Simple Online FPS

Oleg Loshkin

December 31, 2019

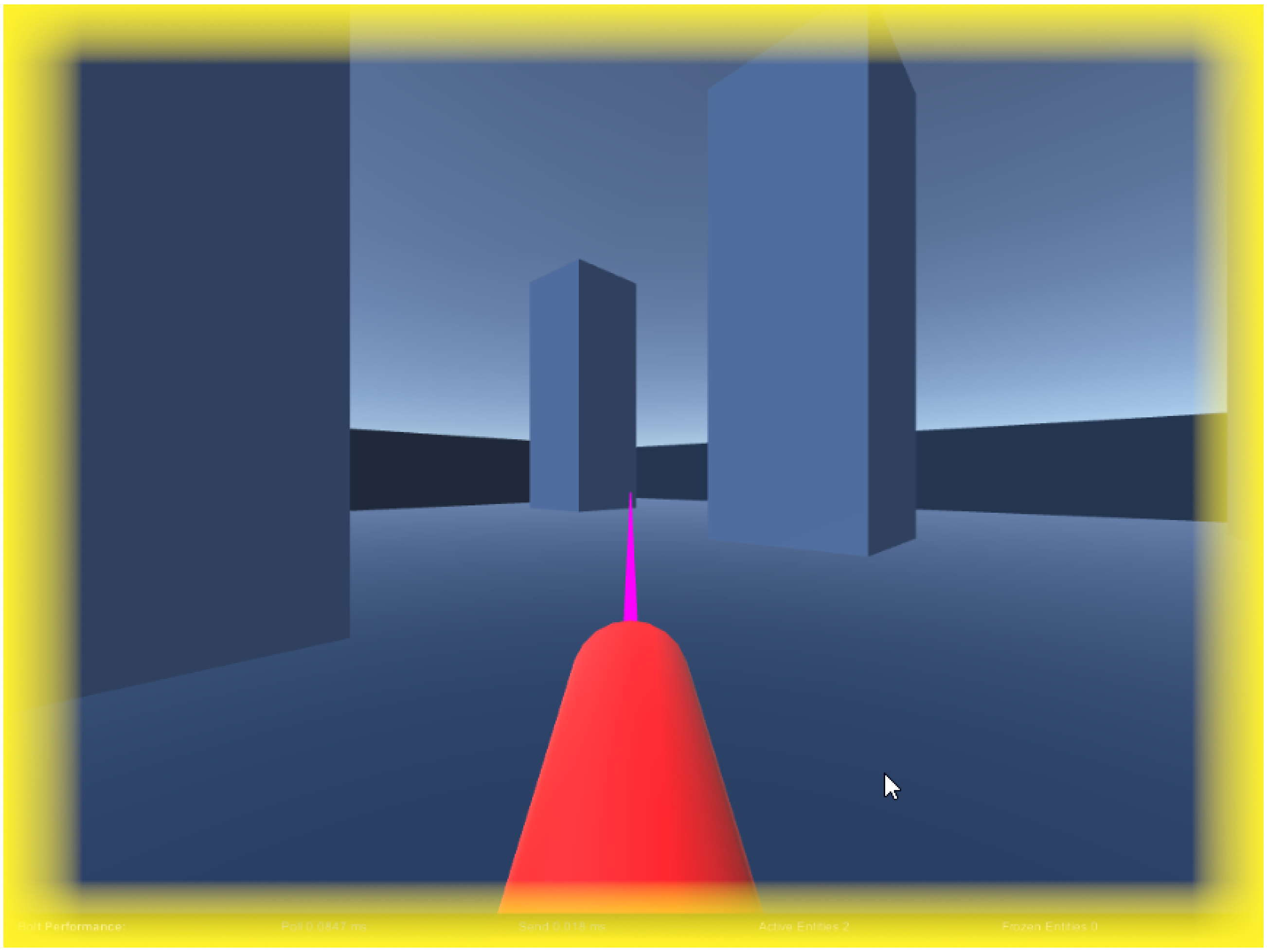
1 Introduction

This game is the result of the **SAE's repeat session for the GPR5100 Game Networking module**. **The aim of this project was to make a simple but functional online game** using Unity and a networking technology of our choice.

