Poseidon

Why Do you need this?

A typical Shooting Game will have hundreds of scattered projectiles and explosion particle systems that may slow down your game as not only the number of instantiated objects doesn't have a ceiling; but as C# garbage collectors kick in at some point to clean up idle memory allocations causing a negative spike in the gameplay framerate.

Poseidon makes it easier for game programmers and designers to pool any game object regardless of its type

Implementation

Download and Import from Unity Package Manager like any other asset.

Project Setup

1. In your Projects Window, Right click, Create, Poseidon and select GameObject pool

Projectile Object

2. Open any class component attached to your projectile game object and implement the `IGameObjectPooled` interface It should add the line: `public GameObjectPool Pool { get; set; }`
3. Write your projectile end of life logic and at the point where the object is normally destroyed, write: `Pool?.ReturnToPool(this.gameObject);`

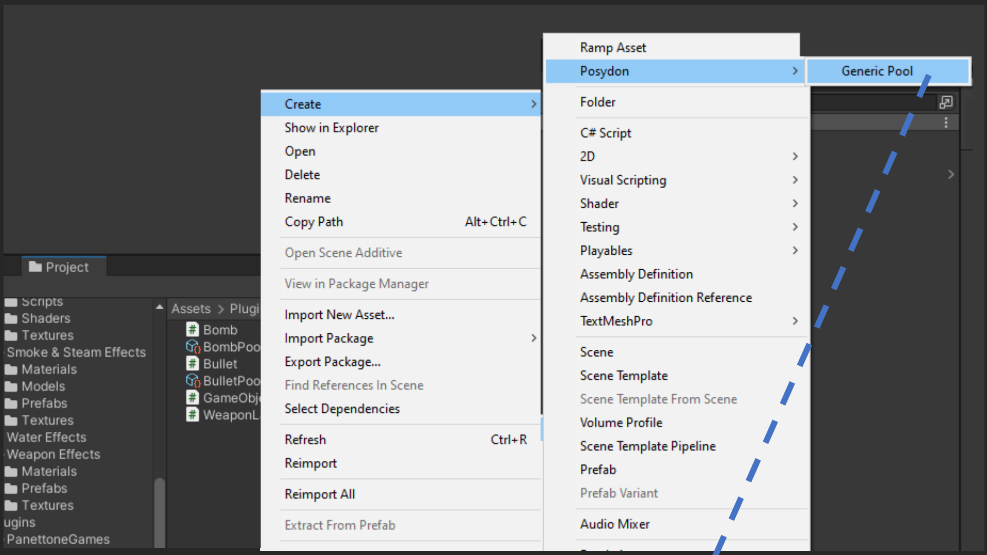
Launching Object

4. In your projectile launching class:
   - get the Pool: `[SerializeField] private GameObjectPool projectilePool;`
   - Pre-Warm it: `private void Awake() => projectilePool.PreWarm();` and
   - Shoot: `var shot = projectilePool.Get().SetActive(true);`

Unity Editor

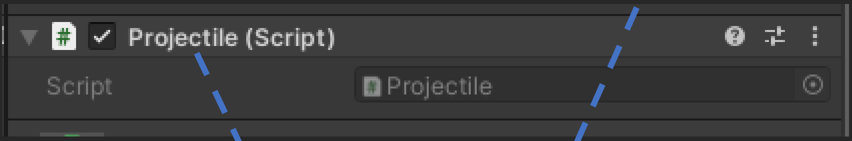
5. Back to unity Interface:
   - drag the projectile GameObject to your project folder to create a prefab
   - Select the pool scriptable object from your project folder and drag the projectile prefab to the GameObjectPool field in the inspector
   - Select the launching GameObject and drag the pool scriptable object to the ProjectilePool field in the.

Steps are illustrated on the next page. Watch the tutorial video if you need to or refer to the two Demo scenes included.



