Nat Manley, u1906046

CS324 Coursework Supporting Document

(905 words)

1 Introduction

The project is about controlling a fixed wing glider as it flies through a randomly generated city. It is loosely based on a wingsuit game called Superflight [1]. The goal is to avoid the buildings and explore the city.

To run the game, run a local web server to host the files (for instance, python3's http.server). Both a bash script and a python script have been provided that can be used for this.

2 Features

All essential features were implemented:

- **Interactivity** was achieved through standard JavaScript event handling rather than any of Threejs' built-in control methods due to the desire to have more unique controls. These then adjusted the glider's pitch and roll (Using Object3D.rotateOnAxis [2] to rotate the glider around its relative cardinal directions). Yaw was modelled as a function of roll rather than considering the centre of mass compared to centre of lift.
- Two **camera** views are offered, a third person view behind the glider that adjusts depending on the glider's speed, and a first-person view that just hides glider model. They use the same camera object, but just change the method by which the optimum camera position is found.
- Two **menus** are included, an initial welcome menu as well as a death menu. These were implemented with HTML on top of the game canvas, and navigation through them acted similar to a state machine.
- **Levels** are randomly generated and so there are functionally infinite possibilities, after crashing, a user can decide to replay the same level or switch to a new one. The buildings in the cities all have BoxGeometry as if they were square skyscrapers.
- One **model** of a glider was created (see later section for details).
- **Lighting** was included, both a directional global light to represent the sun (as well as a hemisphere light to model the ambient reflected light), and a spotlight in front of the glider pointing forwards as if it were a headlight of sorts. Shadows were enabled, but due to hardware constraints, the shadows are baked in, only being calculated once when each level is created, and so the glider does not have a shadow.
- **Textures** were created for the buildings, to add some degree of variation, two alternatives were provided when choosing a building's texture. Textures were loaded with the Threejs texture loader and applied to a MeshPhongMaterial.

On top of that, several optional features were included:

- A **HUD** that displays both height and speed (achieved with a HTML overlay).
- **Sounds** of wind rushing past the glider that get louder as speed increases. The sound was taken from Mixkit [3] and loaded using Threejs' AudioLoader. The sound volume follows a simple linear function, as it serves its purpose fine, and I couldn't find any sources discussing wind volume as a function of speed.
- **Collision detection** with the ground and other meshes in the scene implemented as a cube bounding box of rays around the glider's centre. I decided that a simple bounding box was good enough for the speed at which the glider goes, and so a more accurate collision method would be both unnecessary and less fun for the user due to the decreased leeway they would have.
- **Physics** to model the glider's acceleration. These don't aim to be completely realistic, as I felt that implementing the real-world aerodynamics would be unnecessary for this project, but they simulate something close. The physics were all calculated using the glider's rotation, and made extensive use of Vector3 transformed with both quaternions and Threejs' ApplyAxisAngle method.

4 Modelling Process

The reference used for the model is shown below:

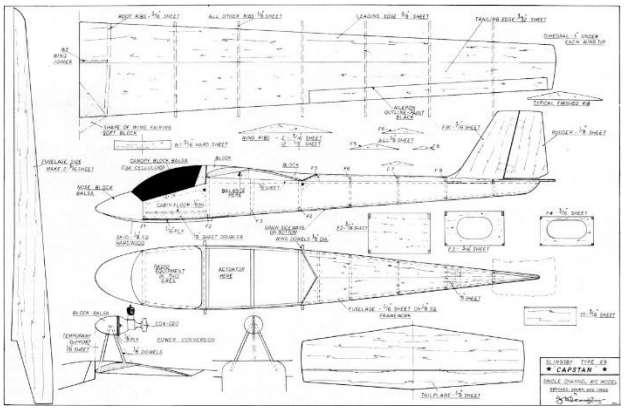

Figure 1 Slingsby Type 49 Capstan Glider Schematics [4]

This modelling process started with crafting the body of the glider, based on the diagram above, but done by eye. The basic shape is similar to an aerofoil, and so there were some challenges in getting the curves realistic. Due to the symmetric nature of the model, I chose to reflect it in the y axis so only one half of the glider had to be modelled.

