AirTarget Writeup – Nick Lochner

CS498 SL (Virtual Reality - Oculus Rift Development)

The project's goal was to create a collision avoidance simulator for a study to determine if flight training with VR is beneficial to help pilots improve their avoidance skills.

The Oculus Rift was essential to this project, as the project would not be possible without some sort of VR.

Briefly, the steps necessary for the project were:

1. Creating a user interface for the main menu and options.
2. Spawning oncoming aircraft with various paths.
3. Calculating and displaying dynamics data for debugging and later analysis.
4. Creating the interface for the user to indicate if they spotted the oncoming aircraft, and if it is a threat.
5. Scenery and graphical elements.

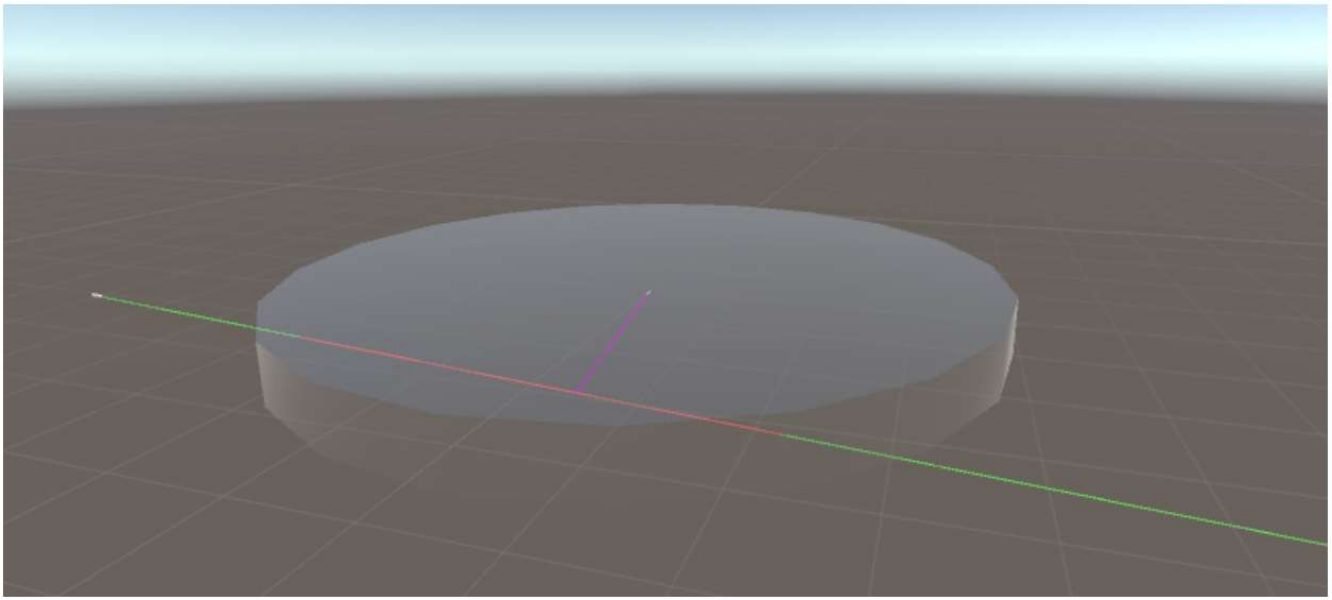
Dynamics Data

I worked on the dynamics data portion, later integrating with the oncoming aircraft spawning code.

The dynamics data portion tracks and displays the following information in the inspector:

x-coordinate, y-coordinate, z-coordinate, Heading, Bank, Pitch, Speed, Altitude, Aircraft Type, Score, CameraRig x-rotation, CameraRig y-rotation, CameraRig z-rotation, Closest Point x-coordinate, Closest Point y-coordinate, Closest Point z-coordinate, Bearing Horizontal, Bearing Vertical, Aspect Horizontal, Aspect Vertical, Target Aircraft x-coordinate, Target Aircraft y-coordinate, Target Aircraft z-coordinate, Target Aircraft Heading, Target Aircraft Bank, Target Aircraft Pitch, Target Aircraft Speed, Target Aircraft Altitude, Target Aircraft Type, Airspace Violated

