317097742 Group 7

Technical manual
Apple store

Index

Objectives

The main objective of this project is to develop an interactive virtual environment that faithfully replicates the Apple Store located in Antara, Polanco (Mexico City) using Maya as the software for the modeling and detailing of the architectural structure and internal objects of the store. Also, involving OpenGL for the implementation of interactive functionalities, dynamic animations and lighting systems.

The project seeks to offer an immersive experience that allows users to explore the space virtually. This manual has been designed to provide a more explored understanding of each interaction, element and process involved in the creation of this animation.

Project Scope

The objective of this project is to develop an interactive virtual environment that simulates the Apple Store located in Antara, providing a visual, functional, realistic and detailed experience; modeled in Maya and animated in OpenGL.

The following points are the scope of the project:

- **3D modeling:**
  - Detailed modeling of the store's exterior frontage.
  - Complete modeling of the store interior, including fixed furniture such as display tables and shelves.
  - Creation of digital replicas of typical Apple products found in the store, such as iPhones, iPads and MacBooks.

- **Texturing and Materials:**
  - Application of textures and materials for all internal and external objects and structures, based on photographs of the actual store.

- **Lighting:**
  - Implementation of a lighting system that reflects both natural and artificial lighting conditions within the environment.

- **Animations:**
  - Simple animations of interactive objects, such as opening and closing doors, product movements and other animated elements. Also complex animations applied to store objects.

- **Interactivity:**
  - Development of basic controls for navigation within the virtual environment using keyboard and mouse.
  - Interactivity with some objects within the environment, such as operating electronic devices and opening doors.

- **Optimization and Performance:**
    - The virtual environment is fluent and runs efficiently on computers with medium specifications.

Constraints

The development of an animation project using Maya software for an Apple store faced a number of technical complications that required creative and adaptive solutions. One of the main difficulties was importing the glass animation directly from Maya. In this process, a recurring problem arose when we tried to import an object to OpenGL, textures were added that did not correspond, generating a bug in the visualization.

The solution to this problem involved a modification of the fragment shader and the activation of the alpha channel to allow transparency. Once these adjustments were applied, We tried again importing a solid model to OpenGL and applying transparency, which finally allowed displaying the transparent models correctly.

Another significant limitation is related to the poor optimization of the models as, this delayed the startup of the project, the computer used for the project has a Ryzen 5 3500 series processor, 8 GB RAM, 512 GB SSD integrated graphics, the computer showed difficulties at times when loading the visualizations in Maya, especially with regard to the front end of the project. In addition, delays were experienced when loading the project in Visual Studio. However, once the models, textures and uv maps were optimized, the behavior was much improved.

