PUN Multiplayer Add-On

Table of Contents

PUN Multiplayer Add-On

The PUN Multiplayer Add-on will synchronize your character and items using version 2 of the Photon Unity Network (PUN). The add-on sits on top of PUN and the Ultimate Character Controller.

Target Audience

The PUN Multiplayer Add-on is not a PUN template or complete project. **You are expected to script your own game logic and should be experienced with PUN scripting**. If you do not have PUN scripting experience then this add-on is not for you. You can get started with PUN scripting by taking a look at this page.

PUN uses a client authoritative networking architecture which is not suitable for all game types. With a client authoritative architecture the local client runs its own logic for determining where the character should move, shoot, spawn, etc. There is no validation with a central server to determine if the client's actions are valid. This makes it more susceptible towards cheating compared to a server authoritative architecture. As a result, this addon is geared towards more a casual or coop game where the possibility of cheating does not matter as much.

Folder Structure

The PUN Multiplayer Add-On is imported into the Opsive/UltimateCharacterController/Add-Ons/Multiplayer/PhotonPUN folder. All of the synchronization scripts are located within the Scripts folder. The Demo folder contains the demo scene and the main Ultimate Character Controller demo scene must be imported in order for this demo scene to work.

Getting Started

Importing

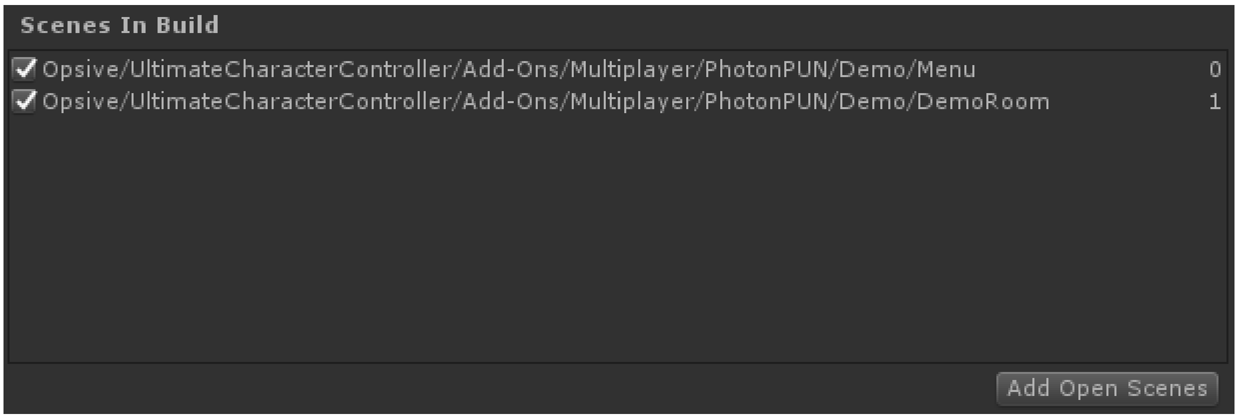
Before you import the PUN Multiplayer Add-On ensure that you have first imported:

- Any of the Opsive Character Controllers version 2.1.5+.
- PUN version 2.13.

Demo Scene

The PUN Multiplayer Addon contains two demo scenes: a main menu and the actual demo scene. The Ultimate Character Controller demo assets are required by these demo scenes. There are a few steps required before being able to play the demo scene:

1. Ensure you have set your Photon App ID.
2. **Open the DemoRoom scene and save the file.** An editor script will automatically run ensuring all of the correct references are setup.
3. Add the MainMenu and DemoRoom scenes to the Unity Build Settings.


Scenes In Build

✓ Opsive/UltimateCharacterController/Add-Ons/Multiplayer/PhotonPUN/Demo/Menu	0
✓ Opsive/UltimateCharacterController/Add-Ons/Multiplayer/PhotonPUN/Demo/DemoRoom	1

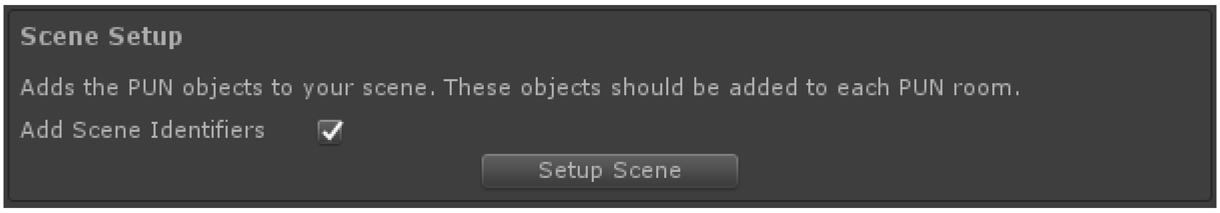
Lightmapping

In order to reduce the download size the demo scene does not contain any lightmapping. When the scene first opens it will appear dark and this can be corrected by baking the scene. If the lights are not baked before hitting play they will automatically be disabled.

