****

**Computer Vision and Graphics EEE2041**

**Flight Simulator**

**Report**

**Rafal Guz**

URN: 6534571

**Introduction**

The objective of the project was to develop a 3rd person flight simulator game using OpenGL and GLUT. The goal of the game was to navigate a plane around an aerial course using the keyboard and mouse. The course should have consisted of a series of rings within the game environment. The plane should have been flown through in order to accumulate points. Hitting or missing a ring entirely should result in the game being over.

I have achieved desired result using plenty of different techniques and parameters. I was focusing on user freedom of interactions and interface, as in my opinion they are the key things in this kind of games.

I was inspired by AAA games, as they define current “top class” of this market. That relates to the approach of gaining user engagement, physics imitation, and freedom in gameplay, variety of options like difficulty levels or high standard of immersion.

The game was tested several times with audience made from people from different age groups, professions and familiarity with the computer games. That triggered some adjustments and changes in the game which I mentioned in related paragraphs.

**Method**

Implementation of the game can be described in 5 stages. I will look deep into each of them, detailing approach and techniques used.

***Game Environment***

* Ground

The game environment, in particular the ground, is made from 9 planes. Visualisation of their relative positions is shown on the Figure 1.

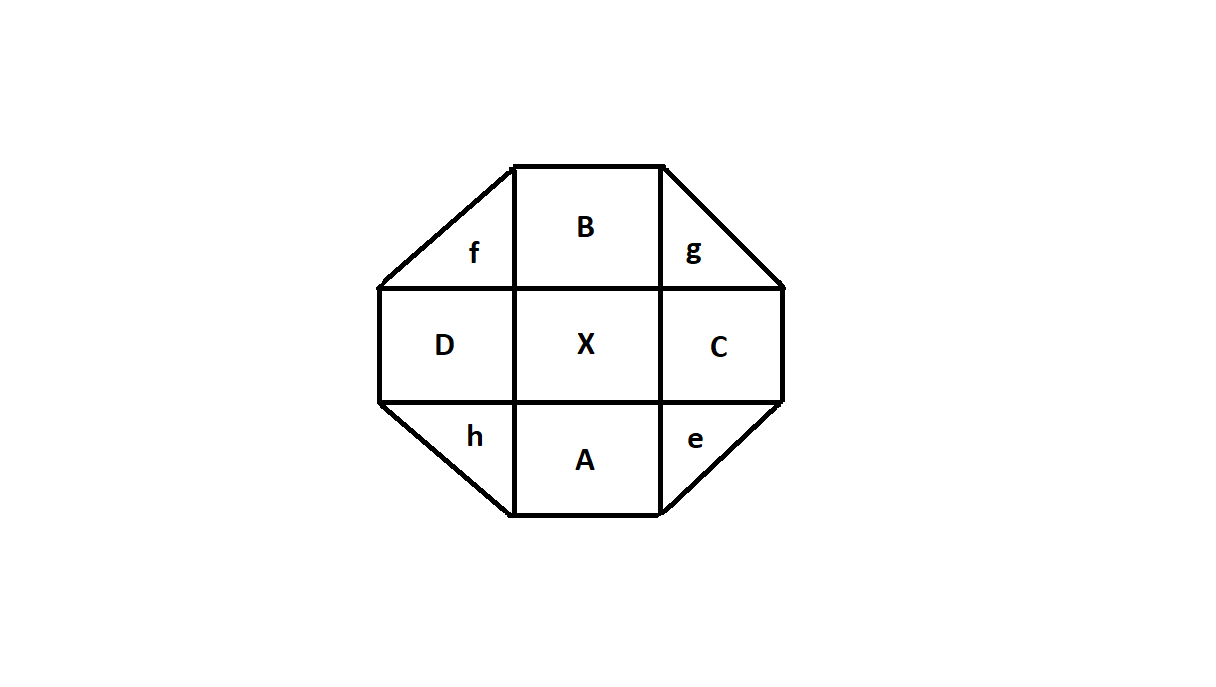


Figure 1. Game arena visualisation.

