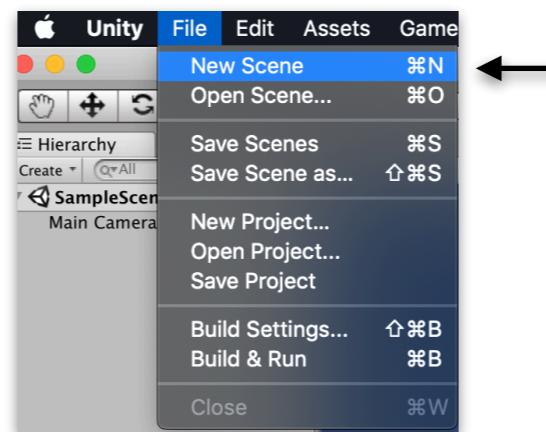


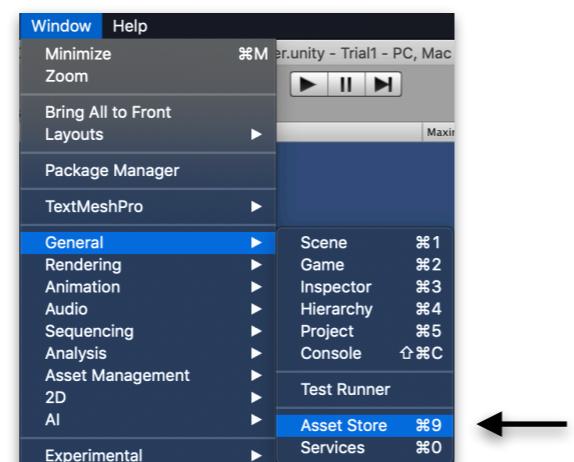
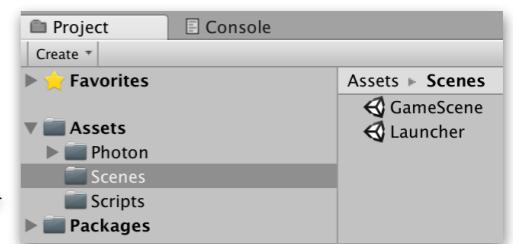
Unity Project Setup.



- Download the latest unity version. (I am using Unity 2018.2.16f1)
- Create a 2D unity project.
- In editor Create a new Scene.



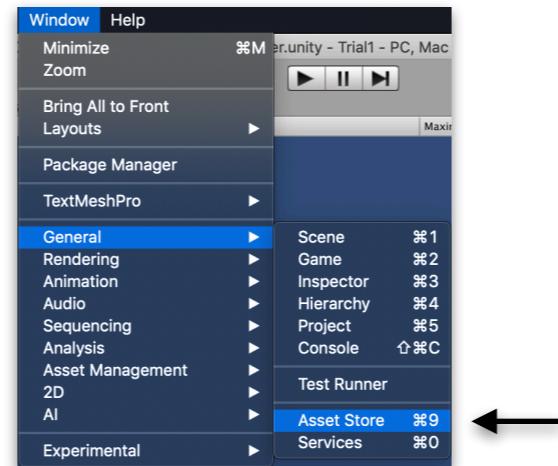
- Save the scene and name it 'Launcher'
- Repeat the above two step and create 'GameScene' →
- Double click 'Launcher' scene to open it.
- Open Asset store in unity.



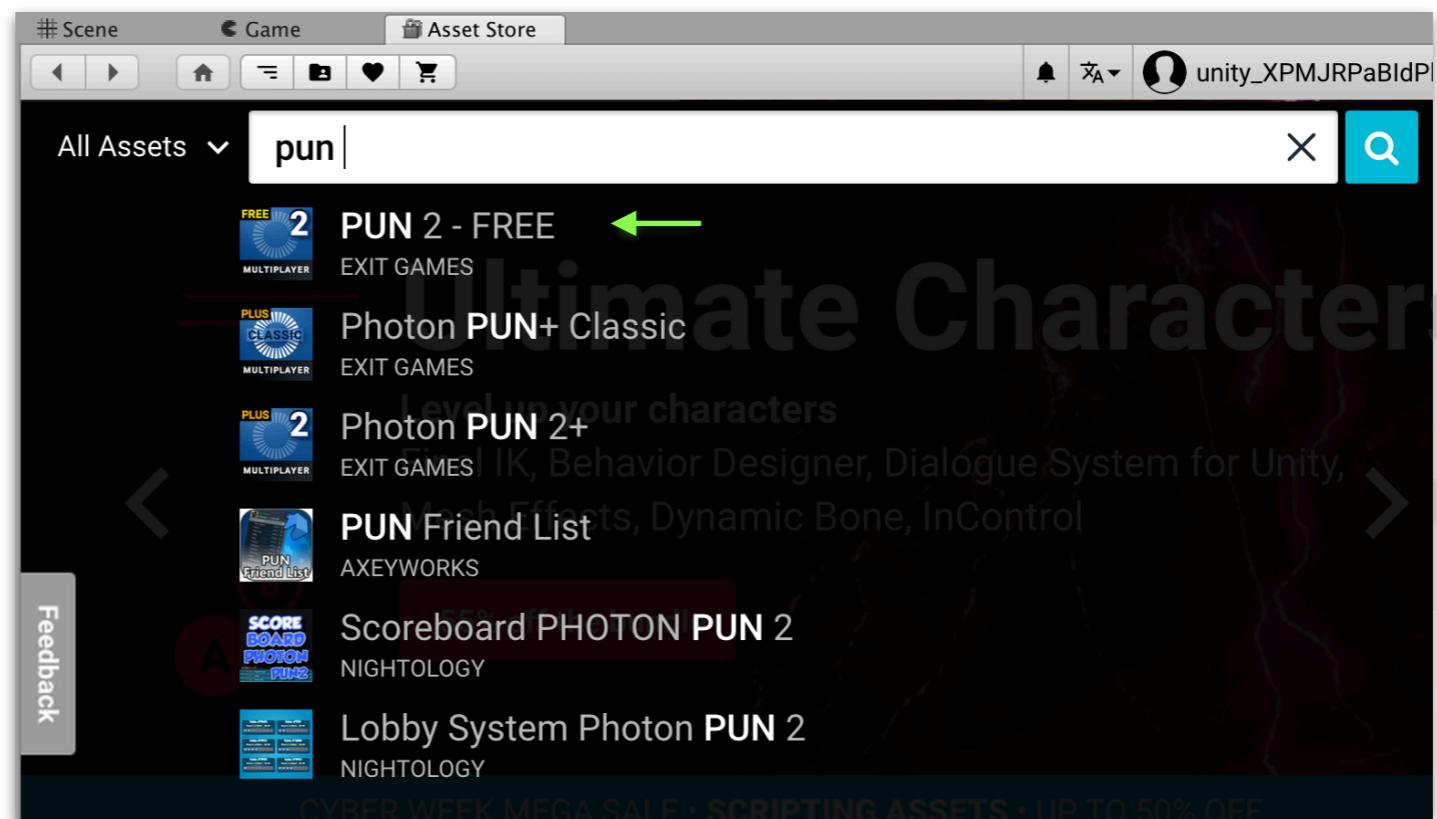
Photon Setup



- Open Asset store in unity.