Setup

Scene

Similar to the Scene Setup window, your scene must be setup for PUN in order for it to work over the network. The scene can be setup through the PUN Multiplayer Manager. The PUN Multiplayer Manager can be accessed from the Tools -> Opsive -> Ultimate Character Controller -> Add-Ons Manager -> PUN Multiplayer.


Scene Setup

Adds the PUN objects to your scene. These objects should be added to each PUN room.

Add Scene Identifiers ✓

The Setup Scene button will add the following components:

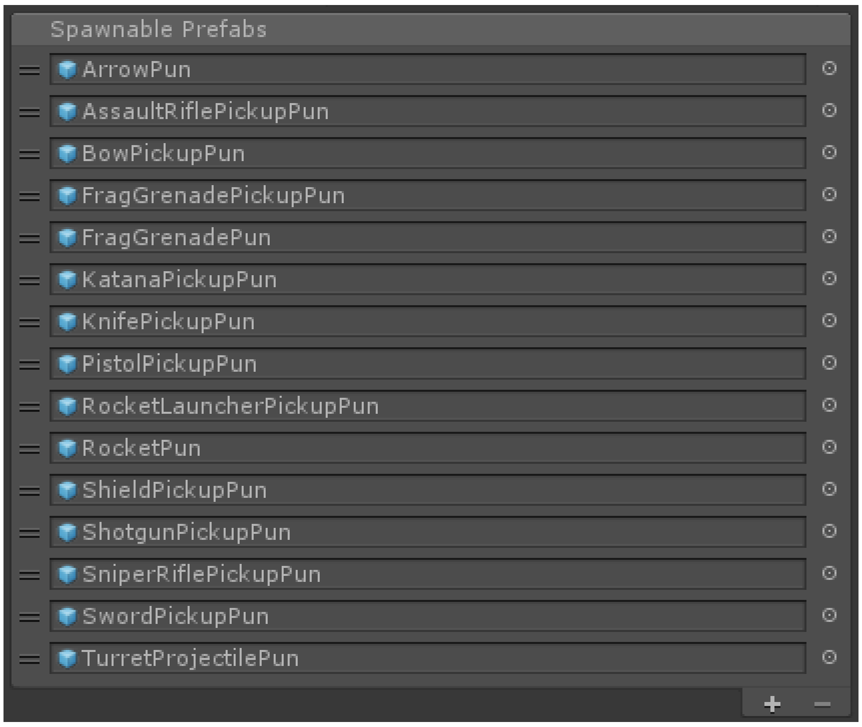
- PunObjectPool: Provides a way to synchronize pooled objects over the network.
- RuntimePickups: Adds all runtime pickups to the character after they joined the room.
- SpawnManager: Manages the character instantiation within a PUN room.
- PunStateManager: Ensures the states are synchronized when a new player joins the room. StateManager.SendStateChangeEvent must be enabled for this component to work.
- ItemTypeTracker: Maps the Item Type ID to the Item Type over the network.

These components are added to the new “PunGame” GameObject within the scene.

If *Add Scene Identifiers* is selected the manager will add the PunObjectIdentifier component to every collider within the scene. When a melee weapon collides with a scene object all of the other clients need to know which scene object was hit. Normally you’d add the PhotonView component to this scene object to make it identifiable over the network but in this case the Photon View component is overkill for what we need. The Pun Object Identifier component inherits the Object Identifier and ensures the melee weapon can identify which object was hit. Note that you must **rerun the scene setup every time you add a new collider to the scene.**

PUN Object Pool

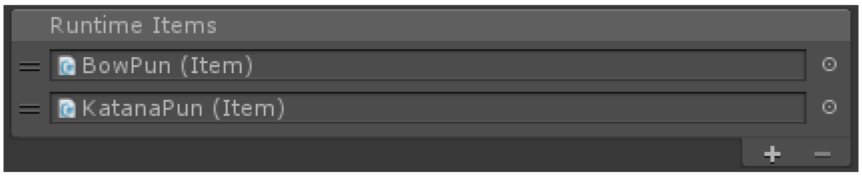
The PUN Object Pool synchronizes pooled objects over the network. Any object that should be synchronized needs to first be defined within the *Spawnable Prefabs* list of the PUN Object Pool. Spawnable Prefabs include objects such as projectiles, grenades, pickups, etc. Objects that are not synchronized over the network, such as decals, particles, or other effects, do not need to be specified within the Spawnable Prefabs list.