Software methodology

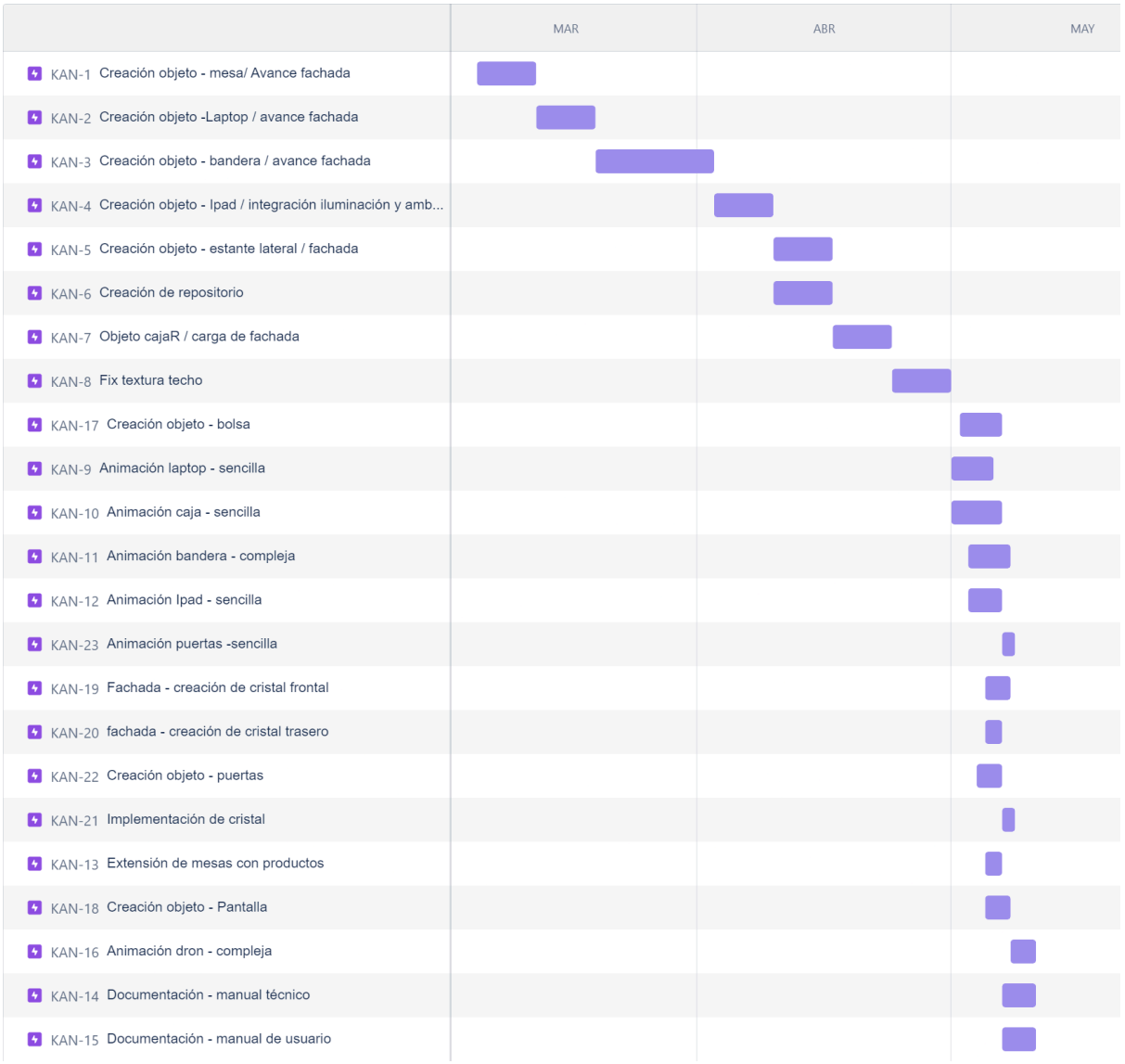
For the development of the Apple Store project, we chose to use an Incremental Development methodology. This decision was based on the need to have a good management of the complexity of modeling and programming in OpenGL, allowing the necessary flexibility to adapt to changes and improvements throughout the development process. This choice was important to integrate continuous feedback from the teacher and adjust the project according to the needs and expectations of the initial proposal.

- **Phase organization:**
  It was divided into several incremental phases, each with specific objectives and clear deliverables. This structure allowed us to focus on completing segments of the environment in manageable stages, ensuring steady and visible progress.

- **Weekly deliverables:**
  A weekly delivery cycle was established, where significant project progress was presented each week. These presentations ranged from new 3D models to improvements in animation and interactivity. This regularity in the deliverables helped to keep the ongoing project under constant evaluation and also allowed for the timely implementation of changes suggested by the reviewers.

- **Continuous improvement:**
  With each weekly delivery, the technical results and feedback received by the teacher were evaluated. This allowed for technical adjustments that improved efficiency and quality in the development of the project.