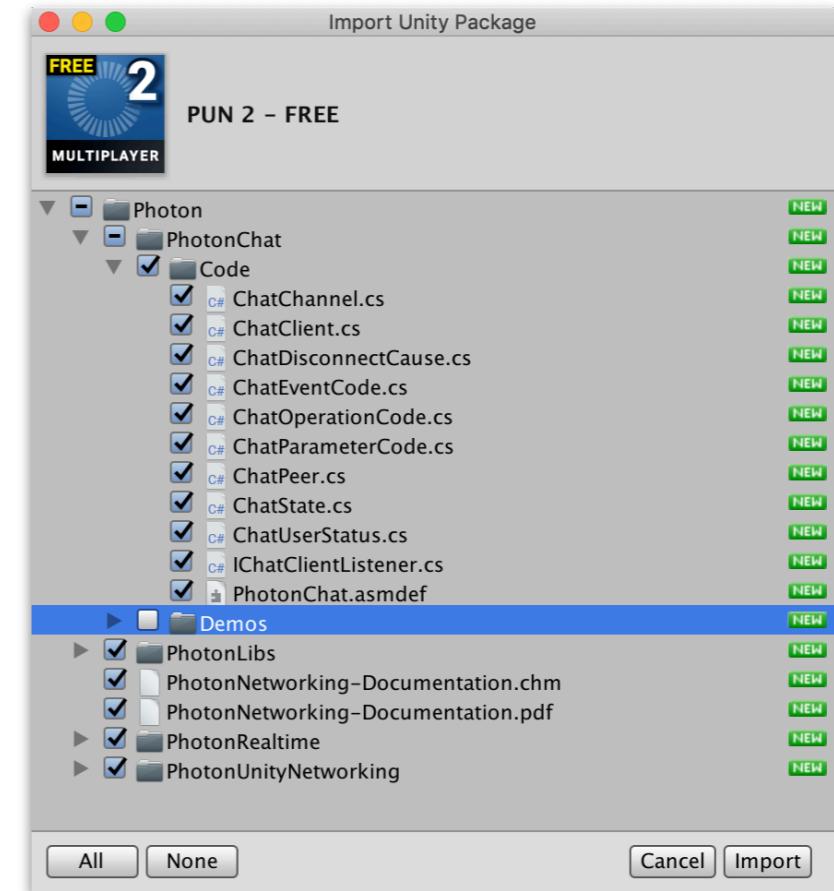
- In Asset store search for Photon plugin Or simply type 'pun'



Photon Setup



- Import Photon plugin to your project



- After Import the PUN Setup Wizard will auto launch.



- Register your mail id on Photon website to access its dashboard.

<https://dashboard.photonengine.com/en-US>

Photon Setup



- Create a new App in photon dashboard.

Your Photon Cloud Applications

Show in Status Sort

All Apps Active Pea

CREATE A NEW APP

Create a New Application

The application defaults to the **Free Plan**. You can change the plan at any time.

Photon Type *

Photon PUN

Name *

GDC2018Workshop

- Choose Photon PUN and provide a name (Give project name)

PUN 20 CCU

GDC2018Workshop

App ID: 7dd1b639... Peak Current Month 0 CCU →
Peak Previous Month 0 CCU
Rejected Peers 0

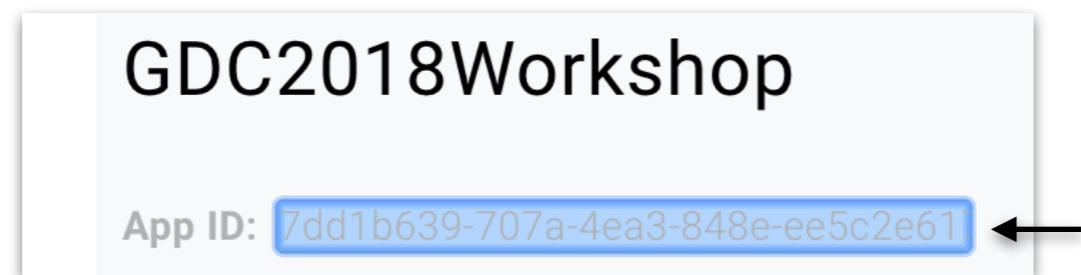
ANALYZE MANAGE CHANGE CCU ADD COUPON

On Success you can see your new app like this.

Photon Setup



- **Copy the AppID**



- **Go back to unity editor and paste the AppId in to the field specified. And Click Setup Project.**



Photon Setup



- Hurray ...

