EZ Object Pools v1.4.1

Created By Kevin Somers

Contents

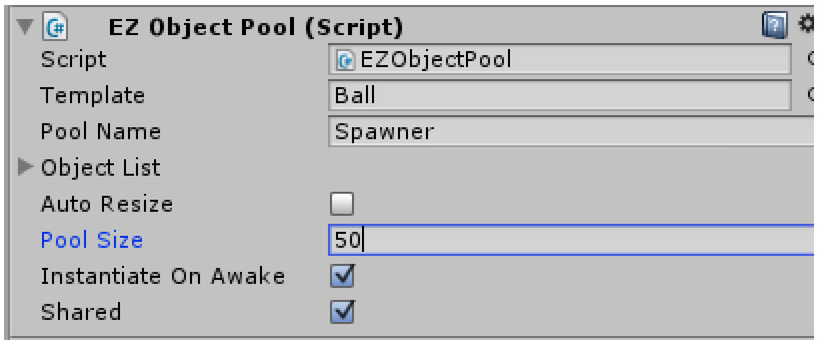
EZ Object Pools Overview

When creating and destroying objects at runtime, the easiest solution is to use the Instantiate() and Destroy() methods. However, these methods can be very intensive in terms of memory and processing, especially when they are being done in rapid succession.

The solution to this problem is Object Pooling, a much more optimized way of handling objects that must be instantiated and destroyed at runtime. The idea behind Object Pools is this: Instantiate a large number of objects on startup and deactivate them. When an object is needed, we simply activate an object from the pool instead of instantiating a brand new one. Deactivating an object will put it back into the pool for use again.

The EZ Object Pool system is designed to be simple to set up and easy to integrate into existing projects.

Object Pool Properties



EZ Object Pool (Script)

Script	EZObjectPool
Template	Ball
Pool Name	Spawner
Object List	
Auto Resize	☐
Pool Size	50
Instantiate On Awake	✓
Shared	✓

Object Pools have several properties:

- **Template:** The GameObject (usually a prefab) that this pool will make copies of.
- **Pool Name:** The name of this pool. It is used to identify pools that are set to be shared. Pools of the same name will be shared if the shared flag is set to true upon creation.
- **Object List:** This shows the list of objects instantiated by this pool.
- **Auto Resize:** Whether or not the pool should instantiate new objects if the pool is empty. Enabling this will ensure that when you request an object, you will get one. You would probably want to enable this for object pools that spawn important objects.
- **Pool Size:** How many objects are instantiated at startup.
- **Instantiate On Awake:** Should this pool be instantiated when the scene loads?
- **Shared:** Any scripts that request a pool with the same name as this pool will receive a reference to this pool, rather than creating a new one. This only works if the **Shared** parameter is also true in the **CreateObjectPool()** method.

Setting Up Object Pools

Object pools can be set up two ways. In this section I will walk you through both of these ways.

The Scripting Way

The first and recommended way to set up a pool is to create it in either the Start() or Awake() function in your scripts. This way requires 2 things: A reference to the pool and a line of code to create the pool:

```
EZObjectPool objectPool;

void Awake ()
{
        objectPool = EZObjectPool.CreateObjectPool(template, "TEST", 100, false, true,
        false);
}
```

The parameters for this method are:

- **Template:** The GameObject (usually a prefab) that this pool will make copies of.
- **Pool Name:** The name of this pool. It is used to identify it in the hierarchy as well as identify it for shared pools.
- **Pool Size:** How many objects are instantiated at startup.
- **Auto Resize:** Whether or not the pool should instantiate new objects if the pool is empty. Enabling this will ensure that when you request an object, you will get one. You would probably want to enable this for object pools that spawn important objects.
- **Instantiate Immediate:** Whether or not the pool should instantiate its object as soon as its created. You can instantiate the pool at any time in your scripts.
- **Shared:** Scripts using shared pools will draw their objects from a single pool in the scene, rather than having a pool for each instance of the script. Useful for scripts that may have many instances in a scene.

The method returns a reference to the created object pool.

The GameObject Way

The second way to set up a pool is to attach the script to a GameObject in your scene:

1. On any GameObject in your scene, click the Add Component button and click on *"EZ Object Pools"*, and click the Object Pool script.
2. From here you can assign the values as needed.

In your scripts, you must have an inspector field that you can drag and drop this object onto:

```
public EZObjectPool objectPool;
```

IMPORTANT!!! If you have a shared pool set up using this method, any calls to CreateObjectPool() that are expected to receive the shared pool MUST be done in the Start() method, not the Awake() method.

Retrieving Objects from Object Pools

Retrieving objects from an object pool is very simple. There are 2 methods you can use.

The first method is the simplest, and requires only one line of code:

```
objectPool.TryGetNextObject(position, rotation);
```

The parameters for this method are:

- **Position:** The position the object should be moved to when it is enabled.
- **Rotation:** The rotation the object should be set to when it is enabled.

This method can be used when a reference to the spawned object is not needed. If the pool is empty (and not set to auto-resize), then it will do nothing.