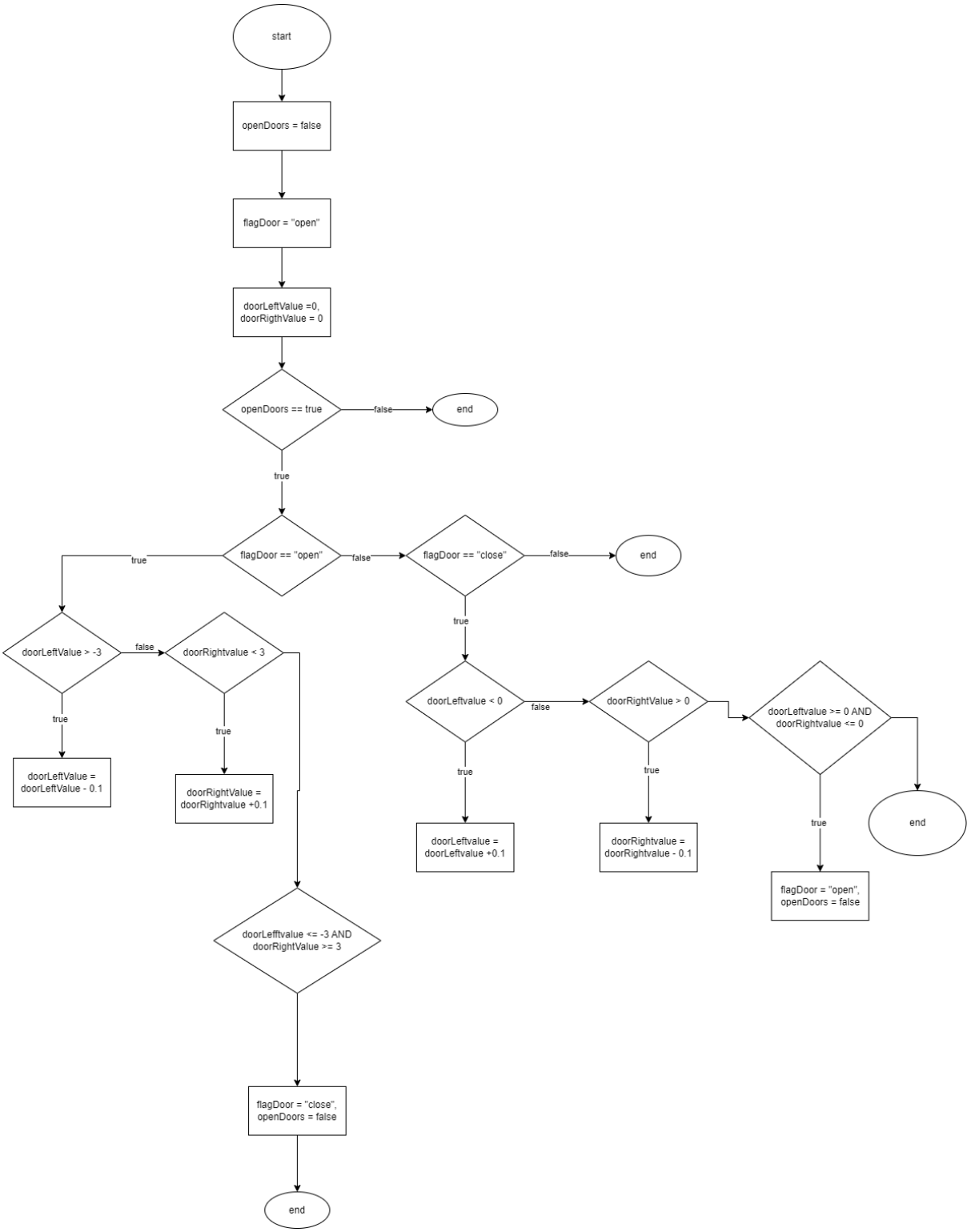
Gantt Chart



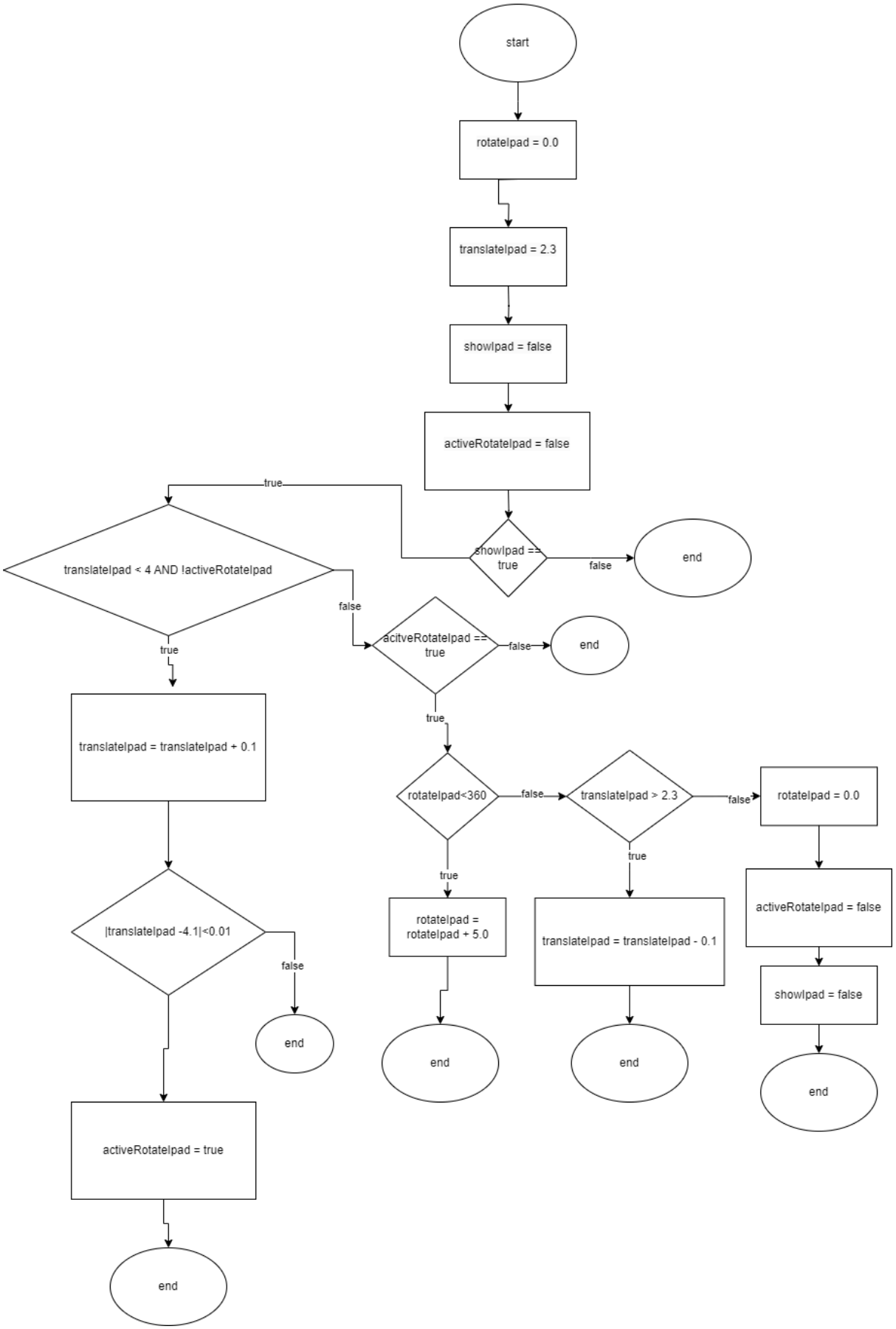
flow diagrams

The simple animations were defined inside the DoMovement function, since it is executed inside the while and this helps to have always listening to the behavior of the variables to activate or deactivate x animation.