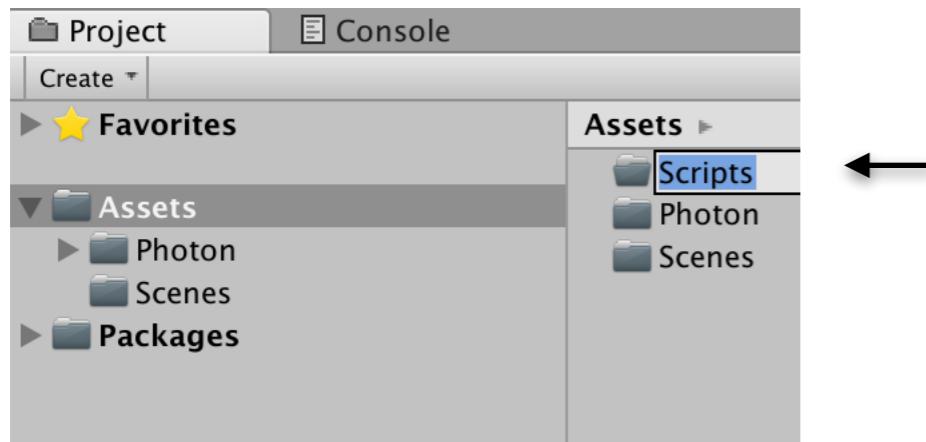


- Close the wizard view

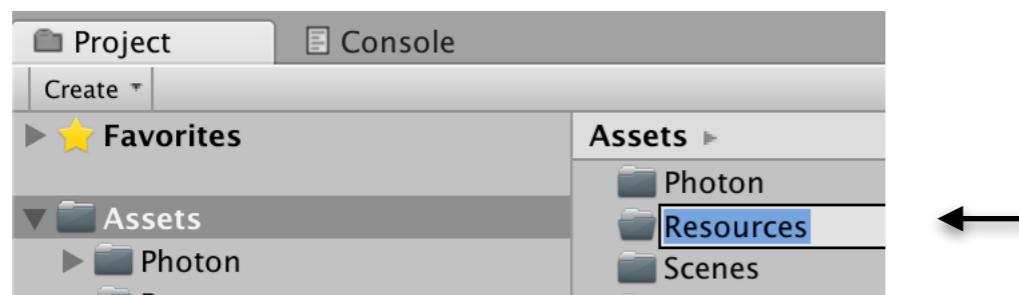
Asset Folders



- Create a ‘Scripts’ folder under ‘Assets’ to store all our source code.



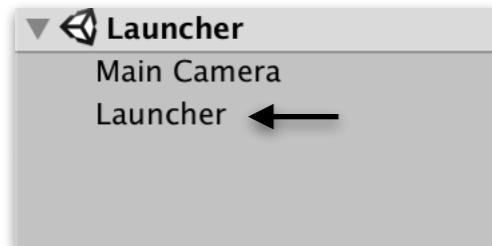
- Create Resources folder for Photon serialisable objects.



Launcher Script



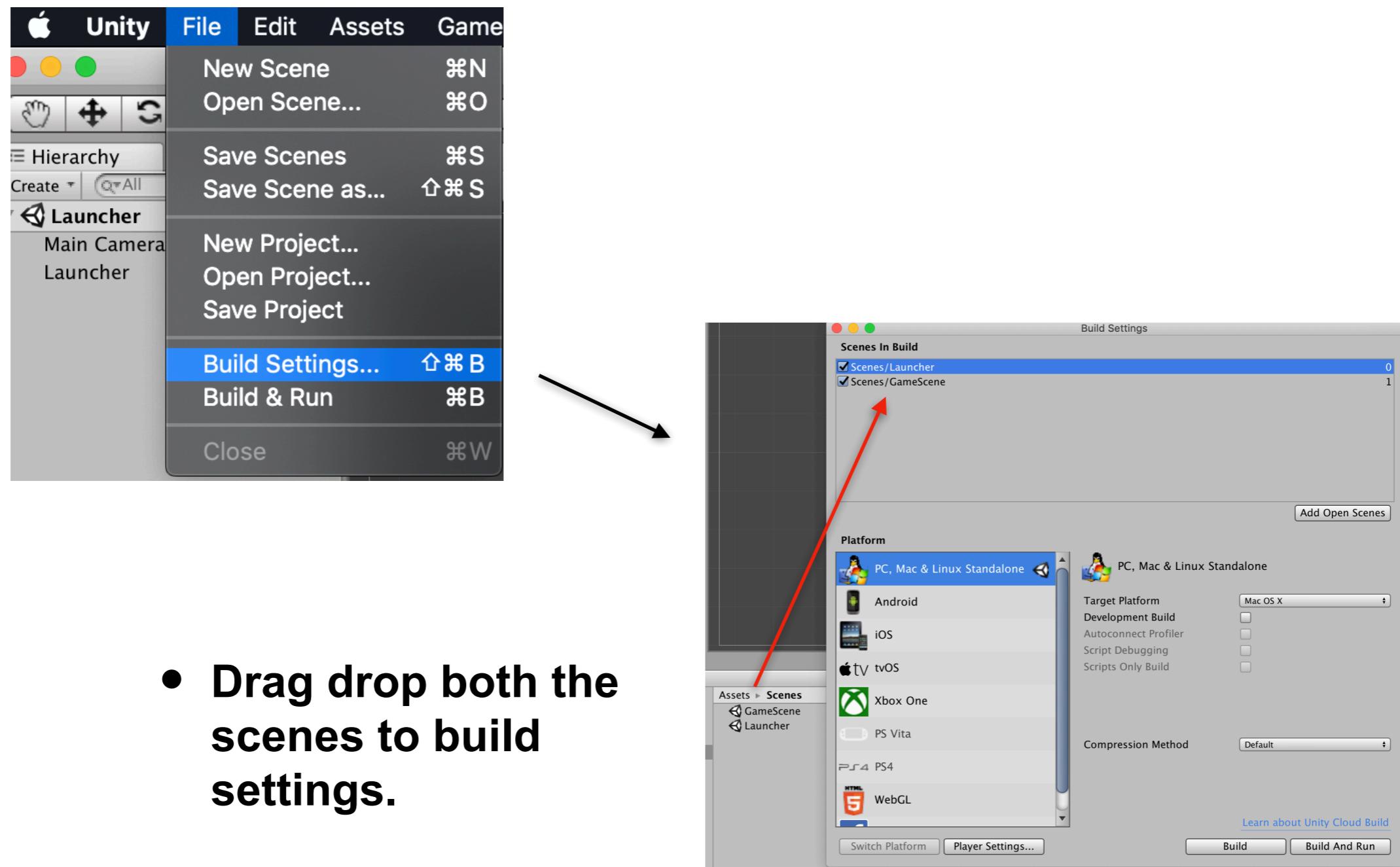
- Create a ‘Launcher.cs’ script inside ‘Scripts’ folder.
- Create child name ‘Launcher’ in hierarchy view and attach ‘Launcher.cs’ script to it.



Scene setup



- Add both Launcher and GameScene to build settings.