X plane is the main one for gaming. Above this area the player will find at least the majority of the rings, as well as there the game will start. A, B, C, D planes are defining end of the game world. They are creating a slope, where the valley is the X plane. Planes e, f, g, h are connecting neighbours in such a way to avoid discontinuity of the ground.

That approach helps to increase the immersion as the player does not see the end of the game world. Detection system (described in more details in Plane Movement section) was made to prevent player to go beyond specified area.

* Rings

Position and diameters of rings are defined before the game starts. It is done by reading from the matrix and assigning values to created “Torus” structure, which contains all needed parameters defining the rings. They are dependent on a difficulty level, chosen by the user. Rings can have different height and rotation for larger variety of obstacles. However, there was not created enough accurate pointing system to enable user to fly through the aerial courses with concise manner of collision detection. Therefore, the angle of rotation was reduced to 0, leaving the consistent score algorithm and easily adjustable structure if the proper algorithm will be derived.

After proper flying through ring, points are accumulated, number of remaining rings is reduced, and the ‘validity’ of that ring is turn to false, which results in no possibility of gaining points twice using the same object. That also affects drawing function, as it will draw only the valid courses.

***Model data***

All textures used in the game were provided at the beginning of the assessment. The plane and torus meshes were provided as well, however, the ground, due more complex structure, was made from the beginning in a separate mesh file.

Rings scaling is adjusted to the difficulty level. That has an impact on the collision detection system as well.

***Plane Movement***

Plane is moving in all directions using keyboard inputs. Each change of the direction of the flight is achieved by the combination of three ways (pitch, yaw & roll) and then applying a thrust to move in the direction which it is facing. The position of a plane is determined by the speed and direction of the movement. Player has a control under the direction as well as the speed of the flight. If the airplane moves into area defining the edges of the game world, firstly it results in the warning shown in the HUD and if that does not force the user to turn back, then it will terminate the game.

There is also implemented a key buffering, which allows multiple keys to be registered at once. That enables player to for example decrease the velocity while he is turning in a chosen direction.

***Camera and HUD***

The camera follows the plane as it moves through the environment from a fixed distance. Primarily, the user was able to rotate camera around the plane using right click of the mouse without interrupting the flight. The angle at which the camera was looking at the plane, was adjusted in the run time. However, that was making confusion among not experienced with that approach individuals. Therefore, the camera is static, and always looking from the back of the plane.

The time of the game, number of rings remaining to pass through and points gained are shown at the top of the screen. Current velocity of the aircraft (in Ma) is shown in left down corner. A provided function to draw 2D text was used to draw the whole game HUD.

***Additional functionality***

* Soundtrack

From short games, the most reminded aspect is audio-visual experience. Hence, the main theme and while-game audio was introduced. Music that appears in the menu (described later on) creates a specific atmosphere and makes the game more enjoyable. It is joyful and simple, so that the game is more casual. On the other hand, when the user gets on the plane, the wind sound can be heard. It can give a closer feeling of experiencing the flight in real life.

* Levels of difficulty

There are implemented 2 levels of difficulty - EASY and HARD. It is worth a note, that the decision between them will impact significantly the gameplay. It will not only change the size of the rings but whole pointing system as well. Number of gained points changes from linear dependency to the logarithmic one. The place where the centre of the plane is passing ring changes from just within the inner ring to that one, which makes sure that no one point on the plane would ever touch the ring.

* Menu

The crucial part of User Interface is menu. The gaming industry has made a convention that the menu interactions are made via mouse clicks. Therefore, it is also implemented this way in the Flight Simulator.

In the game, the main menu allows the player to see controls – familiarise with the keys before gaming. Each game has a bit different key set responsible for interactions. It is essential to show it to the user beforehand, avoiding not needed confusion and frustration (i.e. aggression).

The next thing added in the menu is the difficulty settings. As already mentioned, there are 2 levels of difficulty – EASY and HARD. The user is able to change it (from default EASY) until the game will start. After this point it will be not possible to do so unless player will win or lose, which will trigger return to menu and resetting the game.

