Network Programming Assignment

Tiger Martin

GITHUB REPO LINK: https://github.com/Laktostolerant/Arclight

I want to start off by saying that this is not my first networked game, I just did not have as much time to work on this one so I made it very basic while aiming to fill the criterias. If you want a more accurate representation of my network code abilities, I made an 8 player mini-golfing game that has a scoreboard and synced player positions and such. https://www.youtube.com/watch?v=MI_ipiAraf8

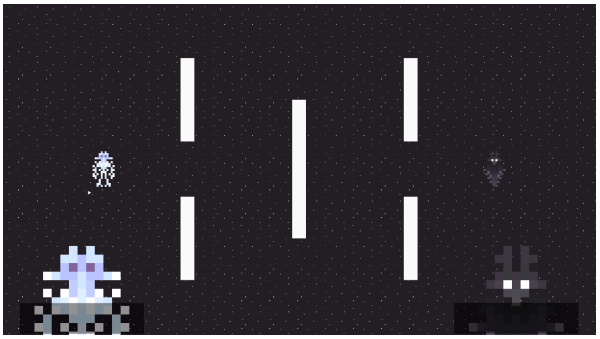
Game Design

Important to know: When starting the game as a host, the screen will go black. Join another instance as a client for the game to start.

I made a simple top down 1v1 shooter game that runs endlessly. There is no direct goal to the game as it was more of a demo if anything. I wanted to give the game a retro feeling so I designed the characters to be pixelated and shoot simple beams of light that destroy players and objects they touch, with a short cooldown. On top of the ability to shoot, the player can also place down “blockers” that spawn for both players and act as a single-use block against incoming attacks.

I wanted to keep it barebones so other than the players and the blockers, every other component in the game is single coloured. This includes the UI, the attack beams, the particles, the background and obstacles.

The gamemode itself is kept symmetrical, the players have the same abilities and the map is also identical, mirrored on each side. It is meant to be similar to an arcade game, but the stakes are kept low as there is no direct counter for deaths and there is no game over. The players choose themselves when they want to end the game.



Network Features

The players and their positions are synced on the network. Additionally, they are able to shoot beams of light that are shown on both clients. They also have the ability to place down “blockers” that block said beams. These blockers are also spawned on

the network, meaning that if a player shoots one to destroy it, it calls the server to tell it to be destroyed and despawned. Whenever a player dies, they also call the server to tell it to end the round and start a new one, teleporting the players to their spawn positions.

There is also a chat system implemented in the game, letting players have some form of communication between them. When a player presses “Q”, it displays a positive message, whereas it displays a negative message if a player presses “E”. The message displays for both players and decays automatically. The contents of the message are also synced. I made it so that the player sends over an int representing the message index, and the server then grabs said pre-defined message from a bank containing every possible interaction.

I originally had a health system for keeping track of player health, but I removed that as I wanted to keep the game endless. There is a network variable on the match manager that tells the player whether the current round is active or not. This way it also makes the host unable to move unless another player has joined.

For the teleportation, the players are allowed to teleport themselves to their spawn locations. This was because I ended up using ClientNetworkTransform due to the time constraints of the assignment. Given the nature of the game type, it is a negligible downside, as hacking serves no real purpose.

Challenges

The main challenge of this project was time constraint. There are a lot of things I wanted to do and could have done, if I had more time. The main issue was getting the players to their spawn locations. Using the normal NetworkTransform, I had problems teleporting the players and getting it to be synced for both players. Using NetworkTransform.Teleport is only allowed server side, but when I tried to teleport them using it in a ServerRpc, it refused to work and said that it was not allowed client side, which I was not even trying to do. I ended up cheaping out and switched to CNT because it took shorter to implement and had negligible differences on my game.

The main issue that persisted in the game is that players would occasionally have the same spawn location. ClientNetworkTransform fixed this by making the server tell the players where to teleport to.

There was a minor issue that caused fading in and out to not work, but that was caused by using LeanTween, which is async. This caused very inconsistent results. I ended up scrapping that feature and went for a coroutine, which worked well enough to fulfil its purpose.

Another issue that kept happening was that the player portraits and chat messages would sometimes be on the wrong side, but that was fixed by making the server manage more of the things such as which portrait belongs to who.

Reflection

I did not learn much new in terms of network coding, although I was able to adapt changes from Unity 5 to Unity 6 as there were quite a lot of things that were reworked or deprecated. An example of this is how the RPC system was reworked from being just ServerRpc to ClientRpc, the new system is a bit confusing but I am getting used to it.
Overall I feel like the game concept has some potential to it, if I were to continue development at some point. This would require more time, which I do not have at the moment.