3D FLIGHT SIMULATOR  
ASSIGNMENT

Alp Bora Orgun, 6597661

EEE2041 COMPUTER VISION AND GRAPHICS

Contents

[1. ABSTRACT 1](#_Toc71589064)

[2. METHODS 2](#_Toc71589065)

[2.2 ENVIRONMENT 2](#_Toc71589066)

[2.2.1 GAMEPLAY 2](#_Toc71589067)

[2.2.2 MAPS 2](#_Toc71589068)

[2.2.3 SKYBOX 2](#_Toc71589069)

[2.2.4 LIGHTING 2](#_Toc71589070)

[2.3 MODEL DATA 2](#_Toc71589071)

[2.4 PLANE MOVEMENT 3](#_Toc71589072)

[2.5 CAMERA 3](#_Toc71589073)

[2.5.1 THIRD PERSON CAMERA 3](#_Toc71589074)

[2.5.2 FIRST PERSON CAMERA 3](#_Toc71589075)

[2.6 HUD 3](#_Toc71589076)

[3. RESULTS 3](#_Toc71589077)

[4. CONCLUSION 4](#_Toc71589078)

# 1. ABSTRACT

3D Flight Simulator project was a project made to learn about implementation of 3D realistic simulations in OpenGL using GLUT. Different techniques were used to implement realistic features such as realistic lighting, physics and texturing. The project explored how logic is used to control things like plane movement and collisions and how that combines with environment to create realistic simulation.

# 2. METHODS

## 2.2 ENVIRONMENT

### 2.2.1 GAMEPLAY

Gameplay consists of flying a plane in 3D space through rings scattered around a map which have a spinning coin in the centre. Passing through the rings and collecting them add to score. If all the coins in a level have been collected the next level is loaded until all the provided levels are finished. There also is a timer displayed showing the player how long it took for them to finish a level.

Collisions were also added as an obstacle to the player. The ring collisions were done by comparing the plane’s location to 3 separate limits. Whether if the plane is intersecting the rings x dimensions and if it is in between the inner and outer radius of the sphere. This provided a very accurate collision detection for if the plane is inside the rings or not. The same principle was also applied to the ground collision detection but much simpler by only checking the altitude of the plane.

### 2.2.2 MAPS

Map loading was designed so that it can take any MxN sized matrix as a valid course. The matrix is provided as a text file where any integer represents a ring with its value representing the height of the ring and any 0s represent an empty spot.

Which maps to load and which order to load them in is provided while running the simulator through the terminal where each argument provided is the location of the map file in the file system of the computer.

The map loading works by reading the file row by row, counting the rows and then reading the spaces between each integer to count the columns. Then a matrix is created using the row and column values and then the file is re-read to fill the matrix with the values from the text file.

A vector of 4x4 Matrices is then generated with the size of how many rings are read from the file. These are then translated into there places as determined by the input map file.

### 2.2.3 SKYBOX

A skybox is created using 6 planes all with a texture of a side of a cube map texture. Although not using cube map texturing methods, it essentially acts as a cube map skybox with seamless edges. The skybox is centred onto the plane, so it acts as if you fly infinitely into the horizon.

### 2.2.4 LIGHTING

Lighting was already provided through shaders in the setup. However, to improve on this the view matrix and the model matrix were separated. This allowed the lighting to be separate from the camera and be static on models instead allowing for lighting to change as the camera moved around.  
  
A second shader was also implemented in order to remove the lighting effects from the skybox which does not need to have lighting.

## 2.3 MODEL DATA

Models are all done the same way. An object is initialised, and a texture is loaded. The shader is set and then the texture is bound to GL\_TEXTURE\_2D. Then the object is rendered. After rendering the object GL\_TEXTURE\_2D is cleared.

## 2.4 PLANE MOVEMENT

Plane movement was done by using two matrices, a transformation matrix and a rotation matrix. Forward transform is applied to the transformation matrix, and each rotation, yaw, pitch and roll are separately applied to the rotation matrix. These matrices are then multiplied and then multiplied with the plane’s matrix which is then set equal to the plane’s matrix translating and rotating the plane as intended. This gives a close to realistic plane movement with the only flow being the plane instantly snapping instead of decelerating when a key is lifted.

2.5 CAMERA

### 2.5.1 THIRD PERSON CAMERA

Camera implementation is done by extracting the forward direction and up direction from the plane’s matrix then setting the camera position to the plane’s position minus the forward direction times a constant plus the up direction times a constant which places the camera position to a certain distance behind the plane. Then the lookAt function is used on the view matrix with the eye set as the camera position, target set as the plane and the global up direction given as 1 on the y axis. This rotates the camera so that it constantly looks at the plane.

