Lab Report
Lab 05 – UNET Networking
Darrick Hilburn
9/30/15

Introduction

In the early years of game development, players could gather around a console and play a multiplayer game. With the internet, players are now able to play together on their own consoles. Game networking is vital for modern multiplayer games.

In this lab, we will create a simple Networking example that showcases how to synchronize player data across games. We will write C# scripts that sync player data across a network in a Client-Server manner.

Methods

Three scripts were written for this lab: one for initializing client player data, one for synchronizing transform positions, and one for synchronizing rotations.

Before writing the scripts, two scenes were built: a main game scene and an initial menu scene. The initial menu scene contains the object that holds the network controller object, which contains a Network Manager script along with a Network Manager HUD script. The main game scene simply contains a large plain with a couple environment objects, in this case cubes, and several player spawn points.

This lab also required creating a Player Character prefab. We use the Standard Assets First Person Character object for the player. We add a capsule to the player along with a small cube to indicate facing direction. On the player, we disable the Character Controller and First Person Controller components. Within the player, we disable the camera and audio listener components on the First Person Character object and the Capsule Collider on the Capsule object. The three scripts are attached to the Player object when they are ready.

The Player Network Setup script enables most of the disabled components on the player object. It first checks if the player object is a local player before re-enabling all disabled components except for the capsule collider.

The player position synchronization script uses fixed updates to convey to the server where players are in the game scene. Player position synchronization uses a server command function to update the player position within the server and a client callback function to send the command. On fixed intervals, the player sends their position data to the server and interpolates the position of all other players in the scene so that players appear to move smoothly.

The player synchronization script is almost identical to the position script, only instead of working with positions, this script works with rotations.

Conclusion

I have learned through this lab basic methods for synchronizing player data across a server, namely the position and rotation. This is a good start for getting into networking since it's impossible to do networking without making the player controller objects run correctly. The concepts presented in this lab can be further expanded upon to incorporate more than just movement as well.