Assignment 3 - Scriptable Objects/Editor Tools

By: Samuel R. Mortimer

Part 1 – Intro to Scriptable Objects

As it stands, the Quest Manager uses “Race.cs”, which itself uses “TimedWaypoints.cs” to define what each different race contains. The idea here will be to keep “TimedWaypoints.cs” as a reference for Race, but turn Race itself into a scriptable object, that way we can create and save different races without needing to use an object or prefab of any sort. “TimedWaypoints.cs” does not need to be a scriptable object as it will never be in the scene at all, only used as a reference in our asset folder.

Step 1: Create a “RaceData.cs” script. This will create a new item in the asset menu that will make a scriptable “race” object for us, filename “Race”, MenuName “Data/Race”

Step 2: After creating and filling in all 3 races, change “QuestManager.cs” to reference “RaceData.cs” instead of the “Race.cs” script.

Step 3: Creating 2 new features. The first feature will be to add health to the player. If a bullet hits the player, they lose some health, and fail the race if all health is lost. The scriptable object will keep track of how much damage a bullet does each level. Second, we will give the player a boost option. While right mouse button is held, the player will move faster. The scriptable object will hold how much available boost the player has each level and how fast it replenishes.

Part 2 – Designer Focused Layout

At present, each field within the RaceData should be able to be modified by a designer, given each value has import to the race’s difficulty. In a larger data file however, a coder may wish to hide aspects the designer does not need to see, or aspects that could break the game in some way. As an example, if the data structure has reference to an “animator” that needs to be accessed by the code, you wouldn’t want a designer potentially changing that animator or removing it, so you would remove it from being visible. There will be an example of such a removed variable placed in the scriptable object. Further descriptions of each interface field and its design will be commented in the RaceData.cs script.

Part 3 – Gizmos and Handles

The Gizmos themselves act as a useful tool on their own as anyone working on the scene will now have a visual representation of where each of the waypoints are. What’s more, they can be modified to show which have been passed and which are next, or even a progressive color change to show the order from first to last. Adding in handles makes this all the more useful as these will let designers physical push and pull the waypoints around to create the scene, rather than having to rely on a guess and check variation. Each waypoint should have its own gizmo, along with a handle to move it around, actively changing waypoint values. White gizmos have yet to be reached, blue ones are completed waypoints, and green shows the next waypoint to reach.

Handles had to be placed within the QuestManagerEditor script rather than the RaceDataEditor for each to be drawn and moved properly, but otherwise can be dragged around to change the waypoints of each race.