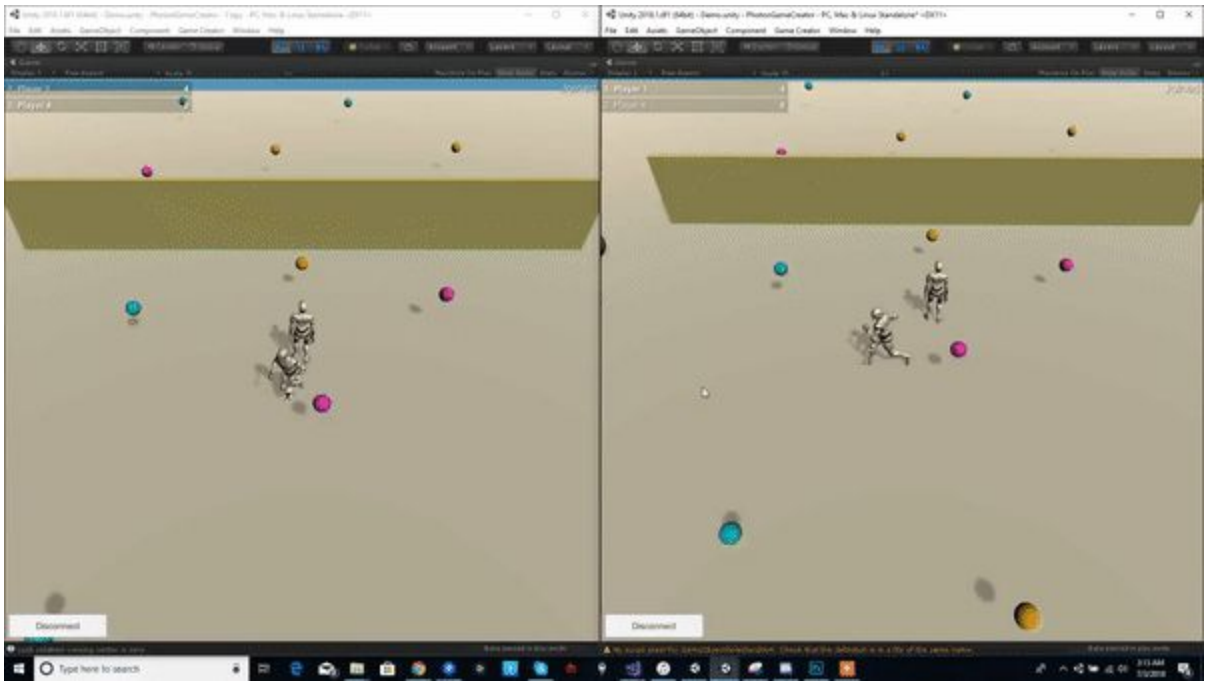
Photon Network

This is a module made for **Game Creator** that seamlessly integrates **Photon Unity Networking** and allows you to have networking in your game with just few clicks and without writing a single line of code.



(Preview of character attachments and movement network synchronization)

To learn more about Photon Unity Networking head out to their website.

Try the **Demo** now!

Key features

- Complete **Character's synchronization** (Position, rotation, speed, jump etc.)
- Full control of **Player Properties** and **Room Properties** via actions.
- **Synchronize Actions** with a single click.
- Character **Attachments** synchronization.
- Cached prefab list editor, easily add prefabs for Network instantiation.
- Item **pickup system**. Useful for power-ups or Game Creator's **Inventory** or **Quest** module.
- Packed with **Actions** and **Conditions** that can be used on any other Game Creator module.
- Built-in editor **debugging tools** to see connection status, player properties, room properties and more.

All actions and conditions are compatible with other Game Creator modules.

Setup

You can get **Photon Unity Networking** from **here**.
Download the package from **itch.io** or the Unity Asset Store (soon). You'll first need to have Game Creator and Photon Unity Networking installed.
Then, bring up the *Module Manager Window* and click the Photon's **Enable** button.

This module requires **Game Creator** and **Photon Unity Network 2** and won't work without it. Don't attempt to extract the package inside the Plugins/ folder as it will throw some errors.

Overview

Basic Concepts
Before you get started using this module let's get to understand some basic concepts.

PUN
PUN is short for **Photon Unity Network** this is the base of this module.

RPC
From here we will be talking about RPC's this the short for **Remote Procedure Call**. Can be a term for Operations (calling methods on the server) but in most cases it refers to calling a method on remote clients within PUN games.

