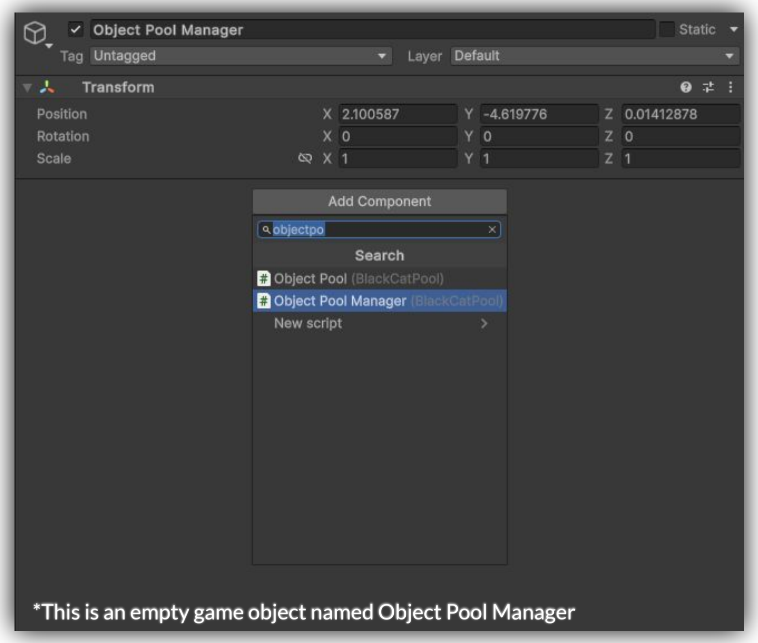
Simple Object Pooling

By The Black Cat

Content

Set Up - Object Pool Manager

Create an empty game object, or in whatever game object that is always active, attach the **Object Pool Manager** component.



Object Pool Manager Static

Tag Untagged Layer Default

Transform

	X	Y	Z
Position	2.100587	-4.619776	0.01412878
Rotation	0	0	0
Scale	1	1	1

Add Component

objectpo

Search

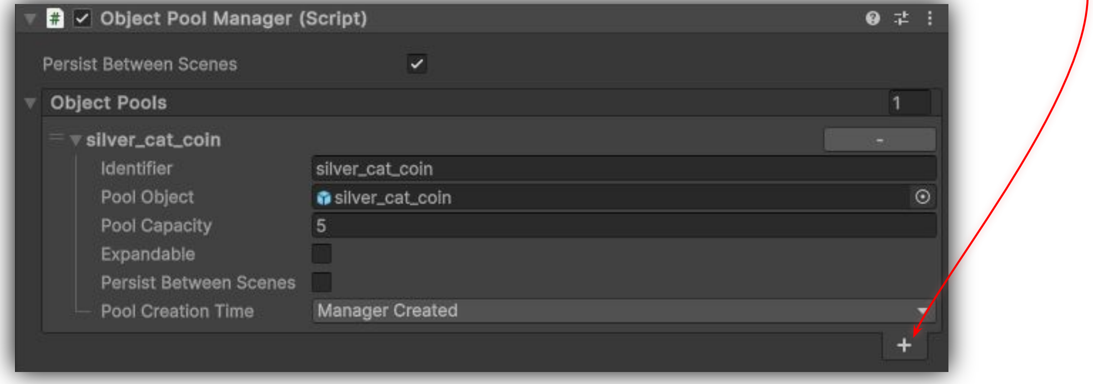
- Object Pool (BlackCatPool)
- Object Pool Manager (BlackCatPool)
- New script

*This is an empty game object named Object Pool Manager

Object Pool Registration - Register Pools

You can register pools in the manager, or create them in code. Registered pools in the manager will be created when the game starts.

To register a pool, in the **Object Pools** field, add a new pool:


Object Pool Manager (Script)

