Design and implementation report of Network Programming Assignment

Preface

The project complies with simple design principles: to make things work and make them work correctly. Multiplayer gaming experience over network is challenging in many technical aspects, e.g. anti-cheat, synchronization and network performance.

By using Unity’s Netcode, we don’t pay too much attention to technical details and other low-level implementations. Instead, developers can focus on best practices in game design and development.

Design of the game project

The main theme for this project is spaceships’ shooting. One player control one spaceship with automatic firing of bullets, and when a player hits another player’s spaceship with bullet, it scores.

Implementation of network features

NetworkManager

NetworkManager controls the “identity” of each player, whether it will be as host or client. By default, the first player will always join the game as “host”, while it maintains the player’s identity; the second player should be “client”, which will yield its authority to host/server, and keep synced with host.

Network Transform on spaceship and bullet

For keeping the “location” synced, the transform of each object must keep synced. For this game, too high tickrate will lead to excessive waste of processing power and network utilization. I prefer to leave it to 30 ticks per second, which should be enough for this simple game; as a reference, most competitive FPS games like CS2 and PUBG use 60 ticks or higher, LAN tournaments usually use 128 ticks.

Collision

Bullet and spaceship all have RigidBody2D, which is fine for local play. However, in Unity’s network architecture, especially in client-hosted hierarchy, server has absolute authority over physics.

However, this creates a small problem, the bullet fired by a ship will collision with the ship itself. This is not right in “real” world and of course it will create incorrect scoring of a ship. To handle this issue, two approaches are made:

1. Spawn point of bullet should be farther from ship’s collider, and make sure they don’t collide upon spawning.
2. When a collision happens, game logic will detect which player the bullet hit, and who is the owner of that bullet, then compare the ownership to decide who will score.

Object spawning

When a new player joins the game, the client’s spaceship will spawn and synchronize its position continuously with server. Bullets are also important objects that have to be synchronized with everyone.

In singleplayer game or server side, spawning the bullet seems rather easy, since the server can do it without too much concerns. But for client side, things become complicated. Client usually have no authority to spawn any network object in scene, and such can only be done by server RPC.

FireBullet() function will check if current caller is server or client. If it is server, it will skip any server-side RPC call, directly spawn the bullet; if not, it will invoke server RPC and the server will do it instead.

Challenges

Although I have extensive experience and knowledge in computer networks, writing codes with Unity still is hard for me. I am not quite familiar with how to accomplish something in Unity, although I have a clear idea how things (especially in low-level) are done. I struggled many hours to understand the idea and methodology of Unity.

(scoreboard)

Reflection

I suppose this game project is far from perfect, it has many places to further improve, but I have submitted this assignment later than the deadline and I understand that I am not doing well in this project. Should I have more time to further polish this work, I would make it more robust and potent in network performance.