Checking PUN Integration



- Now Save and Run the project. Make sure that you are in ‘Launcher’ scene before running the project.
- You can see the following output in console on success.

```
[!] [21:24:52] Connecting...
UnityEngine.Debug:Log(Object)

[!] [21:24:53] Received your UserID from server. Updating local value to: e42ccfda-4a64-4990-b709-3160edce7f7b
UnityEngine.Debug:Log(Object)

[!] [21:24:53] PUN Basics Launcher: OnConnectedToMaster() was called by PUN. Now this client is connected and could join a room.
Calling: PhotonNetwork.JoinRandomRoom(); Operation will fail if no room found

[!] [21:24:53] PUN Basics Launcher:OnJoinRandomFailed() was called by PUN. No random room available, so we create one.
Calling: PhotonNetwork.CreateRoom

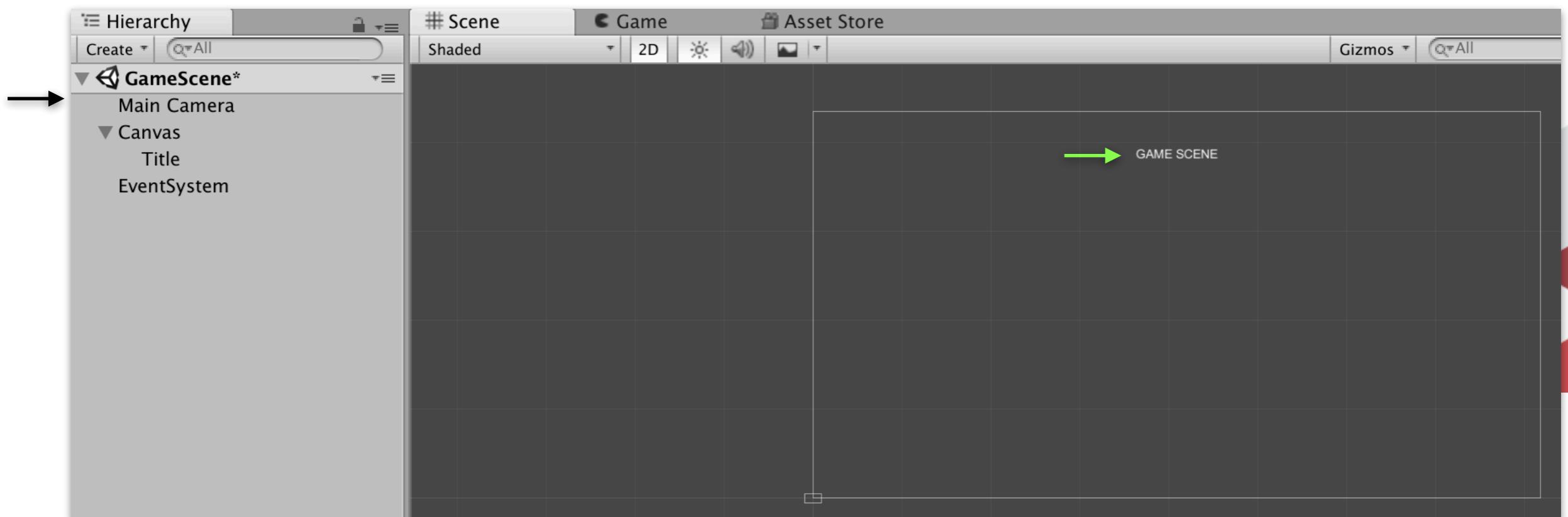
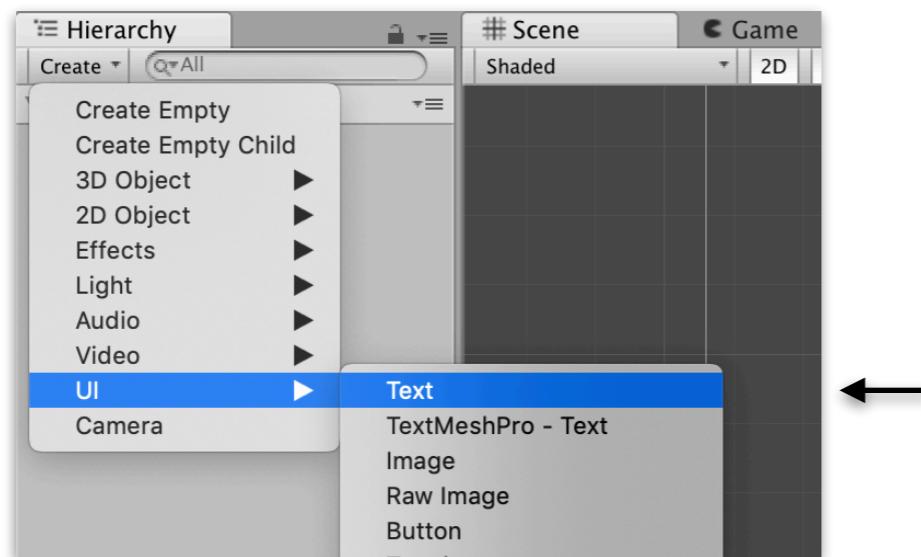
[!] [21:24:54] PUN Basics Launcher: OnJoinedRoom() called by PUN. Now this client is in a room.
From here on, your game would be running.

[!] [21:24:54] We load the 'Room for Player 1'
UnityEngine.Debug:Log(Object)
```

Game Scene



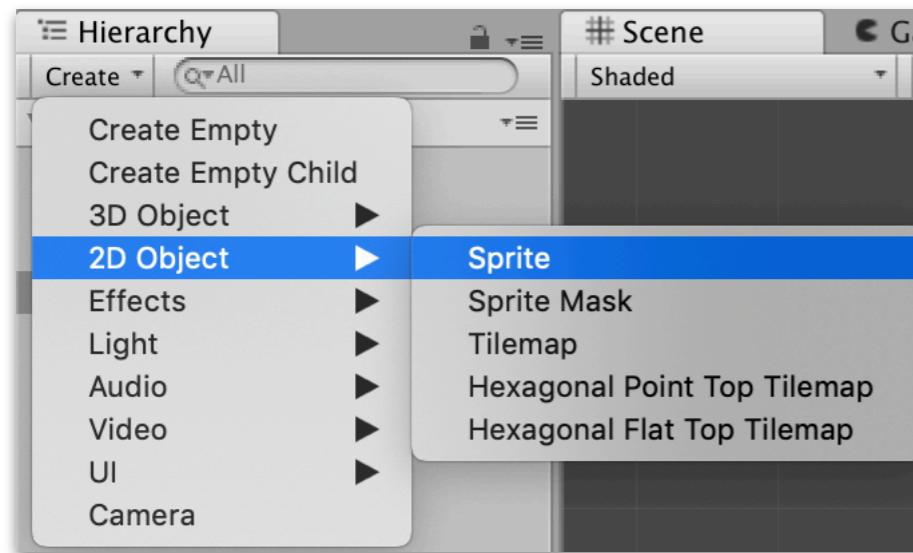
- Create a text in hierarchy view for our title