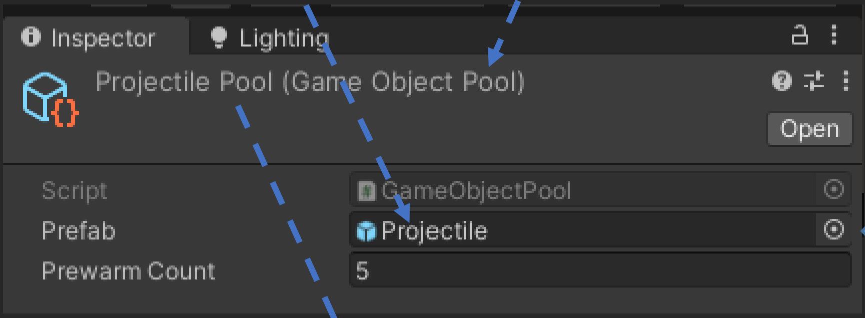
Create The Pool

Step 1


Projectile (Script)
Script: Projectile

Implements IGameObjectPooled

Step 2


Inspector | Lighting

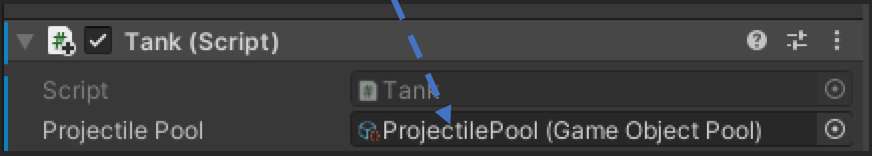
Projectile Pool (Game Object Pool)

Open

Script: GameObjectPool
Prefab: Projectile
Prewarm Count: 5

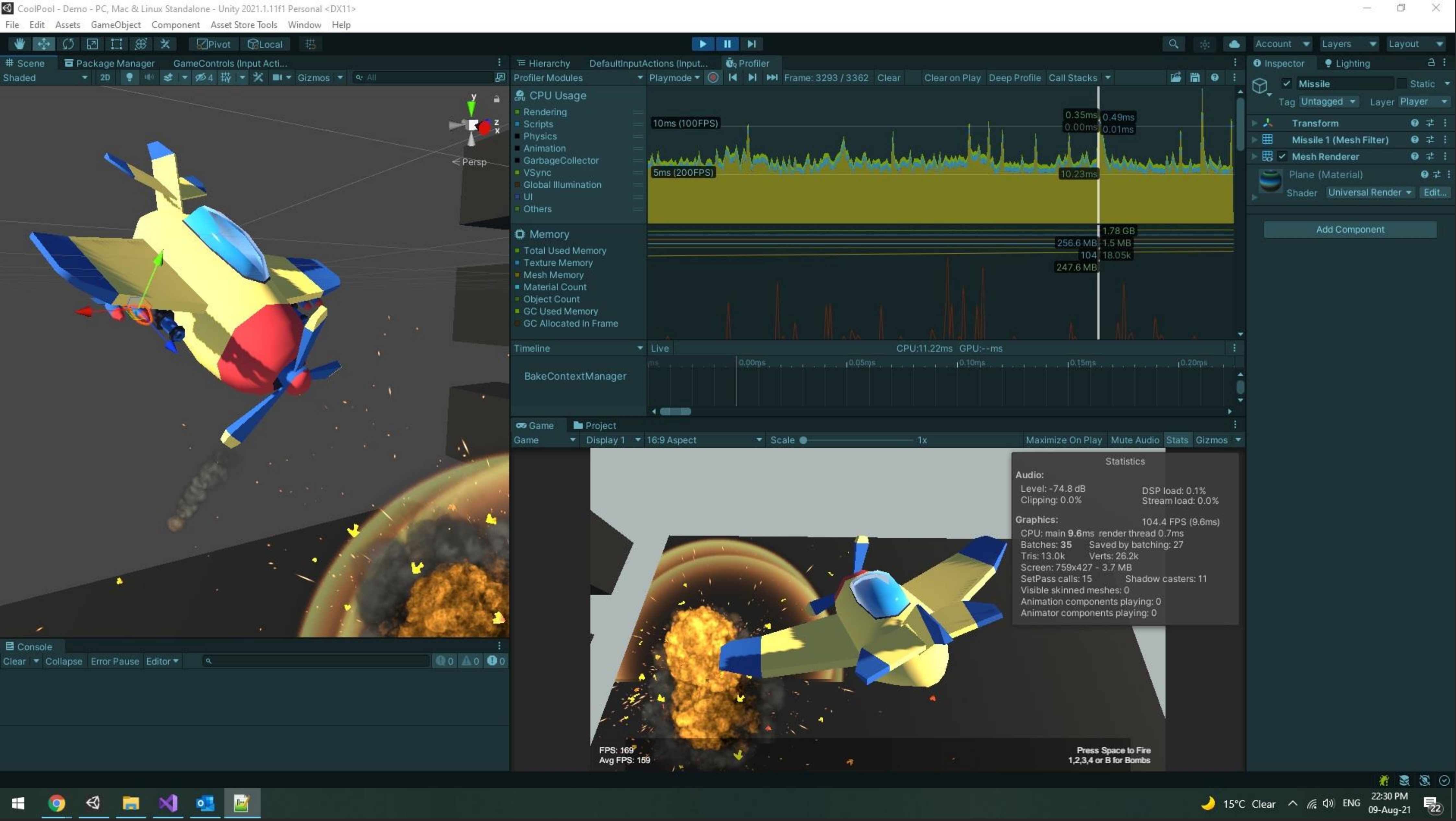
A Prefab of the Object containing Projectile Class (from previous step)

