Coroutine Debugger

The Coroutine Debugger (CD) is a powerful tool designed to empower you in monitoring and controlling coroutines within your game scenes. It provides a coroutine management system that emulates and then adds functionality on top of the Unity system. It also includes a unique debugger that lets you see the coroutines in your scene, their state, their inter-relationships, and much more.

API Documentation

https://aeric.games/coroutinedebugger/api/html/index.html

CoroutineManager

The core of CD is a coroutine management system that is designed to emulate and then exceed the standard Unity system.

Converting Unity standard coroutines

If you have a project already using coroutines then converting them to CD coroutines is simple.

Instead of calling StartCoroutine with your enumerator, you call BeginCoroutine instead. BeginCoroutine is an extension method designed to help with the most common use case of starting coroutines.

```
//Extension method designed to help with converting from Unity coroutines
0 references
public static class StartHelper
{

    7 references
    public static IEnumerator BeginCoroutine(this MonoBehaviour monoBehaviour, IEnumerator enumerator)
    {
        var handle = CoroutineManager.StartCoroutine(enumerator);
        CoroutineManager.Instance.SetCoroutineContext(handle, monoBehaviour.gameObject);
        return enumerator;
    }
}
```

This helper method calls the StartCoroutine method in CoroutineManager and also sets the context for the coroutine to be the GameObject. The context for any coroutine does not have to be the GameObject and coroutines do not have to be connected to GameObjects at all.

Features

Execution controls
The CoroutineManager class contains various methods for controlling coroutine execution:

- StopCoroutine
- PauseCoroutine
- ResumeCoroutine
- ResetCoroutine
- StepCoroutines

All of these methods have a version that accepts a list of coroutines.

Tags / Layers/ Context
Metadata can be associated with coroutines to enable grouping them in various ways that the game requires.
Tags - string metadata
Layer - integer metadata
Context - object metadata (usually the GameObject)

Lists of coroutines can then be retrieved by specifying metadata.

- GetCoroutinesByName
- GetCoroutinesByContext
- GetCoroutinesByLayer
- GetCoroutinesByTag

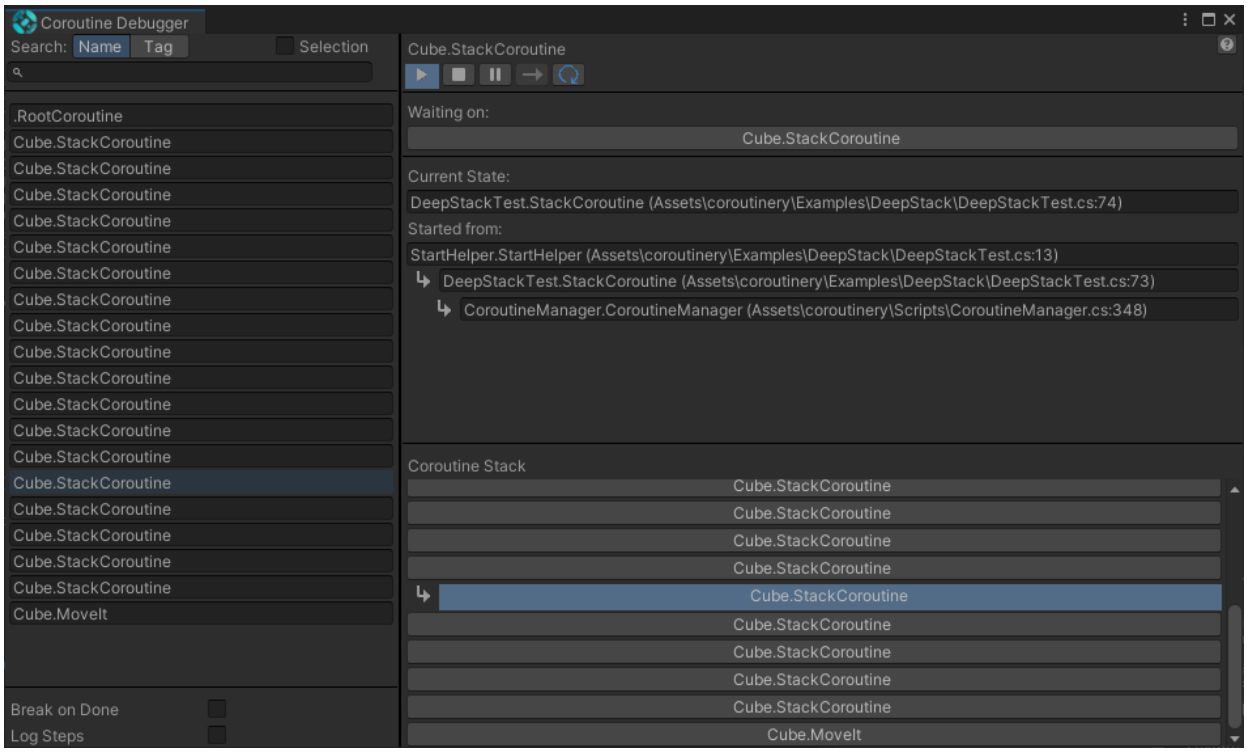
New yields
CD adds new yield instructions:

- WaitForFrames - yields for a specified number of frames
- WaitForLateUpdate - yields execution until the LateUpdate part of the frame

Coroutine Debugger

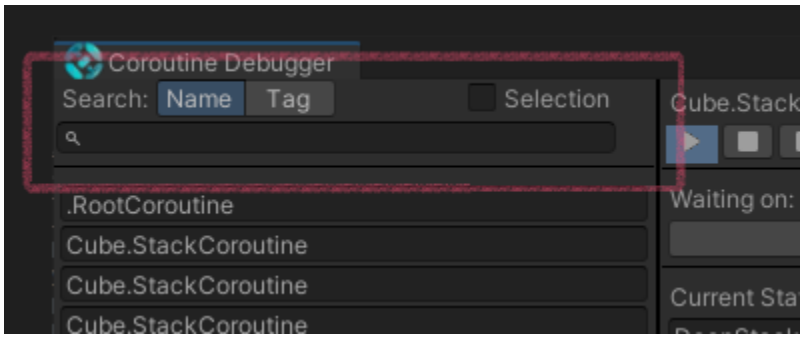
How to Access

Window/Aeric/Coroutine Debugger



Features

Search bar

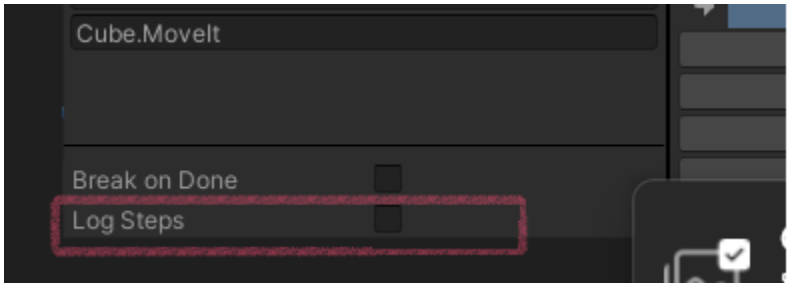


The search bar allows you to filter the coroutines in the scene by name, or by tag. You can also check the Selection box to limit the search to just those coroutines that have a GameObject context included in the scene selection.

Break on Done

This toggle will cause Unity play mode to pause any time a CD coroutine finishes.

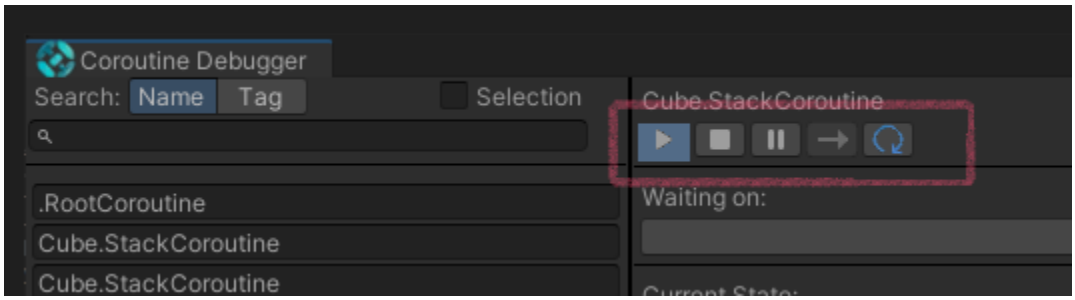
Log Steps



This toggle will cause CD to log all coroutine execution steps to the console.

