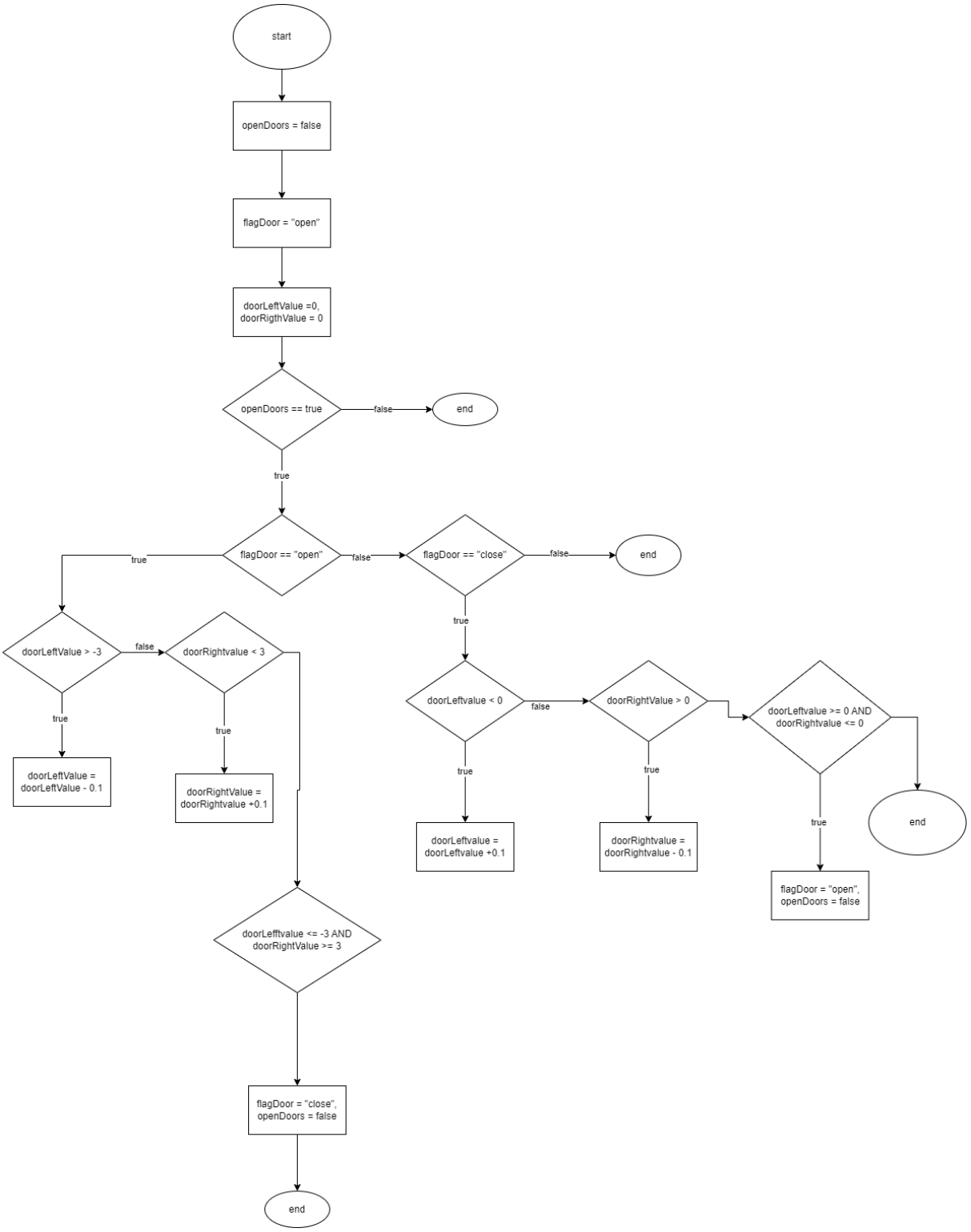


Diagrama de flujo - ipad

