

[首页](#) [资讯](#) [精华](#) [论坛](#) [问答](#) [博客](#) [专栏](#) [群组](#) [更多 ▼](#)

您还未登录！[登录](#) [注册](#)



One thing I know,that is I know nothing. (Socrates Greek)

- [博客](#)
- [微博](#)
- [相册](#)
- [收藏](#)
- [留言](#)
- [关于我](#)

Unity协程（Coroutine）原理深入剖析

博客分类：

- [Unity3D插件学习，工具分享](#)
- [Unity](#)
- [C#](#)

Unity协程（Coroutine）原理深入剖析

By D.S.Qiu

尊重他人的劳动，支持原创，转载请注明出处：<http://dsqiu.iteye.com>

关闭

记得去年6月份刚开始实习的时候，当时要我写网络层的结构，用到了协程，当时有点懵，完全不知道Unity协程的执行机制是怎么样的，只是知道函数的返回值是IEnumerator类型，函数中使用yield return，就可以通过StartCoroutine调用了。后来也是一直稀里糊涂地用，上网google些基本都是例子，很少能帮助深入理解Unity协程的原理的。

本文只是从Unity的角度去分析理解协程的内部运行原理，而不是从C#底层的语法实现来介绍（后续有需要再进行介绍），一共分为三部分：

线程（Thread）和协程（Coroutine）

Unity中协程的执行原理

IEnumerator & Coroutine

之前写过一篇《[Unity协程\(Coroutine\)管理类——TaskManager工具分享](#)》主要是介绍TaskManager实现对协程的状态控制，没有Unity后台实现的协程的原理进行深究。虽然之前自己对协程还算有点了解了，但是对Unity如何执行协程的还是一片空白，在UnityGems.com上看到两篇讲解Coroutine，如数家珍，当我看到Advanced Coroutine后面的Hijack类时，顿时觉得十分精巧，眼前一亮，遂动了写文分享之。

线程（Thread）和协程（Coroutine）

D.S.Qiu觉得使用协程的作用一共有两点：1）延时（等待）一段时间执行代码；2）等某个操作完成之后再执行后面的代码。总结起来就是一句话：控制代码在特定的时机执行。

很多初学者，都会下意识地觉得协程是异步执行的，都会觉得协程是C# 线程的替代品，是Unity不使用线程的解决方案。

所以首先，请你牢记：协程不是线程，也不是异步执行的。协程和 MonoBehaviour 的 Update 函数一样也是在MainThread中执行的。使用协程你不用考虑同步和锁的问题。

Unity中协程的执行原理

UnityGems.com给出了协程的定义：

A coroutine is a function that is executed partially and, presuming suitable conditions are met, will be resumed at some point in the future until its work is done.

即协程是一个分部执行，遇到条件（yield return 语句）会挂起，直到条件满足才会被唤醒继续执行后面的代码。

Unity在每一帧（Frame）都会去处理对象上的协程。Unity主要是在Update后去处理协程（检查协程的条件是否满足），但也有写特例：



关闭



从上图的剖析就明白，协程跟Update()其实一样的，都是Unity每帧都会去处理的函数（如果有的话）。如果MonoBehaviour 是处于激活（active）状态的而且yield的条件满足，就会协程方法的后面代码。还可以发现：如果在一个对象的前期调用协程，协程会立即运行到第一个 yield return 语句处，如果是 yield return null，就会在同一帧再次被唤醒。如果没有考虑这个细节就会出现一些奇怪的问题『1』。

『1』注 图和结论都是从UnityGems.com 上得来的，经过下面的验证发现与实际不符，D.S.Qiu用的是Unity 4.3.4f1 进行测试的。经过测试验证，协程至少是每帧的LateUpdate()后去运行。

下面使用 yield return new WaitForSeconds(1f); 在Start, Update 和 LateUpdate 中分别进行测试：

C#代码  

```

1. using UnityEngine;
2. using System.Collections;
3.
4. public class TestCoroutine : MonoBehaviour {
5.     private bool isStartCall = false; //Makesure Update() and LateUpdate() Log only once
6.     private bool isUpdateCall = false;
7.     private bool isLateUpdateCall = false;
8.     private bool isStartCall = false;
9.     private bool isUpdateCall = false;
10.    private bool isLateUpdateCall = false;
11.    private void Start()
12.    {
13.        StartCall Begin";
14.        StartCoroutine(StartCoutine());
15.        Debug.Log("Start Call End");
16.        isStartCall = true;
17.    }
18.
19. }
20. IEnumerator StartCoutine()
21. {
22.
23.     Debug.Log("This is Start Coroutine Call Before");
24.     yield return new WaitForSeconds(1f);
25.     Debug.Log("This is Start Coroutine Call After");
26.
27. }
28. // Update is called once per frame
29. void Update () {
30.     if (!isUpdateCall)
31.     {
32.         Debug.Log("Update Call Begin");
33.         StartCoroutine(UpdateCoutine());
34.         Debug.Log("Update Call End");
35.         isUpdateCall = true;
36.     }
37. }
38. IEnumerator UpdateCoutine()
39. {
40.     Debug.Log("This is Update Coroutine Call Before");
41.     yield return new WaitForSeconds(1f);
42.     Debug.Log("This is Update Coroutine Call After");
43. }
44. void LateUpdate()
45. {
46.     if (!isLateUpdateCall)

