Find The Source
A VR Maze and Puzzle game for the Samsung Gear VR
by Walter Thornton

Design

On the advice of others, I decided to use Unity to create my VR project, because it is a powerful game engine, free, and there seemed to be a great deal of information about it. Although this was the best choice, much of the information regarding VR support for Unity is outdated. When updating from v4 to v5, Unity incorporated native support for Oculus's VR package, where before v5, Oculus's tools needed to be imported manually. Due to this change, much of the available information on setting up Unity for VR was old, including Samsung's own development site.
'
Once I had all of the software installed and configured, including Java SE Development Kit, Android Studio with SDK, the Samsung Not 5 drivers, the Oculus signature file and Unity itself, I began my app. Knowing nothing about game engines, or VR at all, I spent a few weeks just exploring Unity and the gearVR. I decided to make a puzzle type game set on a destroyed and scorched planet. The first level, which I implemented for this project, is a maze which the player must traverse in order to get to a tree which serves as a portal to the next level, ideally another similar puzzle. Choosing the destroyed world setting, allowed me to be somewhat free in my choice of features and enabled my to use some very cool assets from the unity store. Although some of these assets are prefabs from the Unity store, I built and edited the maze and terrain.

The main camera, the OVR player controller, is a joint character controller and first person camera designed for VR. Although this C# script comes standard with Unity's v5 VR tools, I edited it to suit my particular game. I did not want the player to be able to jump, thus enabling the player to cheat during the maze. I also wanted my character to go faster than the standard and be able to head look without actual head movement. I discovered that virtual reality is actually a bit cumbersome sometimes, having to turn your head quite a lot, especially in the maze which contains many right angles. I had also began a script that allowed walking forward by looking at an object using a raycaster. I wanted to be able to use the app without a controller, and also to make it available fr the Google cardboard. This turned out to not work so well, as you need an actual actionable object to look at. In other words, this would not work well for exploring terrain that lacked many actionable objects.

Upon compiling, my major issue became lightmapping. My builds would hang in the middle. After much research, the problem is due to Unity v5's in depth lightmaps and baking. This, if done improperly, can be very computationally expensive and was therefore crashing my builds. Eventually I overcame this problem by changing the rendering and decreasing my resolutions. Finally I compiled using ETC2 (GLES 3.0) texture compression.

What's next

I would like to create a second level. Although not fully implemented, there is a script attached to the source “tree” that will “teleport” the player to the next level. The script is a scene change script, though I do not have a second level yet. It will be another maze, except in a subterranean environment. I would additionally like to add some system of clues to my mazes to give the player hints, and I would also like to create a coherent backstory to accompany the game.

Thanks for reading!