Contents

Photon Unity Networking

A Photon Unity Networking (PUN) integration script is included with Pure Pool, to help you use object pooling in your networked game. Please follow the installation instructions for the integration script before continuing.

Integration Script Usage

The integration script takes the form of a bridge class called PrefabPool and an associated component, PunPoolingSetup. The PrefabPool bridge class implements the IPunPrefabPool interface (PUN docs and PUN2 docs) required by PUN, and uses the `Manager` property to acquire and release objects from pools. The PunPoolingSetup component automatically attaches the PrefabPool bridge to PUN in the Start method.

All that's needed to enable Pure Pool with PUN is to attach the PunPoolingSetup component to a GameObject in your scene. Once that's done, all networked instantiation and destruction should now occur through Pure Pool.

General Usage

Once the `PunPoolingSetup` component has been added to an object in your scene, you can begin using Photon Unity Networking in the same way you usually would, following the documentation for PUN.

Objects that should be networked should be spawned using `PhotonNetwork.Instantiate`, and destroyed using `PhotonNetwork.Destroy`. The bridge class in the PunPoolingSetup component will take care of redirecting the PUN requests to your NamedGameObjectPoolManager and its associated GameObjectPoolManager, so everything Photon does will be using pooled objects.

Of course, it's still important that you properly implement your notification mode to recycle and reset your objects.

PUN-specific Requirements

While `OnEnable` will be called when an object is taken from the pool, as well as whatever notification mode you've chosen, the networking values will not have been updated yet when that happens. Code inside `OnEnable` or the notification mode methods will likely have outdated values for PhotonView (isMine, etc.).

However, PUN will call `OnPhotonInstantiate` (see IPunCallbacks for PUN and IPunInstantiateMagicCallback for PUN2) after the values have been updated. This should be used to setup the re-used object with regards to networking values / ownership.

PUN2 requires that objects acquired from the pool are disabled, which means when your notification mode occurs the object is disabled. This is the opposite of the usual case, where the object will be at whatever enabled state the prefab is in. This may result in errors, such as trying to play a disabled audio source. Any code that cannot be used on a disabled object should be moved to the `OnEnable` or `OnPhotonInstantiate` methods.