Simple Networked FPS

Sébastien Feser

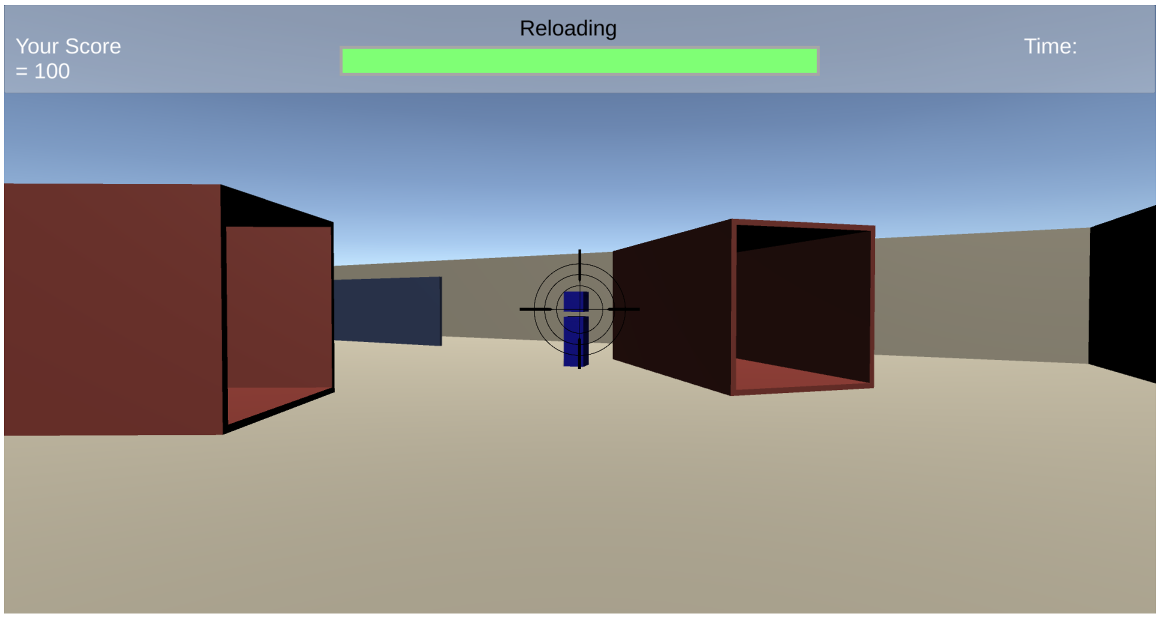
January 21, 2020

1 Introduction

Welcome to the technical document of the Simple FPS game. This game was created for the GPR5100.1 module of the Geneva SAE Institute Games Programming module. The goal of this module is to create a small multiplayer game.

This game was Developed with Unity 2019.3.0f3 with the Photon Pun 2.16 technology.

2 Game Mechanics



The player starts in a map. His goal is to shoot his opponents. Each kills grants 100 points to the player. The first player that reaches 1'000 points wins the game.

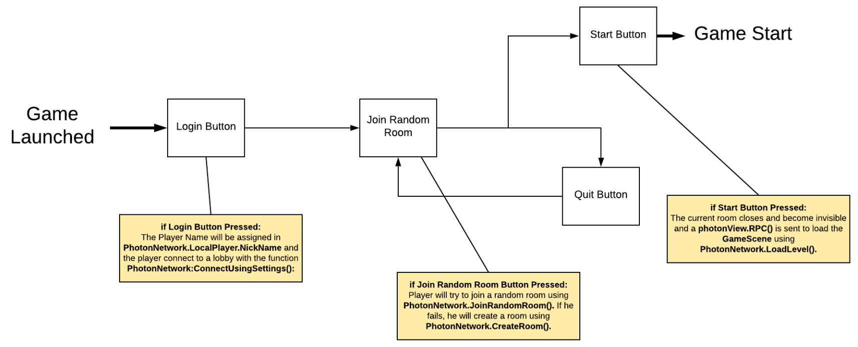
- **Shooting:** The player can shoot balls in the direction he aims. The balls are going straight forward. Each ball one shot the other player. When a player gets killed, it grants points to the player who shot the ball.

2.1 Controls

- **Movement:** The player can move using the keys W,A,S and D.
- **CameraRotation:** The player can move the camera with the mouse.
- **Shooting:** The player can shoot using the left click of the mouse.