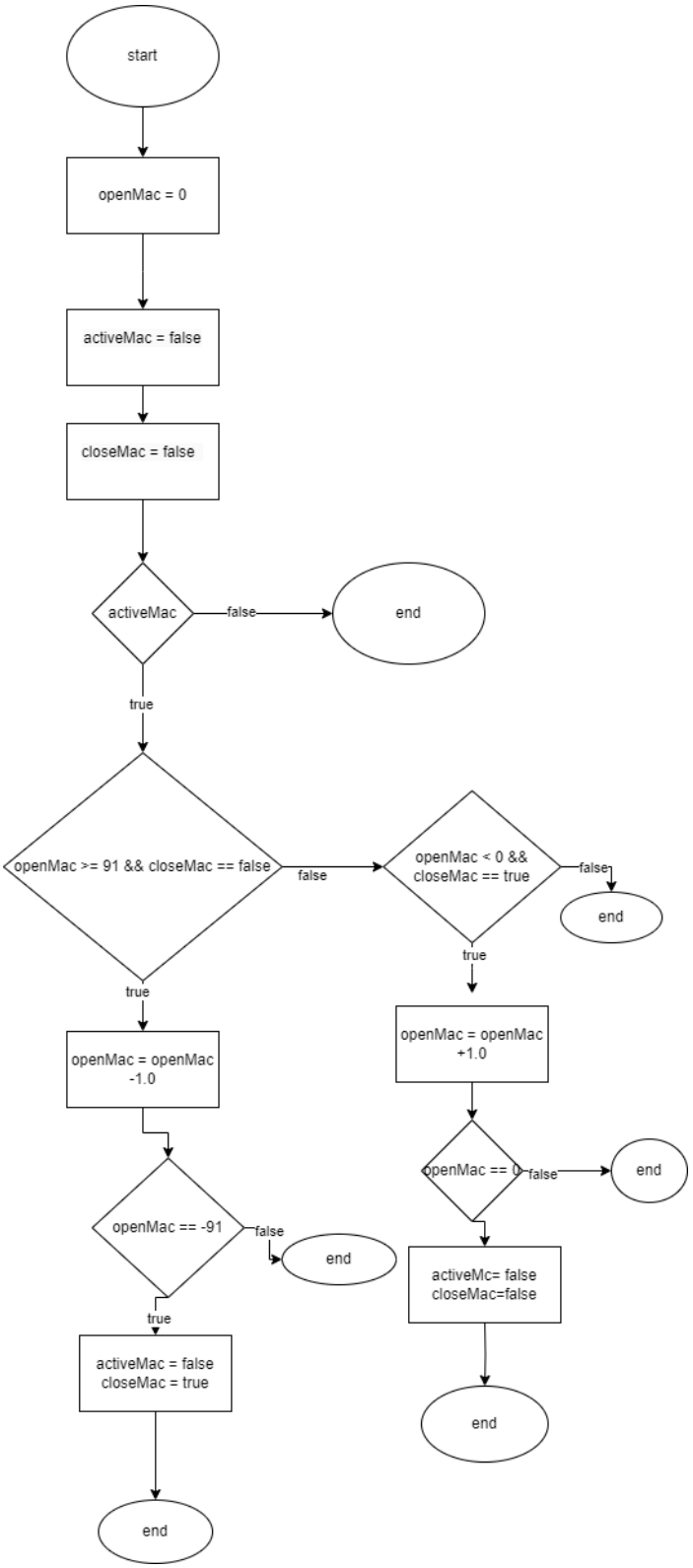
flow diagram - doors



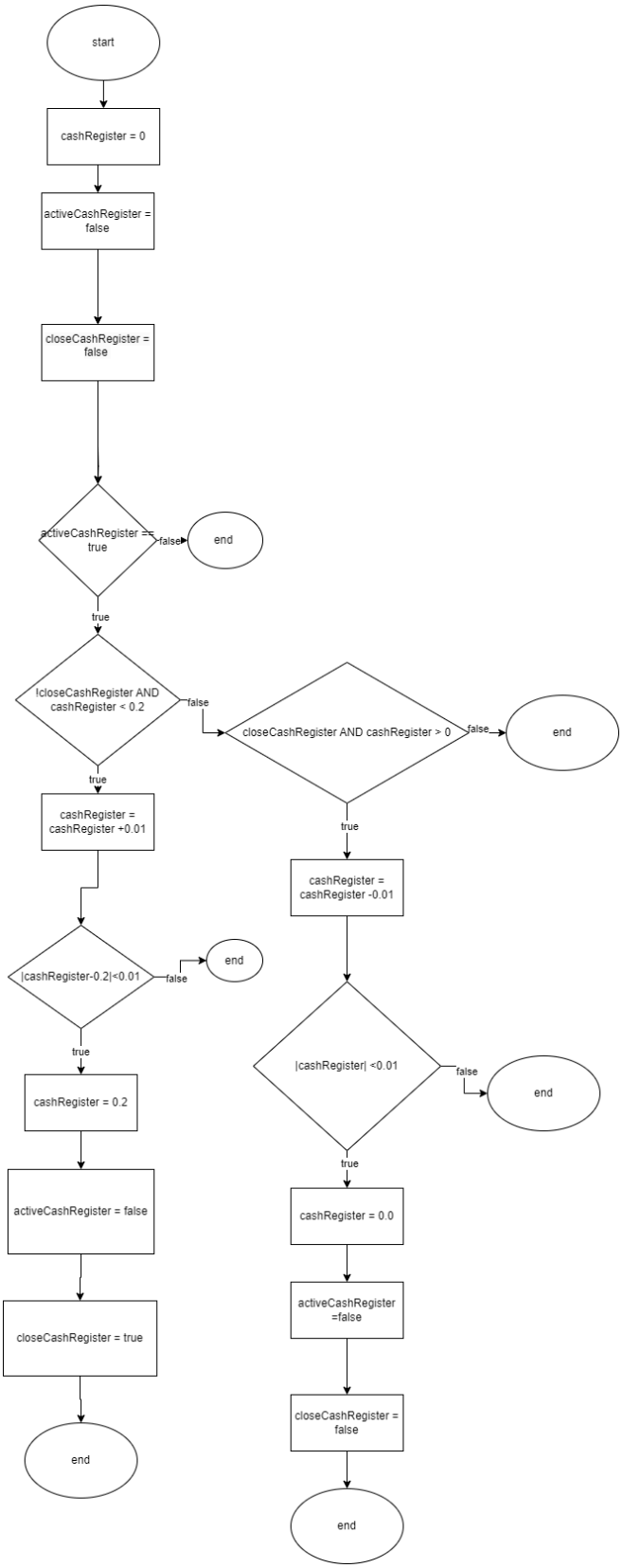
flow diagram - ipad



Flow diagram - mac



flow diagram - cash register



Code documentation

The cpp file is sectioned with comments to define each of the elements that define the project.

variable definition

Variable	Definition
WIDTH	Defines the screen width
HEIGHT	Defines the screen height
SCREEN_HEIGHT	Current screen dimensions
SCREEN_WIDTH	Current screen dimensions
camera	Camera type object that manages the view and position of the camera
lastX, lastY	Last recorded mouse position
keys	Array of booleans for handling key state
firstMouse	Indicator to determine if the mouse has moved for the first time
lightPos	Position of the light source
active	Boolean state to keep the light points always on
spotLightPosition	Position of the spot type light
spotLightDirection	Direction of the spot type light
time	Takes time from the processor
openMac	Controls activation to open the computer
activeMac	Indicates if the variable is active
closeMac	Indicates if the action to close the Mac is active
cashRegister	Increment or decrement value for cash register animation
activeCashRegister	Control variable when the drawer has to be opened
closeCashRegister	Control variable when the drawer has to be closed
rotateIpad	Controls the rotation of the iPad
translateIpad	Controls the translation of the iPad
showIpad	Control to activate the animation of showing
activeRotateIpad	Control to activate the rotation
openDoors	Control variable to activate or deactivate the animation
flagDoor	Control flag to indicate whether to open or close
