Contents

Mirror Networking

A Mirror Networking 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 Setup

The integration script takes the form of a single component called MirrorPooling. You should add this component to any GameObject in your scene, and set the GameObjectPoolManager value to the one in your scene.

The MirrorPooling component registers each prefab from the GameObjectPoolManager with the Mirror `NetworkClient` (previously `ClientScene`, which has been made obsolete), specifying custom pool-based spawn and unspawn handlers. This process occurs automatically if the `AutoRegister` value on the component is set to true. If set to false, you should manually call one of the MirrorPooling `RegisterSpawnHandler` overloads, the `RegisterSpawnHandlers` method, or the `RegisterNetworkedPrefabsForAllPools` method.

AutoRegister

The `AutoRegister` variable on the MirrorPooling component controls whether pools attached to the GameObjectPoolManager are automatically registered with the Mirror spawning system on Awake, in addition to whether the Registered Spawnable Prefabs from the Mirror NetworkManager are registered for pooling support.

Setting AutoRegister to true is essentially equivalent to calling RegisterNetworkedPrefabsForAllPools and RegisterSpawnHandlers during a component's Awake method.

RegisterSpawnHandlers

The RegisterSpawnHandlers method will enable pooling for all prefabs that are in the Network Manager's "Registered Spawnable Prefabs" list, whether they were added in the inspector or by

code. If the prefabs have already been registered, they will first be unregistered, and then registered again with pooling support.

When this method completes, the "Registered Spawnable Prefabs" list (and NetworkManager.singleton.spawnPrefabs list) will be cleared. This prevents Mirror from trying to register them itself, which it would do without pooling support.

RegisterSpawnHandler(GameObject prefab)

The RegisterSpawnHandler method will register the specified prefab with the Mirror spawning system, with pooling support. If the prefab has already been registered, it will first be unregistered, and then registered again with pooling support.

RegisterSpawnHandler(Guid assetId)

The RegisterSpawnHandler method will register the specified asset ID with pooling support. For this to work, the asset ID that you register must be present in the Prefabs dictionary, linking to the GameObject that will be spawned.

CreatePoolsForNetworkedPrefabs

The CreatePoolsForNetworkedPrefabs method will create a pool for each prefab that has been registered with the Mirror spawning system with pooling support. Specifically, every prefab in the Prefabs dictionary will have a pool created for it, if it doesn't yet exist.

RegisterNetworkedPrefabsForAllPools

The RegisterNetworkedPrefabsForAllPools method will ensure the source prefabs from each pool in the manager have been registered with the Mirror spawning system with pooling support. Only pools that have been initialised can be registered, so you should wait until after initialisation to call this method.

Integration Script Usage

To spawn a networked object, you can call `NetworkServer.Spawn` as usual, which will cause clients to spawn the object by taking an instance from the object pool, via the GameObjectPoolManager. To remove a networked object, you should call `NetworkServer.UnSpawn`, which will return the object to the pool. Do not use the `NetworkServer.Destroy` method, which will bypass the pool and destroy the object.

Please note that when calling NetworkServer.Spawn, you should provide an instance from the pool, not the prefab itself. Your code to spawn should look like this:

```
GameObject instance =
GameObjectPoolManager.Instance.Acquire(this.Prefab);
NetworkServer.Spawn(instance);
```

Given that you now have an instance that you've acquired locally, when you call NetworkServer.UnSpawn, you should also return the instance back to the pool, like this:

```
NetworkServer.UnSpawn(instance);
GameObjectPoolManager.Instance.Release(instance);
```

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