LazyCoroutines

Links

Unity Asset Store

JOIN OUR DISCORD

Buy me a coffee

Navigate

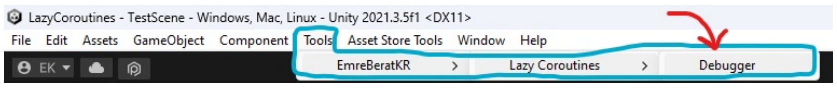
- About
- How to Install
- Debugger
- API

About

An Open-source Extension Library for Unity Coroutines

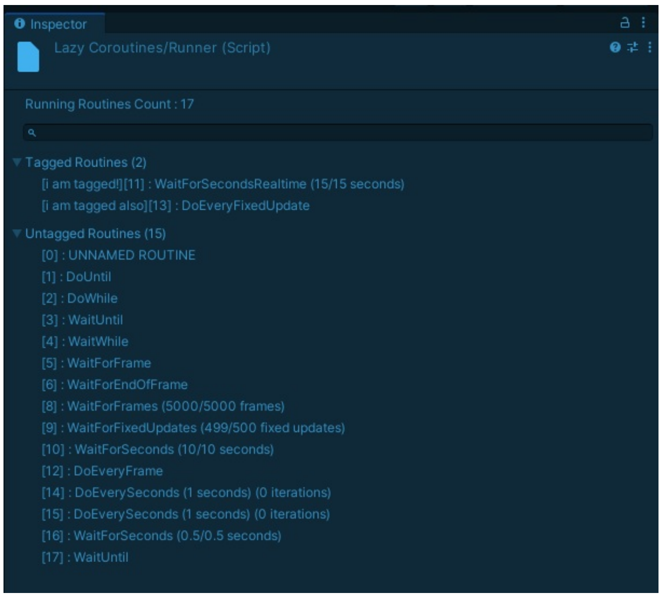
Debugger

How to Access to Debugger Panel



- From Unity Menu Item's `Tools`>`EmreBeratKR`>`Lazy Coroutines`>`Debugger`

Debugger Panel



```
Running Routines Count : 17

Tagged Routines (2)
    [i am tagged!][11] : WaitForSecondsRealtime (15/15 seconds)
    [i am tagged also][13] : DoEveryFixedUpdate
Untagged Routines (15)
    [0] : UNNAMED ROUTINE
    [1] : DoUntil
    [2] : DoWhile
    [3] : WaitUntil
    [4] : WaitWhile
    [5] : WaitForFrame
    [6] : WaitForEndOfFrame
    [8] : WaitForFrames (5000/5000 frames)
    [9] : WaitForFixedUpdates (499/500 fixed updates)
    [10] : WaitForSeconds (10/10 seconds)
    [12] : DoEveryFrame
    [14] : DoEverySeconds (1 seconds) (0 iterations)
    [15] : DoEverySeconds (1 seconds) (0 iterations)
    [16] : WaitForSeconds (0.5/0.5 seconds)
    [17] : WaitUntil
```

API

- StartCoroutine
- StopCoroutine
- StopAllCoroutines

Do Prefix

- DoEveryFrame
- DoEveryFixedUpdate
- DoEverySeconds
- DoEverySeconds (with Func)
- DoWhile
- DoUntil

Wait Prefix

- WaitForFrame
- WaitForFrames
- WaitForFixedUpdate
- WaitForFixedUpdates
- WaitForEndOfFrame
- WaitForSeconds
- WaitForSecondsRealtime
- WaitWhile
- WaitUntil

StartCoroutine

- Starts a new coroutine and associates it with a unique ID.
- Returns the started coroutine.

```
using System.Collections;
using UnityEngine;
using EmreBeratKR.LazyCoroutines;



public class Test : MonoBehaviour
{
    private void Start()
    {
        LazyCoroutines.StartCoroutine(Routine());


    <span class="hljs-function">IEnumerator <span class="hljs-title">Routine</span>(<span class="hljs-params"></span>)
    </span>{
        <span class="hljs-keyword">yield</span> <span class="hljs-keyword">return</span> <span class="hljs-literal">null</span>;
        Debug.Log(<span class="hljs-string">"some routine"</span>);
    }
}



}
```