The game starts from the menu; therefore there is a provided button for that action, which additionally tells the player about current difficulty chosen. Exit button was added as well.

That covers all needed functionality of the main menu. It greatly increases easiness of use and clarity of the setting for the player.

There are also 2 other menu-like interfaces in the game. The first one is the “Game Over” interface. It will appear after the user will lose. It can be either by going beyond the defined game world or hitting the ring. The interface summarises all-important information from that particular game: difficulty level, time spent, rings left and number of point gained.

The other one is a “Won game” interface. That will appear in opposite situation – when the user will successfully fly through the last ring. Similarly to “Game Over” interface, it will summarise the game in relevant parameters. It is also a way to gain player attention and give a “not said challenge” to reach the maximum number of points as quick as possible.

* Win != End of Game

There are many games after finishing which players are enabled to move more freely around the game world. There is a similar situation with this game. When the user will properly fly through the last remaining ring the “WIN menu” will appear. There will be a chance to exit the current game to start new one or go back and continue free flight with no restrictions apart from just staying within gameable area.

* Imitating Physics

As in professional flight simulators, there is a certain imitation of real physics. That relates to several parameters. There are specified maximum and minimum velocities. The navigation of the plane is done by combining pitch, yaw and roll. Each movement has a certain limit of freedom, just like it is in reality. The velocity cannot be changed rapidly; therefore the player needs to aware that major increase of speed can result in worsening navigability.

**Results**

Figure 2. Main Menu Figure 3. Difficulty Menu

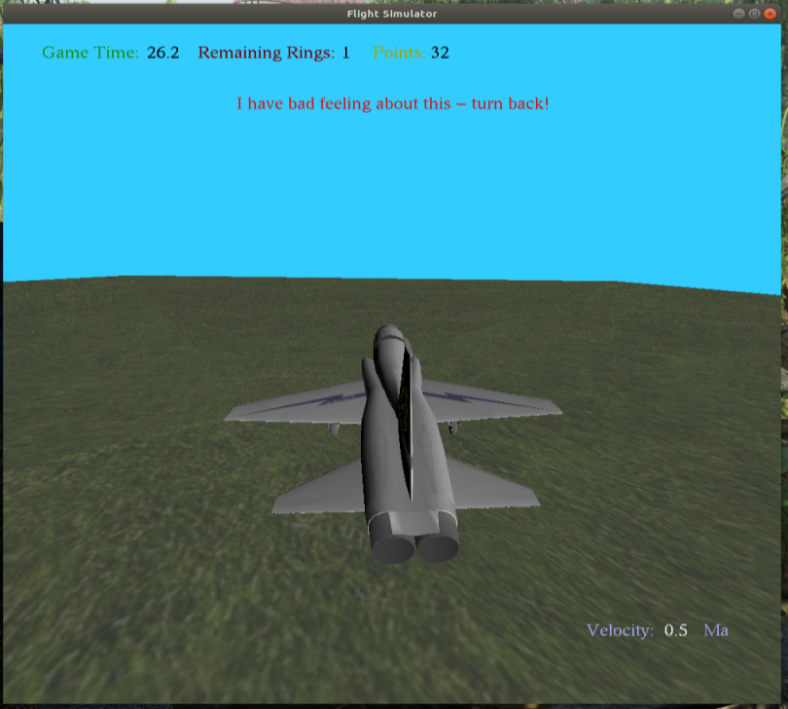
 

Figure 4. Controls Menu Figure 5. HUD and warning

Figure 6. WON Interface Figure 7. LOST Interface

**Conclusion**

The project was a fantastic opportunity to test and enhance greatly the abilities in programming using OpenGL, as well as plenty of other connected with Computer Graphics or games in general. There were many occasions to free creative thinking and implement brave ideas. The approach to the problem needed to be deliberated in advance to enable the code for optional future enhancements.

Obstacles encountered improved debugging skills, where the more complex algorithms like collision detection, taught me about analytical thinking and problem solving techniques.

Having a possibility to play the game, without issues and crashes is the best feeling after plenty of hours spent on thinking, implementing ideas and debugging them.