Photon Player
Summarizes a "player" within a room, identified (in that room) by actorID.
Each player has an actorId (or ID), valid for that room. It's -1 until it's assigned by server.
Each client can set it's player's custom properties with Set Properties Action, even before being in a room. They are synced when joining a room.

Room
Players meet in rooms to play a match or communicate. Communication outside of rooms is not possible. Any client can only be active in one room.

Photon Targets

With **Photon Unity Network** there is the option to send to different group of targets or single player.

- **All**: Sends the RPC to everyone else and executes it immediately on this client. Player who join later will not execute this RPC.

- **Others**: Sends the RPC to everyone else. This client does not execute the RPC. Player who join later will not execute this RPC.

- **MasterClient**: Sends the RPC to MasterClient only. Careful: The MasterClient might disconnect before it executes the RPC and that might cause dropped RPCs.

- **AllBuffered**: Sends the RPC to everyone else and executes it immediately on this client. New players get the RPC when they join as it's buffered (until this client leaves).
- **OthersBuffered**: Sends the RPC to everyone. This client does not execute the RPC. New players get the RPC when they join as it's buffered (until this client leaves).

- **AllViaServer**: Sends the RPC to everyone (including this client) through the server. The server's order of sending the RPCs is the same on all clients.

- **AllBufferedViaServer**: Sends the RPC to everyone (including this client) through the server and buffers it for players joining later. The server's order of sending the RPCs is the same on all clients.

Do you want to know more about Photon Network? Click Here.

The Editor

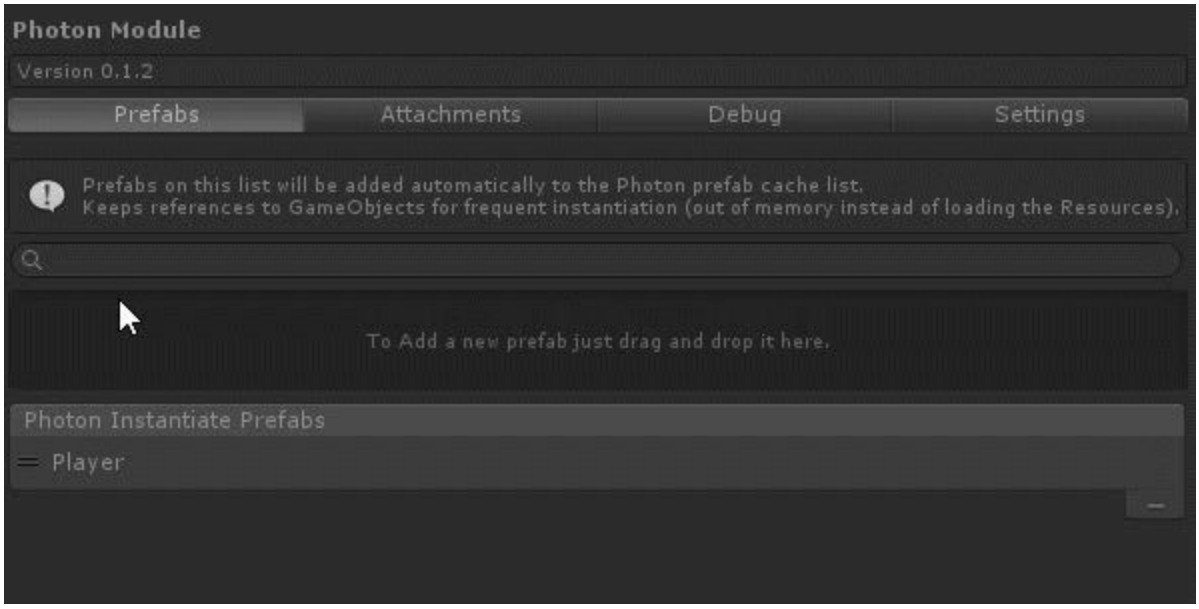
The Photon module comes with a preference window where you can manage:

- **Prefabs**: this is a predefined list of prefabs for network instantiation. This is used by the *Photon Instantiate Action.* You can only use prefabs and they requires to **PhotonView** attached.

- **Attachments**: this is a list of predefined attachments to use with the GameCreator's Character Attachment system.

- **Debug**: from here you can preview **Photon State connection**, Photon *Player Properties* and *Room Properties*.

- **Setting**: under settings you can tweak some Photon Network values like **Send Rates.** Only change this if you know what you are doing. This section it's still under development.



If you try to attach an object to a **Character** and if not part of the attachment list it won't *synchronize the attachment*.