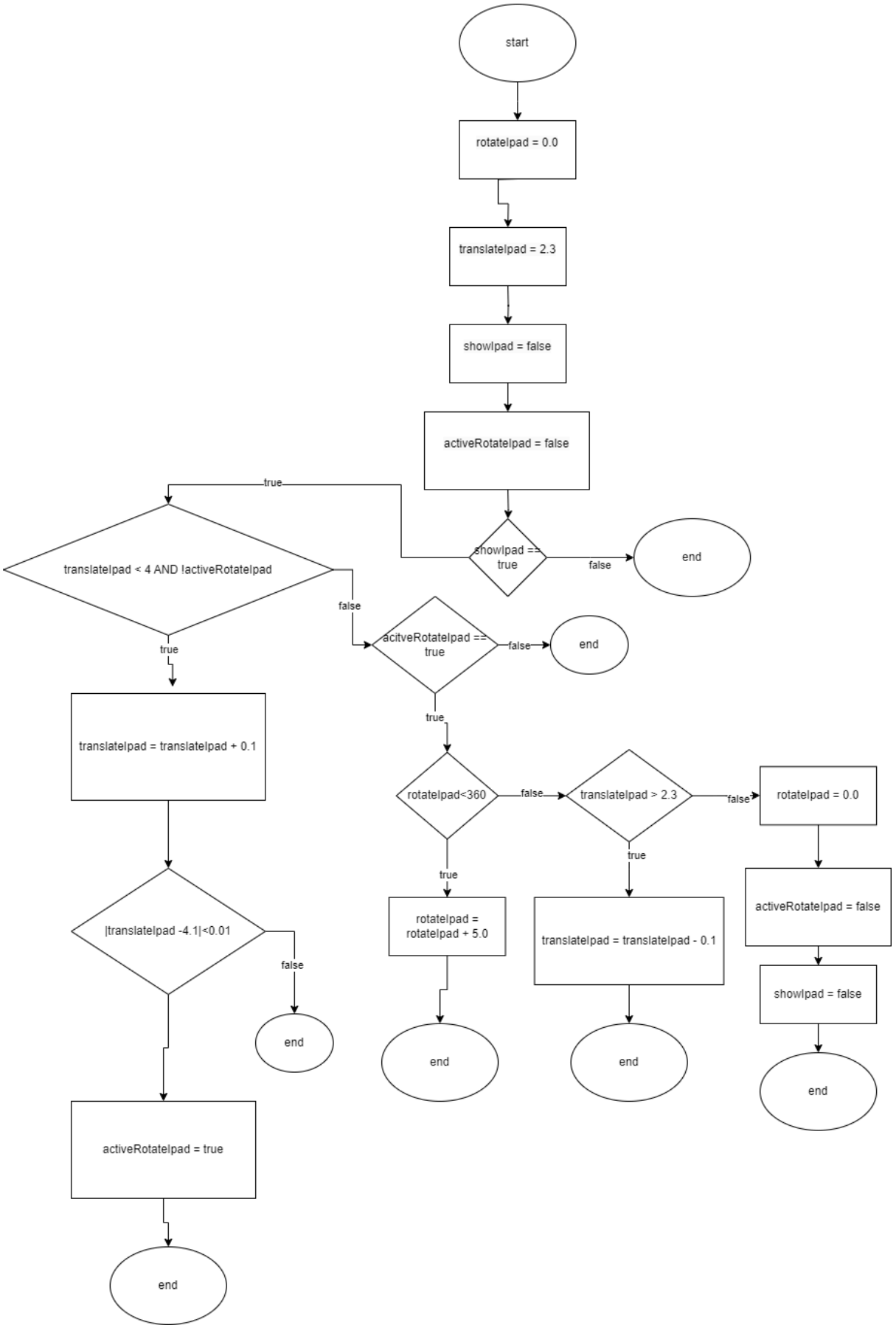


Diagrama de flujo - mac