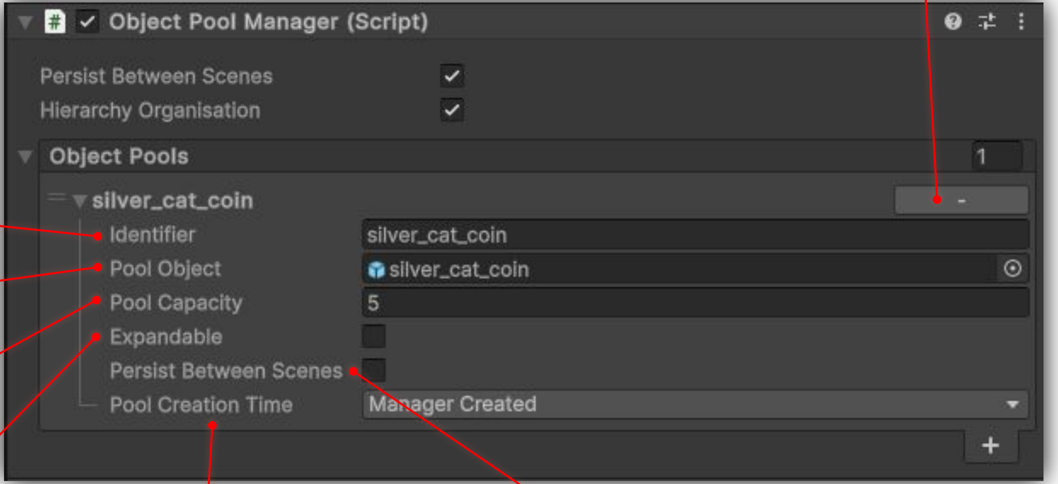
Persist Between Scenes ✓

Object Pools 1

silver_cat_coin

Identifier	silver_cat_coin
Pool Object	silver_cat_coin
Pool Capacity	5
Expandable	
Persist Between Scenes	
Pool Creation Time	Manager Created

Object Pool Registration - Register Pools



Object Pool Manager (Script)

Persist Between Scenes	✓
Hierarchy Organisation	✓

Object Pools: 1

silver_cat_coin

Identifier	silver_cat_coin
Pool Object	silver_cat_coin
Pool Capacity	5
Expandable	
Persist Between Scenes	
Pool Creation Time	Manager Created

Remove registry

The identifier (name) of the pool

The object to pool

The initial capacity of the pool

Whether the pool can expand and pool more objects than the capacity

The time this pool should be created

Whether the pool will persist when scene changes

Object Pool Registration - Pool Creation Time

In the **Object Pools**, there is an enum field **Pool Creation Time** at the bottom of each registration.

Pools that picked **Manager Created** will all be created as soon as the manager is created.

Pools that picked **Scene Loaded** will all be created when a specific scene is loaded, the scene name is specified below the enum field.

Pools that picked **Custom** will not be automatically created until you manually create them through code, see this page.

By default, pools will be create on manager creation.

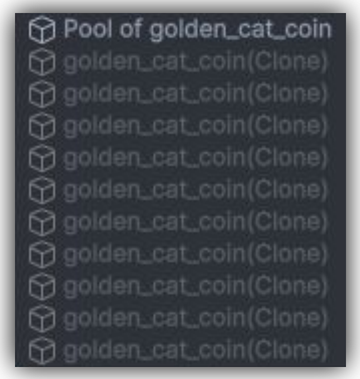
Object Pool Manager - Persist Between Scenes

You can decide whether the manager should persist when scene changes. Note that the manager persisting between scenes does not make the pools persist. In order to make the pools persist as well, you can to toggle it on inside each registrations.

Object Pool Manager - Hierarchy Organisation

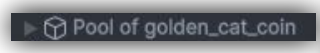
When this is toggled on, every pooled object in the scene, when pooled and inactive, will be set to be their pools' children. This ensure the organisation of your scenes.

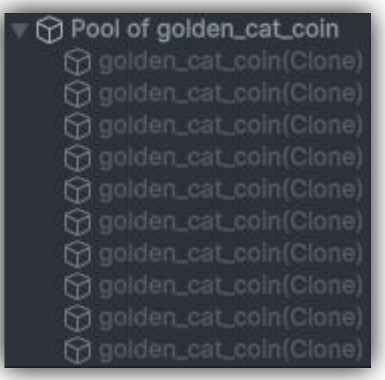
However, setting parents frequently can lead to bad performance, therefore this organisation will only work in the editor. You won't see the hierarchy in builds anyway.