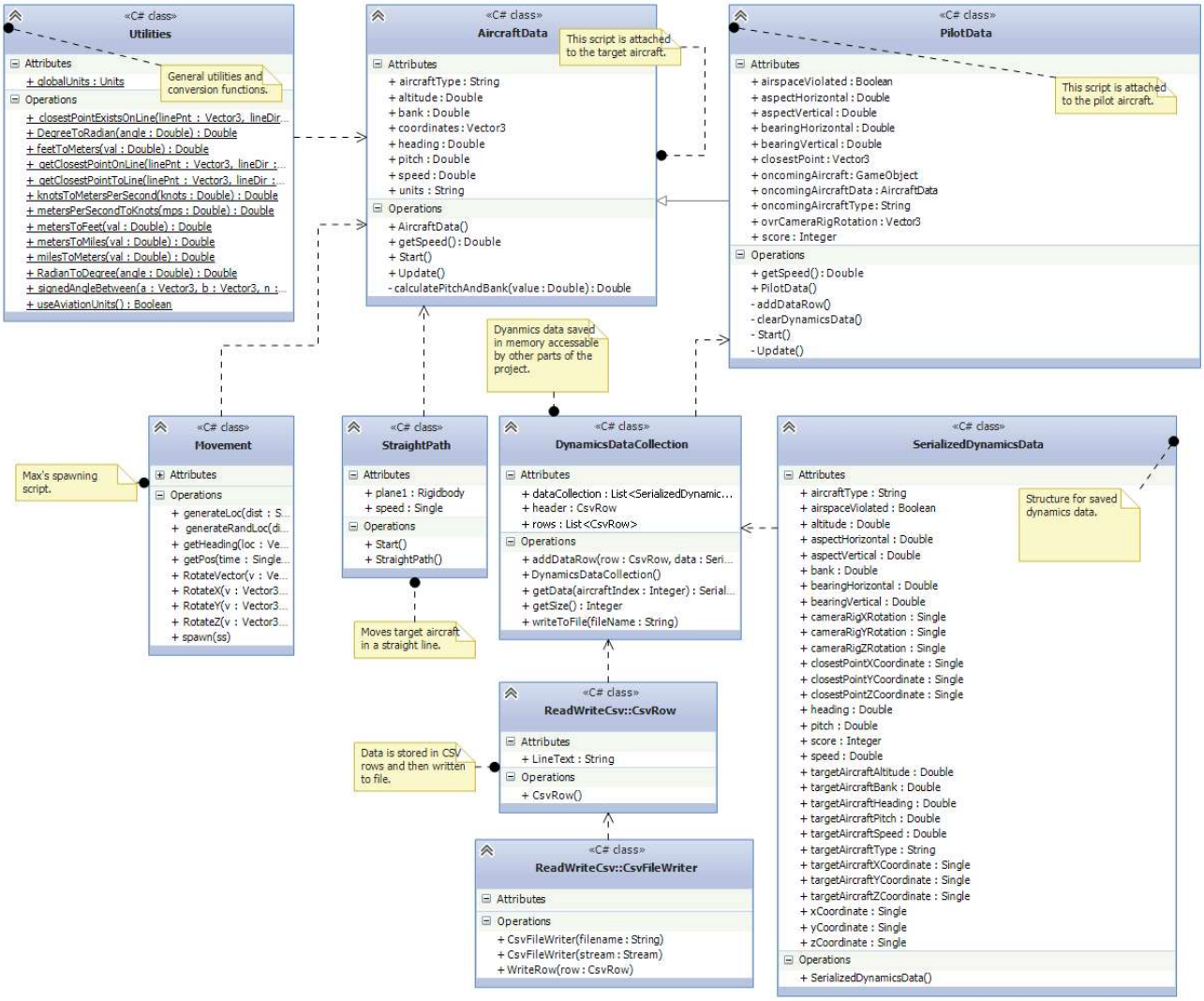
A cylinder defines the violated airspace. The greed/red line displays the oncoming aircraft's path. It is green for where the aircraft would not be in violation, and red for where it would be in violation. The



pink line is the path to the closest point of the two aircraft on the path.

After this I worked on combining my code with Max's spawning code. After it was combined, we worked on a structure to specify a preset spawning order for the target aircraft. Next, I worked on the ability to save this information to a CSV spreadsheet at any point in time.

Code architecture



The AircraftData and PilotData scripts are attached to the target aircraft and pilot aircraft respectively. These scripts calculate the dynamics data. The StraightPath script is used by AircraftData in order to cause the target aircraft to move along a straight path.

Max's Movement script will spawn and initialize the target aircraft, and we created a structure in that

class to specify the order and parameters of spawned aircraft.

The PilotData uses the DynamicsDataCollection, which stores dynamics data in memory for access by other parts of the program, and uses ReadWriteCSV to write the data to a CSV spreadsheet.

Enhancements

We were not sure exactly of the order of Unit's Start() function calls. We created an init() function to use instead of Start() in several classes which were called by dependencies in order to initialize in the correct order. If time allowed, it would be better to understand the ordering of the Start() function calls in order to improve the code architecture. The architecture could also be refactored in many locations to be improved.

Conclusion

The project was enjoyable. We were able to get it finished in time to have a working simulator. Although there were some difficulties writing well structured object oriented code when being unfamiliar with Unity's quirks.

Note

The ReadWriteCSV classes were adapted from from:
http://www.codeproject.com/Articles/415732/Reading-and-Writing-CSV-Files-in-Csharp

License is compatable: http://www.codeproject.com/info/cpol10.aspx

Attribution

This writeup was for my part of the project, which was a collaboration with 6 other students. My code was combined with the code from the other students, listed below, to create the final flight simulator.

•Reia Demont

•Xuan Wang

•Max Watkins

•Yixiang Zeng

•Mingze Gao

•Johnathan (Xusheng) Zhang