Custom Tick System for Unity

Overview

Custom Tick System is a centralized, low-GC tick manager that enables precise, scalable, and flexible time-based function execution in Unity — all without relying on Update() or coroutines.

It supports both attribute-based and code-based registration of ticks, and runs directly within Unity's PlayerLoop for optimal performance. Ideal for gameplay systems like AI loops, timed effects, background tasks, or damage over time.

Key Features

- Register any method or action with a custom tick interval
- Zero hot-path GC allocations during runtime
- Supports delayed starts, one-shot execution, pause/resume
- Use [Tick] attribute or fluent TickBuilder API
- TickManager runs globally — no need to attach components
- Full editor debugging window with live control
- Fully unit tested and documented

Getting Started

1. Attribute-based Tick

Mark any parameterless method in a MonoBehaviour with [Tick(interval, delay)]:

```
public class MySpawner : MonoBehaviour
{
    [Tick( interval: 2.0f)]
    private void SpawnEnemy()
    {
        // Called every 2 seconds
    }

    [Tick( interval: 1.0f,  delay: 0.5f)]
    private void HealOverTime()
    {
        // Called every 1s, starts 0.5s after scene load
    }
}
```

This is automatically detected at scene load.

---

2. Code-based Tick with TickBuilder

Register any action or method at runtime using a fluent API:

```
var handle = TickBuilder.Create(() => DoSomething())
    .SetInterval(1.5f)
    .SetDelay(0.2f)
    .SetDescription("MySystem.DoSomething") // TickBuilder
    .Register();

TickBuilder.Create( target: this, nameof(HandleTick), new object[] { value })
    .SetInterval(0.5f) // TickBuilder
    .Register();
```

Controlling Ticks

Each registration returns a TickHandle, which allows control of the tick's lifecycle:

```
handle.Pause();
handle.Resume();
handle.Unregister();
```

Handles are lightweight and safe. Invalid handles are ignored safely.

Editor Tools

- Open via Window → Energise Software → Custom Tick
- View all active ticks grouped by interval
- Search, sort, pause/resume/remove ticks at runtime
- Debug descriptions show method or delegate source
- Zero-GC update loop verified in the profiler

Use Cases

- AI behavior ticking
- Damage over time / healing effects
- Cooldowns and timers
- Background sync, polling, or logic updates
- Lightweight ECS-like update patterns
- Runtime orchestration of scheduled tasks

Performance

- Zero GC allocations in runtime ticking
- Internally uses Unity's PlayerLoop for consistent frame-time integration
- Handles thousands of concurrent ticks efficiently
- Attribute scan is done once per scene load

Advanced Features

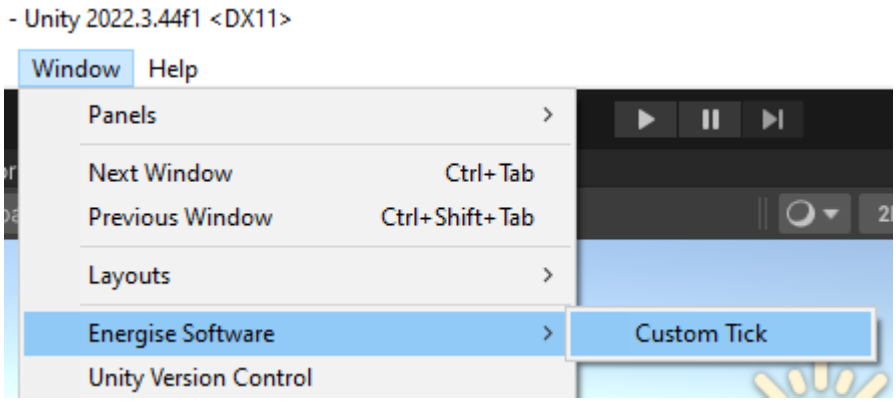
- One-shot ticks: run once then auto-unregister
- Delayed start: offset first execution independently of interval
- Full unit test suite included for coverage & reliability
- Editor-only debug data (type, source, description)

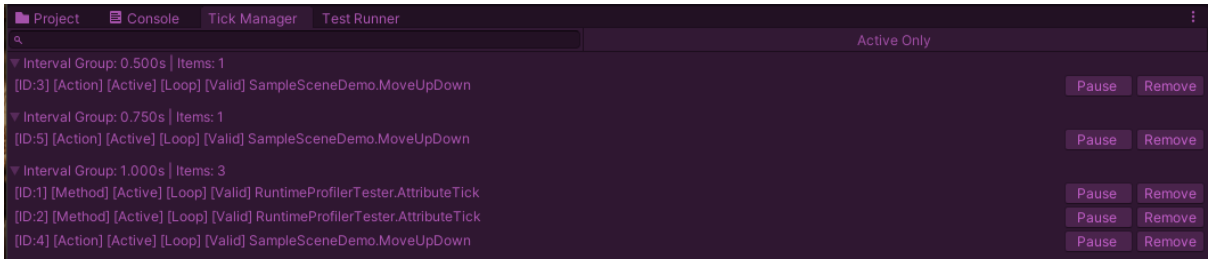
Requirements

- Unity 2021.3 LTS or newer
- Compatible with URP, HDRP, and Built-in
- No third-party dependencies

Installation

1. Import the package into your Unity project
2. Optionally add the [Tick] attribute to methods
3. Or use TickBuilder to register ticks in code
4. Open the Tick Manager window for runtime inspection and control





Known Limitations

- [Tick] attributes only apply to parameterless methods
- Methods with dynamic parameters must be registered manually

---

7. Support & Feedback

Any questions, comments, requests, please contact:

energisetools@gmail.com

https://discord.gg/vpCbqQMdPJ