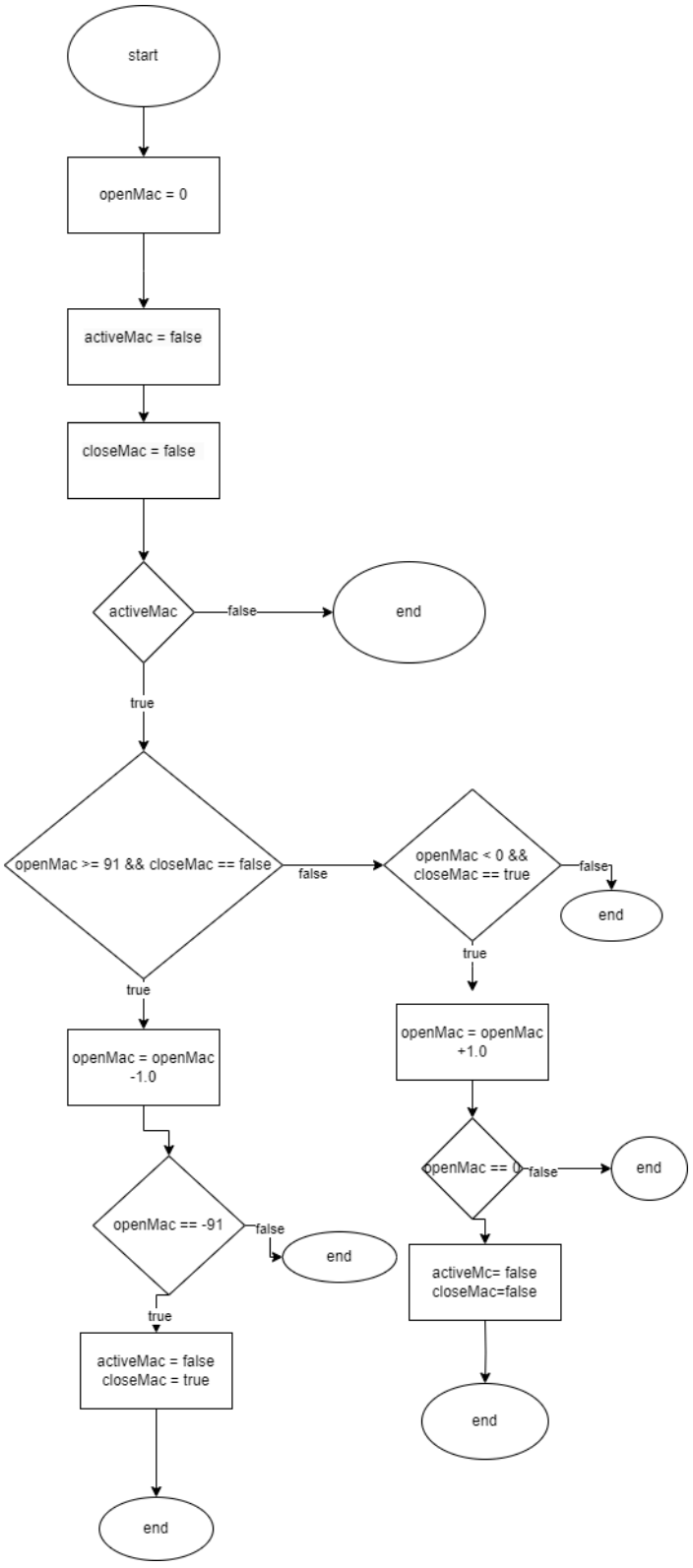
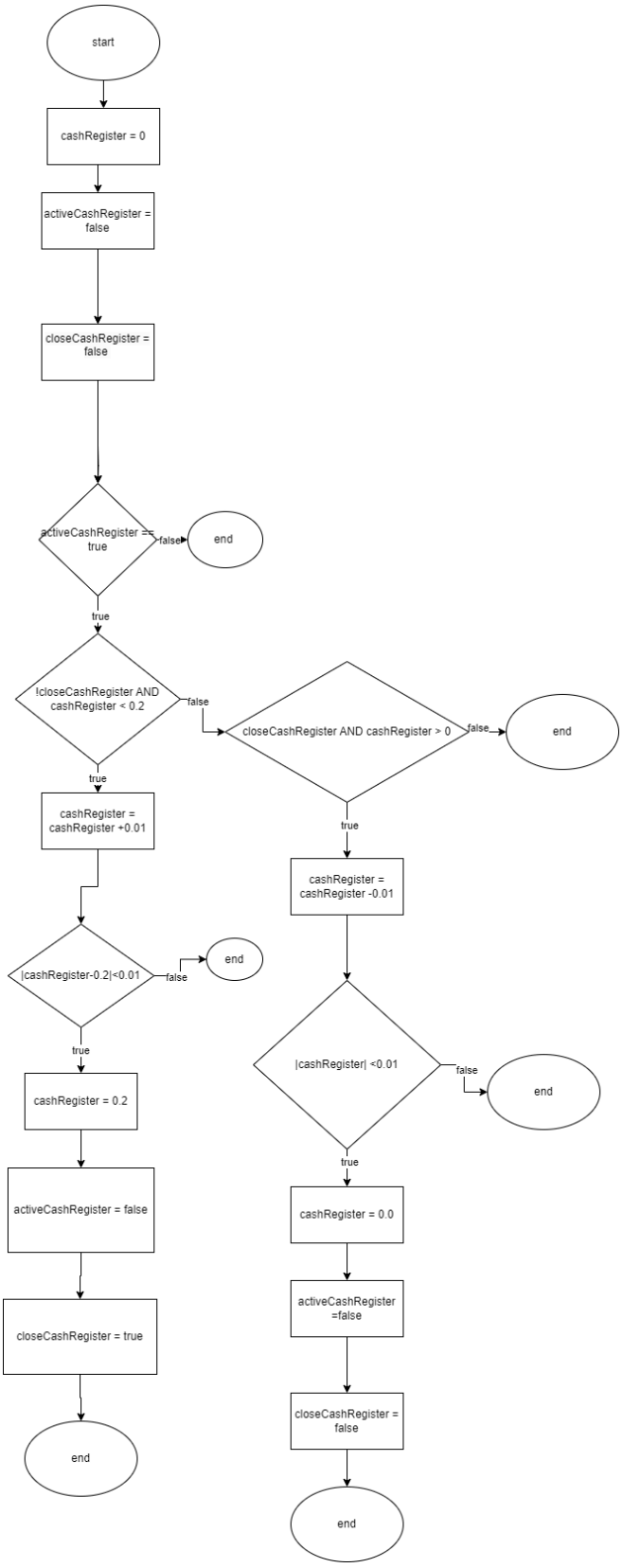


