# **Simple Pool**

## **Information**

Simple Pool is an object pooling solution for Unity. Designed to avoid the memory management issues associated with instantiating and destroying objects during runtime, Simple Pool has an easy-to-use, custom editor for the setup of object pools. Objects can then be taken from the pool, used in the scene and returned back to the pool during runtime, rather than being instantiated and destroyed on every use.

## **Scripts**

### ObjectPoolController.cs

#### **Description**

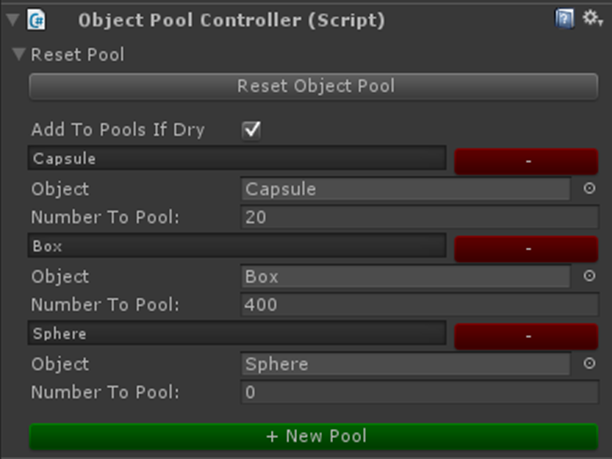
ObjectPoolController controls the overall ObjectPool. The overall ObjectPool contains multiple pools of single GameObjects. The initial setup of the ObjectPool is organised through the ObjectPoolController’s editor interface, as shown in figures 1 - 4 below.

Adds a new pool to the inspector.

Select to allow ObjectPoolController to add additional objects to the Object Pool during run time if there are no objects of the type required remaining in the pool.



**Figure 1. Empty Object Pool Controller**



Name of this pool type

Removes this particular pool type.

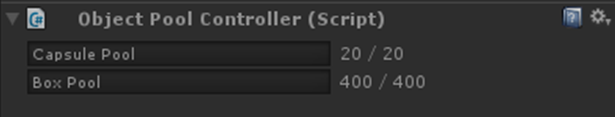
The number of this type of object to be spawned in the pool on scene awake.

Selection of object type to be pooled.

Resets the whole object pool.

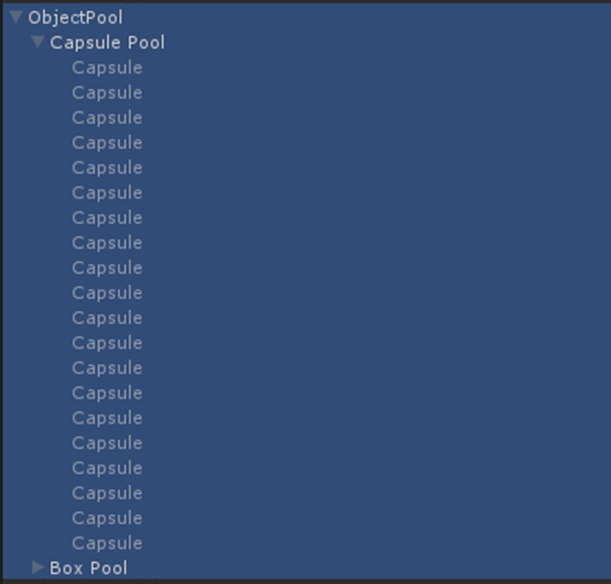
**Figure 2. Object Pool Controller in Use**

The number of objects left in this particular pool/total number of objects the pool contained on Awake.



**Figure 3. ObjectPoolController during play**

The Overall ObjectPool GameObject



Pool type

Individual objects held within this pool

**Figure 4. Objects pooled in the scene**

#### **Public Properties**

public static ObjectPoolController Pool

* Singleton instance of the ObjectPoolController. To use the ObjectPoolController methods, use the syntax:

ObjectPoolController.Pool.*MethodName();*

public List<ObjectPool> listOfObjectPools

* The list of all ObjectPools. In the ObjectPool represented in Figure 4 listOfObjectPools would contain 2 ObjectPools, one for the Capsule Pool and one for the Box Pool.

#### **Events**

ObjectPoolController.PoolInitialised

* Called once the overall Object Pool has been initialised. Used to control the order of execution for events that are required to wait for the Object Pool to be initialised.

#### **Methods**

**AddNewPool**

public void AddNewPool(ObjectPool objPool)

* Adds a new pool of a single object type to the listOfObjectPools within the ObjectPoolController.

**AddToExistingPool**

public void AddToExistingPool(GameObject objToPool)

public void AddToExistingPool(GameObject[] objsToPool)

* Adds either a single GameObject or multiple GameObjects to an existing pools of objects of the type specified through the “objsToPool” parameter.

**TakeFromPool**

public GameObject TakeFromPool(string objectType)

public GameObject[] TakeFromPool(string objectType, int numObjectsToTake)

* Removes either a single GameObject or multiple GameObjects from the relevant pool of objects.

**DestroyPool**

public Void DestroyPool(string poolType)

* Removes a pool of objects of the specified type if they are no longer required in the overall object pool.

**DestroyOverallObjectPooler**

public Void DestroyOverallObjectPooler ()

* Destroys the overall object pool GameObject, removing it from the scene.

### ObjectPool.cs

***Description***

ObjectPool represents a pool of a single type of GameObject. It contains a number of public properties that can only be read, but not assigned to externally. Its methods are used by the ObjectPoolController to manipulate the state of an ObjectPool.

### ObjectPoolEditor.cs

***Description***

ObjectPoolEditor controls the inspector interface for the ObjectPoolController. This should not be interacted with.