doorLeftValue	Displacement of the left door
doorRightValue	Displacement of the right door
activeDron	Boolean in charge of activating the animation
rotateDron	Drone rotation upon reaching a corner
rotateHeli	Rotation of the propellers
elevateDron	Increment/decrement to elevate the drone
desZ	Displacement in z
desX	Displacement in x
r1, r2, r3, r4, r5	Booleans that help to switch states
lightDron	Vector containing the color of the light corresponding to the drone
lightColorDron	Vector containing the data for the spot light
pointLightPositions	Array containing the positions of the light points
deltaTime	Time control between frames

function definition

Function	Definition
KeyCallback	Handles keyboard inputs
MouseCallback	Handles mouse inputs
DoMovement	Updates the camera position and manages animations based on user input.
FlyDron	Controls the drone's flight and animation
CircuitDron	Manages the drone's states to perform the route
main	The main function that initializes GLFW and GLEW, sets up the window and shaders, and loads lights, models, and facades.

Complex animation - flag

A shader was defined to animate a flag by adding a ripple effect to the vertices of the flag mesh.

The shader modifies the z-coordinate of each vertex of the flag mesh based on a sine function, which changes with time and the x-position of the vertex. This creates a ripple effect that makes the flag appear to wave. In the C++ code, you update the shader's time variable at each frame to continuously animate the flag. In addition, transformations are set up to correctly position and orient the flag in the scene.

