Assignment: Developing a Simple Networked Game Using Unity's NGO

Future Games

Student
Pablo Schmidt

23-08-2024

1. **Overview of the game design.**
2. **Implementation details of the network features.**
3. **Challenges faced and solutions implemented.**
4. **A reflection on the learning experience.**

1-Overview of the game design.

The game is based on the classic Galaxian game from 1979. It is a space-themed fixed shooter. The player controls a starship and the objective is to destroy as many of the enemy spaceships. The enemy spaceships spawn from 3 different positions and they follow the nearest player, if they collide with a player, the player dies, but the rest of the players still survive. There is a “team score”; all the players contribute to increase the score.

With time, the enemies spawn faster, increasing the difficulty of the game.

The game includes a chat to talk with other players.

2-Implementation details of the network features.

For this assignment specifically, we used the Netcode for GameObjects library built for Unity. It's the first time I worked on a multiplayer game using networking so it was very challenging and difficult.

Here I remark some of the most important parts of the network features:

2,a Synchronization

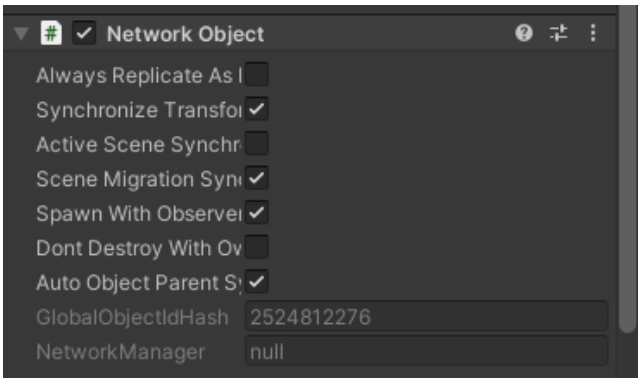
```
[Rpc(SendTo.Server)]
2 references
private void ShootRPC()
{
    NetworkObject bullet = Instantiate(objectToSpawn, aFirePoint.position, aFirePoint.rotation).GetComponent<NetworkObject>();
    bullet.GetComponent<NetworkObject>().Spawn();

    Rigidbody2D rb = bullet.GetComponent<Rigidbody2D>();
    rb.AddForce(aFirePoint.up * speed, ForceMode2D.Impulse);
}

[Rpc(SendTo.Server)]
1 reference
private void MoveRPC(Vector2 data)
{
    moveInput.Value = data;
}
```

This is an example of how to use `[Rpc(SendTo.Server)]` (Remote procedure call): An RPC is initiated by the client, which sends a request message to the server and this one executes a procedure with parameters provided by the client. Then, the server sends a response to the client and so on.

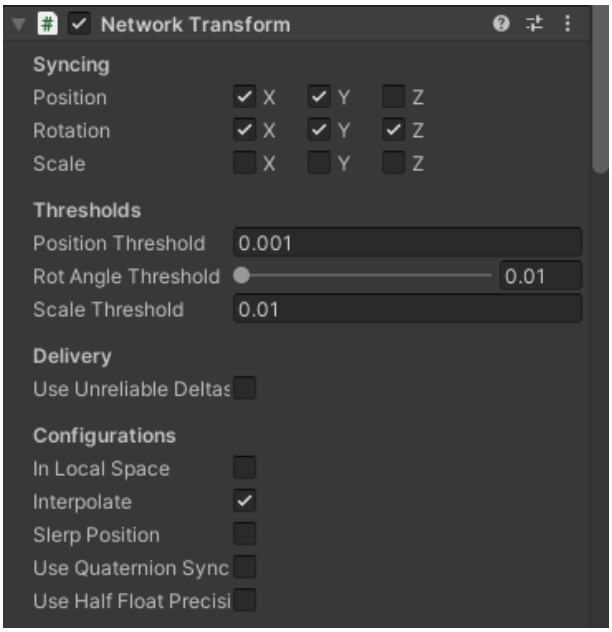
The `MoveRPC` method, for example, is marked with `[Rpc(SendTo.Server)]`, which ensures that the movement input is sent to the server, which then updates the player's position.

This was a big lesson to understand.

2,b - To ensure that what needs to be synchronized across all clients is synchronized, meaning that data is shared with all clients, it is necessary to add a “Network Object” component in the Unity inspector in each prefab.



2,c - To synchronize the position, rotation, and scale and make sure that these values are reflected across the network is necessary a NetworkTransform component .



2,d -Server authority

To ensure server authority is necessary to use `[Rpc(SendTo.Server)]` , for actions like moving and shooting. The risk of cheating is reduced and ensures that the game state remains consistent across all clients.

On the other hand, when the server invokes `[Rpc(SendTo.Everyone)]`, the data is sent to every client, and each client will execute the method.

2,e - Optimizations

After some testing, I noticed that the player prefab who was “joining” was moving with some lag. I could improve the movement of it by checking “Interpolate” in the unity inspector.

Without interpolation, the Joined player moved like “jumping” from one position to another. With interpolation, it is still slower in reacting to the input, but it moves smoother.

I would like to learn some more about optimizations in the future.

3- Challenges faced and solutions implemented.

3,a- The game was going to be a different game, it was going to be Player 1 vs Player 2, trying to kill each other, but since I couldn't not find a proper way to synchronize the mouse position when shooting , the solution was to add enemies and the players shooting in Y only. This solution of course brought more complications.

3,b - One of the first problems that appeared with the implementation of the enemies, is that they started to follow only one of the players, at that time I didn't figure out how to differentiate 2 players. The solution was to make the enemies follow any player that had the tag “Player” and who was the nearest.

3,c - Something that I wanted to avoid and was not planned, was to create a game manager, to count basically how many players were still alive before the Game Over condition. I could not avoid doing it.

3,d - Normally both players have different score counters, in my game for simplification of the networking tasks, the score is a “Team Score”.

4- A reflection on the learning experience

Making this game was totally different from the previous ones. At the beginning I didn't even understand what it was to have 2 players in 2 different screens, or how to synchronize them. Now I learned what is to send data to a server, and how this server is sending the data to the clients.

I did not know anything about Networks before this assignment, but I was always interested in them,since it is something basic for multiplayer games. Summing up, [Rpc(SendTo.Server)], Network Object, Network Transform were something that I took time to understand. Thanks to the mentor that helped me a lot, I could understand some basics and I am very sure I will use all this knowledge very soon.