Easy Object Pool

Object Pooling System for Unity

Document History

Date	Author	Version
05 March 2015	Preet Kamal Singh Minhas	1.0, Initial Release

Introduction

Object pooling is a simple technique used to keep a pool of pre-instantiated objects. Object instantiation & destruction are very heavy operations and may slow down your game if a lot of objects are being created & destroyed rapidly. Using an object pool optimizes your game against such issues.

Integrating Easy Object Pool in your Unity project

Add the files **EasyObjectPool.cs** & **PoolObject.cs** to your project. If you are writing your scripts using JS, add these files to the 'Standard Assets' folder.

Configuring Easy Object Pool

1. Attach the EasyObjectPool script to a gameobject in the scene.
2. Set the pool info size to the number of pools that you want to create.
3. For each pool, define the following 4 fields:
    a. Pool Name: Assign a unique name for this pool.
    b. Prefab: The prefab or gameobject used to create this pool.
    c. Pool Size: The initial size of the pool. That is, the number of objects to create when this pool is created.
    d. Fixed Size: Whether this pool should remain at fixed size or expand whenever it is empty.

Using Easy Object Pool in your scripts

1. EasyObjectPool is defined under the scope of the MarchingBytes namespace. Import the namespace using the *using* or *import* directive based on your language of choice.
2. In order to retrieve an object from the pool, use the following method: *GameObject go = EasyObjectPool.instance.GetObjectFromPool(poolName,position,rotation);* *GetObjectFromPool()* will return a gameobject from the pool or null in case the pool is of fixed size & does not have any objects available.
3. In order to return an object to pool, use the following method: *EasyObjectPool.instance.ReturnObjectToPool(gameObject);*
4. **Important**: Object Pooling plays around with the active state of the gameobject. Hence, if you have functionality which was being handled in Awake() or Start(), move it to OnEnable(). Any code that was a part of OnDestroy() should be moved to OnDisable().

Handling special cases

- Reset rigidbody velocity & angularVelocity values in the OnEnable() method.

- TrailRenderer might exhibit streaking when used with object pool. The streaking can be eliminated by setting the TrailRenderer.time = -1 in OnDisable() & TrailRenderer .time=<requiredValue> in OnEnable().

Time Complexity

The methods *GetObjectFromPool() & ReturnObjectToPool()* have a time complexity of O(1).

Suggestions/Queries?

Please write in to contact@marchingbytes.com