← When toggled off

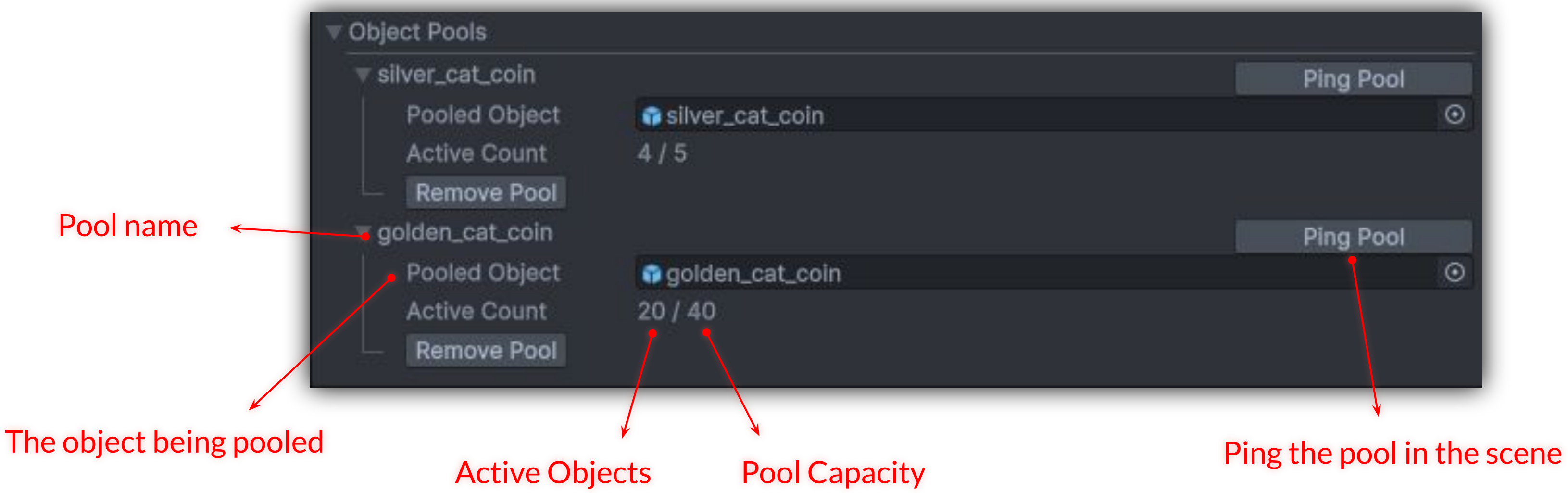
When toggled on →





Object Pool Manager - Pool Details

You can view details of the pools in the scene while in Play Mode. It is at the bottom of the manager.


Object Pools

silver_cat_coin

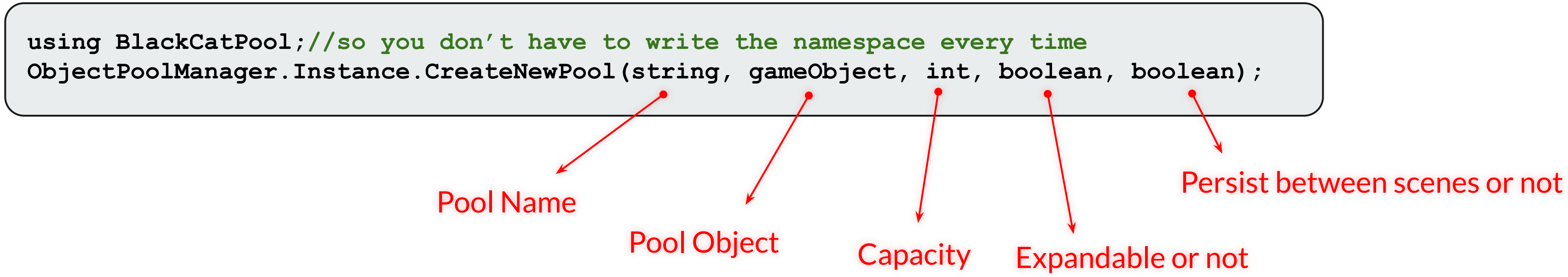
Pooled Object: silver_cat_coin
Active Count: 4 / 5

golden_cat_coin

Pooled Object: golden_cat_coin
Active Count: 20 / 40

Usage - Create a New Pool

To create a new pool in code, call this function:



```
using BlackCatPool;//so you don't have to write the namespace every time
ObjectPoolManager.Instance.CreateNewPool(string, gameObject, int, boolean, boolean);
```

*Of course, if the object to be pooled is a prefab, registering it in the manager is a better choice.

Time complexity: O(N) where N is the pool capacity

Usage - Create a Registered Pool

If **Pool Creation Time** is set to **Custom**, you can create it manually through code. Call this function:

```
ObjectPoolManager.Instance.CreateRegisteredPool(string);
```

And just pass the pool name in (which you provided in **Identifier** field).

Time complexity: O(N) where N is the pool capacity

Usage - Get Object Pool

To obtain an object pool, call this function:

```
ObjectPoolManager.Instance.GetPool(string);
```

And pass the pool name into the function.

Or call this function:

```
ObjectPoolManager.Instance.TryGetPool(string, out ObjectPool);
```

Just like other TryGet functions, this returns a boolean and out the pool if it exists.

Time complexity: O(1)

Usage - Remove Pool

When a pool is no longer in use and you wish to remove it, call this function:

```
ObjectPoolManager.Instance.RemovePool(string);
```

And pass the pool name into the function.

Or you can also call this direction from the pool:

```
ObjectPool pool = //Create Pool or Get Pool from manager
pool.RemovePool();
```

*When a pool is **disabled**, **every** active object obtained from it will be **returned** back to the pool as well. If the pool is **removed**, then the pooled objects will **also be removed**.