Spawnable Prefabs

- ArrowPun
- AssaultRiflePickupPun
- BowPickupPun
- FragGrenadePickupPun
- FragGrenadePun
- KatanaPickupPun
- KnifePickupPun
- PistolPickupPun
- RocketLauncherPickupPun
- RocketPun
- ShieldPickupPun
- ShotgunPickupPun
- SniperRiflePickupPun
- SwordPickupPun
- TurretProjectilePun

Runtime Pickups

When a new player joins the room all of the possible items are added to the character. This ensures that if a new player joins and the remote player has already picked up a runtime item that new player will be able to synchronize that runtime item. After the runtime item has been created it should be added to the Runtime Items list of the Runtime Pickups component.


Runtime Items

- BowPun (Item)
- KatanaPun (Item)

Item Tracker

Similar to the Runtime Pickups component, the Item Tracker component is used to ensure the network is aware of all of the Item Types that can be equipped. The Item Tracker component has a *Item Collection* field which allows all of the Item Types to be loaded when the room starts.

Spawn Manager

The Spawn Manager is responsible for instantiating new characters when a player joins. By default the Single Character Spawn Manager component will be added and it will always spawn the character specified by the *Character* field. If you'd like to spawn a different character prefab you can override the SpawnManagerBase.GetCharacterPrefab method. Use cases for this may be if you'd like each player to have a unique character model or if you are spawning the character into a team based game.

```
using UnityEngine;
using
Opsive.UltimateCharacterController.AddOns.Multiplayer.PhotonPun.Game;
```

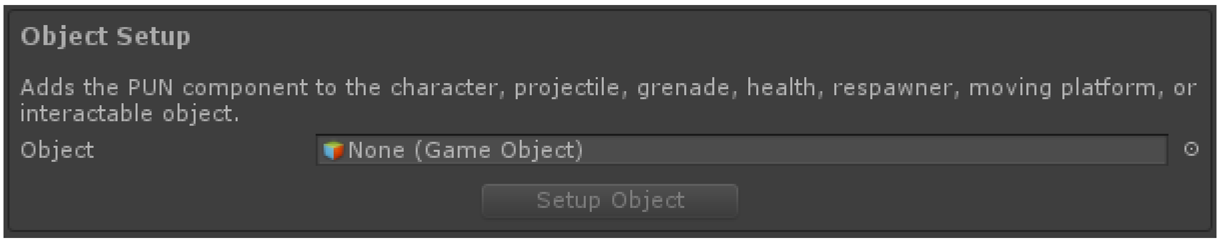
```
using Photon.Realtime;

public class MySpawnManager : SpawnManagerBase
{
    /// <summary>
    /// Virtual method which allows for a character to be spawned
based on the game logic.
    /// </summary>
    /// <param name="newPlayer">The player that entered the
room.</param>
    /// <returns>The character prefab that should spawn.</returns>
    protected override GameObject GetCharacterPrefab(Player newPlayer)
    {
        return null; // Replace with your own game logic.
    }
}
```

GetCharacterPrefab is run on each client instance so it should should return a deterministic value. If it does not then each client may have a unique character model for that player.

Character

Before your character can be used by the PUN add-on it must first be created using the Character Creation workflow. After your character has been created it can then be used by the PUN Multiplayer Manager. The PUN Multiplayer Manager can be accessed from the Tools -> Opsive -> Ultimate Character Controller -> Add-Ons Manager -> PUN Multiplayer.

