Photon PUN2 with Realistic Car Controller V3.4

# Importing Packages

First, download and import **Photon2**. Pass your AppID to Photon setup (expalined below), and then import “**RCC\_PhotonNecessaryScripts**” in **Scripts/Photon** folder. Now you can test the Photon demo scene.

There should be a new scene named “**RCC City Photon2**” in **Demo Scenes** folder after importing the integration package. Same scene with regular city scene. Only difference is, this scene has **RCC\_PhotonManager** with script attached.

Well, it’s quite easy I guess. If you don’t know anything about Photon, or even multiplayer based game, here is the events for how it works;

**1 –** First, you have to connect to server. In this case, our server is **Photon Cloud Server**.

**2 –** If your connection to server succeeded, you will be in lobby. Now you can create your room, or join someones room, or join randomly room here.

**3 –** Congrats, now you are in an online multiplayer based room in realtime.

As soon as you have imported it to your project, it will ask your **AppID**. Pass it. And now you are ready to develop your realtime based multiplayer levels. It’s free. And of course you are limited with 20 CCU.

Photon2 has many simple methods in their API. It’s extremely easy to understand. Let me explain how the demo scene works;

Demo scene has a gameobject with script named “**RCC\_PhotonManager**”. This script handles multiplayer section of the scene. Uses these methods (You can find all methods from Photon2’s docs);

**Photon.Pun.PhotonNetwork.ConnectUsingSettings ();**

We are connecting to the server first. We can listen which connection status we are on in OnGUI() method. Like this;

GUILayout.Label("State: " + Photon.Pun.PhotonNetwork.NetworkClientState.ToString());

Once connection to the server established, we have to join lobby.

**Photon.Pun.PhotonNetwork.JoinLobby ();**

As soon as we are connected to lobby, we want to join a random room by;

**Photon.Pun.PhotonNetwork.JoinRandomRoom();**

If it fails, this means there are no any active other room. We are creating the new room by **Photon.Pun.PhotonNetwork.CreateRoom(null);** This method needs room name. I didn’t use it, because there are no any room list in demo.

I take a string that belongs to player here. And set it by **Photon.Pun.PhotonNetwork.NickName = name;** Enabling/disabling few UI gameobjects depends on connection state. That’s basically, how the demo scene works.

For vehicle sync, each vehicle has **PhotonView.cs** and **RCC\_PhotonNetwork.cs**. These scripts are necessary for each vehicle. **RCC\_PhotonNetwork.cs** is observed by **PhotonView.cs**.

**RCC\_PhotonNetwork.cs** is synchronizing all control inputs, transform position, rotation, and rigid velocity smoothly. If vehicle is our vehicle, it will broadcast your data to the server. If vehicle is not our vehicle, it will receive all data from server.

These vehicles are not instantiated or destroyed with regular **GameObject.Instantiate** or Destroy. You have to do it with **PhotonNetwork.Instantiate** or **Destroy**. Unfortunately, it won’t work with your prefab. It accepts only strings for your vehicle. That means, it will use **Resources** folder for accessing your vehicles. Your vehicle prefabs must be at **Resources** folder. Therefore, there are two canvases in resources folder. One of them is using **RCC\_Demo.cs**, other one is using **RCC\_PhotonDemo.cs**.