Time complexity: O(N) where N is the pool capacity

Usage - Get Pooled Object (By Manager)

You can obtain a pooled object by calling this function:

```
ObjectPoolManager.Instance.GetObject(string, boolean);
```

Pool Name

Whether to activate the object
(default false)

*If the pool is **unexpandable** and is **empty** when obtaining a pooled object, it returns **null**. So if your pool is unexpandable, **adding a null check before modifying the pooled objects is recommended**.

*When the pool **expands** its capacity, the capacity is **doubled**.

Time complexity: average O(1), or O(N) where N is the pool capacity if an expansion is needed

Usage - Get Pooled Object (By Pool)

If you hold the object pool, you can get the object directly from it.

```
ObjectPool pool = //Create Pool or Get Pool from manager
pool.Get(boolean);
```

Whether to activate the object

*If the pool is **unexpandable** and is **empty** when obtaining a pooled object, it returns **null**. So if your pool is unexpandable, **adding a null check before modifying the obtained objects is recommended**.

*When the pool **expands** its capacity, the capacity is **doubled**.

Time complexity: average O(1), or O(N) where N is the pool capacity if an expansion is needed

Usage - Try Get Pooled Object

If requiring null checks doesn't feel safe to you, you can try to get objects instead. Call this function:

```
ObjectPoolManager.Instance.TryGetObject(string, bool, out GameObject);
```

Or if you hold the object pool:

```
ObjectPool pool = //Create Pool or Get Pool from manager
pool.TryGet(bool, out GameObject);
```

The parameters are the same as simply getting them. These return true if an object is successfully obtained, or false otherwise (unexpandable & empty, for example). The pooled object is out.

Time complexity: average O(1), or O(N) where N is the pool capacity if an expansion is needed

Usage - Pool Object (By Manager)

You can return an object to the pool by manager or by pool. To pool it by manager, call this function:

```
ObjectPoolManager.Instance.PoolObject(gameObject);
```

The object to be returned

*It **automatically** finds the **correct** pool and return it back to the pool.

Time complexity: O(1)

Usage - Pool Object (By Pool)

If you hold the object pool, you can return the object directly to it.

```
ObjectPool pool = //Create Pool or Get Pool from manager
pool.Pool(gameObject);
```

Just pass the game object into it and it will be returned to the pool.

*If you returned the object to the **wrong** pool, it will **detect** it and **warn** you, and return it back to the **correct** one. **Please fix it though, it can affect the performance if it happens too frequently.**

Time complexity: O(1)

Usage - Get Multiple Pooled Objects

You can obtain multiple pooled objects at once. If you are doing it by the manager, call this function:

```
ObjectPoolManager.Instance.GetMultipleObjects(string, int, boolean);
```

Or if you are doing this direction from the pool, call this function:

```
ObjectPool pool = //Create Pool or Get Pool from manager
pool.GetMultiple(int, boolean);
```

The parameters are all the same as when you are getting a single object, except you have to specify the number of pooled objects you want to obtain at the **int** parameter.

*If the pool is **unexpandable**, it only returns the **remaining** pooled objects in the pool if the target amount is higher.

Time complexity: average O(M), or O(M+N) where M is the amount of obtainment and N is the pool capacity if an expansion is needed

Extension Method - Get Pool Of an Object

If you have a game object which you are sure it is a pooled object, and you want to obtain its pool, call this:

```
the_gameObject.GetPool();
```

This returns the pool if it is a pooled object, or null if it isn't or the pool doesn't exist.

*Avoid calling this frequently since this calls a **TryGetComponent()** and can have an impact on performance if there are too many objects. **Caching the pool on set up is suggested.**

Events - onPooledObjectCreated

This event is invoked when a new pooled game object is instantiated. This returns the pooled game object and the pool.

Events - onWillDestroyPooledObject

This event is invoked when a pooled game object is going to be destroyed. This returns the pooled game object and the pool.

Events - onObjectObtained

This event is invoked when a pooled object is obtained from the pool. This returns the pooled game object and the pool.

Events - onObjectPooled

This event is invoked when a game object is pooled in a pool. This returns the pooled game object and the pool.

Events - onPoolCreated

This event is invoked when a pool is created. This returns the created pool.

Events - onWillDestroyPool

This event is invoked when a pool is going to be removed and destroyed. This returns the pool.

Thank you for reading until here.

This asset is not perfect. Any suggestions, bug reports, reviews and advice will be highly appreciated.

If you have any problems, please contect me through:
Email: heinokchow314@gmail.com
Github: https://github.com/The-best-cat/Unity_asset-Simple_object_pooling

Good luck with your projects, and have a nice day.