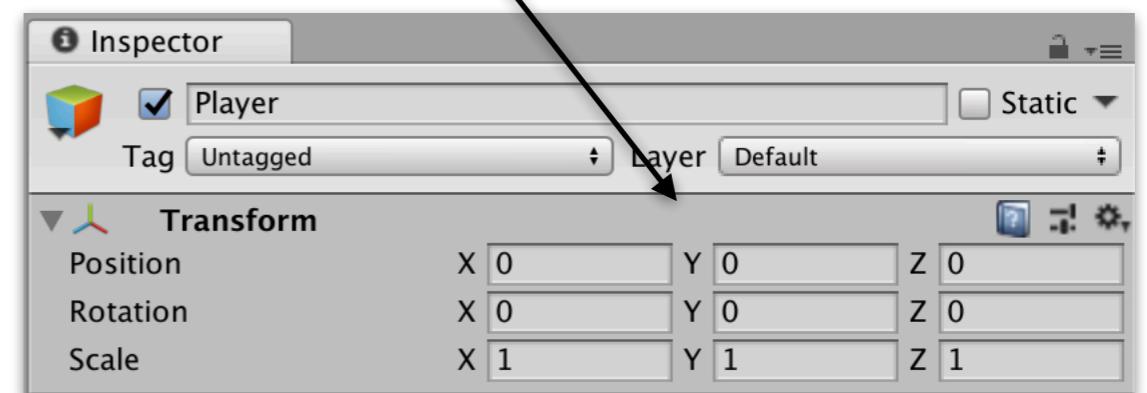
Create our player



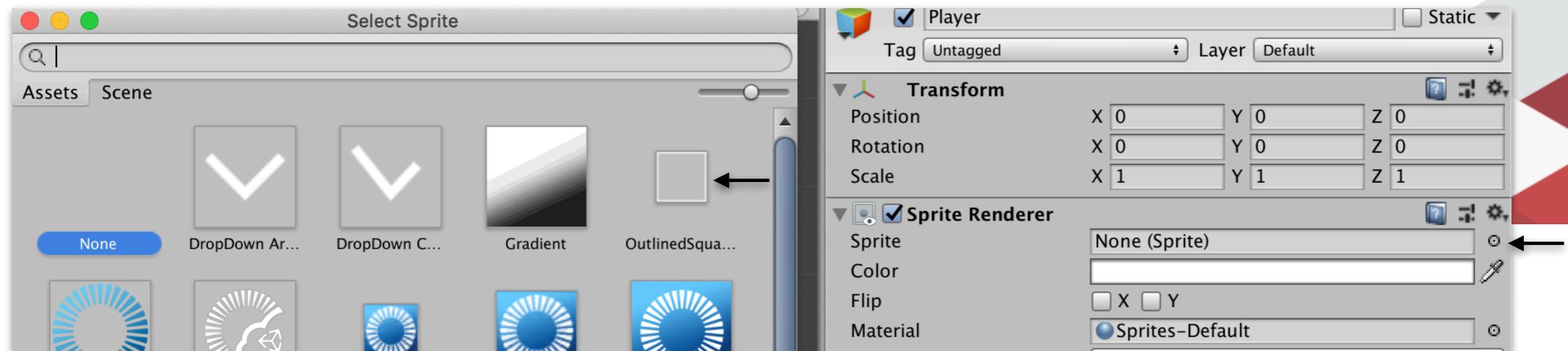
- Create a sprite object.



- Make sure the transforms are identity.



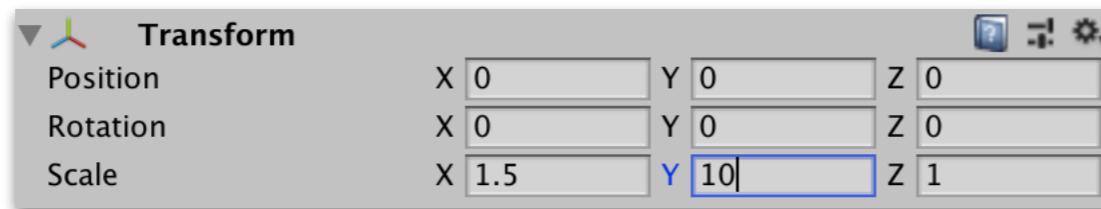
- Select OutlinedSquare sprite for our player.



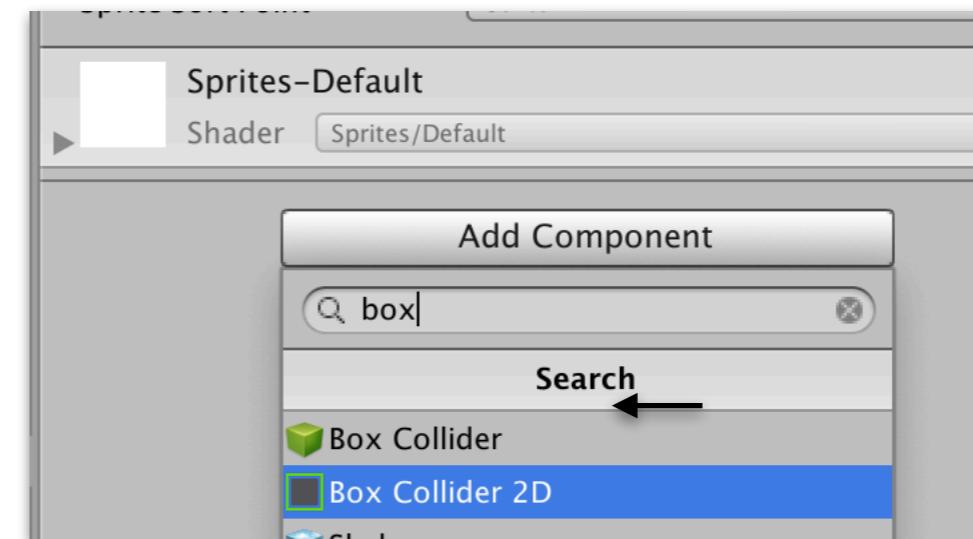
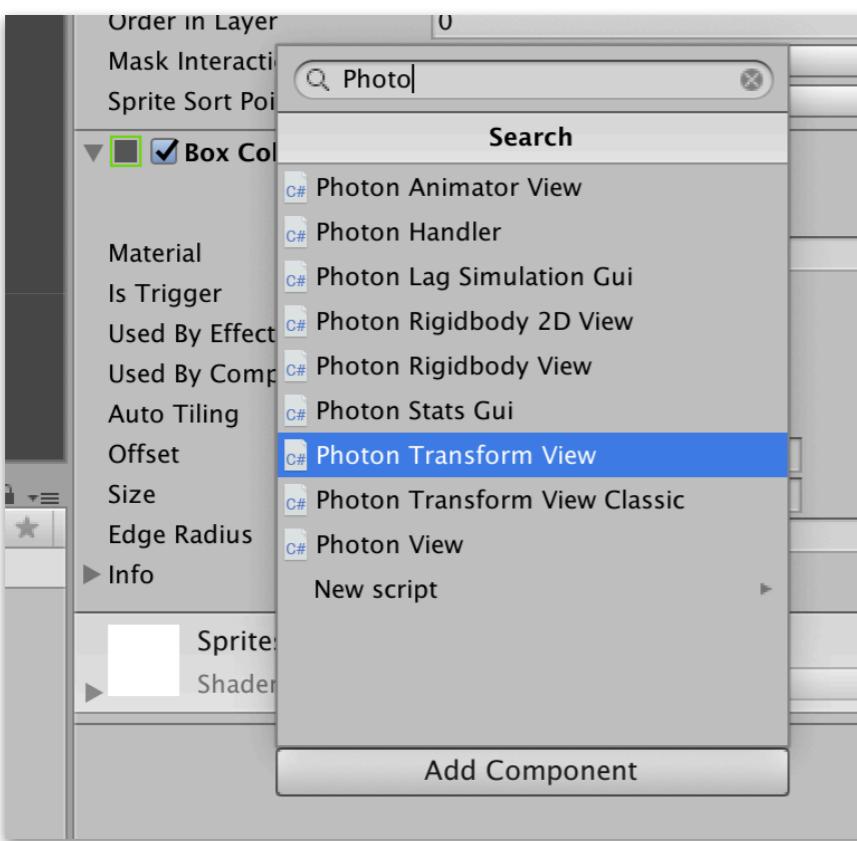
Create our player