2 Technologies used

- **Unity 2018.4.9f1:**
  Latest available LTS version of Unity available at the time the project had started.
- **Photon Bolt:**
  High level game networking API for Unity. **The API provides some features that match perfectly the requirements of an online FPS**, namely lag compensating hitboxes and raycasting which negates the need to implement this complex feature manually.
- **Github:**
  Github was used for versioning. **The project's repository is available** here .
  The repository was temporarely split between two branches mid project to explore a possible migration to Photon Pun. This idea was abandonned and game was implemented using Photon Bolt.

3 Gameplay



- **The Lobby:**
  The player can either **create a new game session** and name it **or join a random game session currently in progress**. Upon the creation of a session, the host joins it as a listen server.
  Note that the session's name is currently unused as no list of sessions is visible anywhere, that sessions cannot be password protected and the number of maximum players per session cannot be established.

- **The Arena:**
  The arena is a **square area surrounded by walls and dotted with pillars that server as cover**. **Spawn points are located on the outer periphery of the arena close to the pillars**.

- **The Controls:**
  Players can **shoot with Left Mouse Button**, **look around horizontally only using the mouse** and **move around using WASD keys**.

- **The Fight:**
  A 2 second cooldown is enforced between each shot a player takes. **A player can take two hits from an adversary before dying**. No ways to regenerate health exist. **Upon death, the player respawns at the spawn point furthest away from the average position of all other players**.

- **The Feedbacks:**
  - A **shooting sound** is played upon any shot.
  - A **reloading sound** is played just before the end of the shooting cooldown and **the sound's end indicates the end of the cooldown**. Only the local player hears this sound.
  - **Getting hit** triggers a **grunting** sound from the player. Only the player being hit hears this sound.
  - **Upon landing a shot** on a player, **a ringing sound is played** for every player except the one being hit.
  - Upon **being hit once**, the player's **screen edges turn yellow**. **Being hit a second time** turns the **edges turn red**. A third hit kills the player and the health indicating overlay resets.

4 Movement

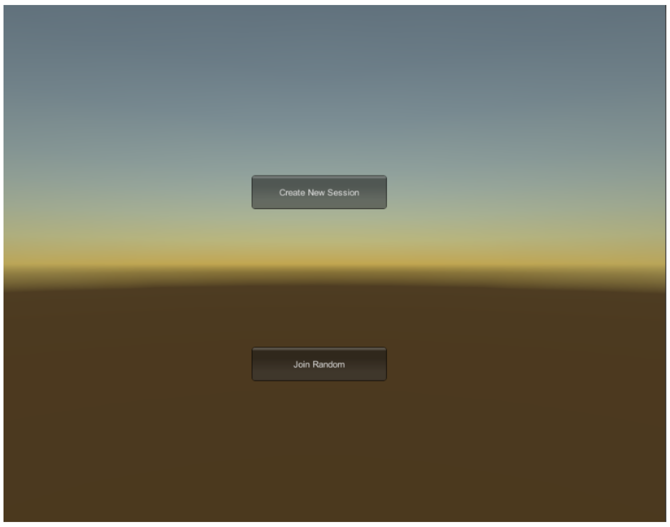
The movement in the game is **client authoritative** and uses **interpolation**. It is implemented using **Photon Bolt's Transform State and Commands**. While using RigidBodies is impossible in conjunction with Photon Bolt, Unity's "Character Controller" component has provided a good replacement for movement and collisions.

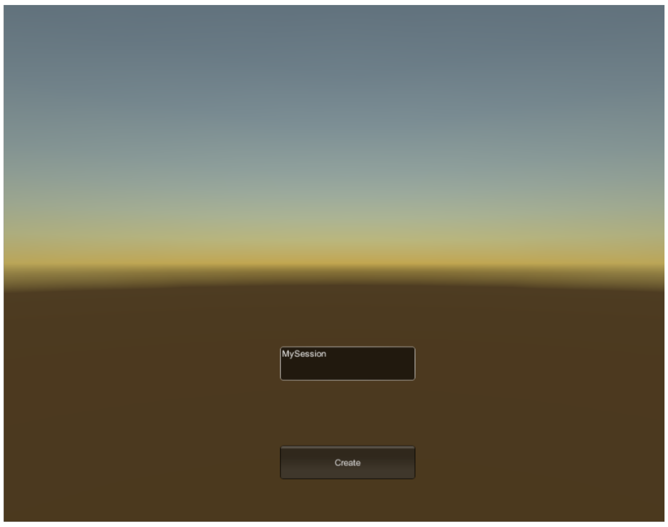
Client authoritativity not being ideal in online FPS games, a server authoritative movement was originally planned. However, after much efforts trying to make it work, Photon Bolt's rollback and lack of documentation of it's inner workings has made this idea too difficult to execute.