Characters

Synchronizing Characters

Synchronizing a **Character** it's just as easy as a single click. Once the module it's installed you will notice a new tab called "**Network Settings**" below the **Character's** or **PlayerCharacter's** *Inspector* in order to make this work just go ahead and enabled the option "**Sync Character**" that's pretty much it (well that was actually 2 clicks). This auto-magically synchronize every property of a character: *Position*, *Rotation*, *Speed*, *Jump Height* etc.

(Example of the Character Network Settings)

Notice that this will add a **Photon View** component to the game object.

By enabling **Sync Character** option you will be allowed to tweak some values like:

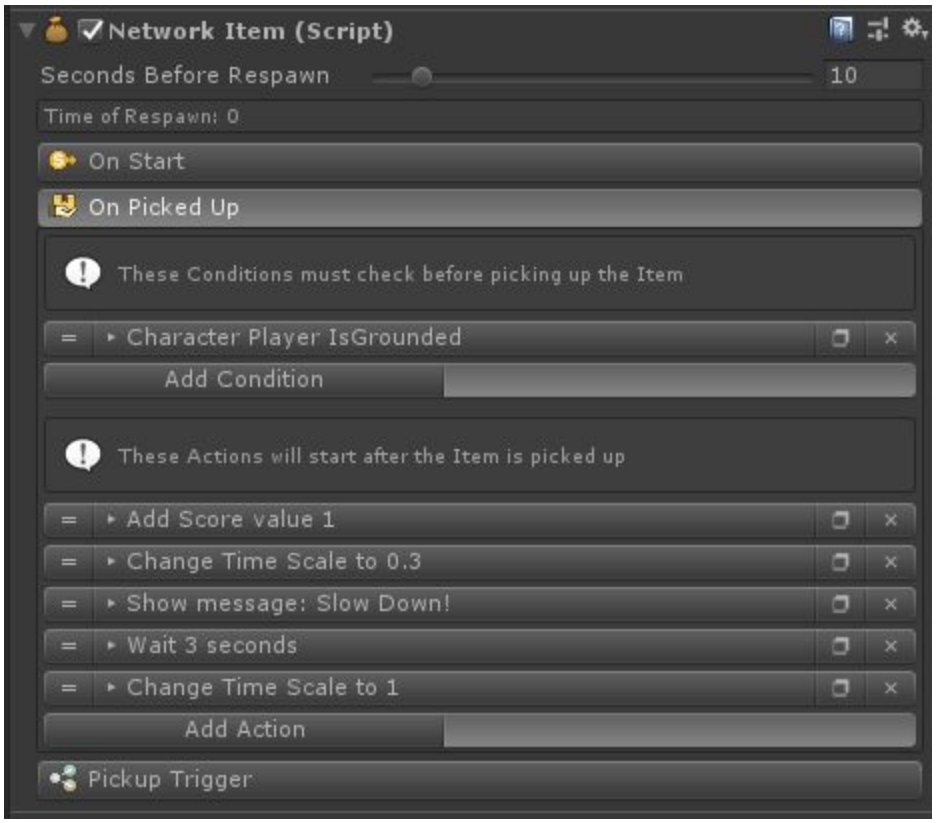
- **Teleport Distance**: if the character's position is too far from this value (distance) it will teleport to the position.
- **Rotation Dampening**: how smooth it should rotate.
- **Sync Attachments**: whether it should synchronize attachments or not. Just make sure to add any attachments do you want to synchronize to the list as stated here.

This requires to be within a **room** to work. Nothing happens if not.

Network Items

Network items are pickup-able objects that are allowed to re-spawn after someone pick it up. These items re-spawn on the same place it was before.
If you don't want an item to re-spawn, set **Seconds Before Respawn** to 0.
This component it's extended from Game Creator's trigger so you are able to select how do you want this item to be picked up.

Network item **actions** are only executed for the **Local Player**



(Example of a Network item that gives score change time scale for the local player only)

A re-spawning item should stick to a fixed position. It should not be observed at all (in any **PhotonView**). It can only be consumed and can't be dropped somewhere else (cause that would double the item)

Actions

The **Photon** module comes with a set of **Actions** that complement **Game Creator**'s. All actions can be used on any other module