- Put the scale to (1.5, 10, 1) for our player game object.



- Add a Box collider 2D component.

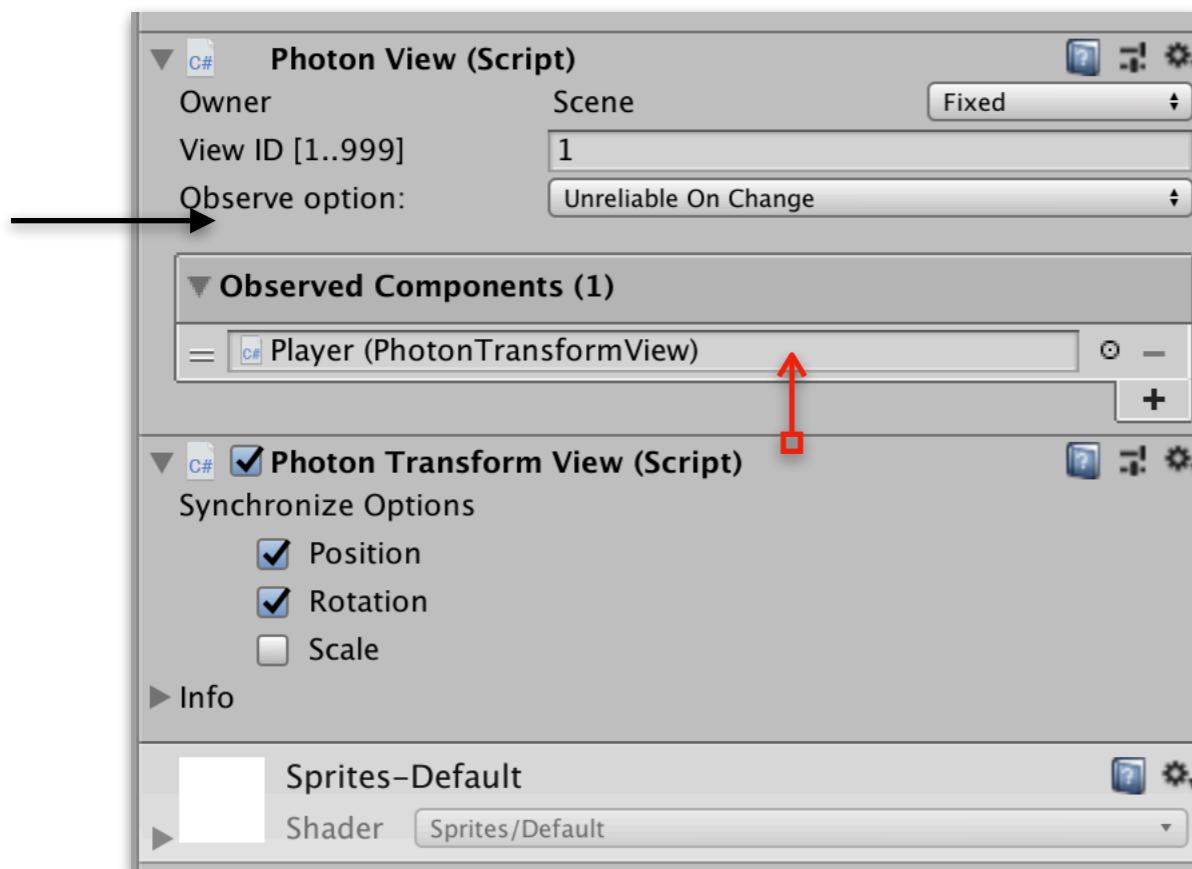


- Add a Photon Transform View to our player.

Create our player

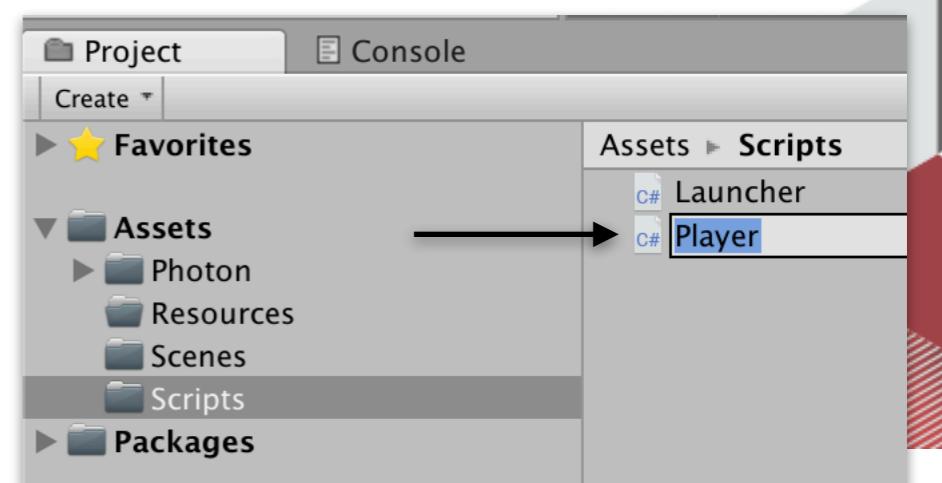


- Drag drop the Photon Transform View script to Observed Components list.