### 2.5.2 FIRST PERSON CAMERA

The first-person camera uses the same principles as the third person camera however the camera position is set to the nose of the plane instead of behind the plane and the lookAt function is changed so that the camera looks at point in front of the plane also set using the forward direction of the plane. A key bind is set so that it is switching between the two camera locations is possible.

## 2.6 HUD

Hud was made by rendering text on the screen using the built-in glut bitmap character function. The text was stored as a string with variables getting added to the end of the string. This string is then iterated through character by character to be printed on to the screen.

# 3. RESULTS

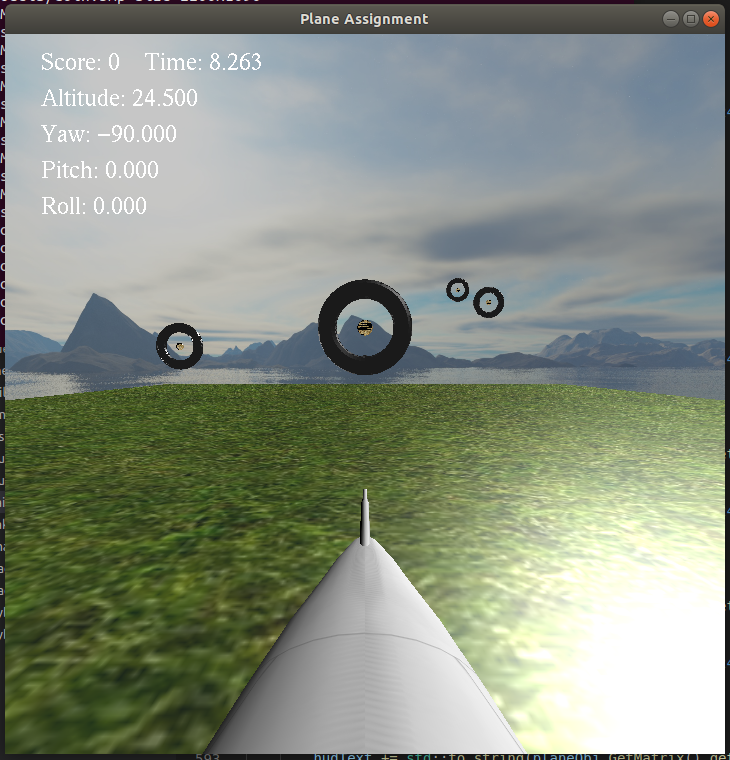
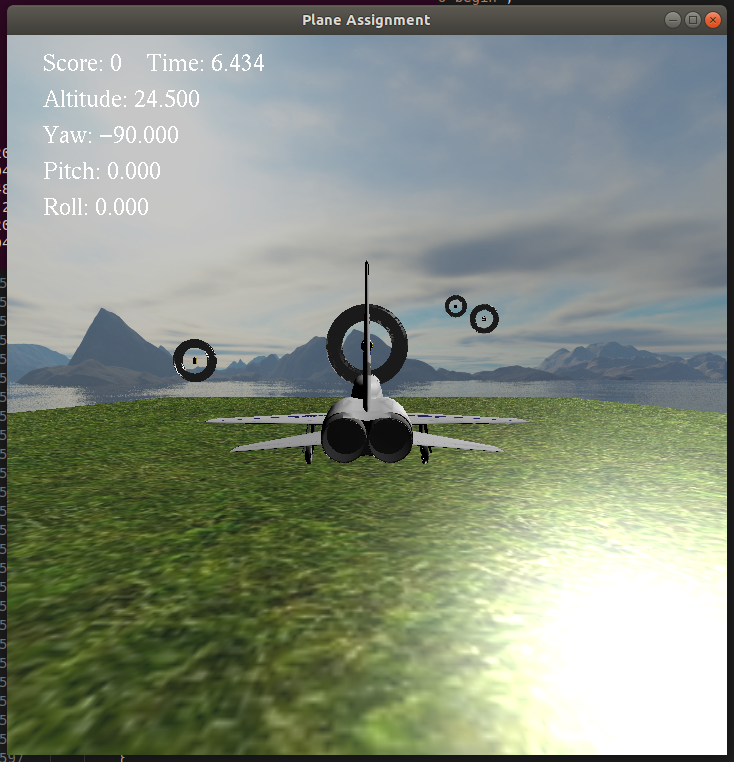


