

EPIC

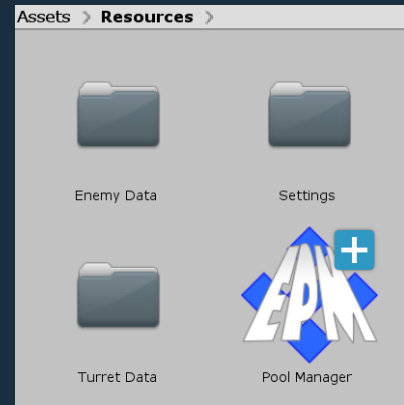
Pool Manager

CONTENTS

Creating The Manager	2
Configuring The Pools.....	2
Creating The Reference Script.....	4
Using the Pools at Runtime.....	4

CREATING THE MANAGER

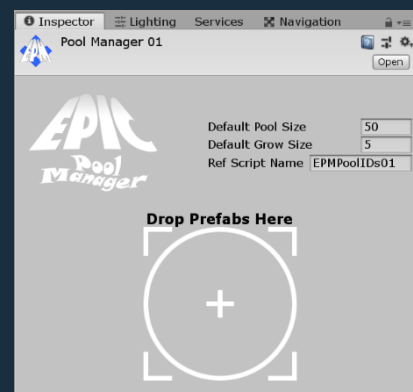
1. Create a *Resources* folder if you don't have one already
2. From the *Assets* file menu or *right-click, create* menu select *Epic Pool Manager/Manager* to create the new asset.



#NOTE: Only one pool manager should be running at a time – the EPM may be changed via script for example; changing levels.

CONFIGURING THE POOLS

1. Select the newly created manager asset and review the options in the inspector
2. Note the options for *Default Pool Size* and *Default Grow Size*. The pool size will be how many instances the pool will spawn in initialize, the grow size will determine how many more instances to add should the pool reach its maximum capacity. The default options will set the value for any new pool, each pool can have their own unique values. Also set the *Ref Script Name* to be something unique

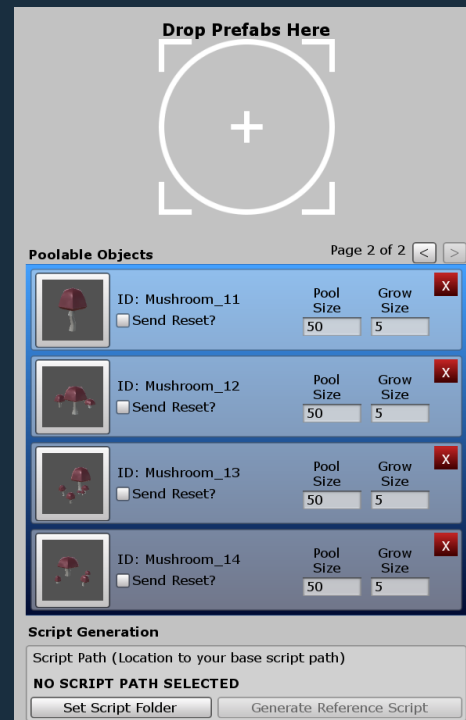


3. Drag and drop prefabs onto the drop zone, this can also accept multiple items in one drop to save you doing it one at a time

4. Note each object will be a unique pool. Each can have an individual pool size and grow size.

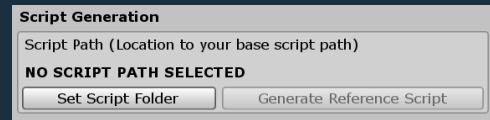
5. The *Send Reset* option can be used to easily reset an object on spawn. Simply tick this box and in your code have a method named *EPMReset*, when the object is spawned it will send a message to call this.

#NOTE: You must script the logic of how this object does the reset, this simply calls the method



CREATING THE REFERENCE SCRIPT

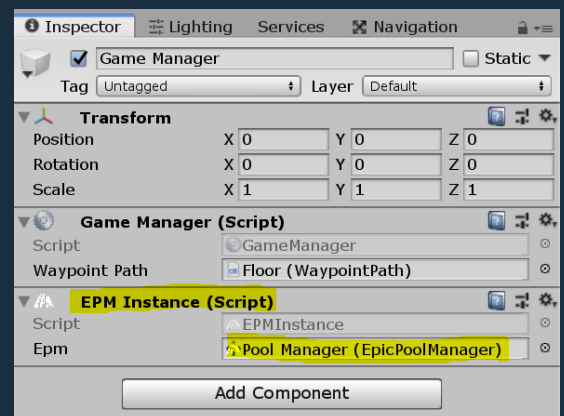
1. Once all the pools are set, the reference script must be generated. Select your base scripts folder by clicking on *Set Script Folder*. This will then ask you to locate a folder where the script will be created
2. Then click on *Generate Reference Script*. This will create a new static script which can be referenced by code when spawning the instances – this is why it is important to have a unique name for the *Ref Script Name* in the options



USING THE POOLS AT RUNTIME

1. On an object in your scene, add the *EPM Instance* component
2. Link the manager asset created earlier

#NOTE: This can also be done via script – the EPM Instance component is simply an easy visual way to initialize the manager. For info on how to do this via script see the ADVANCED TOPICS below



3. The EPM Instance will assign a static reference to the manager, so even if the object the component was added to is destroyed or the game changes scenes, the pool manager is still accessible.
4. To get an instance from a pool, call the `EPMInstance.SpawnObject()` method. It requires two variables, the ID of the object, and the parent it should be assigned to (the parent variable is optional, if omitted the object will child to the root of the scene)

```
void OnFire()
{
    if(controller.attackTarget == null) return;

    Projectile bullet = EPMInstance.SpawnObject
        (EPMPoolNames.BULLET015S, transform).GetComponent<Projectile>();
    bullet.transform.position = transform.position;
    bullet.Fire(controller);
}
```

NOTE: The ID is a constant string stored in the `EPMPoolNames` class generated in the steps above! This will vary based on the name you chose!

