EE267-Final Project
Munir Bshara

There are two aspects of my project the Unity game and the orientation algorithms. They are separate so I will treat them as separate in this README.
This game was tested and developed on Unity 2021.3.45f1 on the Mac M1 Air. It uses many assets, but those are not needed for the actual execution of the game.

1. Please clone the github repo from this website: https://github.com/MBshara/ee267
2. Open the project in Unity, and under Assets/Scenes open up MazeRunner.unity
3. Flash the vrduino.ino in the vrduino folder just downloaded from github to teensy.
4. Make sure to edit the portName variable in the READUSB.cs file under Assets/Scripts to correctly match the port on your device.
5. Build the game!
6. Have fun playing, use WASD to move and it is always relative to your headset direction. Press y to reset the game and/or change the maze.

Gameplay:
Navigate through procedurally generated mazes using head-tracking controls. Avoid the chasing ghost and reach the exit to win. Test different orientation algorithms in real-time to compare their performance and stability by clicking 5 to 8. An example of someone playing the game can be found here.

Orientation Algorithms:
All the algorithms are located under vrduino/OrientationMath.cpp. There are four possible algorithms to try out: Complementary, Madgwick, EKF, and Mahony. In order to test each one press 5 to 8 respectively while playing the game. The methods I created are

1. updateQuaternionEKF
2. updateQuaternionMahony
3. updateQuaternionMadgwick

To modify algorithm parameters, edit the individual function calls in updateOrientation() function under vrduino/OrientationTracker.cpp. REMEMBER TO REFLASH THE ARDUINO!
For a side by side comparison of the algorithms, please look here.

All scripts I created to run the game exist under Assets/Scripts. A total of 6 I edited/created:

1. ReadUSB.cs – added a feature to also serially transmit allowing you to test different algorithms while being in game by pressing 5 through 8.
2. PosControl.cs – controls the position of the viewer in the map
3. MazeGenerator.cs – handles all maze generation

4. EnemyChaser.cs – handles ghost bobbing and how it comes at you.
5. GhostGameOverTrigger.cs – handles death screen.
6. WinTrigger.cs – handles win screen

Default parameters are set for each of these but can be changed. To change these here is a guide:

1. Player Game Object contains scripts:
    a. PosControl.cs – You can change the speed of wasd movement.
    b. GhostGameOverTrigger.cs – You can change the death trigger distance and the death animation
    c. WinTrigger.cs – Change region of win zone.
2. Ghost Game Object contains scripts:
    a. EnemyChaser.cs – Change the speed of the ghost and the bobbing of the ghost.
3. Scene/Maze Game Object contains scripts:
    a. MazeGenerator.cs – Changes the size of everything to do with the maze.