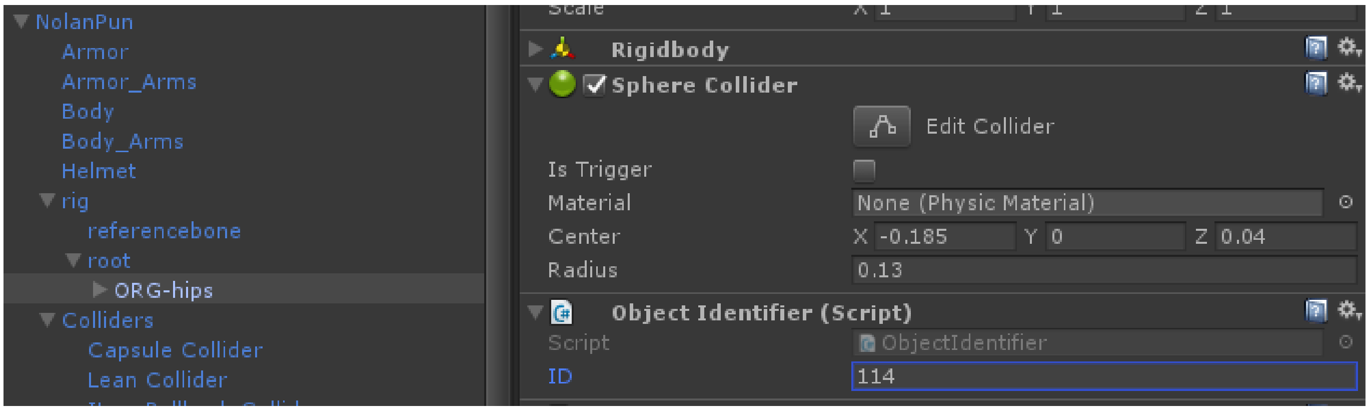

Object Setup

Adds the PUN component to the character, projectile, grenade, health, respawner, moving platform, or interactable object.

Object: None (Game Object)

If you drag the created character into the *Object* field of the Object Setup section the Setup Object button will enable. After the character has been setup it will have all of the PUN components added so it can be used over the network.

When the character is setup the editor will traverse through the character's hierarchy looking for any colliders that may need to be identifiable over the network. This will allow a particular collider to be identified if it is hit by an attack. After your character has been built you'll see the ObjectIdentifier components on the character and ragdoll colliders. If you make a collider change ensure you have rerun the character setup editor script.



NolanPun
- Armor
- Armor_Arms
- Body
- Body_Arms
- Helmet
- rig
  - referencebone
  - root
    - ORG-hips
- Colliders
  - Capsule Collider
  - Lean Collider

Rigidbody

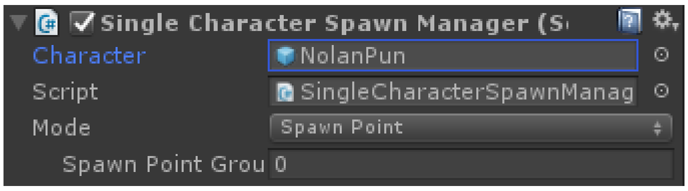
Sphere Collider

	Edit Collider
Is Trigger	
Material	None (Physic Material)
Center	X -0.185 Y 0 Z 0.04
Radius	0.13

Object Identifier (Script)

Script	ObjectIdentifier
ID	114

If your PUN character isn't already a prefab then it should be made into one. This prefab should then be specified in the *Character* field of the Single Character Spawn Manager component. The Spawn Manager is created when you setup the scene.



Single Character Spawn Manager (S

Character	NolanPun
Script	SingleCharacterSpawnManag
Mode	Spawn Point
Spawn Point Grou	0

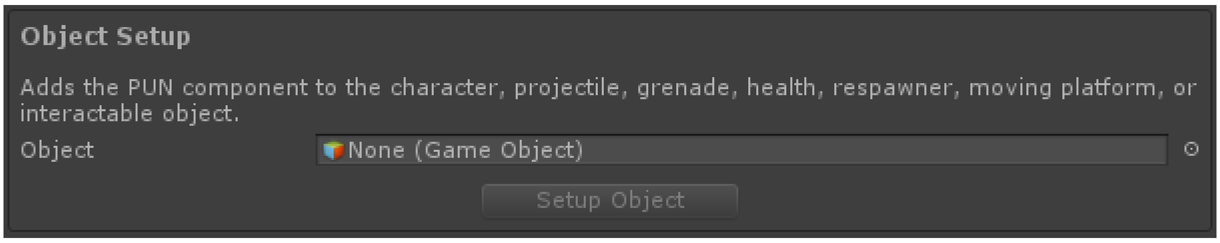
If you'd like a different character to be spawned based on the player you can do so by inheriting the Spawn Manager Base component and implementing the GetCharacterPrefab method. See the bottom of the Scene page for details.

Objects

New components may need to be added to the Ultimate Character Controller objects before they can be used over the network. The following objects must be run through the Object Setup editor:

- Projectile
- Grenade
- Health Respawner
- Moving Platform
- Interactable

The Object Setup editor can be accessed within Tools -> Opsive -> Ultimate Character Controller -> Add-Ons Manager -> PUN Multiplayer. This is the same editor as what was used to create your character.



Object Setup

Adds the PUN component to the character, projectile, grenade, health, respawner, moving platform, or interactable object.

Object	None (Game Object)

Before the Setup Object button is enabled your object must be first created using the standard Ultimate Character Controller workflow. For example, with a moving platform the Moving Platform component must first be added.