StopCoroutine

- Stops the specified coroutine.

```
using System.Collections;
using UnityEngine;
using EmreBeratKR.LazyCoroutines;



public class Test : MonoBehaviour
{
    private void Start()
    {
        var coroutine = LazyCoroutines.StartCoroutine(Routine());


    <span class="hljs-function">IEnumerator <span class="hljs-title">Routine</span>(<span class="hljs-params"></span>)
    </span>{
        <span class="hljs-keyword">yield</span> <span class="hljs-keyword">return</span> <span class="hljs-literal">null</span>;
        Debug.Log(<span class="hljs-string">"some routine"</span>);
    }

    LazyCoroutines.StopCoroutine(coroutine);
}



}
```

StopAllCoroutines

- Stops all running coroutines.

```
using EmreBeratKR.LazyCoroutines;
using UnityEngine;



public class Test : MonoBehaviour
{
    private void Start()
    {
        LazyCoroutines.StopAllCoroutines();
    }
}
```

Do Prefix

DoEveryFrame

- Executes the specified action every frame.
- Returns the started coroutine.

```
using EmreBeratKR.LazyCoroutines;
using UnityEngine;



public class Test : MonoBehaviour
{
    private void Start()
    {
        LazyCoroutines.DoEveryFrame(() =>
        {
            Debug.Log("Log every frame!");
        });
    }
}
```

DoEveryFixedUpdate

- Executes the specified action every FixedUpdate.
- Returns the started coroutine.

```
using EmreBeratKR.LazyCoroutines;
using UnityEngine;



public class Test : MonoBehaviour
{
    private void Start()
    {
        LazyCoroutines.DoEveryFixedUpdate(() =>
        {
            Debug.Log("Log every FixedUpdate!");
        });
    }
}
```

DoEverySeconds

- Executes the specified action every specified number of seconds.
- Returns the started coroutine.

```
using EmreBeratKR.LazyCoroutines;
using UnityEngine;



public class Test : MonoBehaviour
{
    private void Start()
    {
        LazyCoroutines.DoEverySeconds(0.5f, () =>
        {
            Debug.Log("Log every 0.5 seconds!");
        });
    }
}
```

DoEverySeconds (with Func)

- Executes the specified action every specified number of seconds.
- Useful whenever the duration is changing.
- Returns the started coroutine.

```
using EmreBeratKR.LazyCoroutines;
using UnityEngine;



public class Test : MonoBehaviour
{
    private void Start()
    {
        var duration = 0.75f;



    LazyCoroutines.DoEverySeconds(() =&gt; duration, () =&gt;
    {
        <span class="hljs-comment">// change duration randomly</span>
        duration = Random.Range(<span class="hljs-number">0.1</span>f, <span class="hljs-number">1</span>f);
        Debug.Log(<span class="hljs-string">"Log every [duration] seconds!"</span>);
    });
}



}
```

DoWhile

- Executes the specified action while the specified condition is true.
- Returns the started coroutine.

```
using EmreBeratKR.LazyCoroutines;
using UnityEngine;



public class Test : MonoBehaviour
{
    private void Start()
    {
        LazyCoroutines.DoWhile(() => Input.GetKey(KeyCode.Space), () =>
        {
            Debug.Log("Log while space key is pressed!");
        });
    }
}
```

DoUntil

- Executes the specified action until the specified condition is true.
- Returns the started coroutine.

```
using EmreBeratKR.LazyCoroutines;
using UnityEngine;



public class Test : MonoBehaviour
{
    private void Start()
    {
        LazyCoroutines.DoUntil(() => Input.GetKeyDown(KeyCode.Space), () =>
        {
            Debug.Log("Log until space key is pressed!");
        });
    }
}
```

Wait Prefix

WaitForFrame

- Waits for the next frame and then invokes the provided action.
- Returns the started coroutine.

```
using EmreBeratKR.LazyCoroutines;
using UnityEngine;



public class Test : MonoBehaviour
{
    private void Start()
    {
        LazyCoroutines.WaitForFrame(() =>
        {
            Debug.Log("Waited for a frame!");
        });
    }
}
```