The second method can be used when a reference to the spawned object is needed:

```
GameObject outObject;

if(objectPool.TryGetNextObject(position, rotation, out outObject))
{
	//Do stuff with outObject
}
```

Making Objects Available for Use

In most cases, making an object available for retrieval is very easy: Simply disable the object, and the attached **PooledObject** component (see the below section for more info) will add the object back into the parent object pool's available object list automatically.

PooledObjects

A **PooledObject** is a script that automatically keeps track of the pool the object originated from. An object pool will automatically add this script as a component to any object it creates. However, you can also have scripts that inherit from **PooledObject** and attach these to your object instead, thereby gaining access to the object's parent pool. It will behave the same way as the automatically added script.

I have provided some examples of **PooledObject** derived scripts:

- **TimedDisable** will disable an object after the specified amount of time (in seconds).
- **ParticleSystemDisable** will disable an object once the particle system attached to it is finished emitting. You must use the Shuriken particle system for this to work.

Scripting

EZObjectPool

CreateObjectPool (Static)
Will create a new object pool with the specified properties.

Parameters:
- **Template (GameObject):** The GameObject (usually a prefab) that this pool will make copies of.
- **Pool Name (String):** The name of this pool. Its only use is to identify it in the hierarchy; otherwise it does not serve an important purpose.
- **Pool Size (Int):** How many objects are instantiated at startup.
- **Auto Resize (Boolean):** Whether or not the pool should instantiate new objects if the pool is empty. Enabling this will ensure that when you request an object, you will get one. You would probably want to enable this for object pools that spawn important objects.
- **Instantiate Immediate (Boolean):** Whether or not the pool should instantiate its objects when immediately (i.e. when this method is called). You can tell a pool to instantiate its objects at any time in your scripts.
- **Shared (Boolean):** Scripts using shared pools will draw their objects from a single pool in the scene, rather than having a pool for each instance of the script. Useful for scripts that may have many instances in a scene.

Returns:
- A reference to the created pool **(EZObjectPool)**.

TryGetNextObject (Overload 1)
Attempts to retrieve an object from the pool. If successful, it will return **True** and assign a reference to the out variable. Otherwise, it will return **False** and the out object will be set to **null**.

Parameters:
- **Position (Vector3):** The position the object should be moved to when it is enabled.
- **Rotation (Vector3):** The rotation the object should be set to when it is enabled.
- **Obj (out GameObject):** A variable that will be used to store a reference to the retrieved object.

Returns:
- **True** if an object is successfully retrieved.
- **False** if an object is not retrieved.

TryGetNextObject (Overload 2)
Attempts to retrieve an object from the pool. Will do nothing if an object cannot be retrieved.

Parameters:
- **Position (Vector3):** The position the object should be moved to when it is enabled.
- **Rotation (Vector3):** The rotation the object should be set to when it is enabled.

ClearPool
Deletes all objects created by this pool.

DeletePool
Deletes this pool.

Parameters:
- **Delete Active Objects (Boolean):** Whether or not it should also delete any objects that are currently active.

AddToAvailableObjects
Adds the object to this pool's list of available objects. DOES NOT add an object to the ObjectList.

Parameters:
- **Obj (GameObject)**: The object to add.

ActiveObjectCount
Gets the number of objects that are currently active.

Returns:
- The number of active objects **(Int)**.

AvailableObjectCount
Gets the number of objects that are currently available for use.

Returns:
- The number of available objects **(Int)**.

SetProperties
Sets the properties on an object pool instance. Generally, this should only be used by the **CreateObjectPool** method.

Parameters:
- **Template (GameObject):** The GameObject (usually a prefab) that this pool will make copies of.
- **Pool Name (String):** The name of this pool. Its only use is to identify it in the hierarchy; otherwise it does not serve an important purpose.
- **Pool Size (Int):** How many objects are instantiated at startup.
- **Auto Resize (Boolean):** Whether or not the pool should instantiate new objects if the pool is empty. Enabling this will ensure that when you request an object, you will get one. You would probably want to enable this for object pools that spawn important objects.
- **Instantiate On Start (Boolean):** Whether or not the pool should instantiate its object when the scene loads (In the Start() method). You can tell a pool to instantiate its objects at any time in your scripts.

InstantiatePool
Creates the set number of objects at the world origin (0, 0, 0) and sets them to inactive. Deletes any previous objects.

Template (GameObject)
The GameObject this pool will use and make copies of in the **InstantiatePool** method.

PoolName (Int)
The name of this pool. Is used to identify it for use in shared pools, as well as label it in the hierarchy.

ObjectList (List<GameObject>)
The list of all objects owned by this pool, including active and inactive objects.

AutoResize (Boolean)
Should this pool create new objects when it is empty?

PoolSize (Int)
How many objects are instantiated in the **InstantiatePool** method.

Instantiate On Awake (Boolean)
Only used for pools created in the editor. Should this pool be instantiated when the scene loads?

Shared (Boolean)
Only used for pools created in the editor. Any scripts that request a pool with the same name as this one will receive this pool, if they are marked as shared.

PooledObject

ParentPool (EZObjectPool)
The object pool this object originated from.

Disable (OBSOLETE)
Disables the object. Only calls gameObject.SetActive(false), no longer needed.