5. An object can also be spawned using a reference to a prefab instead of the ID string

```
public override void FireRound()
{
    Projectile bullet = EPMInstance.SpawnObject(data.projectilePrefab).GetComponent<Projectile>();
    bullet.transform.position = bulletSpawnLoc.position;
    bullet.transform.rotation = bulletSpawnLoc.rotation;
    bullet.Fire(this, GameManager.Settings.enemyMask, attackTarget.transform);
    lastFire = Time.time;
}
```

NOTE: The manager will log a warning in the console if this prefab is not configured in the database prior. It is strongly recommended to add it via the database ahead of runtime for performance reasons.

6. Every object spawned from the pool manager will have a new component called *EPMRecycleObject*.

When it comes time to destroy/recycle the pooled item, simply call the *Recycle* method on this component. For performance gains, it would be advantageous creating a reference to this component in the *OnEnable* method of your script and storing it for later use.

```
private void OnEnable()
{
    deathTime = Time.time + lifetime;
    if(recycler == null)
        recycler = GetComponent<EPMRecycleObject>();
}
```

```
void Die()
{
    if(Time.time > deathTime)
        recycler.Recycle();
}
```

7. If the *Send Reset* option was enabled for the pool, a message will be sent to the gameObject expecting to find at least one script with the Reset method.

This can be handy for resetting physics etc. This is optional and saves you having to handle the timing of the reset manually.

```
private void Reset()
{
    rigidbody.velocity = Vector3.zero;
    trailRenderer.Clear();
}
```

8. When it comes time to clean up, simply call *EPMInstance.ClearInstance()* – this will remove all subscribed events, gameObjects and parent container for the pool.

```
void ClearLevel()
{
    ///
    /// Other logic here
    ///
    EPMInstance.ClearInstance();
}
```

ADVANCED TOPICS

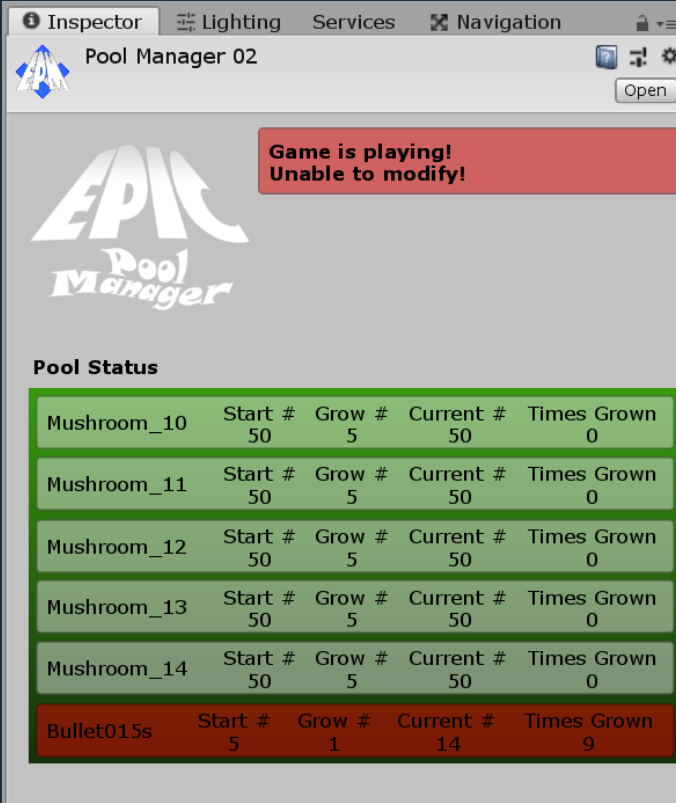
Using the EPMLInstance via Script

The EPMLInstance class doesn't have to be a component on a gameObject. The EPM can be initialized via script with the *EPMLInstance.CreateNewInstance()* method.

This is useful when using different EPMs for different levels/scenes.

Debug Status

When in playmode, the EPM object can display the performance of each pool. This can be very helpful tuning the pools for optimum sizes. Simply select it in the project window and view the inspector during playmode.



Inspector | Lighting | Services | Navigation

Pool Manager 02

Open

**Game is playing!
Unable to modify!**

EPIC Pool Manager

Pool Status

Mushroom_10	Start #	Grow #	Current #	Times Grown
	50	5	50	0
Mushroom_11	Start #	Grow #	Current #	Times Grown
	50	5	50	0
Mushroom_12	Start #	Grow #	Current #	Times Grown
	50	5	50	0
Mushroom_13	Start #	Grow #	Current #	Times Grown
	50	5	50	0
Mushroom_14	Start #	Grow #	Current #	Times Grown
	50	5	50	0
Bullet015s	Start #	Grow #	Current #	Times Grown
	5	1	14	9