Universal Coroutine

Features:
- Works in both Play mode and in Edit mode
- Avoids unnecessary Garbage Collection/memory allocation by using Object Pools.
- Same coroutine methods can be executed in Play mode and Edit mode*
- Supports nested coroutines and most of the other functions of the normal Unity coroutines

*Not all yield options are supported in Edit mode (they will be ignored in Edit mode)

Supported in Play mode:
- Start coroutine(s)
- Stop coroutine(s)
- Pause coroutine(s)
- Resume coroutine(s)
- yield return null
- yield return break
- yield return WaitForSeconds
- yield return WaitForFixedUpdate
- yield return WaitForLateUpdate
- yield return WWW

* yield return WaitForEndOfFrame not supported, because Unity doesn't expose this event

Supported in Edit mode:
- Start coroutine(s)
- Stop coroutine(s)
- Pause coroutine(s)
- Resume coroutine(s)
- yield return null
- yield return break
- yield return WaitForSeconds
- yield return WWW

How to use:

There are some differences between regular Unity coroutines and Universal Coroutines:

1. **Managed differently:**
   - Unity coroutines are linked to MonoBehaviours
   - Universal coroutines are globally managed by the CoroutineManager class
2. **Syntax is different:**
   - Because Universal Coroutines uses Object Pooling, you don't use “new” keyword in the coroutine method.
   See below paragraph for more information.

There are 2 ways to call Universal Coroutines functionality in a script:

NOTES:
- You need to import “UniversalCoroutines” to use the extension methods or access CoroutineManager class.
- yield return null, yield break and yield return www stays the same inside a coroutine method.
- You can just use the WWW class like you normally would do.

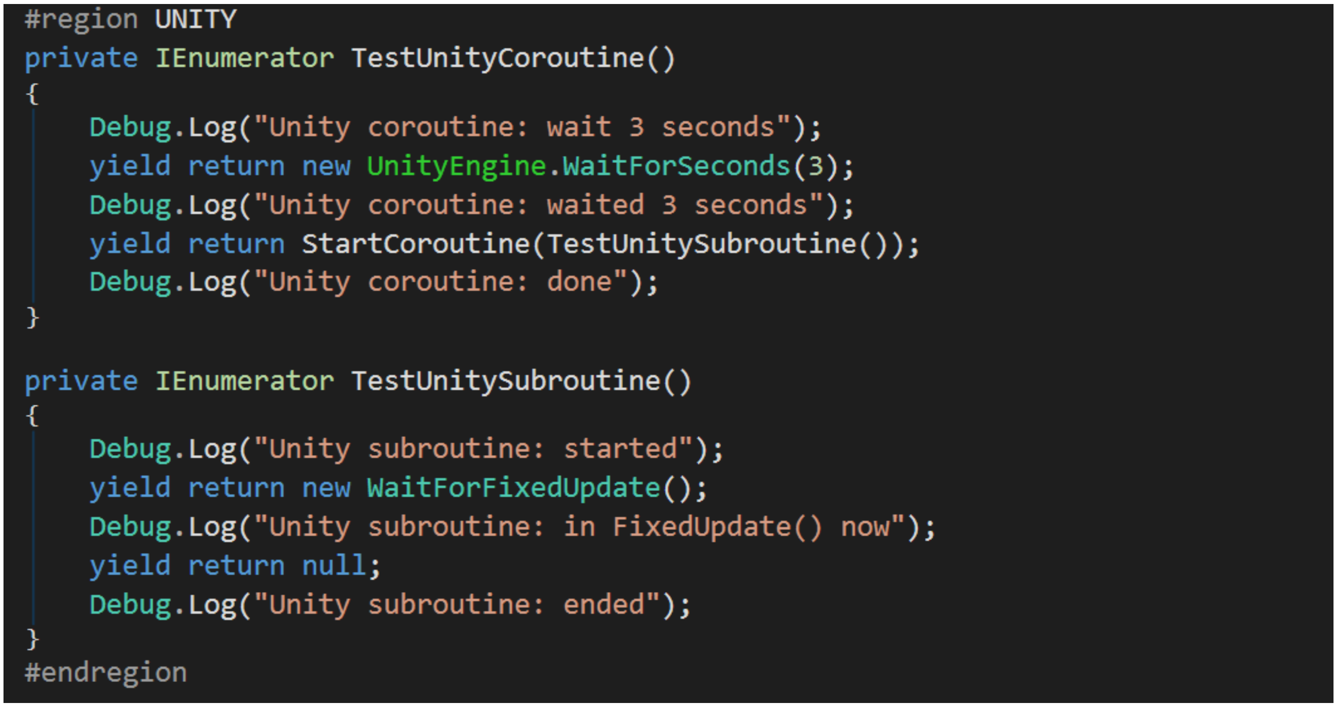
1. If the script is a MonoBehaviour or EditorWindow you can use the extension methods:

All extension methods start with “Uni” for consistency and can be called using “this.Uni...”.
- UniStartCoroutine
- UniWaitForSeconds
- UniWaitForFixedUpdate
- UniWaitForLateUpdate
- UniStopCoroutine
- UniStopAllCoroutines
- UniPauseCoroutine
- UniPauseAllCoroutines
- UniResumeCoroutine
- UniResumeAllCoroutines