Figure 1 - Third Person Camera Figure 2 - First Person Camera

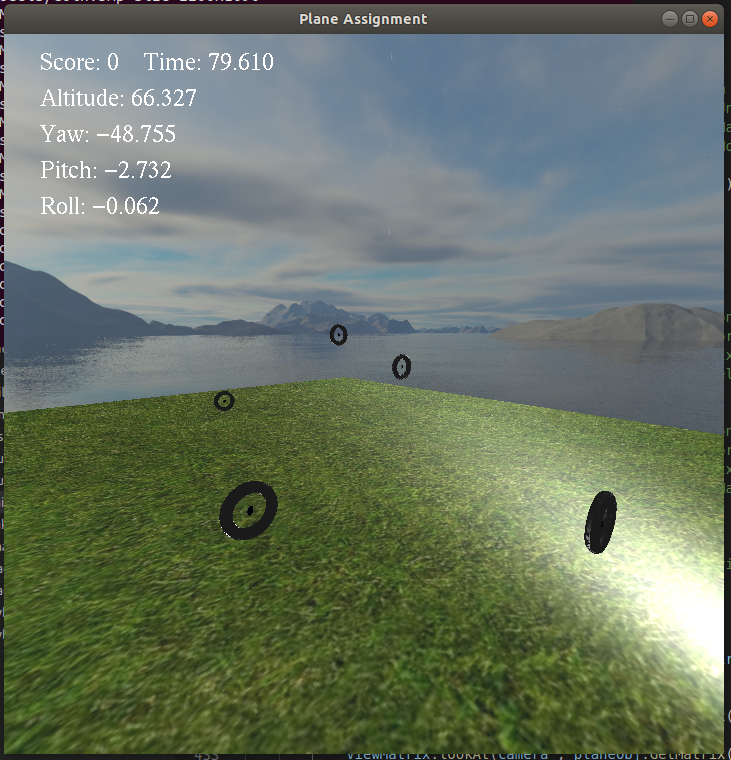


Figure 3 - Example Map 1 Figure - Ring Close-up

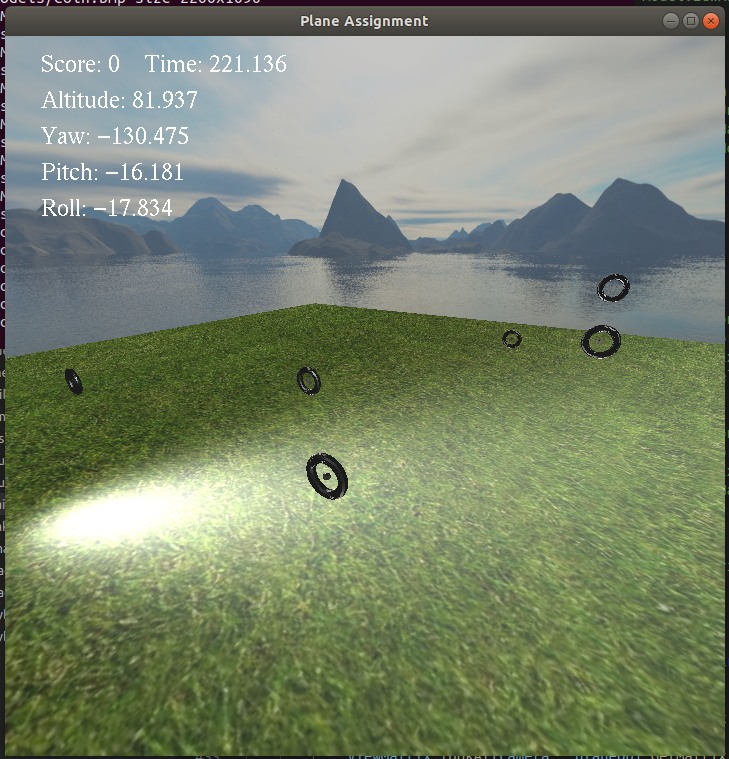


Figure 5 - Example Map 2

# 4. CONCLUSION

The project was a great success from getting a plane to move around semi-realistically to having an environment with a fully functioning skybox. The implantation of all the required features was achieved including, a physics-based movement of the plane, customisable maps that are read from a file, third person and first-person camera options, a full skybox with a separate shader, collision detections of rings and the ground. Score count and a heads-up display. The project was a success for filling out the required criteria and also as a learning tool teaching about how modern game engines work under the hood in terms of implementing graphics relying heavily on vector mathematics.