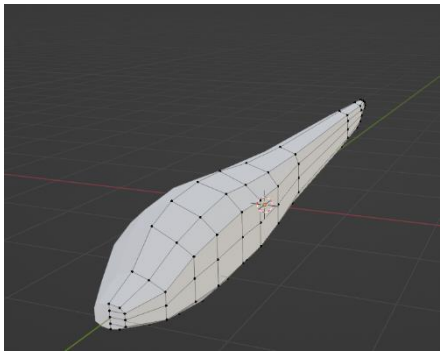

Figure 2 Model Glider Body

The rudder was almost a complete copy of the reference image, but because I didn't like the vertical stabilizer design, I chose to take inspiration from dart fletchings instead.

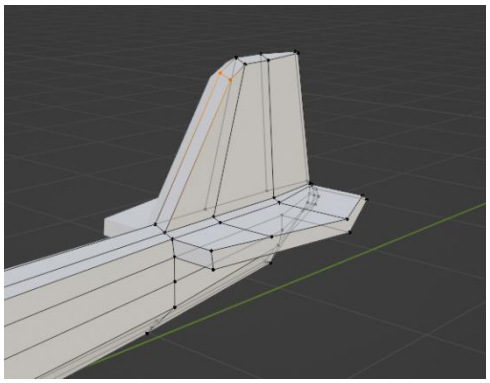

Figure 3 Modelled rudder and vertical stabilizer

The wings were extruded along the x-axis from 4 quads that made up the shape of the hull, they were then further subdivided so that the curve at the end could be achieved:

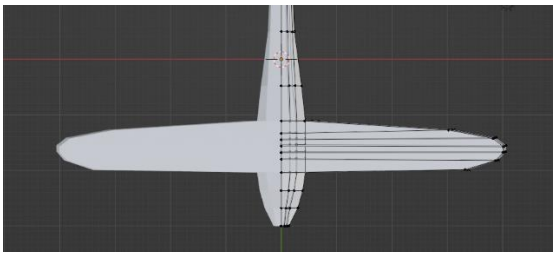

Figure 4 Modelled Wings

I chose to go with a simple paint theme for the glider, and also chose not to apply any different specular attributes for the cockpit glass as I thought it would distract from the simplicity of the model. It also wouldn't be able to be seen much from the perspective in the game.

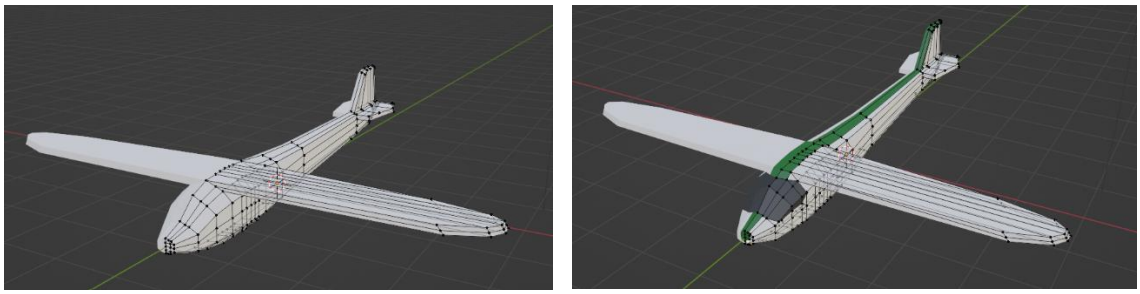

Figure 5 Finished Glider Model with and without texture

The paint scheme required the use of the knife tool to split some quads into smaller sections to achieve the desired result, but there weren't many significant changes apart from that.

References

[1] GrizzlyGames, *Superflight,* GrizzlyGames, 2017.

[2] Threejs, “Threejs documentation,” [Online]. Available: https://threejs.org/docs/#api/en/core/Object3D.rotateOnAxis. [Accessed 16 1 2022].

[3] Mixkit, “Winter Wind Loop,” [Online]. Available: https://mixkit.co/free-sound-effects/wind/. [Accessed 16 1 2022].

[4] K. Blattenberger, “Airplanes & Rockets,” [Online]. Available: https://www.airplanesandrockets.com/airplanes/capstan-slingsby-aug-1972-aam.htm. [Accessed 01 16 2022].

[5] Open Source, “Threejs Github,” [Online]. Available: https://github.com/mrdoob/three.js. [Accessed 16 1 2022].