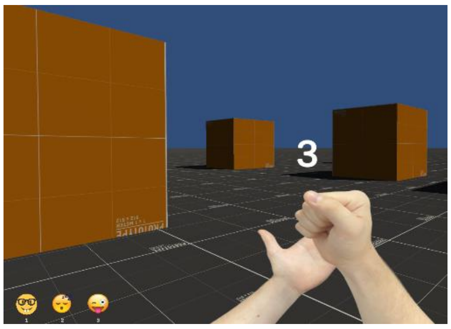
Robin Zeitlin | Network Course Assignment

Game Title -> RockPaperScissor+

Short Description / Summary

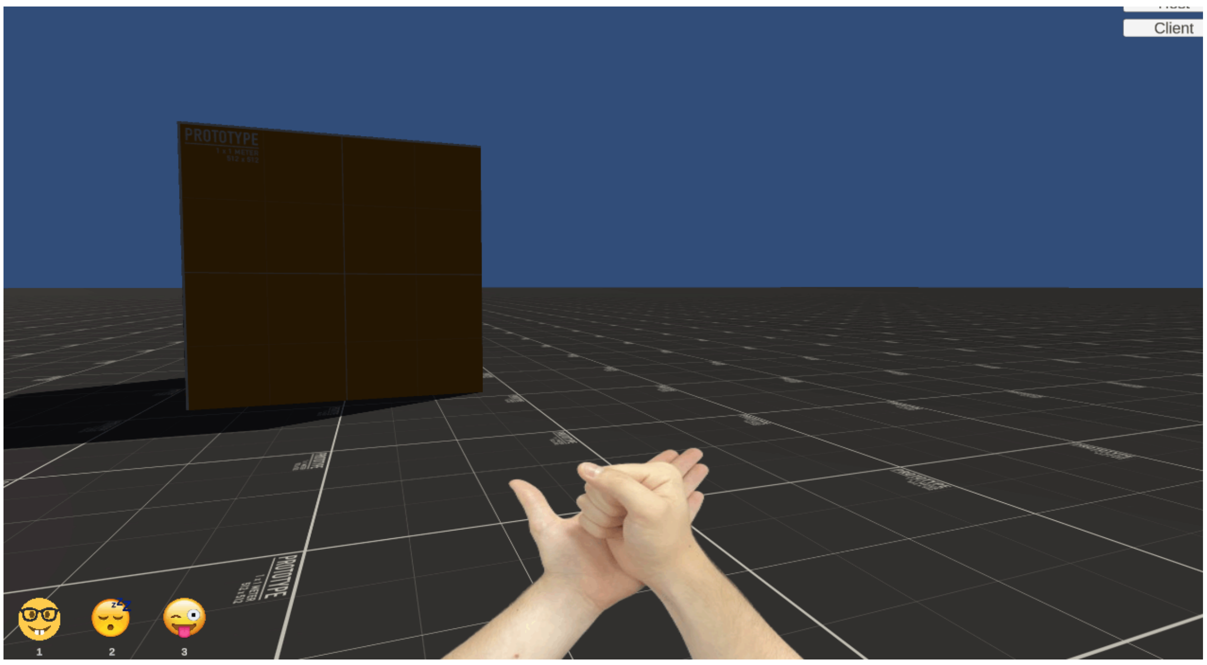
RockPaperScissor+ is an FPS arena shooter where both players charge up the shots by playing rock paper scissor. It utilizes Unity's Netcode to make the experience Multiplayer. The players can communicate with eachother through a series of "Emojis" or "Reactions" by pressing 1-3 on their keyboard to instantiate their reaction above their head. Making the shooter a more interactive experience.



Challenges during development

Due to my previous experience using Photon a Netcode framework very structurally similiar to Unity's I had an easier time picking said framework up and working with it. I did however face a couple of challenges.

The first goal for this project was to make a reaction wheel to let the player choose from the variety of "Reactions". Due to the way the netcode handles spawning objects I had a hard time getting it to work. In the end I managed to find out that instantiating a "Network Object" (in this case the reactions that would appear above the players head). Didnt work unless the object instantiating the object also was a "Network Object". Because I spawned the reactions everytime the player held right click, I felt like it would make too many frequent calls, instead I decided it would be worthwhile to make them keyboard input based.



To make the player shoot I used a Raycast from the camera. I then used a [ServerRPC] supported function on the object hit called "Die" (very self-explanatory name). This made it so the object hit on my side would be called simultaneously on both clients. If I would have taken a step back and made the Raycast shoot out simultaneously it would have caused all players to shoot out at once, which would lead to unwanted bad behavior.

Synchronization

<u>Synchronization was handled through the variety of components that the Netcode framework provides such as:</u>

Network Object – For objects that needed in anyway be synced.

Client Network Transform – To sync the transforms.

Network Rigid-body – To sync the Rigid-body / velocity of objects.

<u>There were also several key things to utilize when coding:</u>

Network Behaviour – Giving me access to a library of important variables such as one of the most important ones "IsOwner" which differentiates your player character from the other characters only letting you control your own. It was also utilized to deactivate your own Mesh and everyone else's Cameras and Meshes.

[ServerRPC] – Was a very important part in the development due to its ability to let me call functions on a server side. Making it possible to (as an example of many) call the "Die" function on the player hit by a "Raycast" on the server side which means that it's also called on everyone's client.

Reflection

Overall, I would consider this project a good learning experience, finally giving me reason to try out a new way of making multiplayer games. I also took some time and implemented stuff that I just personally enjoyed developing making it feel like I learnt more than just networking in this small project.

If I would have continued the development, I would have implemented spawn points, a score system, and a proper death state instead of the reset of position I currently have but due to other time limitations I unfortunately did not have time to spend more time on the project.