Step 3


Tank (Script)
Script: Tank
Projectile Pool: ProjectilePool (Game Object Pool)

The Pool (from previous step)

Step 4


CoolPool - Demo - PC, Mac & Linux Standalone - Unity 2021.1.11f1 Personal <DX11>
File Edit Assets GameObject Component Asset Store Tools Window Help
Pivot Local
Account Layers Layout
Scene Package Manager GameControls (Input Acti...
Shaded 2D Gizmos All
Persp
Hierarchy DefaultInputActions (Input... Profiler
Profiler Modules Playmode Frame: 3293 / 3362 Clear Clear on Play Deep Profile Call Stacks
CPU Usage
Rendering
Scripts
Physics
Animation
GarbageCollector
VSync
Global Illumination
UI
Others
10ms (100FPS)
5ms (200FPS)
Memory
Total Used Memory
Texture Memory
Mesh Memory
Material Count
Object Count
GC Used Memory
GC Allocated In Frame
Timeline Live CPU:11.22ms GPU:--ms
BakeContextManager
Inspector Lighting
Missile Static
Tag Untagged Layer Player
Transform
Missile 1 (Mesh Filter)
Mesh Renderer
Plane (Material)
Shader Universal Render Edit...
Add Component
Console
Clear Collapse Error Pause Editor
Game Project
Game Display 1 16:9 Aspect Scale 1x Maximize On Play Mute Audio Stats Gizmos
FPS: 169
Avg FPS: 159
Press Space to Fire
1,2,3,4 or B for Bombs
15°C Clear ENG 22:30 PM 09-Aug-21

```
Statistics
Audio:
  Level: -74.8 dB                  DSP load: 0.1%
  Clipping: 0.0%                   Stream load: 0.0%
Graphics:                          104.4 FPS (9.6ms)
  CPU: main 9.6ms  render thread 0.7ms
  Batches: 35      Saved by batching: 27
  Tris: 13.0k      Verts: 26.2k
  Screen: 759x427 - 3.7 MB
  SetPass calls: 15          Shadow casters: 11
  Visible skinned meshes: 0
  Animation components playing: 0
  Animator components playing: 0
```

Panettone Games

Productivity Tools for Game Developers