Connect 4 – Documentation

Setup

The setup might be a bit confusing as I started working on this as a Steamworks project but realized that I might need a way to play it locally as well. It will start in “Steamworks” mode which connects to steam and requires two different steam accounts to play with eachother. To switch to local mode, you will need to go into the Scripts folder and find a script called DeveloperMode, in that script there should be a variable called LocalMode, set that true and go into the SetupScene and go to the NetworkManager game object. Make sure the Network Transport is set to null at first and then press the dropdown that appears and click on UnityTransport, if everything was correct, when you host a game, you should see helo on the right side of your screen.

Game design overview

The idea was to find a simple game idea to be made into a multiplayer game, I looked at a bunch of old games such as Pong, Asteroids and even some board games that could be easily made into a multiplayer game. I then ended up with Connect 4, it’s a very simplistic game that has a lot of room for improvements, such as a ranked mode, character and tiles customization as well as spectators being able to watch and many more. I decided to put these functions/features into a wish-list which I would implement if I had time.

The core game mechanic was simple, the game supports two players, each player sees a board infront of them, by hovering over one of the rows in the board and clicking the left mouse button, the player would place a tile and it would be the next players turn, this would continue until one of the players would get 4 tiles in a row in either horizontally, diagonally or vertically. After each win, the player who won would get one point and the game would continue until one of the players quit. Before going into a game, you would need to connect to another player via Steam or NFG (Netcode for Game Objects) if there was no steam connection (mostly for testing purposes). After connecting to a player, the host would have a button to start the match and play the game.

These descriptions below are for the wish-list functions. The character customization works as follows, you have a default mesh that you can pick from but you can also change the different pieces of the mesh such as the eyes, head and other facial features, when you then

join a lobby/game, you and the other player would see your newly created character sitting in front of your opponent. Ranked mode would calculate take place in a best of three game and after a game is over, there would be a system to check how well you did against your opponent or how many times you blundered and lost against the opponent and then add it to your current elo, so for example, if you had a really great game and you blocked a lot of wins from your opponent while also keeping the amount of moves you made to a minimum, you would get a lot of elo, however if the opposite happened, you would lose a lot of elo.

Implementation of network features

I started with all the classes that I would need which in this case was quite simple, I needed a Board class to take care of each individual players board, I would then make functions for placing tiles and calculate which tile would go to which row, so if a player chose to put a tile at column two, the server would calculate the row as it is authortative, the client would send the column it wanted the tile to go to and then the server would do the rest such as spawning the tile, making sure the client knows that the calculated row is correct, the server would also spawn the tile but they would individually calculate how fast the tile would go down to the specified row. I worked with Steamworks at first so the lobby system and connection are handled by steam, whenever a player hosts a lobby, it would setup the buttons as well the chat system and then when a new player joins, the host got a message saying that a player joined and a new button would appear for the host to start the game. While the client is joining, it would send a request to the server to receive and retrieve the customizations options that both players have chosen, so the client would send a request to the server with the data of the customization and receive the servers customization, afterwards we would spawn the player models according to the customizations options both players picked.

Challenges faced

Some of the most difficult challenges was actually the UI, I had a seperate scene where all the UI was handled called the UIScene, and it worked great until I realized that Netcode for gameobjects had a scene syncing function which tries to sync all players scenes with the host, so if the client has loaded the ui scene locally and he joined a session, he would get a duplicate scene with no data in it, so I gave up on that as I couldn’t fit a suitable solution to

fix that. I instead opted for doing it a roundabout way by having the lobby data sent to the game scene through a singleton object, so any data I would need from the lobby would be stored there and then accessed through the singleton.

Overall experience

Overall, this was a very fun experience, it taught me a lot about how difficult it actually is to make a multiplayer game even though it was a very small scale, I learned a lot about different ways to send data besides variables and also some dos and donts, I probably did a lot of donts because I felt like I wanted to do it my way and see what I can make with just documentation.