3 Game Organisation

3.1 Lobby Organisation



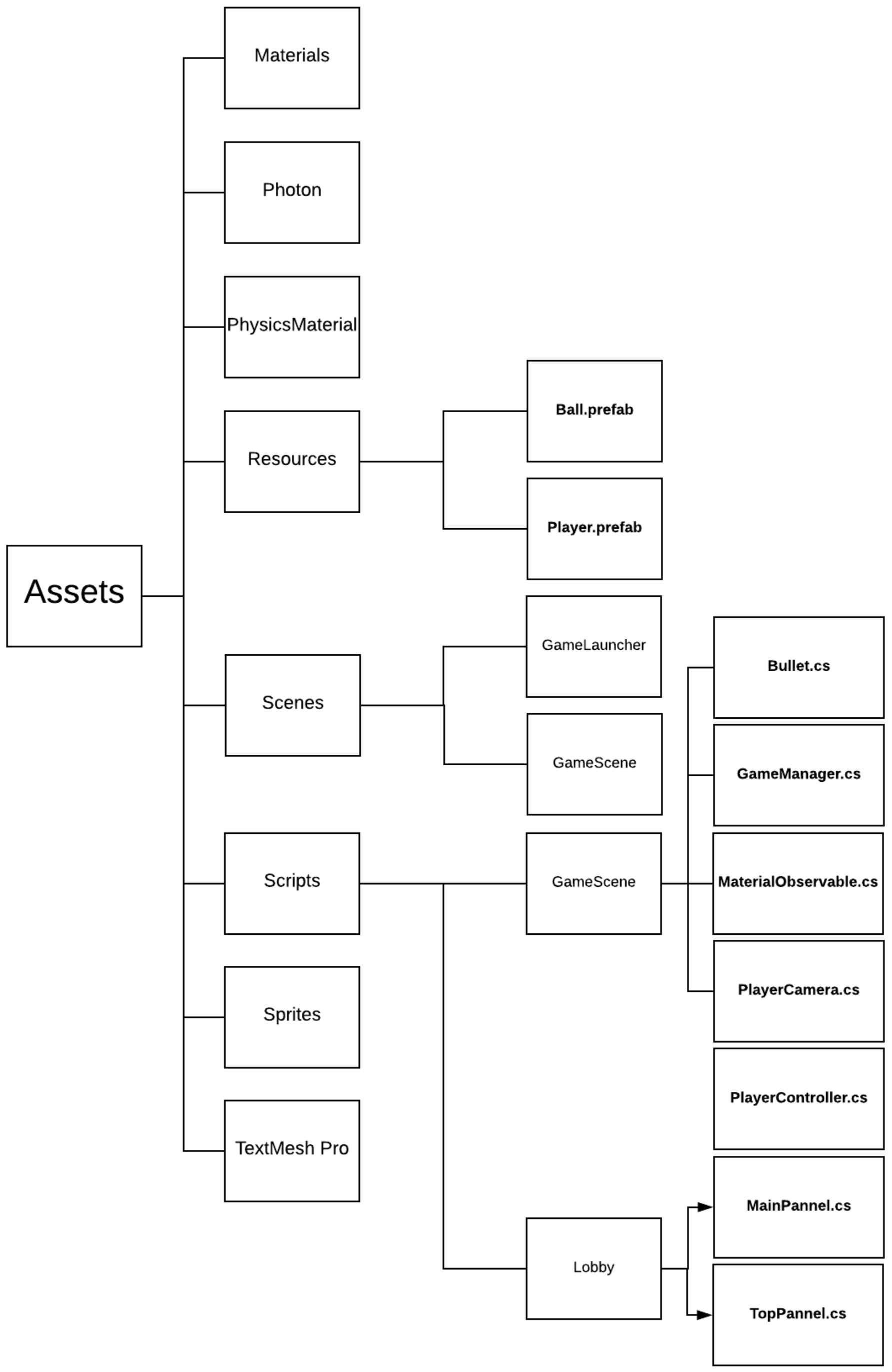
if Login Button Pressed:
The Player Name will be assigned in **PhotonNetwork.LocalPlayer.NickName** and the player connect to a lobby with the function **PhotonNetwork:ConnectUsingSettings():**

if Join Random Room Button Pressed:
Player will try to join a random room using **PhotonNetwork.JoinRandomRoom().** If he fails, he will create a room using **PhotonNetwork.CreateRoom().**

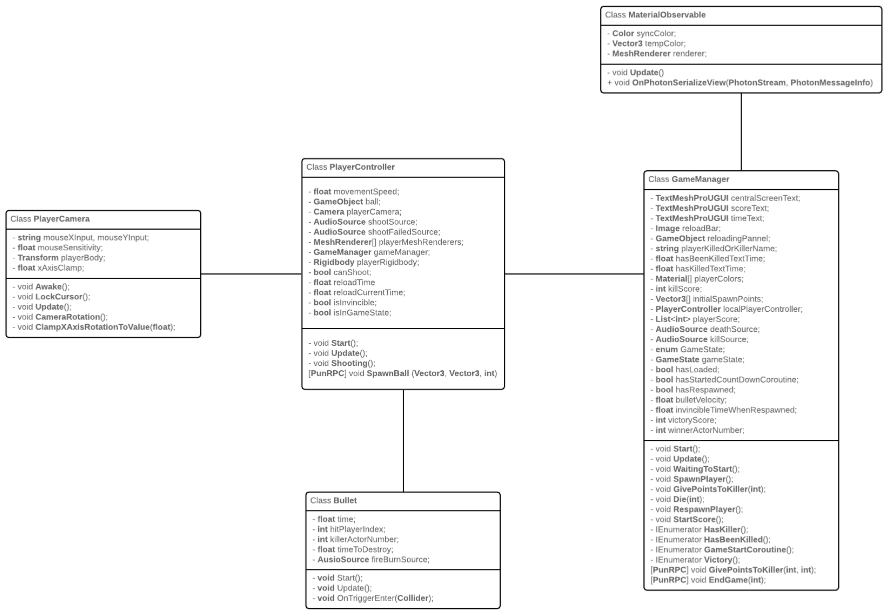
if Start Button Pressed:
The current room closes and become invisible and a **photonView.RPC()** is sent to load the **GameScene** using **PhotonNetwork.LoadLevel().**