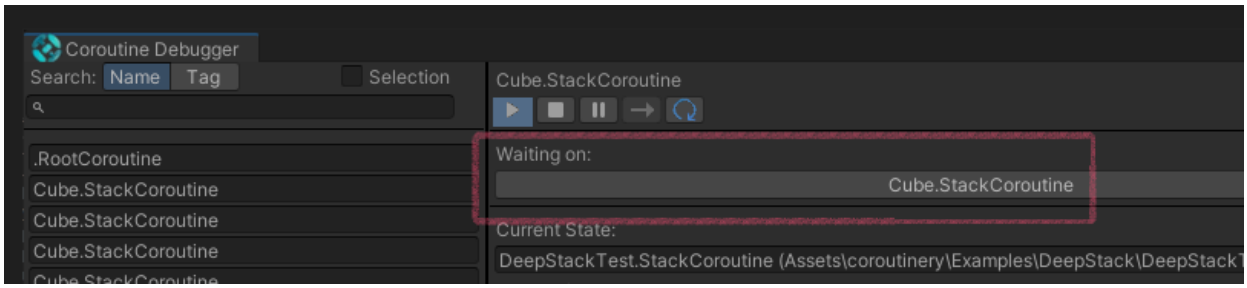
Debug Controls

When a coroutine is selected this set of widgets allows the user to perform various operations related to its execution.



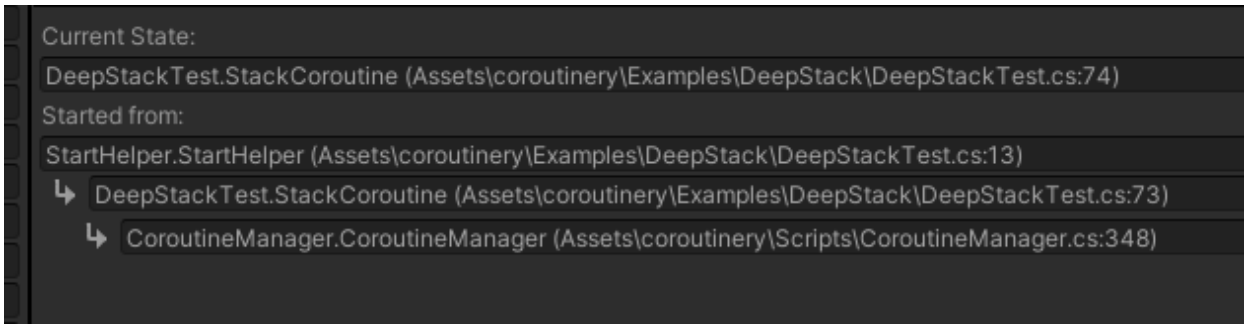
- Pause/Resume
  - The coroutine can be paused, and if paused can be resumed
- Stop
  - The coroutine can be stopped
- Single step
  - If a coroutine is paused then we can run a single “step”, which means it will run to its next yield statement.
- Restart
  - This will reset a coroutines internal state so it starts from the beginning again. Be advised that this only resets the internal state of the coroutine and no other game state, including other coroutines, so it might not always behave as expected.

Waiting state



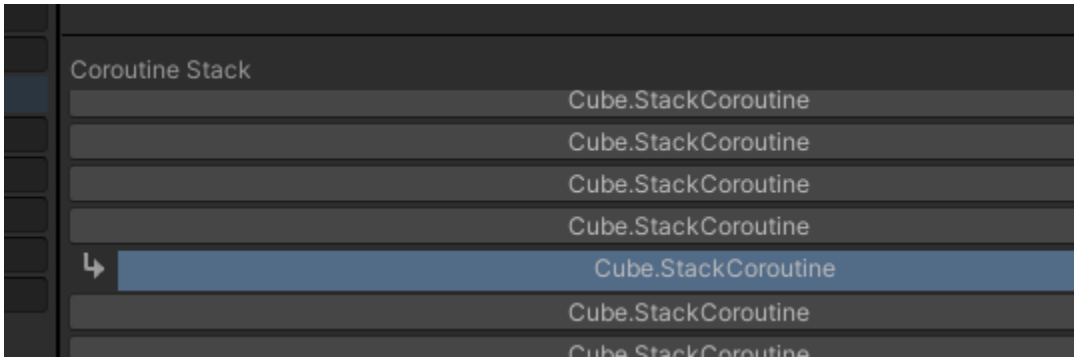
This section of the UI shows what the coroutine is currently waiting on, i.e. what it yielded on. This might be another coroutine, in which case you can then select that coroutine. It might be a WaitForSeconds or WaitForFrames yield instruction, in which case it will display the state of that instruction (e.g. seconds remaining).