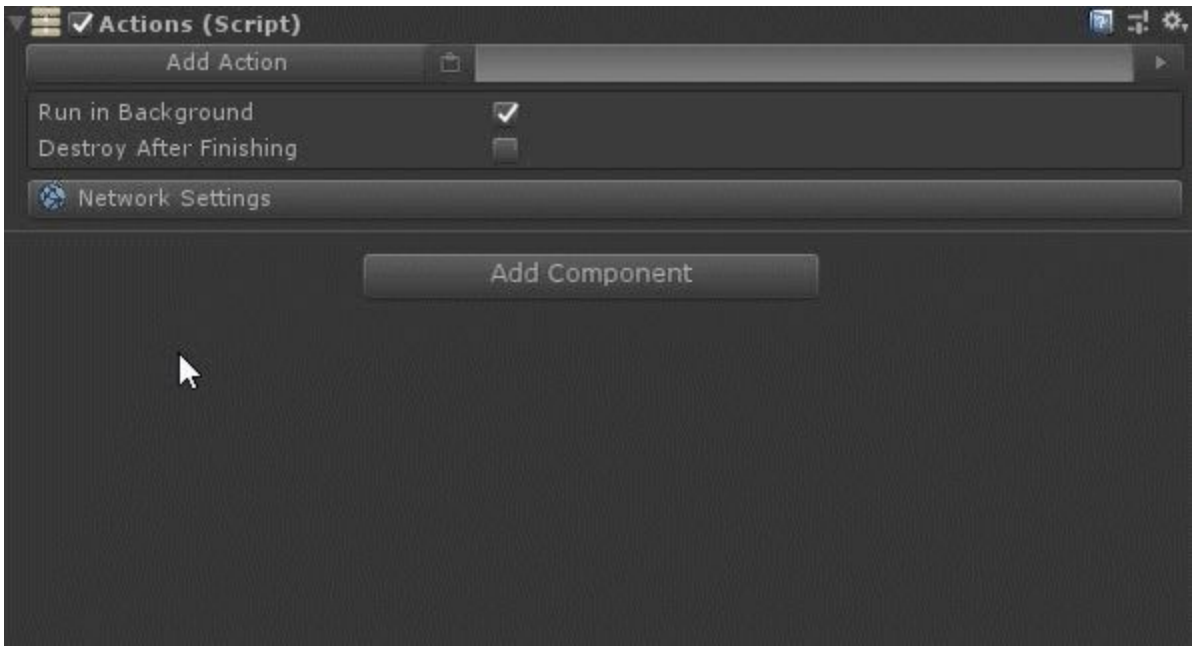
Most of the Actions won't work if you are not connect to the Photon Service unless you have **offline mode** activated.

Synchronizing Actions

Synchronizing Actions it's just as easy as a single click. Once the module it's installed you will notice a new tab called "**Network Settings**" below the *Action's Inspector* in order to make this work just go ahead and enabled the option "**Sync Actions**" that's pretty much it (well that was actually 2 clicks).

This requires to be within a **room** to work. Nothing happens if not.

If you are *synchronizing* **actions** some times you might need to use target **Invoker** instead of **Player**



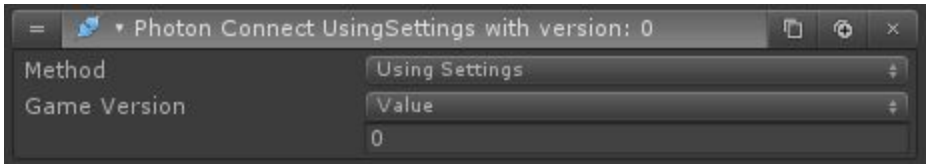
(Example of the built-in sync actions inspector)

Once you enabled **Sync Actions** you can select the [targets] you want to synchronize with.
Notice that this will add a **Photon View** component to the game object.

Action Photon Connect

Allows you connect to the Photon Service, parameters:

- **Game Version**: this client's version number. Users are separated from each other by game version (which allows you to make breaking changes).
- **Methods:** this action allows you to connect using 3 different method types:
    - **UsingSettings**: Connect to Photon as configured in the PUN settings
    - **Region**: Connects to the Photon Cloud region of choice.
    - **BestCloudServer**: Connect to the Photon Cloud region with the lowest ping.



Connecting to the **Photon Cloud** might fail due to:

- **Invalid AppId** calls: OnFailedToConnectToPhoton. check exact AppId value

- **Network issues** calls: OnFailedToConnectToPhoton

- **Invalid region** calls: OnConnectionFail with DisconnectCause.InvalidRegion

- **Subscription CCU limit reached** calls: OnConnectionFail with DisconnectCause.MaxCcuReached. also calls: OnPhotonMaxCccuReached

