Object Pooling Tool

There is one script for the Object Pooling Tool which can be found under:

DangryGames -> ObjectPool -> Scripts -> ObjectPooling -> ObjectPool.cs

There are 3 help scripts which are attached to different game objects in the Demo scene.
They can be found under:

DangryGames -> ObjectPool -> Scripts -> Help

BackToPool.cs - A small example of how to return game objects back to the object pool
Comment.cs - A small script attached to game objects in the Demo scene that gives details of the other scripts attached to the game objects
SpawnObjects.cs - A small example of how to pull objects from the pool for your needs

A prefab of the object pooling game object can be found in the Demo scene or in the prefabs folder found under:

DangryGames -> ObjectPool -> Prefabs -> ObjectPool

The ObjectPool prefab contains two child game objects

1) ObjectsToPool
    a) Contains the ObjectPool script - This will allow you to add game objects to the pool
2) PooledObjects(InPlayMode)
    a) This is an empty game object that will be used as a holder for the pooled objects once the game is in play mode

How to use

1) Please check out the DEMO scene and see how the ObjectPool prefab is set up
2) Check out the Help scripts for how to activate a game object from the pool and also return the object back to the pool
3) You can drag in the ObjectPool prefab to your scene and test it out on your own game objects

The ObjectPool.cs file is commented so that you can follow along with what is happening.
The ObjectPool system works by picking a game object from the pool via the game object Tag. You will need to be very specific with which game object you wish to set active.
Video: https://youtu.be/ZvZCJrzLuqY

Quick fire

Get an object from the pool:

```
GameObject objectToSetActive = DangryGames.ObjectPooling.ObjectPool.SharedInstance.GetPooledObject("TestCube");
objectToSetActive.transform.position = _SpawnPosition.position;
objectToSetActive.transform.rotation = _SpawnPosition.rotation;
objectToSetActive.SetActive(true);
```

Create an object from the pool using the tag of the game object. You can see from the SpawnObjects.cs file how the code will look for a game object with the tag name of TestCube. It will then set the position and rotation to what is desired. For the Demo scene that is an empty game object under the Test game object called SpawnPosTest.
The next part is most important, set the newly pulled object to true.
This will turn on one of the TestCube objects in the pool and set it active in the scene.

Send object back to the pool:

In the Demo scene you will find the BackToPool.cs file attached to the BackToPool(Plane) game object.

This script is very simple and only contains a collision test. If a TestCube collides with the plane object, then send it back to the pool. It is very simple.

```
if (collision.gameObject.CompareTag("TestCube"))
{
  DangryGames.ObjectPooling.ObjectPool.SharedInstance.BackToPool(collision.gameObject);
}
```

The BackToPool method needs to be passed a GameObject. The method will then simply turn off the GameObject that is passed. You do not need to use this method, you can simply just turn off the GameObject in the collision.

```
public void BackToPool(GameObject GO)
{
    GO.SetActive(false);
}
```