Source References



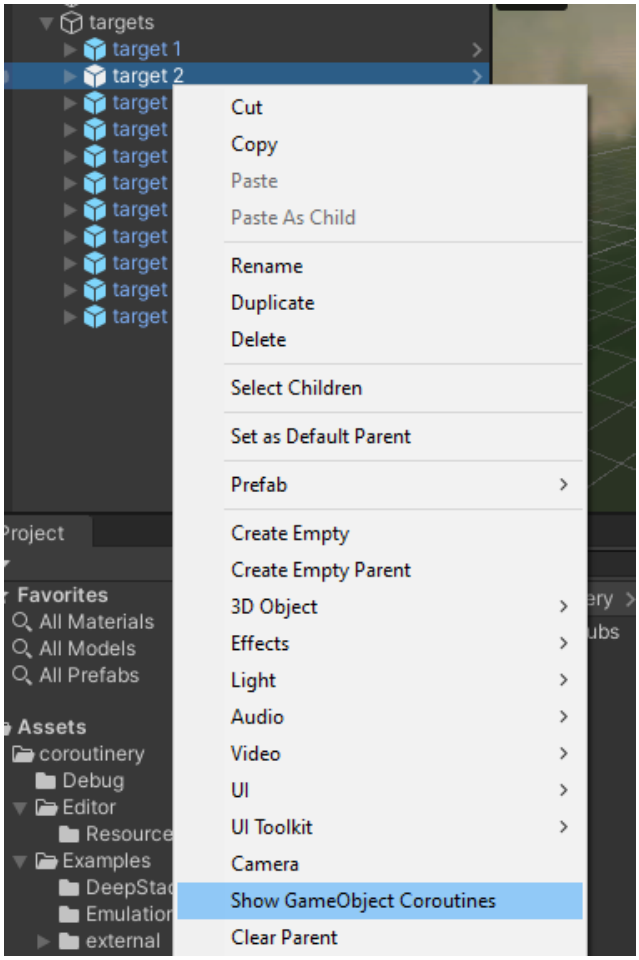
The Current State shows the last line of execution where the coroutine yielded. The Started From section shows the call stack from where the coroutine was initiated.

Coroutine Stack



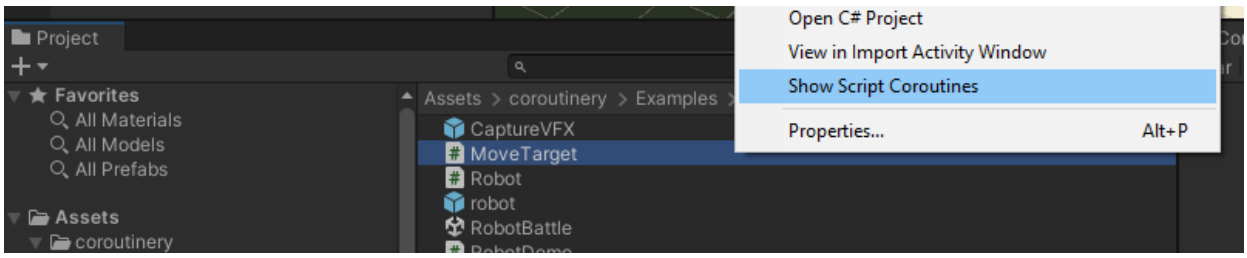
A coroutine may be waiting on another coroutine, which in turn may be waiting on a coroutine. This means that a coroutine can be part of a chain of coroutines each waiting on the next. This section will show all the coroutines in the current stack, highlighting the current coroutine, and allowing each to be selected.

Show Object Coroutines



This is a context menu option on scene objects that tells the debugger window to show all coroutines that have this GameObject as their context.

Show Script Coroutines



This is a context menu option on script files that tells the debugger to show all coroutines that were initiated by this script.

Demos

All the demos can be accessed via the menu in the Assets/coroutinery/Examples/menu.unity scene.

- **Emulation**
    - This demo has 2 cubes, one animated using standard Unity coroutines, and the other using CoroutineManager coroutines. This demo is designed as a test to ensure that the CoroutineManager system is emulating the standard Unity behavior with regards to coroutines, yields, and the various Wait* yield instructions. At the end of the test we verify that the timings and transforms match.
- **Pause**
    - This demo shows the ability to pause and resume coroutines.
- **Deep Stack**
    - This demo is a stress test of the CoroutineManager system that has a deep chain of nested coroutines.
- **Robot Battle**
    - This demo shows a game-like scene where AI robots are battling to capture points on a map.
- **Labels**
    - This demo shows the ability to specify a label for coroutines and apply operations to them as a group.

Source Mapping

CD has the option to build source mappings when code is recompiled. This is what allows CD to show source listings for the coroutines. This is automatically enabled by default, and typically is fast enough that it shouldn't even be noticeable. CD does provide an option to disable this though in the menu should the user which to do that. Note that disabling source mapping will cause some parts of the debugger window not to function.

Stack traces are collected when coroutines are started, when playing in the editor. This is not done at runtime, and can also be disabled via the menu if desired.

Menu options