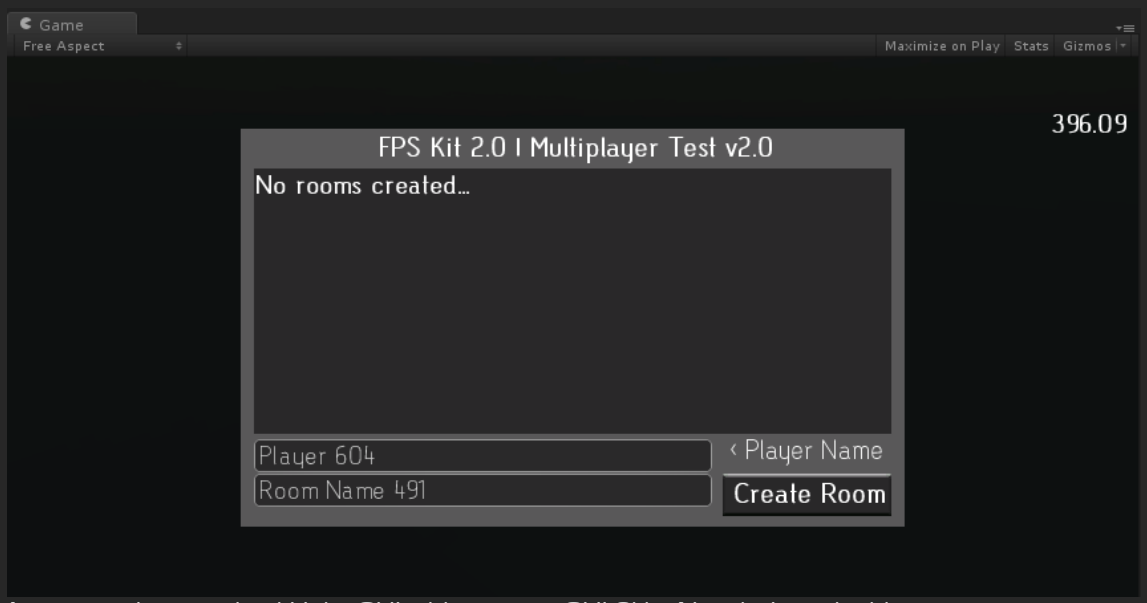
FPS Kit | Version 2.0 + Multiplayer

- This documentation is called to show user the main aspects of Photon Cloud programming and explain the way how FPS Kit 2.0 was integrated with photon cloud service. Package also contains complete Photon Cloud documentation. To follow this steps, you need have basic-medium programming knowledge (C#);
- NOTE: Before starting make sure you setup Photon cloud ID by registering here https://cloud.exitgames.com/. After registration you will have free plan for 20 player and personal ID which you have to paste in Unity 3D Photon Cloud panel.

Create, Join, Leave Rooms, Spawn Player

- So the first thing we start from is connection menu (aka Main Menu). There we will connect to Photon Cloud, create new room or connect to existing rooms;
- FPS Kit 2.0 Multiplayer contain example script called ConnectMenu.cs. This how it looks:



As we see its standard Unity GUI with custom GUI Skin. Now let's go inside ConnectMenu.cs and see how it was done;

- First of all we need to define our script to work with Photon, for this we add Photon.MonoBehaviour instead of just MonoBehaviour:

```
public class ConnectMenu : Photon.MonoBehaviour {
```

- After previous step is done we could start edit our script. First we need to connect to Photon Cloud server to be able create/join rooms etc. For this we add next lines into void Start(), Awake() or Update() (I put it into update since sometimes connection could be interrupted, and back to stable again so script trying to connect every frame) :

```
        //Connect to Photon Cloud
        if(!PhotonNetwork.connected)
            PhotonNetwork.ConnectUsingSettings("v1.0");
```

- Now we could move to GUI content. Here we check if we connected or not. If we connected to Photon Cloud, display GUI content:

```
        if(PhotonNetwork.room != null && PhotonNetwork.connected)
            return;
        GUI.Window (0, new Rect (Screen.width/2 - 250, Screen.height/2 - 150, 500, 300), connectMenu, "FPS Kit 2.0 | Multiplayer Test v2.0");
```

- To create room in Photon use next line:

```
                    PhotonNetwork.CreateRoom(newRoomName, true, true, 20);
```

Part from Photon documentation (which also included in this package)

Creates a room with given name but fails if this room is existing already.

- static void CreateRoom (string roomName, bool isVisible, bool isOpen, int maxPlayers, Hashtable customRoomProperties, string[] propsToListInLobby)

- After room is created, next function is called:

```
    void OnJoinedRoom(){
```

Here you can setup everything you need, such as load map, spawn player etc.

- Let's assume we created new room, and want to spawn our player somewhere. FPS Kit 2.0 Multiplayer gives next example. After we joined the room, OnJoinedRoom () were called, here how it looks:

```
    void OnJoinedRoom(){
        print ("Joined room: " + newRoomName);
        connectingToRoom = false;
        transform.GetChild(0).gameObject.camera.farClipPlane = 600;
        pmn.enabled = true;
        mc.enabled = true;
        pmn.isPaused = true;
        connectingToRoom = false;
        this.enabled = false;
    }
```

Here we setup everything we need, like enabling spectator camera, enable RoomMultiplayerMenu.cs (This one is used to control gameplay inside created room).

- So let's move to RoomMultiplayerMenu.cs which handle player spawning and pause menu. To spawn player we use PhotonNetwork.Instantiate():

```
PhotonNetwork.Instantiate(playerPrefab.name, spawnPoints[temp].position, spawnPoints[temp].rotation, 0);
```

Part from Photon documentation:

Instantiate a prefab over the network. This prefab needs to be located in the root of a "Resources" folder.

- static GameObject Instantiate (string prefabName, Vector3 position, Quaternion rotation, int group, object[] data)

- To find all existing rooms use:

```
RoomInfo[] allRooms;
allRooms = PhotonNetwork.GetRoomList();
```

The count of rooms currently in use. When inside the lobby this is based on PhotonNetwork.GetRoomList().Length. When not inside the lobby, this value updated on the MasterServer (only) in 5sec intervals (if any count changed).

- To join certain room, use:

```
//Join a room
PhotonNetwork.JoinRoom(room.name);
```

Join room by room.Name. This fails if the room is either full or no longer available (might close at the same time).

- static void JoinRoom (string roomName)

- To leave room, use next line:

```
PhotonNetwork.LeaveRoom();
```

Here is part from documentation:

- static void LeaveRoom ()

Leave the current room

Player position sync

- To be able to sync character position over network, use OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)

```
void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info){
```

See example scrip PlayerLocationSync.cs

Weapon shot sync

- Weapon sync system consists in passing RPC messages over network from local to remote player. This system contain 2 scripts WeaponSync.cs and WeaponSync(Catcher)JS.js. WeaponSync.cs play a role of WeaponScript on remote player, it receives messages from WeaponSync(Catcher)JS.js, on the other hand WeaponSync(Catcher)JS.js receives messages from WeaponScript.js, such as Fire and Reload.

For better understanding of each script, try to open them and investigate carefully, there are a lot of explanation comments. Also check PhotonNetwork-Documentation.pdf.

Documentation by Max A.V. (NSdesignGames)