Change to 'Reliable Data Compressed'

- Create 'Player.cs' file under 'Scripts' folder.

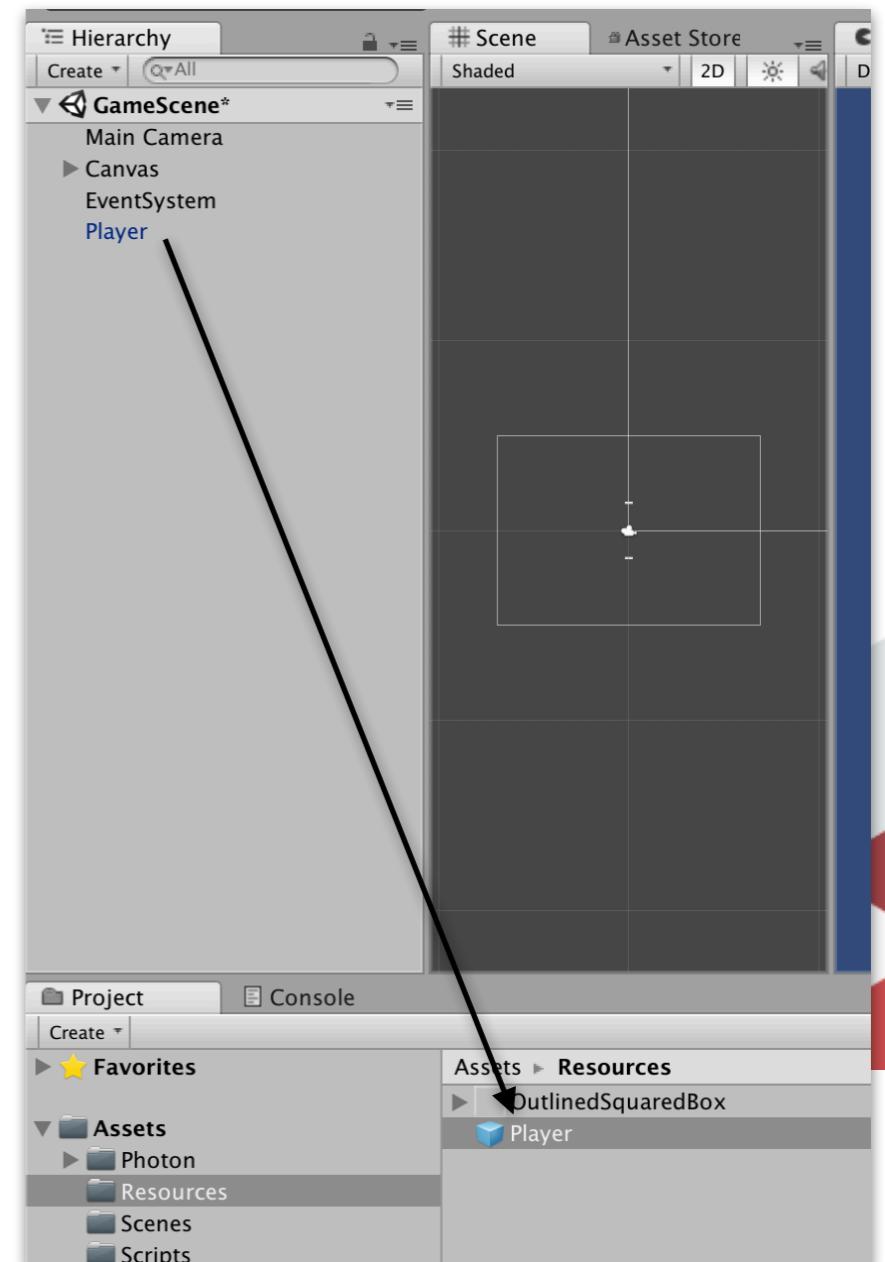
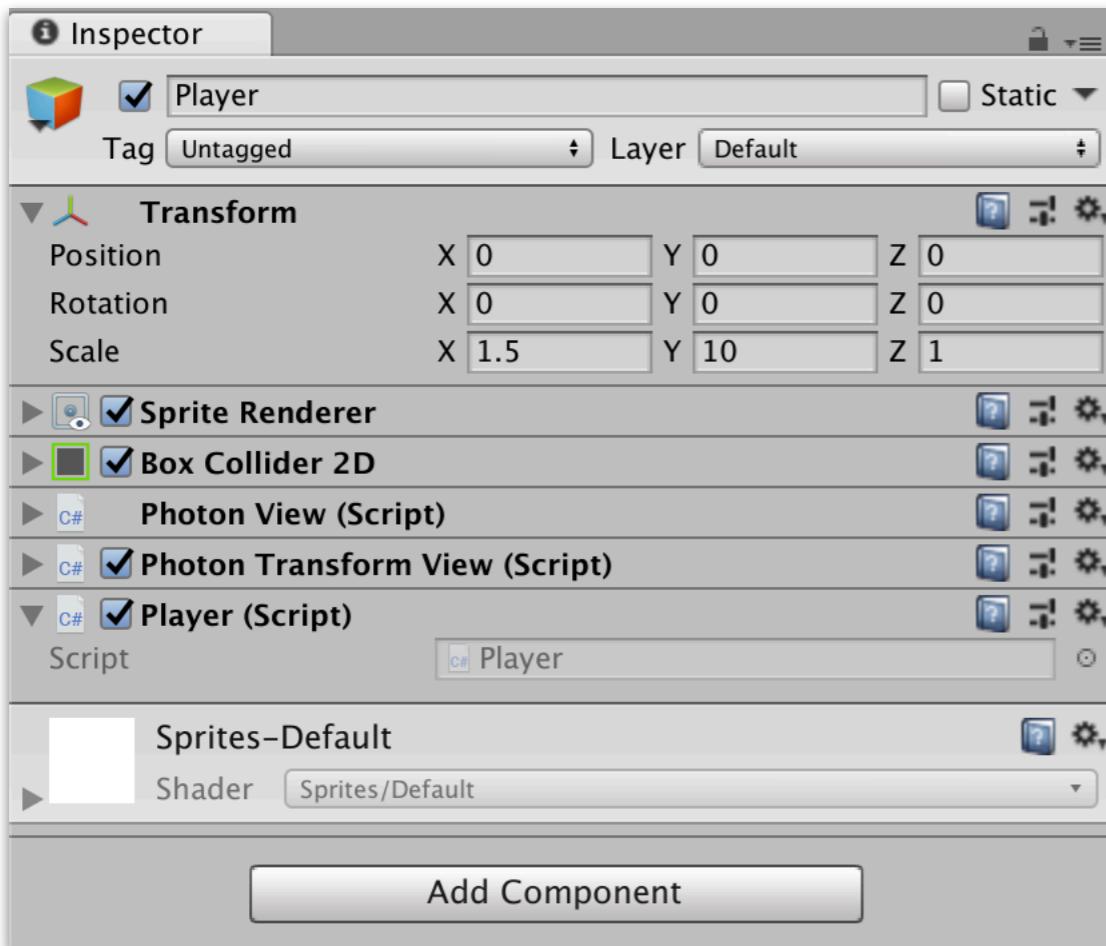


- Attach 'Player.cs' script to our player using 'Add component' in property panel.