Diagrama de flujo - caja registradora



Documentación de código

El archivo cpp está seccionado con comentarios para definir cada uno de los elementos que definen el proyecto.

Definición de variables

Variable	Definición
WIDTH	Define el ancho de la pantalla
HEIGHT	Define el alto de la pantalla
SCREEN_HEIGHT	Dimensiones actuales de la pantalla
SCREEN_WIDTH	Dimensiones actuales de la pantalla
camara	Objeto de tipo Camera que gestiona la vista y posición de la cámara
lastX, LastY	última posición registrada del mouse
keys	Arreglo de booleanos para el manejo de estados de las teclas
firstMouse	Indicador para determinar si el mouse se ha movido por primera vez
lightPos	Posición de la fuente de luz
active	Estado booleano para tener los puntos de luz siempre encendidos
spotLightPosicion	posición de la luz de tipo spot
spotLightDireccion	Dirección de la luz tipo spot
tiempo	Toma el tiempo proveniente del procesador
openMac	controla la activación para abrir la computadora
activeMac	Referente si está activa la variable
closeMac	Referente si está activa la acción de cerrar mac
cashRegister	valor de incremento o decremento para animación de caja R
activeCashRegister	variable de control cuando se tiene que abrir el cajón
closeCashRegister	variable de control cuando se tiene que cerrar el cajón
rotateIpad	controla la rotación del ipad

translateIpad	controla la traslación del ipad
showIpad	control para activar la animación de mostrar
activeRotateIpad	control para activar la rotación
openDoors	variable de control para activar o desactivar la animación
flagDoor	bandera de control para señalar si abren o cierran
doorLeftValue	desplazamiento de la puerta izquierda
doorRightValue	desplazamiento de la puerta derecha
activeDron	booleano que se encarga de la activación de la animación
rotateDron	rotación del dron al llegar a una esquina
rotateHeli	rotación de las hélices
elevateDron	Incremento/decremento para elevar el dron
desZ	desplazamiento en z
desX	desplazamiento en x
r1,r2,r3,r4,r5	booleanos que ayudan a cambiar de estados
lightDron	vector que contiene el color de la luz correspondiente al dron
lightColorDron	vector que contiene los datos para la spotLight
pointLightPositions	arreglo que contiene las posiciones de los puntos de luz
deltaTime	control de tiempo entre frames

Definición de funciones

Función	Definición
KeyCallback	Maneja las entradas de teclado
MouseCallback	Maneja las entradas del mouse
DoMovement	Actualiza la posición de la cámara y maneja las animaciones basadas en entrada de usuario.
FlyDron	Controla el vuelo y la animación del dron
CircuitDron	Controla los estados del dron para poder realizar el recorrido
main	Función principal que inicializa GLFW y GLEW, configura la ventana y los shaders, y contiene la carga de luces, modelos y fachada.

Animación compleja - bandera

Se definió un shader para animar una bandera añadiendo un efecto de ondulación a los vértices de la malla de la bandera.