4 Project Organisation

4.1 File Hierarchy



4.2 class UML


Class MaterialObservable
- Color syncColor;
- Vector3 tempColor;
- MeshRenderer renderer;
- void Update()
+ void OnPhotonSerializeView(PhotonStream, PhotonMessageInfo)
Class PlayerCamera
- string mouseXInput, mouseYInput;
- float mouseSensitivity;
- Transform playerBody;
- float xAxisClamp;
- void Awake();
- void LockCursor();
- void Update();
- void CameraRotation();
- void ClampXAxisRotationToValue(float);
Class PlayerController
- float movementSpeed;
- GameObject ball;
- Camera playerCamera;
- AudioSource shootSource;
- AudioSource shootFailedSource;
- MeshRenderer[] playerMeshRenderers;
- GameManager gameManager;
- Rigidbody playerRigidbody;
- bool canShoot;
- float reloadTime
- float reloadCurrentTime;
- bool isInvincible;
- bool isInGameState;
- void Start();
- void Update();
- void Shooting();
[PunRPC] void SpawnBall (Vector3, Vector3, int)
Class GameManager
- TextMeshProUGUI centralScreenText;
- TextMeshProUGUI scoreText;
- TextMeshProUGUI timeText;
- Image reloadBar;
- GameObject reloadingPannel;
- string playerKilledOrKillerName;
- float hasBeenKilledTextTime;
- float hasKilledTextTime;
- Material[] playerColors;
- int killScore;
- Vector3[] initialSpawnPoints;
- PlayerController localPlayerController;
- List<int> playerScore;
- AudioSource deathSource;
- AudioSource killSource;
- enum GameState;
- GameState gameState;
- bool hasLoaded;
- bool hasStartedCountDownCoroutine;
- bool hasRespawned;
- float bulletVelocity;
- float invincibleTimeWhenRespawned;
- int victoryScore;
- int winnerActorNumber;
- void Start();
- void Update();
- void WaitingToStart();
- void SpawnPlayer();
- void GivePointsToKiller(int);
- void Die(int);
- void RespawnPlayer();
- void StartScore();
- IEnumerator HasKiller();
- IEnumerator HasBeenKilled();
- IEnumerator GameStartCoroutine();
- IEnumerator Victory();
[PunRPC] void GivePointsToKiller(int, int);
[PunRPC] void EndGame(int);
Class Bullet
- float time;
- int hitPlayerIndex;
- int killerActorNumber;
- float timeToDestroy;
- AusioSource fireBurnSource;
- void Start();
- void Update();
- void OnTriggerEnter(Collider);

5 Networking

5.1 Photon View

The Photon View is a tool that has to be on a Networked Gameobject. It is used to send RPCs and to add other components like the Transform View Classic, the Rigidbody View and the OnPhotonSerializeView. It also allows to access to the Photon Actor Number that allows to differentiate every players.

5.2 Transform View Classic

The Transform View Classic provided by Photon Pun is used to interpolate the position and the rotation of the other clients gameObjects. It's placed on the Player prefab.

5.3 RigidBodyView

The Rigidbody view provided by Photon Pun is used to send the player speed to the other clients. It's placed on the Player prefab.

5.4 OnPhotonSerializeView

The function OnPhotonSerializeView provided by Photon Pun is used to synchronise the materials of the players in the MaterialObservable script.

5.5 RPCs

RPCs are used to update informations about the state of the game and to instantiate the bullet.

5.5.1 RPCs in PlayerController.cs

- **SpawnBall(Vector3 position, Vector3 velocity, int killerActorNumber):** Used to instantiate the bullet for each players at the same position.

5.5.2 RPCs in GameManager.cs

- **GivePointsToKiller(int killerActorNumber, int killedActorNumber):** Informs the killer who he has killed and informs everyone that the killer will get his points.
- **EndGame(int winnerActorNumberInRPC):** Inform everyone who won the game to switch to the end game status.

6 Conclusion

For this simple FPS, I was able to use the technology Photon Pun, it's RPCs, the views it provides and the OnPhotonSerializeView to synchronize my different players. The game is basic, you can walk, shoot and move the camera but it gives me a good overview of what I can do in photon pun.

6.1 Personal Impressions

I'm happy about this project, about the fact I finally finished it. I've learn some better techniques for the Networking with Photon Pun. I'm pretty sure that I've reached the objectives of this module.

One thing I regret is that I wasn't able to create a room selection list. I had mainly problems with unity and how the UI works. I've worked on everything but that, knowing it would've been a hard task to do but finally, I wasn't able to do it in times.