Troubleshooting

Compiler errors

If you receive this compiler error:

> Assembly has reference to non-existent assembly 'PhotonRealtime' (Assets/Opsive/UltimateCharacterController/Add-Ons/Multiplayer/PhotonPUN/Editor/Opsive.UltimateCharacterController.AddOns.Multiplayer.PhotonPUN.Editor.asmdef)

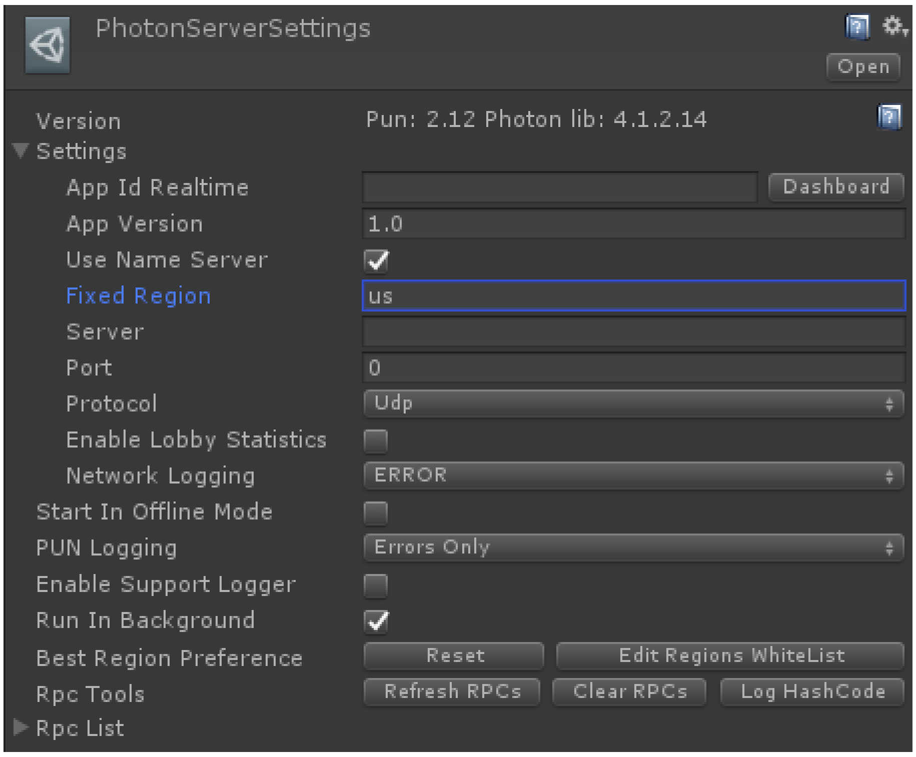
Ensure you have imported PUN 2 from the Asset Store. This compiler error:

> Assets/Opsive/UltimateCharacterController/Add-Ons/Multiplayer/PhotonPUN/Scripts/Game/PunObjectPool.cs(138,47): error CS1502: The best overloaded method match for `Photon.Pun.PhotonNetwork.AllocateViewID(Photon.Pun.PhotonView)' has some invalid arguments

Indicates that you are running an older version of Photon PUN. Ensure you are running version 2.13.

Player doesn't join an already created room

Ensure you are connecting to a server within the same region as the room that is already created. You can force Photon to connect to a specific region by setting the *Fixed Region* of the Photon Server Settings.


PhotonServerSettings

Version	Pun: 2.12 Photon lib: 4.1.2.14
Settings	
App Id Realtime	
App Version	1.0
Use Name Server	✓
Fixed Region	us
Server	
Port	0
Protocol	Udp
Enable Lobby Statistics	
Network Logging	ERROR
Start In Offline Mode	
PUN Logging	Errors Only
Enable Support Logger	
Run In Background	✓
Best Region Preference	
Rpc Tools	
Rpc List	

Unable to Spawn Object

If you receive the following error:

> Error: Unable to spawn *name* on the network. Ensure the object has been added to the PunObjectPool.

The object needs to be added to the PUN Object Pool. See the Scene Setup page for more details.

Object Location/Active State Not Synchronized

If the object position/rotation or the GameObject active state is not being synchronized over the network then the PunGameObjectTransformMonitor component should be added. This component will allow the object to be updated by the owner and the results will be sent to the other clients.

Object Not Sent over the Network

If you receive the following error:

> Error: The object *name* does not contain a PhotonView or ObjectIdentifier. It will not be able to be sent over the network.

Photon is not able to uniquely identify the object in order to be sent on the network. This can be fixed by adding a Photon View or Object Identifier component to that object. If the object is a scene object then the PunObjectIdentifier should automatically be added after the scene has been setup.