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;
}

```

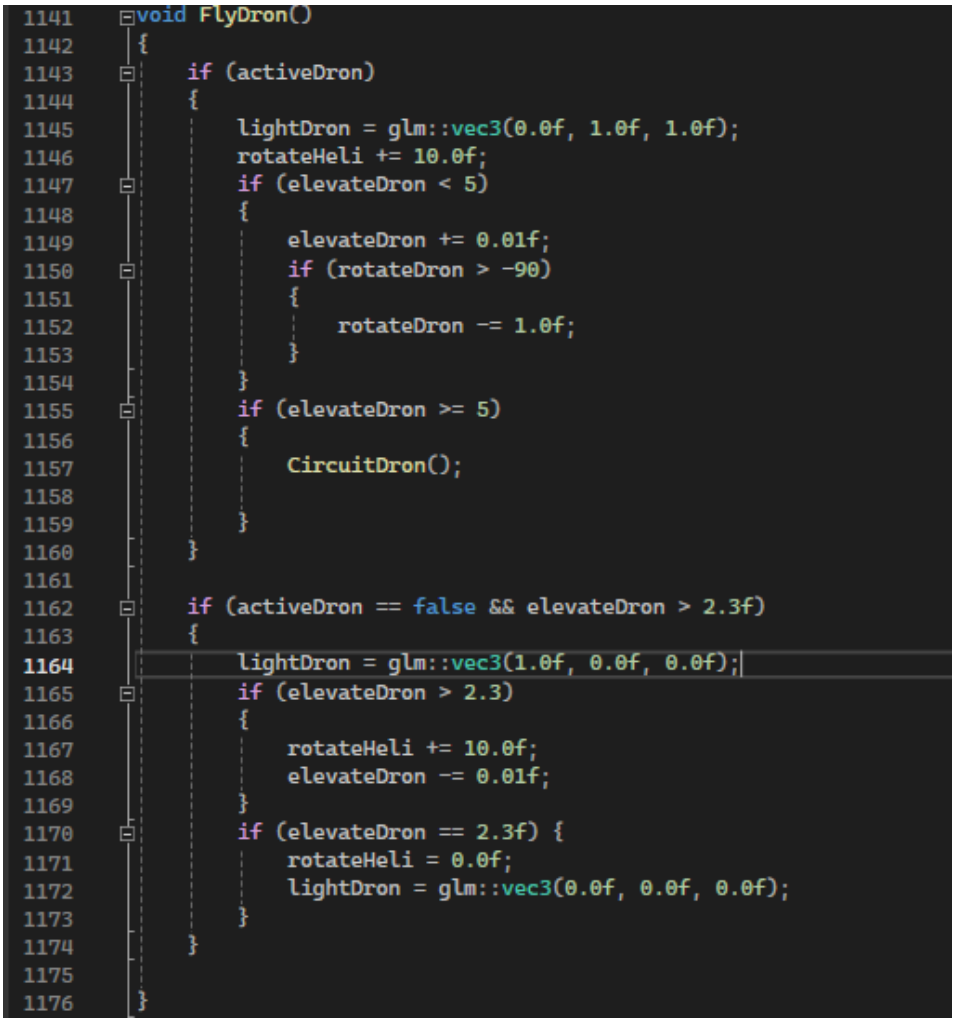
anim.frag

```
#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;

}
```

Complex animation - drone

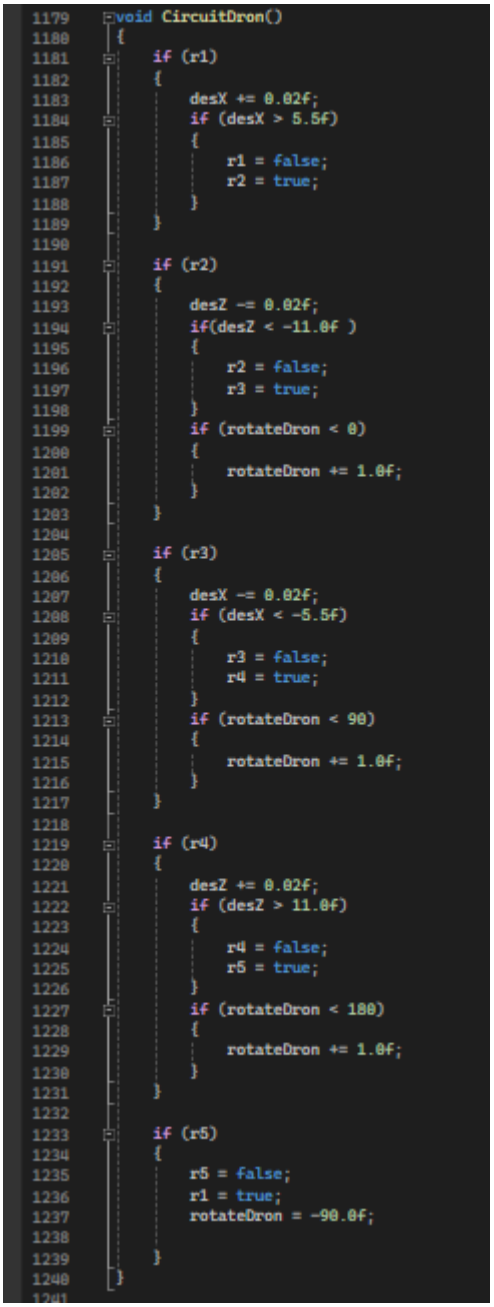
For this animation we created two separate functions which are responsible for animating the drone:

- FlyDron function
  This function handles the activation and vertical movement of the drone, as well as integrating changes in lighting based on the state of activeDron (a Boolean that determines whether the drone is active or not).