El shader modifica la coordenada z de cada vértice de la malla de la bandera basándose en una función seno, que cambia con el tiempo y la posición x del vértice. Esto crea un efecto de onda que hace que la bandera parezca ondear. En el código C++, se actualiza la variable time del shader en cada cuadro para animar continuamente la bandera. Además, se configuran las transformaciones para posicionar y orientar correctamente la bandera en la escena.

anim.vs

```
#version 330 core
layout (location = 0) in vec3 aPos;
layout (location = 1) in vec3 aNormal;
layout (location = 2) in vec2 aTexCoords;

out vec2 TexCoords;

uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;
uniform float time;

const float amplitude = 0.1;
const float frequency = 1.7;
const float PI = 3.14159265359;

void main()
{
    float wave = amplitude * sin(PI * aPos.x * frequency + time);
    vec3 newPosition = vec3(aPos.x, aPos.y, aPos.z  + wave );
    gl_Position = projection * view * model * vec4(newPosition, 1.0);
    TexCoords = aTexCoords;
}

```

anim.frag

```
#version 330 core
out vec4 FragColor;
in vec2 TexCoords;
uniform sampler2D texture1;

void main()
{

  vec4 texColor= texture(texture1,TexCoords);
    if(texColor.a < 0.1)
        discard;
    FragColor = texColor;

}
```

Animación compleja - dron

Para realizar esta animación se crearon dos funciones aparte las cuales se encargar en conjunto de animar el dron:

- funcion FlyDron

  Esta función maneja la activación y el movimiento vertical del dron, además de integrar cambios en la iluminación basados en el estado de activeDron (un booleano que determina si el dron está activo o no).

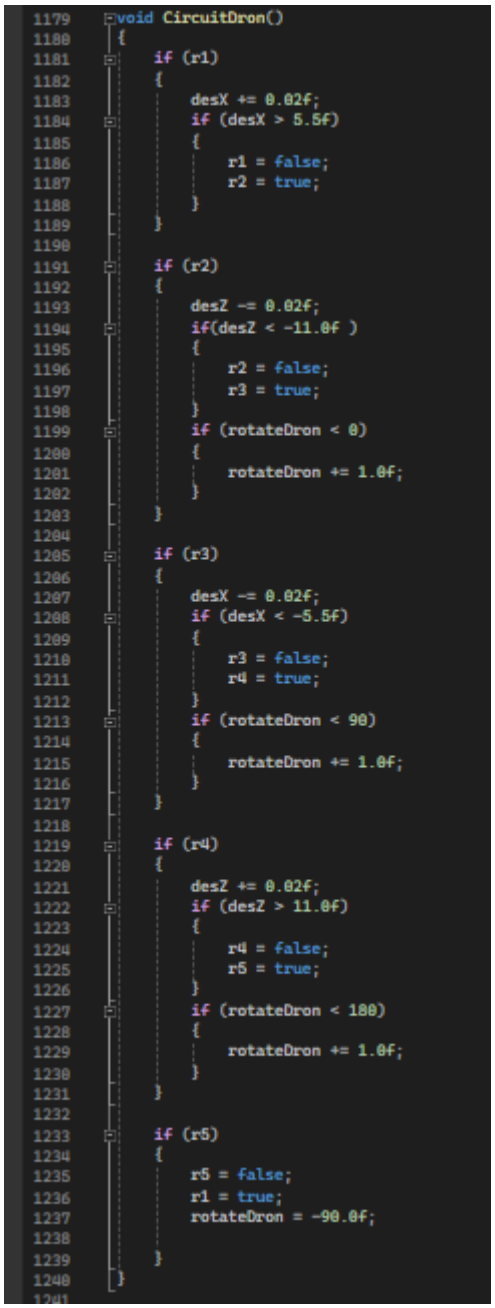
```
void FlyDron()
{
    if (activeDron)
    {
        lightDron = glm::vec3(0.0f, 1.0f, 1.0f);
        rotateHeli += 10.0f;
        if (elevateDron < 5)
        {
            elevateDron += 0.01f;
            if (rotateDron > -90)
            {
                rotateDron -= 1.0f;
            }
        }
        if (elevateDron >= 5)
        {
            CircuitDron();

        }
    }

    if (activeDron == false && elevateDron > 2.3f)
    {
        lightDron = glm::vec3(1.0f, 0.0f, 0.0f);
        if (elevateDron > 2.3)
        {
            rotateHeli += 10.0f;
            elevateDron -= 0.01f;
        }
        if (elevateDron == 2.3f) {
            rotateHeli = 0.0f;
            lightDron = glm::vec3(0.0f, 0.0f, 0.0f);
        }
    }

}
```