5 Hitscanning

Implementation of hitscanning was surprisingly easy using **Photon Bolt's Hitboxes and it's BoltNetwork.RaycastAll() method**. The only caveats that had to be worked around was the inability to filter the targets and the fact that the raycast does not interact with non-Bolt game objects like walls.

6 Lobby





The lobby is comprised of **two tabs**: a main one with two buttons for **creating a new session or to join a random one**, and **a tab for naming the new session before joining it**.
A different lobby was originally planned, one that would display a list of all current game sessions in progress with their name, the number of players and whether or not they are protected by a password. This could not be implemented for the following reasons:

Photon Bolt provides a static property under "BoltNetwork.Sessions" to allow the program to retrieve all ongoing sessions. However, this list of sessions is unexplicablely always empty. The idea of displaying a list of ongoing sessions had therefore to be abandoned.
Instead, Photon Bolt's BoltMatchmaking.JoinRandom() static method is used to join any ongoing game session, bypassing the sessions list completely and successfully finding any existing sessions.

A password protection and maximal number of players per session limit was not implemented either: while the documentation of Photon Bolt goes to length about the usage of it's BoltMatchmaking utility class, it does not provide any concrete examples of implementations of password protection and the documentation does not provide any explicit enough ways to handle session customization.

For these reasons, the lobby is as minimalistic as it is.

7 Feedback

Feedbacks for the gameplay have been implemented using **Photon Bolt's Events**. This allows the various feedbacks to be played on the programs of all players in a session when needed.
Albeit functional, these are still bug prone due to my lack of understanding of the Photon Bolt's API and the way it handles event sending and reception depending on the various confusing configurations that can be used.

8 Potential Improvements

- Clicking "Join Random" in the lobby without any running sessions locks the client into an indefinite state where they can't do anything and that they cannot escape without restarting the game. This is due to the bug described in the Lobby section that prevents the implementation of a check to verify the non existence of any sessions.
- The visual aspect of the game was completely neglected. Any assetwork would improve the player's experience.

- While the player can name a session when creating one, due to the above mentioned "BoltNetwork.Sessions" bug, no session list are displayed in the lobby and the name given by the player upon session creation is not used in any way.
- There are no kill to death ratio counters.
- The game does not perform any checks to verify that the session in progress is not playable (eg: less than 2 players in session).
- Sessions cannot currently be protected by a password.
- The host cannot define a limit on the number of players allowed in a session.
- Bots for testing the game's reaction to lag would have been a good addition to the project (eg. zigzagging, peeking from around corner, etc.).
- Having more than two players in a session creates auditive bugs due to the way network events are handled.

9 Summary

The game fulfils it's basic ambitions: **it is a barebone online FPS that successfully synchronizes players and performs hitscans for firing.**

Photon Bolt has proven very difficult to work with. Each new feature, however simple and small it may be required hours of debugging.
During the development, I've encountered bugs due to possibly inner workings of the API that rendered impossible the implementation of some key features, and weirder still, bugs coming from Unity's side that seemingly came from the interaction of Photon Bolt with the engine (eg: GameObject.FindGameObjectWithTag() breaking).
All these holdbacks made me consider switching networking platforms but doing so would have meant discarding much of the work I had done up to that point.
Despite a somewhat complete documentation provided by Photon Bolt, I've found myself needing more to investigate these strange bugs but could not find any.
Moreover, I was unable to get help to overcome these issues due to my choice of Photon Bolt as the networking technology, my colleagues having picked different ones for their projects and the senior students and teachers being unfamiliar with the API.

Due to all of this, **the game's current status is lacking in both features and polish** and I am left to wonder how much of the problems encountered have it's origin in my own abilities and how much in the framework's choice.