Create our player



- Our player has following components attached.

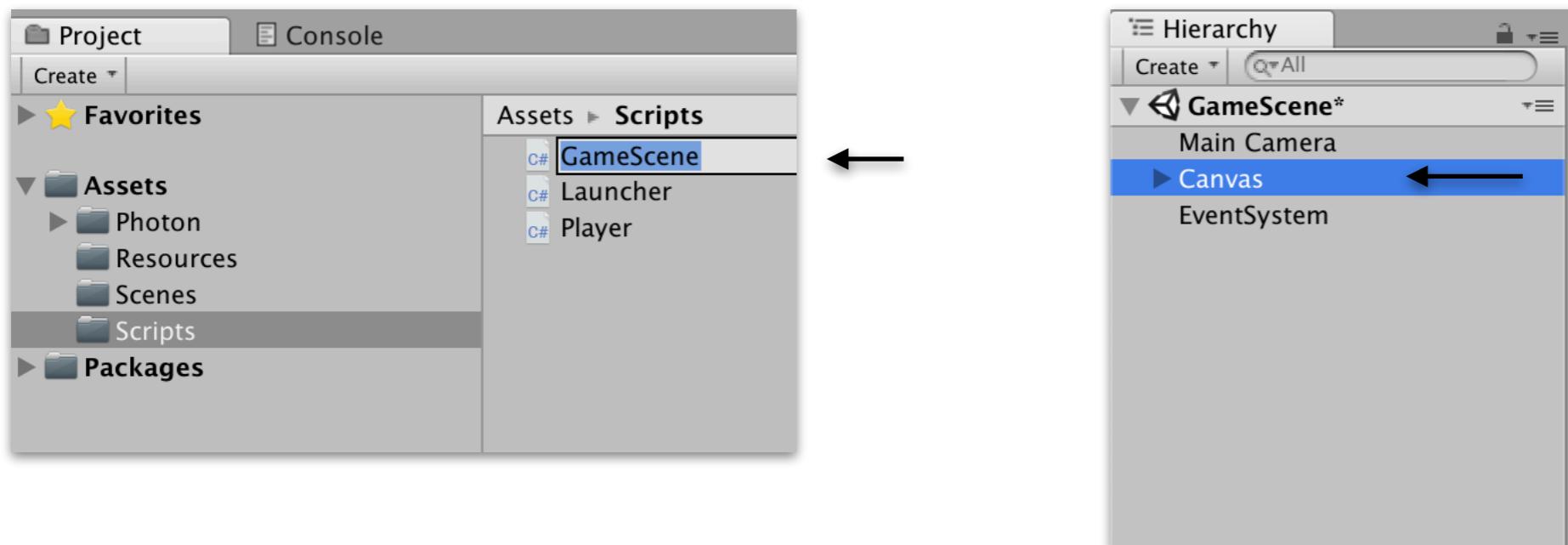


- Create Player prefab of our Player game object in Assets/ → Resources folder.
- Delete the Player from hierarchy view.

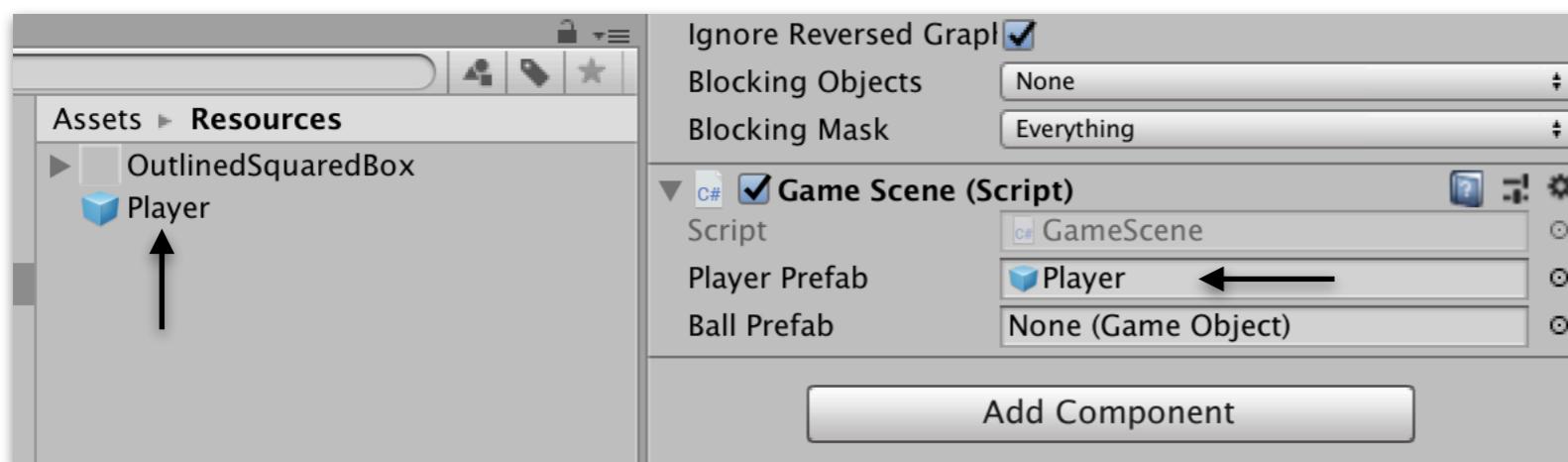
GameScene.cs Script



- Create 'GameScene.cs' file and attach to our Canvas



- Drag-drop 'Player' prefab from Assets/Resources folder to GameScene script's 'Player Prefab' field.



Test Run



- **Save GameScene (Command + S)**
- **Open Launcher scene**
- **Click on Play button to test our project.**