- función CircuitDron
  Esta función controla el movimiento horizontal del dron siguiendo un circuito predefinido a través de cinco segmentos (r1 a r5), donde cada segmento ajusta la posición horizontal del dron (desX, desZ) y su rotación (rotateDron) para alinearse con la siguiente dirección de vuelo.



```
void CircuitDron()
{
    if (r1)
    {
        desX += 0.02f;
        if (desX > 5.5f)
        {
            r1 = false;
            r2 = true;
        }
    }

    if (r2)
    {
        desZ -= 0.02f;
        if(desZ < -11.0f )
        {
            r2 = false;
            r3 = true;
        }
        if (rotateDron < 0)
        {
            rotateDron += 1.0f;
        }
    }

    if (r3)
    {
        desX -= 0.02f;
        if (desX < -5.5f)
        {
            r3 = false;
            r4 = true;
        }
        if (rotateDron < 90)
        {
            rotateDron += 1.0f;
        }
    }

    if (r4)
    {
        desZ += 0.02f;
        if (desZ > 11.0f)
        {
            r4 = false;
            r5 = true;
        }
        if (rotateDron < 180)
        {
            rotateDron += 1.0f;
        }
    }

    if (r5)
    {
        r5 = false;
        r1 = true;
        rotateDron = -90.0f;

    }
}
```

Precio

El costo total para este proyecto se ha fijado en $80,000 MXN (ochenta mil pesos mexicanos). Este precio ha sido cuidadosamente calculado considerando todos los componentes, servicios y recursos utilizados para la ejecución exitosa del proyecto. La suma establecida refleja el valor de las siguientes áreas clave:

- Desarrollo y Diseño: Incluye todas las actividades relacionadas con la planificación, diseño, desarrollo y prueba del proyecto.
- Materiales y Recursos: Comprende todos los materiales físicos y digitales necesarios para la implementación efectiva del proyecto.
- Mano de Obra: Cubre el costo de los profesionales implicados en todas las fases del proyecto, desde la conceptualización hasta la entrega final.
- Gastos Administrativos y Generales: Incluye los costos indirectos asociados con la gestión y supervisión del proyecto.

Este proyecto garantiza la calidad y la sustentabilidad del proyecto, asegurando que se cumplan todos los objetivos establecidos y se entregue un producto final de alta calidad. Estamos comprometidos con la transparencia y la justificación detallada de los costos para la satisfacción y comprensión de todos los encargados involucrados.

Plataforma de gestión y respaldo

Como repositorio del proyecto se utilizó Github, siendo una herramienta versatil y conocida por todos los desarrolladores del equipo.
Aquí se puede revisar el avance a nivel de desarrollo y obtener el proyecto en su totalidad, cuenta con una carpeta ejecutable la cual contiene el archivo listo para ejecutar.

Url del repositorio Aura MarketPlace:

https://github.com/ChrisCedi/aura-marketplace

IMPORTANTE:

Podrá no visualizarse mucho avance o progreso a nivel de commits en el repositorio, pero gran parte del avance se encuentra centrado en el repositorio del laboratorio de gráfica, extraje el proyecto con la gran parte de los avances y lo metí a este repositorio para complementar los elementos restantes que corresponden a mis compañeros, debido a que el repositorio del laboratorio es privado y no habría manera de mostrarlo. Lo dejo anexado de igual manera:

Url laboratorio:

https://github.com/ChrisCedi/317097742_GPO07_PROYECTOFINAL2024-2

Plataforma de gestión

Se utilizó jira para registro de tareas y llevar a cabo un cronograma de actividades. Se necesita solicitar acceso para ver el proyecto en Jira.

Url Jira:

https://christiancedillopalacios.atlassian.net/jira/software/projects/KAN/boards/1/timeline

Conclusiones

Fue un proyecto desafiante, lo supe desde que iba iniciando el curso, pero sin duda fue interesante todo el proceso para adquirir las herramientas que nos ayudarían a construir el proyecto en su totalidad.
Partiendo desde objetos primitivos, movimientos básicos dentro del entorno, la suma de todas estas prácticas dieron como resultado un proyecto satisfactorio, que tuvo sus momentos de complejidad como elaborar la animación compleja del dron o la bandera, me costó trabajo comprender los requisitos necesarios para que una animación se considere compleja, pero dados los resultados se consiguió. Se requiere tiempo y mucha práctica para poder desenvolverse mejor en las tecnologías utilizadas.

Quedaron más claros muchos conceptos teóricos y técnicas vistas en clase, poco a poco iba encontrando áreas de mejora, intentando pulir cada parte del desarrollo para entregar un proyecto de calidad.