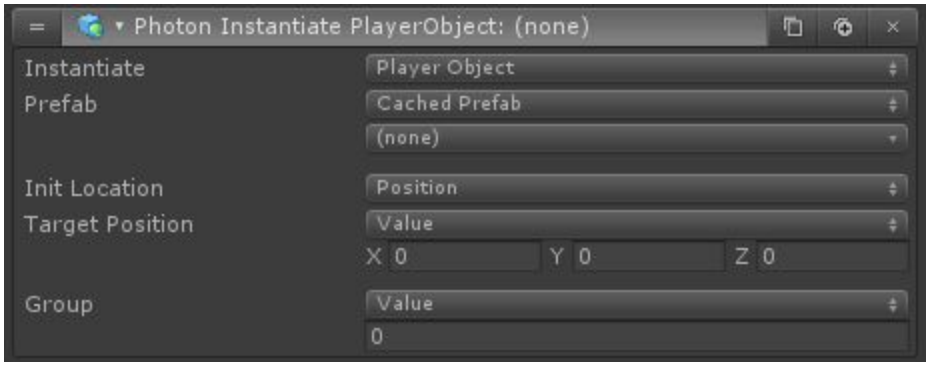
Actions Photon Disconnect

Makes this client disconnect from the photon server, a process that leaves any room and calls OnDisconnectedFromPhoton on completion.

Action Photon Instantiate

Instantiate a prefab over the network. There is 2 types of instantiations:

- **Player Object**: the owner of the object is the player. Normally when player disconnects all instantiated player objects will be destroyed.
- **Scene Object**: the owner of the object is the scene. The object will always be there unless someone destroys it.



This prefab needs to be defined in **Prefabs list** from **Photon Preferences Settings** or located in the root of a "Resources" folder.

Conditions

The **Photon** module comes with a set of **Conditions** that complement **Game Creator**'s and allows to check if the Player is yours or Player is the Master Client etc. All conditions can be used on any other module and everywhere.

Most of the Actions won't work if you are not connect to the Photon Service unless you have **offline mode** activated.

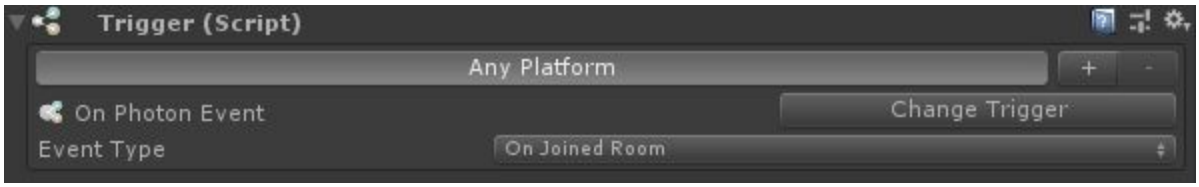
Triggers

The **Photon** module comes with a set of **Igniters or Triggers** that are called when when photon events happens like On Connected, On Joined Room etc.

Trigger On Photon Event

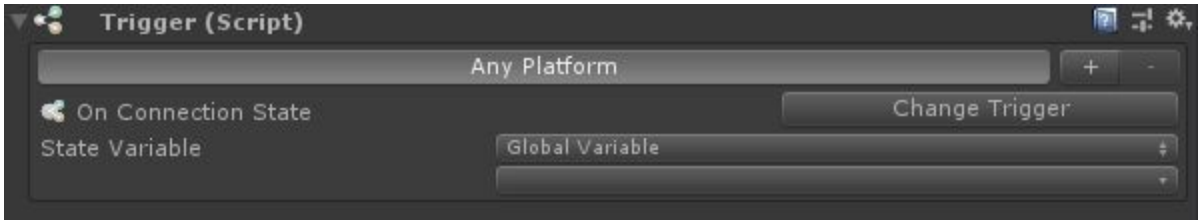
This event handle pretty much all Photon Events. Some of them like:

- **OnConnectedToPhoton**: Called when the initial connection got established but before you can use the server. *OnJoinedLobby* or *OnConnectedToMaster* are called when PUN is ready.
- **OnLeftRoom**: Called when the local user/client left a room.
- **OnCreatedRoom**: Called when this client created a room and entered it. *OnJoinedRoom* will be called as well.
- **OnJoinedRoom**: Called when entering a room (by creating or joining it). Called on all clients (including the Master Client).
- And more.. To see the full list head to the [Photon Events page](#).



Trigger On Connection State

This event it's called when the Photon Network connection state changes and also store the current state in a local or global variable.



