Object Pool Manager

What is the ObjectPoolManager

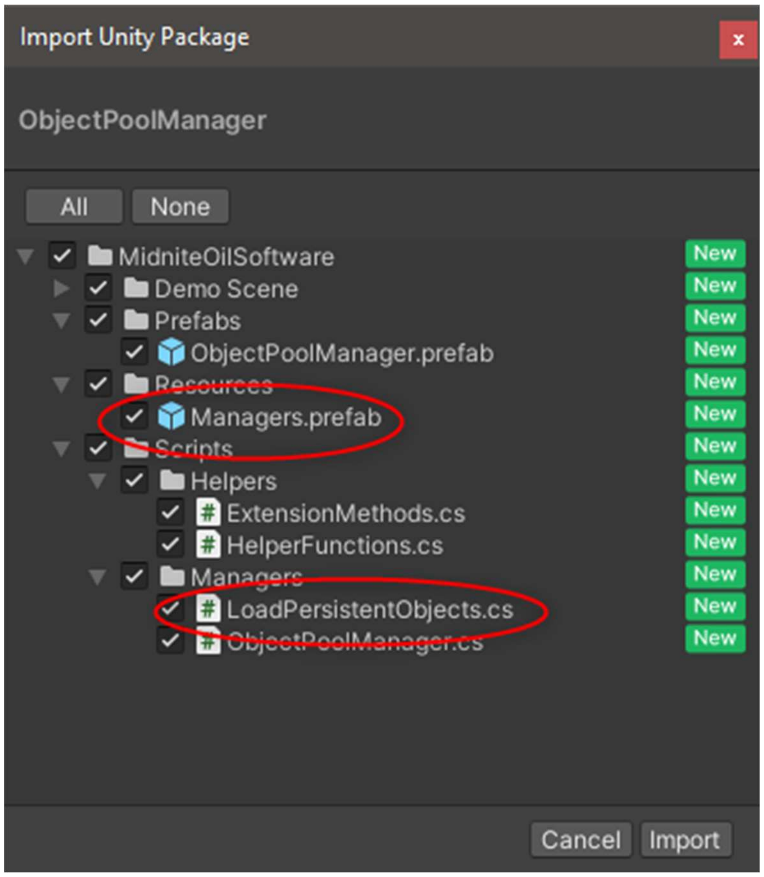
The ObjectPoolManager is a lightweight, easy-to-use system for adding object pooling to your game. It is designed specifically to work with prefabs (unlike the built-in object pooling system provided by Unity which requires you to use exact types).

Objects are only instantiated as needed and only take up storage in the pool when they are despawned. This makes retrieving pooled objects extremely performant because you don't have to search for inactive objects in the pool.

Installation

To install the Object Pool Manager simply download and import the package. The full package includes a sample scene that demonstrates how to use the manager.

If you want the manager to be automatically loaded when your game starts (and persist across scenes) make sure you import the Managers prefab and the LoadPersistentObjects.cs script.



```
✓ MidniteOilSoftware                 New
  ✓ Demo Scene                       New
  ✓ Prefabs                          New
    ✓ ObjectPoolManager.prefab       New
  ✓ Resources                        New
    ✓ Managers.prefab                New
  ✓ Scripts                          New
    ✓ Helpers                        New
      ✓ ExtensionMethods.cs          New
      ✓ HelperFunctions.cs           New
    ✓ Managers                       New
      ✓ LoadPersistentObjects.cs     New
      ✓ ObjectPoolManager.cs         New
```

If you exclude those files you will need to explicitly add the ObjectPoolManager prefab to a scene to use it.

Using the pool manager

The ObjectPoolManager is a static class which follows the singleton pattern. It is designed to work with prefabs. These can be as simple as a SerializedField of type GameObject:

```
[SerializedField]
GameObject _projectilePrefab;
```

Instead of directly instantiating the GameObject you can use the ObjectPoolManager.SpawnGameObject() method. There are multiple overloads of this method which takes optional parameters akin to the GameObject.InstantiateMethod().

```
GameObject projectile =
ObjectPoolManager.SpawnGameObject(_projectilePrefab,
transform.position, Quaternion.Identity);
```

To despawn an object simply call the ObjectPoolManager.DespawnGameObject() method passing in the object to be despawned.

```
public class DespawnAfterDelay : MonoBehaviour
{
   [SerializeField] [Range(5f, 30f)] float _despawnDelay = 10f;

   private void OnEnable()
   {
      Invoke(nameof(Despawn), _despawnDelay);
   }

   private void Despawn()
   {
      ObjectPoolManager.DespawnGameObject(gameObject);
   }
}
```

There are also methods to permanently destroy a GameObject and to clear the Object Pool.

```
Public static void PermanentlyDestroyGameObjectsOfType(GameObject
prefab);
```

```
Public static void EmptyPool();
```