typhoon.unity.pool使用说明

```
//案例一

private void Demo_1()
{
    //构建池
    var pool = PoolManager.CreatePool(() =>
    {
        var clone = new GameObject();
        return clone.AddComponent<PoolObject>();
    });

    //获取池对象
    var obj = pool.GetPoolObjectAs<PoolObject>();

    //回收池对象
    obj.RecycleToPool();
}
```

```
//案例二

private void Demo_2()
{
    //加载预制物资源
    var prefab = Resources.Load<GameObject>("path/of/pool_prefab");
    //构建池
    var pool = PoolManager.CreatePool<PoolObject>(prefab);
    //获取池对象
    var obj = pool.GetPoolObjectAs<PoolObject>();
    //回收池对象
    obj.RecycleToPool();
}
```

```
//案例三
//自定义池对象
```

```
3  public class MyPoolObject : PoolObject
4  {
5      public MeshRenderer Renderer;
6
7      //池对象被生成时触发
8      protected override void OnPoolObjectCreate()
9      {
10         base.OnPoolObjectCreate();
11         //获取MeshRenderer组件
12         Renderer = transform.GetComponent<MeshRenderer>();
13     }
14
15     protected override void OnPoolObjectOnWakeUp()
16     {
17         base.OnPoolObjectOnWakeUp();
18         //被唤醒（从池子拿出来）时触发
19         Renderer.enabled = true;
20     }
21     //回收逻辑执行前触发
22     protected override void OnPoolObjectBeforeRecycle()
23     {
24         base.OnPoolObjectBeforeRecycle();
25         Debug.Log("回收逻辑执行前触发");
26     }
27
28     //回收逻辑执行后触发
29     protected override void OnPoolObjectAfterRecycle()
30     {
31         base.OnPoolObjectAfterRecycle();
32         Debug.Log("回收逻辑执行后触发");
33         Renderer.enabled = false;
34     }
35 }
36
37 private void Demo_3()
38 {
39     //加载预制物资源
40     var prefab = Resources.Load<GameObject>("path/of/poolObject");
41     //构建池,使用拓展的池对象
42     var pool = PoolManager.CreatePool<MyPoolObject>(prefab);
43     //获取池对象
44     var obj = pool.GetPoolObjectAs<MyPoolObject>();
45     //隐藏renderer
46     obj.Renderer.enabled = false;
47     //回收池对象
48     obj.RecycleToPool();
49 }
```

```
//案例四
private void Demo_4()
{
    //构建池分组
    var group = new PoolGroup<string>();
    //加载预制物资源
    var car1Prefab = Resources.Load<GameObject>("path/of/car1Prefab");
    var car2Prefab = Resources.Load<GameObject>("path/of/car2Prefab");
    var car3Prefab = Resources.Load<GameObject>("path/of/car3Prefab");
    //构建池，并且添加到分组
    group.AddPool("car1", PoolManager.CreatePool<PoolObject>(car1Prefab));
    group.AddPool("car2", PoolManager.CreatePool<PoolObject>(car2Prefab));
    group.AddPool("car3", PoolManager.CreatePool<PoolObject>(car3Prefab));
    //根据key快速获取池对象
    var car1 = group.GetPoolObjectAs<PoolObject>("car1");
    var car2 = group.GetPoolObjectAs<PoolObject>("car2");
    var car3 = group.GetPoolObjectAs<PoolObject>("car3");
}
```