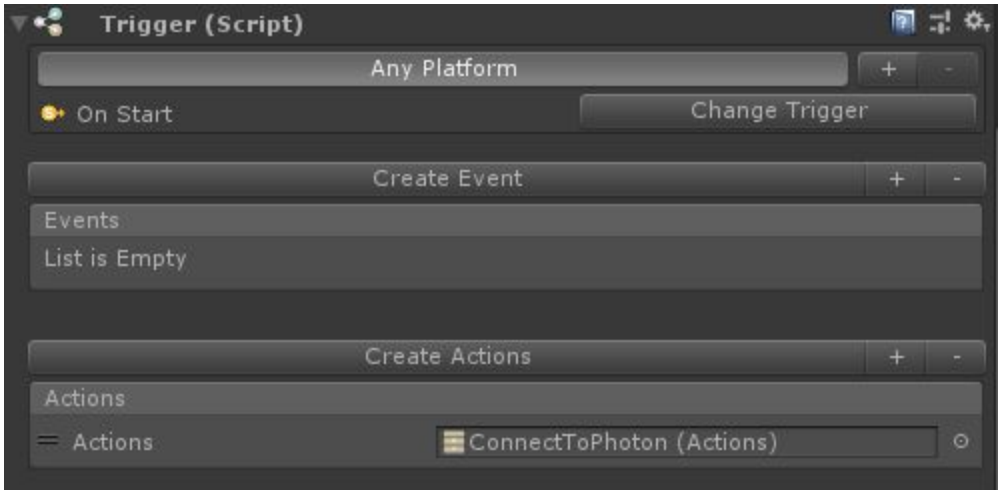
How to Connect

Connect to Photon

Connect to photon is really simple, here is a quick guide of how to connect, create a room and finally instantiate a character:

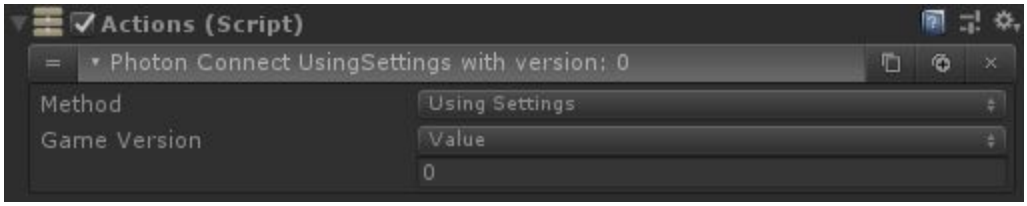
1. First create a trigger with Start igniter

After you create the trigger go ahead create actions.



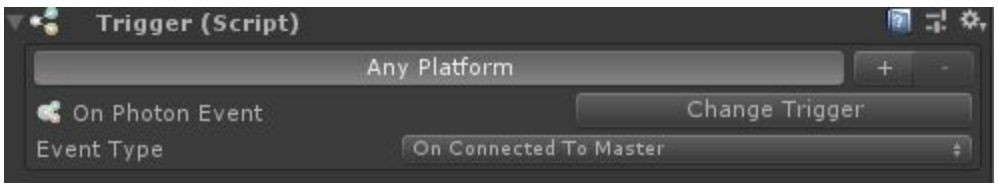
2. Add Photon Connect Action

You can leave parameters as they are.



3. Create another trigger with On Connected To Master event

When photon is connected this event is going to be called.

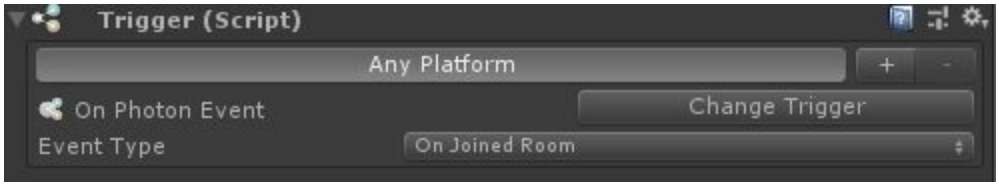


4. Let's add Join Or Create Action

We will join a room o create if it doesn't exists.

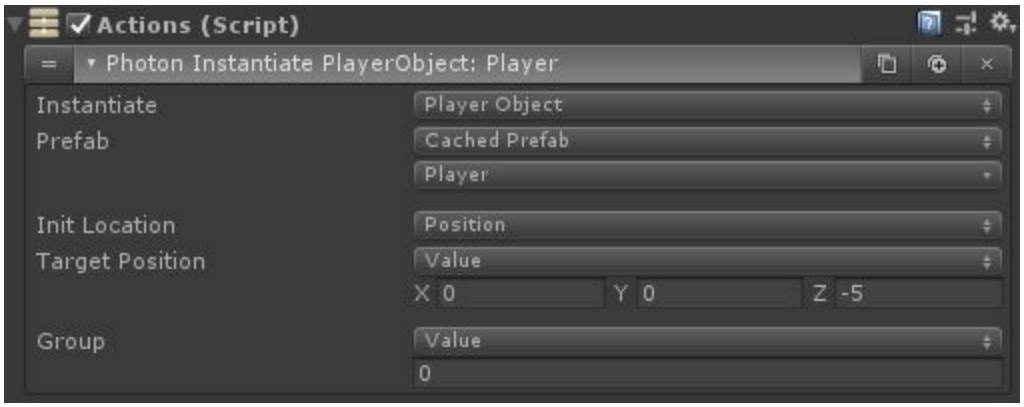
5. Now we need another trigger

We are connected to a room.