WaitForFrames

- Waits for a specified number of frames and then invokes the provided action.
- Returns the started coroutine.

```
using EmreBeratKR.LazyCoroutines;
using UnityEngine;


public class Test : MonoBehaviour
{
    private void Start()
    {
        LazyCoroutines.WaitForFrames(10, () =>
        {
            Debug.Log("Waited for 10 frames!");
        });
    }
}
```

WaitForFixedUpdate

- Waits for a FixedUpdate and then invokes the provided action.
- Returns the started coroutine.

```
using EmreBeratKR.LazyCoroutines;
using UnityEngine;


public class Test : MonoBehaviour
{
    private void Start()
    {
        LazyCoroutines.WaitForFixedUpdate(() =>
        {
            Debug.Log("Waited for a FixedUpdate!");
        });
    }
}
```

WaitForFixedUpdates

- Waits for a specified number of FixedUpdates and then invokes the provided action.
- Returns the started coroutine.

```
using EmreBeratKR.LazyCoroutines;
using UnityEngine;


public class Test : MonoBehaviour
{
    private void Start()
    {
        LazyCoroutines.WaitForFixedUpdates(5, () =>
        {
            Debug.Log("Waited for 5 FixedUpdates!");
        });
    }
}
```

WaitForEndOfFrame

- Waits until the end of the current frame and then invokes the provided action.
- Returns the started coroutine.

```
using EmreBeratKR.LazyCoroutines;
using UnityEngine;


public class Test : MonoBehaviour
{
    private void Start()
    {
        LazyCoroutines.WaitForEndOfFrame(() =>
        {
            Debug.Log("Waited for 5 end of the frame!");
        });
    }
}
```

WaitForSeconds

- Waits for a specified amount of time in seconds and then invokes the provided action.
- Returns the started coroutine.

```
using EmreBeratKR.LazyCoroutines;
using UnityEngine;


public class Test : MonoBehaviour
{
    private void Start()
    {
        LazyCoroutines.WaitForSeconds(3.67f, () =>
        {
            Debug.Log("Waited for 3.67 seconds");
        });
    }
}
```

WaitForSecondsRealtime

- Waits for a specified amount of real time in seconds and then invokes the provided action.
- Returns the started coroutine.

```
using EmreBeratKR.LazyCoroutines;
using UnityEngine;


public class Test : MonoBehaviour
{
    private void Start()
    {
        LazyCoroutines.WaitForSecondsRealtime(10, () =>
        {
            Debug.Log("Waited for 10 seconds");
        });
    }
}
```

WaitWhile

- Waits while a given condition is true and then invokes the provided action.
- Returns the started coroutine.

```
using EmreBeratKR.LazyCoroutines;
using UnityEngine;



public class Test : MonoBehaviour
{
    private bool m_IsLevelComplete;



<span class="hljs-function"><span class="hljs-keyword">private</span> <span class="hljs-keyword">void</span> <span class="hljs-title">Start</span>(<span class="hljs-params"></span>)
</span>{
    LazyCoroutines.WaitWhile(() =&gt; !m_IsLevelComplete, () =&gt;
    {
        Debug.Log(<span class="hljs-string">"Wait while level is not completed yet!"</span>);
    });
}



}
```

WaitUntil

- Waits until a given condition is true and then invokes the provided action.
- Returns the started coroutine.

```
using EmreBeratKR.LazyCoroutines;
using UnityEngine;



public class Test : MonoBehaviour
{
    private int m_CoinCount;



<span class="hljs-function"><span class="hljs-keyword">private</span> <span class="hljs-keyword">void</span> <span class="hljs-title">Start</span>(<span class="hljs-params"></span>)
</span>{
    LazyCoroutines.WaitUntil(() =&gt; m_CoinCount &gt; <span class="hljs-number">5</span>, () =&gt;
    {
        Debug.Log(<span class="hljs-string">"Wait until we have more than 5 coins!"</span>);
    });
}



}
```