My Pooler – Object Pooler - Documentation

Welcome to the documentation of ***My Pooler – Object Pooler***! Firstly, I would like to tell you I am very, very happy that you gave my asset a chance, and I really hope you enjoy it!
For any questions, bug reports or suggestions please contact me at my discord "Carlinhu#3159".
I'd be tremendously grateful if you could rate my package in your asset store downloads or leave a review on this asset page.

Getting Started

Setting up

The asset contains one main script called ObjectPooler.cs that controls all the functions of your pools, and a secondary script called PooledObjInterface that will handle some functionalities of the pool. All the scripts are detailed below.

- ObjectPooler.cs: This will be the main script added to your ObjectPooler object. It will handle all the pools that you want to create and its functionalities.
- PooledObjInterface.cs: This is the interface that will handle the method that will be called at the moment an object is called from any pool. This is useful to take care of fields that need to be reseted.

ObjectPooler.cs

Properties:

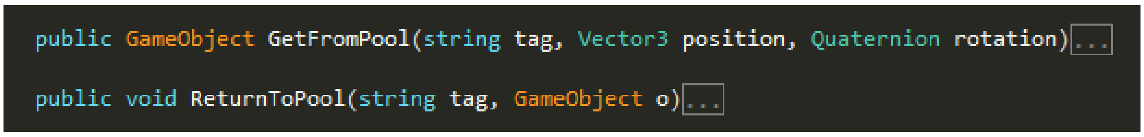
Property	Type	Description
shouldDestroyOnLoad	bool	If you want to destroy the ObjectPooler object or not during scenes transitions.
pools	List<Pools>	A list that holds all the different pools created by the user.
Pool		
size	int	The number of pools the user wants.

Property	Type	Description
tag	string	The exact name used to identify this pool.
prefab	GameObject	The prefab of the object that the user wants on the pool.
amount	int	The number of objects the user wants the pool to start with.
souldExpandPool	bool	If the pool should be incremented in case it reaches the its own limit.
extensionLimit	int	The limit of incrementations it can have if shouldExpandPool is set to true. Set to zero or less for unlimited.

- You can get the ObjectPooler object on the Prefabs folder. This asset is supposed to work as a generic object pooler, where you can have different pools and handle them easily. From there, all you need to do is setup your pool and create your own logic to handle the pool functionality. You can see a simple example below of how the system works.

Example

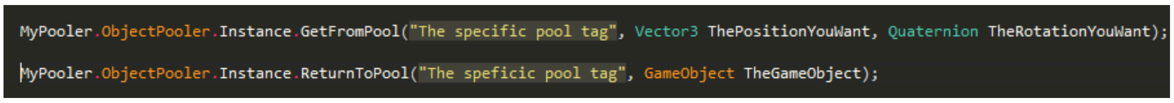
Methods: Those are the methods you will call get the object and return it to the pool. Both are in located **ObjectPooler.cs**:



```
public GameObject GetFromPool(string tag, Vector3 position, Quaternion rotation)...

public void ReturnToPool(string tag, GameObject o)...
```

You can call them like this:

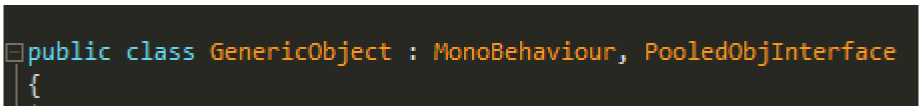


```
MyPooler.ObjectPooler.Instance.GetFromPool("The specific pool tag", Vector3 ThePositionYouWant, Quaternion TheRotationYouWant);

MyPooler.ObjectPooler.Instance.ReturnToPool("The speficic pool tag", GameObject TheGameObject);
```

If you have checked the “ShouldExpandPool” check box, in case your pool runs out of objects, it will automatically create another object and increment the pool size as it needs.

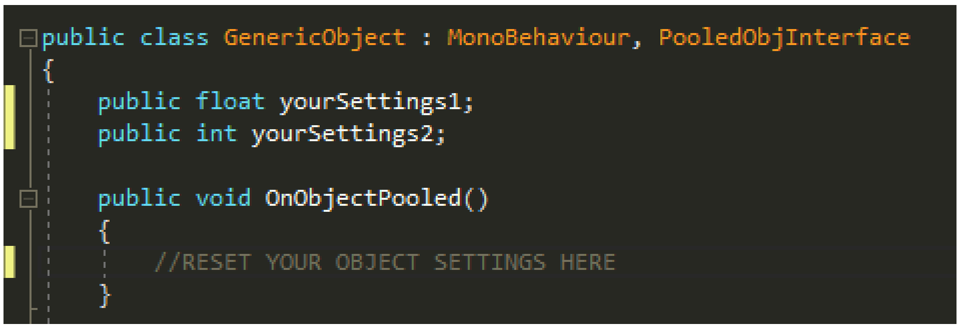
In case you need to reset some of your pooled object settings, you will need to used the “OnObjectPooled()” method from **PooledObjInterface.cs**.



```
public class GenericObject : MonoBehaviour, PooledObjInterface
{
```

Your object class must inherit from PooledObjInterface, and must have implemented the method “OnObjectPooled()”.

Inside the method, you will reset all your settings. This method will automatically be called when you get an object from the pool.



```
public class GenericObject : MonoBehaviour, PooledObjInterface
{
    public float yourSettings1;
    public int yourSettings2;

    public void OnObjectPooled()
    {
        //RESET YOUR OBJECT SETTINGS HERE
    }
```

There will be a simple demonstration on the project to clear things up, but if you have any question about it, feel free to contact me through Discord. I will be very happy to help you! 😉

This asset and documentation were created by Carlos Menezes Concencio