Contents

The Basics

How to access the manager, and replace your existing calls to Object.Instantiate and Object.Destroy with pool-friendly versions.

Accessing The Manager

Accessing the Manager in your scene can be done in any of the usual Unity ways. The example below shows how you can assign the `PoolManager` field in the Unity inspector, or leave it blank to have it automatically located when the script's `Awake` method is executed.

```
public class ExampleScript : MonoBehaviour {

        public GameObjectPoolManager PoolManager;

        private void Awake() {
                // Find the manager if one hasn't been specified.
                if (this.PoolManager == null) {
                        this.PoolManager =
Object.FindObjectOfType<GameObjectPoolManager>();
                }
        }

}
```

You can also access the Manager statically, either using the Instance property, or using one of the provided Static scripts (StaticPoolManager, StaticNamedPoolManager, StaticComponentPoolManager). The Static scripts and Instance property will automatically find the Manager in the scene, and cache it for future use.

```
public class ExampleScript : MonoBehaviour {
```

```
        public GameObject AsteroidPrefab;

        private void Start() {
                // Access the manager statically, and create a new
pool.
                GameObjectPoolManager.Instance.CreatePool(new
GameObjectPoolSettings {
                        Source = this.AsteroidPrefab,
                        InitialSize = 85,
                        MaximumSize = 100,
                        InitialiseOnStart = true,
                        LogMessages = LogLevel.Information,
                        DontDestroyOnLoad = false
                });

                // Acquire an instance statically. This is a shortcut
for GameObjectPoolManager.Instance.Acquire.
                StaticPoolManager.Acquire(this.AsteroidPrefab);
        }

}
```

Replacing Instantiate

Your original calls to `Object.Instantiate` can be replaced with calls to the Manager's `Acquire` methods, as shown in the example below.

```
public class ExampleScript : MonoBehaviour {

        public GameObject ExplosionPrefab;
        public GameObjectPoolManager PoolManager;

        private void Awake() {
                // Find the manager if one hasn't been specified.
                if (this.PoolManager == null) {
                        this.PoolManager =
Object.FindObjectOfType<GameObjectPoolManager>();
                }
        }

        private void Start() {
                // Explode in 5 seconds.
```

```
                this.StartCoroutine(this.ExplodeLater());
        }

        private IEnumerator ExplodeLater() {
                // Wait for 5 seconds.
                yield return new WaitForSeconds(5);

                // Acquire (pool-based replacement for Instantiate) an
explosion.
                this.PoolManager.Acquire(this.ExplosionPrefab,
this.transform.position, this.transform.rotation);

                // The line above would originally have looked like
this:
                //Object.Instantiate(this.ExplosionPrefab,
this.transform.position, this.transform.rotation);

                // Or, accessed statically:
                //StaticPoolManager.Acquire(this.ExplosionPrefab,
this.transform.position, this.transform.rotation);
        }

}
```

Replacing Destroy

Your original calls to `Object.Destroy` can be replaced with calls to the Manager's `Release` method, as shown in the example below.

```
public class ExampleScript : MonoBehaviour {

        public GameObject ExplosionPrefab;
        public GameObjectPoolManager PoolManager;

        private void Awake() {
                // Find the manager if one hasn't been specified.
                if (this.PoolManager == null) {
                        this.PoolManager =
Object.FindObjectOfType<GameObjectPoolManager>();
                }
        }
```

```
        private void Start() {
                // Explode in 5 seconds.
                this.StartCoroutine(this.ExplodeLater());
        }

        private IEnumerator ExplodeLater() {
                // Wait for 5 seconds.
                yield return new WaitForSeconds(5);

                // Acquire (pool-based replacement for Instantiate) an
explosion.
                var explosion =
this.PoolManager.Acquire(this.ExplosionPrefab,
this.transform.position, this.transform.rotation);

                // Wait for another 5 seconds.
                yield return new WaitForSeconds(5);

                // Release (pool-based replacement for Destroy) the
explosion.
                this.PoolManager.Release(explosion);

                // The two lines above would originally have looked
like this:
                //Object.Instantiate(this.ExplosionPrefab,
this.transform.position, this.transform.rotation);
                //Object.Destroy(explosion);

                // Or, accessed statically:
                //StaticPoolManager.Acquire(this.ExplosionPrefab,
this.transform.position, this.transform.rotation);
                //StaticPoolManager.Release(explosion);
        }

}
```