```
1141  void FlyDron()
1142  {
1143      if (activeDron)
1144      {
1145          lightDron = glm::vec3(0.0f, 1.0f, 1.0f);
1146          rotateHeli += 10.0f;
1147          if (elevateDron < 5)
1148          {
1149              elevateDron += 0.01f;
1150              if (rotateDron > -90)
1151              {
1152                  rotateDron -= 1.0f;
1153              }
1154          }
1155          if (elevateDron >= 5)
1156          {
1157              CircuitDron();
1158
1159          }
1160      }
1161
1162      if (activeDron == false && elevateDron > 2.3f)
1163      {
1164          lightDron = glm::vec3(1.0f, 0.0f, 0.0f);
1165          if (elevateDron > 2.3)
1166          {
1167              rotateHeli += 10.0f;
1168              elevateDron -= 0.01f;
1169          }
1170          if (elevateDron == 2.3f) {
1171              rotateHeli = 0.0f;
1172              lightDron = glm::vec3(0.0f, 0.0f, 0.0f);
1173          }
1174      }
1175
1176  }
```

- CircuitDron function
  This function controls the horizontal movement of the drone following a predefined circuit through five segments (r1 to r5), where each segment adjusts the horizontal position of the drone (desX, desZ) and its rotation (rotateDron) to align with the next flight direction.



```
1179    void CircuitDron()
1180    {
1181        if (r1)
1182        {
1183            desX += 0.02f;
1184            if (desX > 5.5f)
1185            {
1186                r1 = false;
1187                r2 = true;
1188            }
1189        }
1190
1191        if (r2)
1192        {
1193            desZ -= 0.02f;
1194            if(desZ < -11.0f )
1195            {
1196                r2 = false;
1197                r3 = true;
1198            }
1199            if (rotateDron < 0)
1200            {
1201                rotateDron += 1.0f;
1202            }
1203        }
1204
1205        if (r3)
1206        {
1207            desX -= 0.02f;
1208            if (desX < -5.5f)
1209            {
1210                r3 = false;
1211                r4 = true;
1212            }
1213            if (rotateDron < 90)
1214            {
1215                rotateDron += 1.0f;
1216            }
1217        }
1218
1219        if (r4)
1220        {
1221            desZ += 0.02f;
1222            if (desZ > 11.0f)
1223            {
1224                r4 = false;
1225                r5 = true;
1226            }
1227            if (rotateDron < 180)
1228            {
1229                rotateDron += 1.0f;
1230            }
1231        }
1232
1233        if (r5)
1234        {
1235            r5 = false;
1236            r1 = true;
1237            rotateDron = -90.0f;
1238
1239        }
1240    }
1241
```

Conclusions

It was a challenging project, I knew it since I was starting the course, but it was certainly interesting the whole process to acquire the tools that would help us to build the project in its entirety.
Starting from primitive objects, basic movements within the environment, the sum of all these practices resulted in a satisfactory project, which had its moments of complexity such as elaborating the complex animation of the drone or the flag, it was hard to understand the requirements necessary for an animation to be considered complex, but given the results it was achieved. It takes time and a lot of practice to be able to manage better in the technologies used.
Many theoretical concepts and techniques seen in class were clearer, little by little I was finding areas of improvement, trying to polish each part of the development to deliver a quality project.