2. Directly call the CoroutineManager:

Instead of the “wrapper” extension methods you can call the CoroutineManager methods directly. These can be called from any script:
- CoroutineManager.StartCoroutine
- CoroutineManager.WaitForSeconds
- CoroutineManager.WaitForFixedUpdate
- CoroutineManager.WaitForLateUpdate
- CoroutineManager.StopCoroutine
- CoroutineManager.StopAllCoroutines
- CoroutineManager.PauseCoroutine
- CoroutineManager.PauseAllCoroutines
- CoroutineManager.ResumeCoroutine
- CoroutineManager.ResumeCoroutines

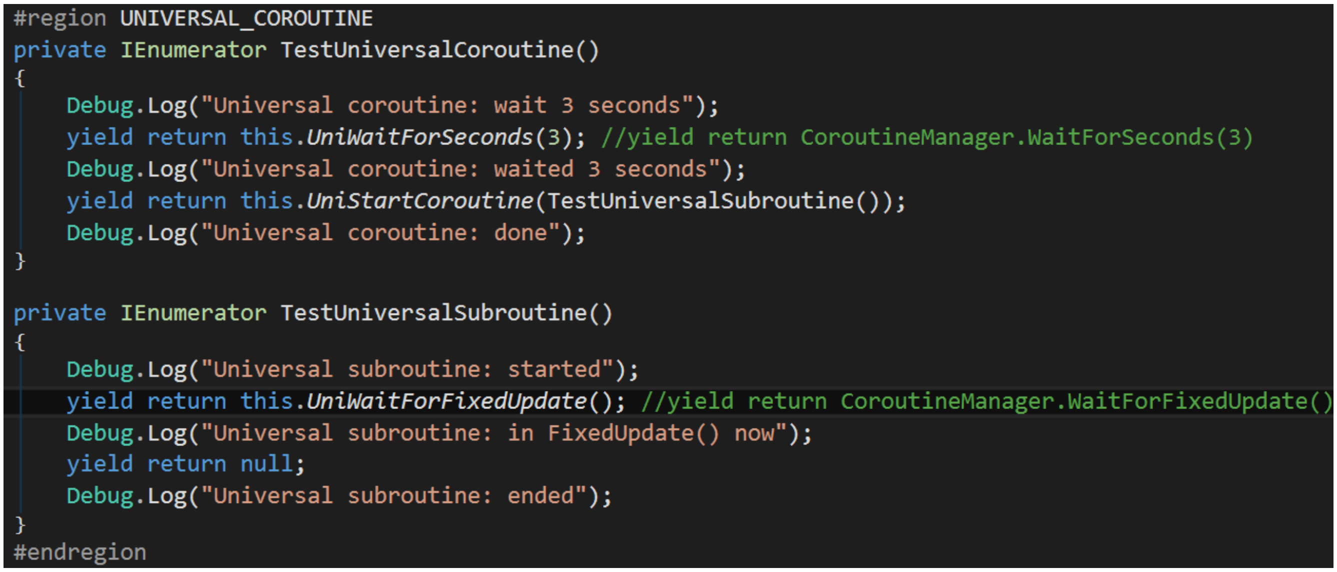
Example: From Unity Coroutine format to Universal Coroutine format (see CoroutineComparison.cs for full code):



```
#region UNITY
private IEnumerator TestUnityCoroutine()
{
    Debug.Log("Unity coroutine: wait 3 seconds");
    yield return new UnityEngine.WaitForSeconds(3);
    Debug.Log("Unity coroutine: waited 3 seconds");
    yield return StartCoroutine(TestUnitySubroutine());
    Debug.Log("Unity coroutine: done");
}

private IEnumerator TestUnitySubroutine()
{
    Debug.Log("Unity subroutine: started");
    yield return new WaitForFixedUpdate();
    Debug.Log("Unity subroutine: in FixedUpdate() now");
    yield return null;
    Debug.Log("Unity subroutine: ended");
}
#endregion
```

This becomes:



```
#region UNIVERSAL_COROUTINE
private IEnumerator TestUniversalCoroutine()
{
    Debug.Log("Universal coroutine: wait 3 seconds");
    yield return this.UniWaitForSeconds(3); //yield return CoroutineManager.WaitForSeconds(3)
    Debug.Log("Universal coroutine: waited 3 seconds");
    yield return this.UniStartCoroutine(TestUniversalSubroutine());
    Debug.Log("Universal coroutine: done");
}

private IEnumerator TestUniversalSubroutine()
{
    Debug.Log("Universal subroutine: started");
    yield return this.UniWaitForFixedUpdate(); //yield return CoroutineManager.WaitForFixedUpdate()
    Debug.Log("Universal subroutine: in FixedUpdate() now");
    yield return null;
    Debug.Log("Universal subroutine: ended");
}
#endregion
```

See the Test scene and CoroutineTester.cs file if you want to see a fully implemented class that uses all the functionality.