

Advanced Object Pooler for Unity - Documentation

Advanced Object Pooler for Unity

A high-performance, expandable, and modular object pooling system to reuse GameObjects at runtime improving performance, reducing GC pressure, and enhancing game stability.

Features

- ScriptableObject-based configuration for easy pool setup
- Multiple pool support
- Dynamic pool expansion
- Lifetime-based auto-return
- Manual return-to-pool support
- IPooledObject interface for custom spawn/despawn behavior
- Plug-and-play compatible with any MonoBehaviour
- DontDestroyOnLoad option via singleton
- Minimal GC allocations

Folder Structure

ObjectPoolingSystem/

| | |
|------------------|---------------------------------|
| PoolManager.cs | Singleton pooling manager |
| PooledObject.cs | Auto-despawns based on lifetime |
| PoolConfig.cs | ScriptableObject config |
| IPooledObject.cs | Optional lifecycle interface |

Demo/

ExampleProjectile.cs Sample bullet with custom logic

Getting Started

1. Create the Pool Manager

1. In your scene, create an empty GameObject called `PoolManager`
2. Attach the `PoolManager.cs` script

2. Create Pool Configs

1. Right-click in the Project window

2. Choose Create Pooling Pool Config

3. Fill out the fields:

- Tag Unique string ID (e.g. "Bullet")
- Prefab Prefab to instantiate
- Initial Size How many objects to preload
- Expandable Can this pool grow beyond initial size?

3. Add PooledObject Component

Add the `PooledObject.cs` script to any prefab you want automatically returned to the pool after a delay.

4. (Optional) Implement IPooledObject

Add the `IPooledObject` interface to your prefabs script to respond to pooling events.

Spawning and Returning Objects

Spawn Object from Pool

```
GameObject obj = PoolManager.Instance.Spawn("Bullet", spawnPos, spawnRot);
```

Return Manually to Pool

```
PoolManager.Instance.ReturnToPool("Bullet", obj);
```

Example

```
if (Input.GetKeyDown(KeyCode.Space))  
{  
    PoolManager.Instance.Spawn("Bullet", transform.position, transform.rotation);  
}
```

```
private void OnCollisionEnter(Collision other)  
{  
    PoolManager.Instance.ReturnToPool("Bullet", gameObject);  
}
```

Tips & Best Practices

- Use clear and unique pool tags.
- Use `OnObjectSpawn()` to reset velocity, animations, etc.
- Use pooling for bullets, enemies, VFX, decals, UI popups.
- Use `DontDestroyOnLoad(PoolManager)` if pooling across scenes.

Support

For additional features like editor GUI, demo scenes, etc., contact the developer.