6. Let's add a Character

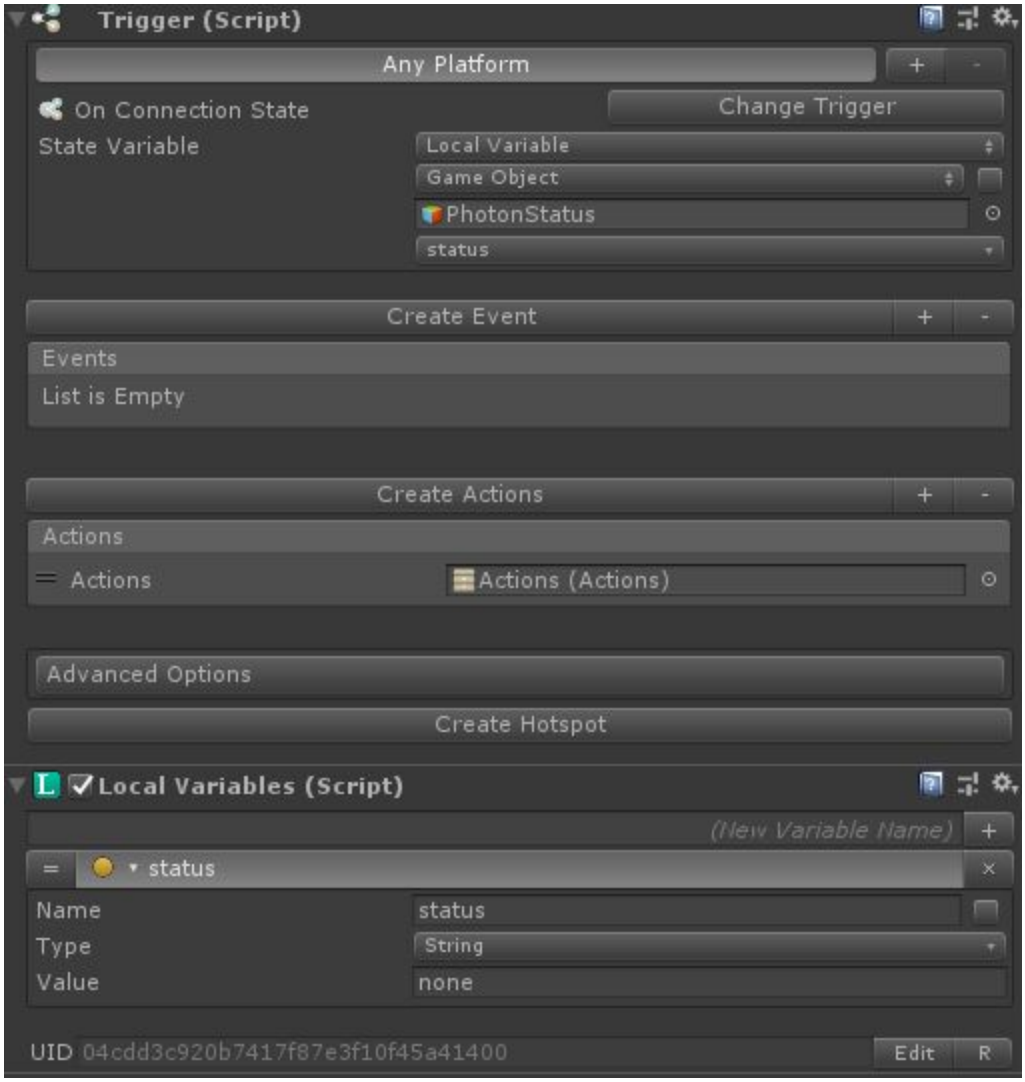
Within the On Joined Room event now add Photon Instantiate action and select or assign your Character Prefab.



(On this example we use a Player prefab that was previously added to Prefabs list in Photon Preferences)

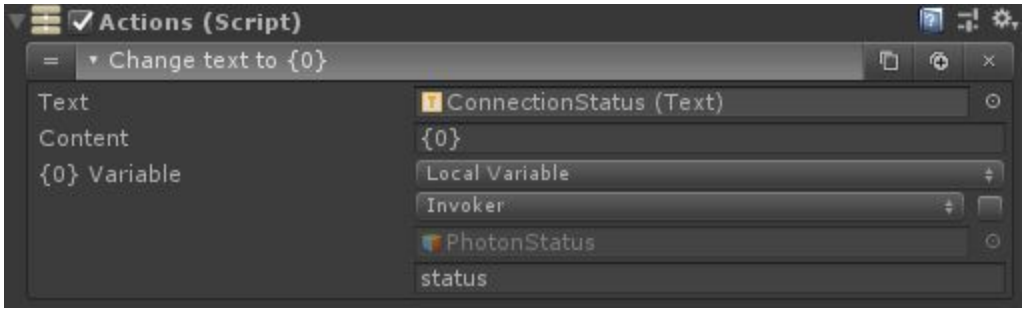
7. (Optional) You might want to have some feedback

Let's add a text component to get the Photon Status but first we need the event. We can store the status/state in a local or global variable to later use this.

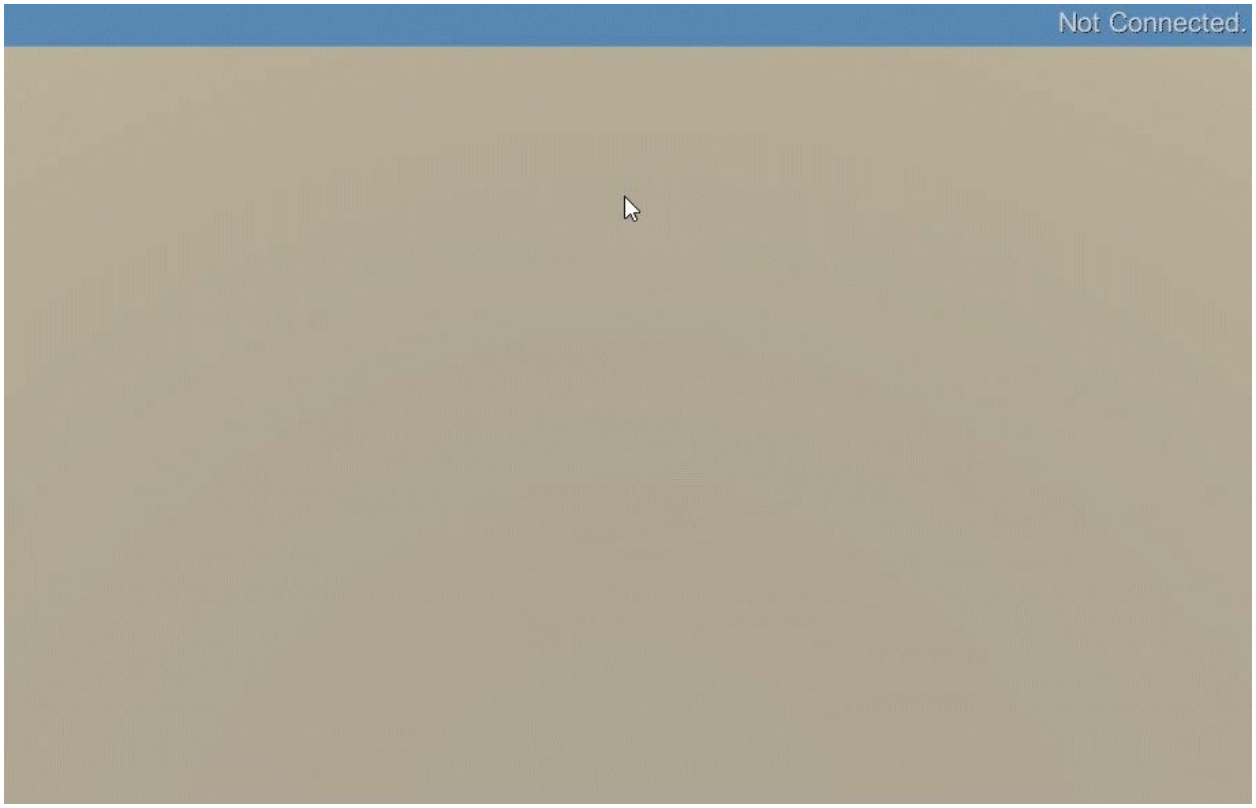


7b. (Optional) Show some feedback

Now add the text action and to show what we saved in a local variable.



8. Voila. This is how it should look.



Have any question? Feel free to drop us a line at support@ninjutsugames.com.

Click here to see the last online version of this documentation.