```

广告

关闭

```

47.     {
48.         Debug.Log("LateUpdate Call Begin");
49.         StartCoroutine(LateCoutine());
50.         Debug.Log("LateUpdate Call End");
51.         isLateUpdateCall = true;
52.     }
53. }
54. IEnumerator LateCoutine()
55. {

```

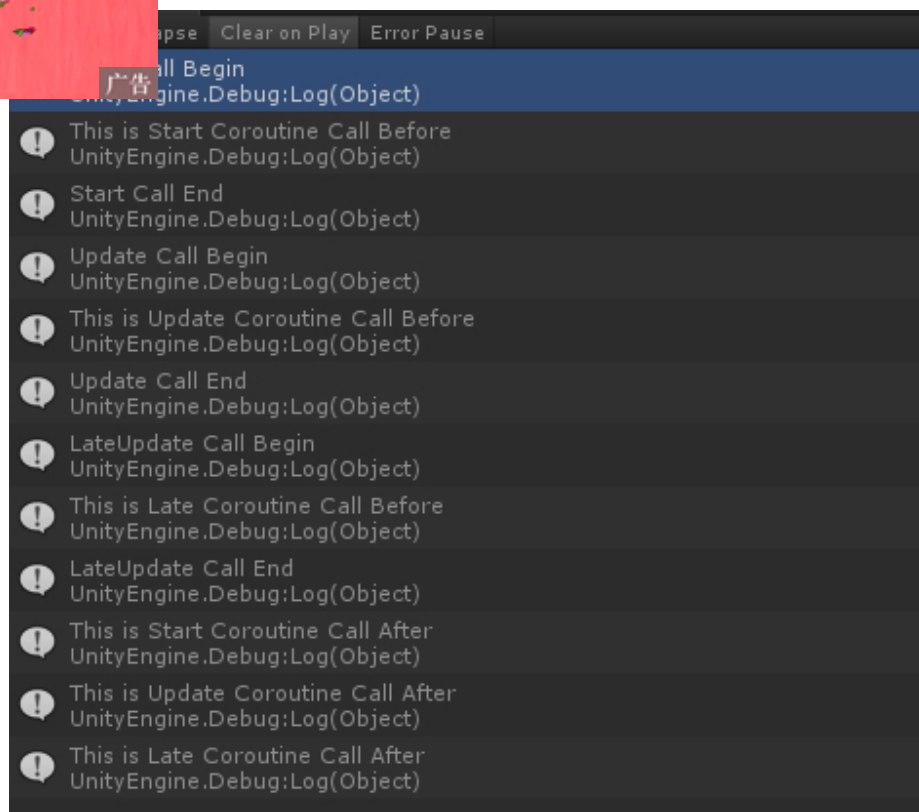
官网自助下单
免费得239元包鼠!



```

        Debug.Log("This is Late Coroutine Call Before");
        yield return WaitForSeconds(1f);
        Debug.Log("This is Late Coroutine Call After");
    }
}

```



关闭

然后将yield return new WaitForSeconds(1f);改为 yield return null; 发现日志输入结果和上面是一样的，没有出现上面说的情况：

C#代码

```

1. using UnityEngine;
2. using System.Collections;
3.
4. public class TestCoroutine : MonoBehaviour {
5.
6.     private bool isStartCall = false; //Makesure Update() and LateUpdate() Log only once
7.     private bool isUpdateCall = false;

```

```

8. private bool isLateUpdateCall = false;
9. // Use this for initialization
10. void Start () {
11.     if (!isStartCall)
12.     {
13.         Debug.Log("Start Call Begin");
14.         StartCoroutine(StartCoutine());
15.         Debug.Log("Start Call End");
16.         isStartCall = true;
17.     }
18. }
19. IEnumerator StartCoutine()
20. {
21.     Debug.Log("This is Start Coroutine Call Before");
22.     yield return null;
23.     Debug.Log("This is Start Coroutine Call After");
24. }
25.
26.
27. }
28. // Update is called once per frame
29. void Update () {
30.     if (!isUpdateCall)
31.     {
32.         Debug.Log("Update Call Begin");
33.         StartCoroutine(UpdateCoutine());
34.         Debug.Log("Update Call End");
35.         isUpdateCall = true;
36.     }
37. }
38. IEnumerator UpdateCoutine()
39. {
40.     Debug.Log("This is Update Coroutine Call Before");
41.     yield return null;
42.     Debug.Log("This is Update Coroutine Call After");
43. }
44. void LateUpdate()
45. {
46.     if (!isLateUpdateCall)
47.     {
48.         Debug.Log("LateUpdate Call Begin");
49.         StartCoroutine(LateCoutine());
50.         Debug.Log("LateUpdate Call End");
51.         isLateUpdateCall = true;
52.     }
53. }
54. IEnumerator LateCoutine()
55. {
56.     Debug.Log("This is Late Coroutine Call Before");
57.     yield return null;
58.     Debug.Log("This is Late Coroutine Call After");

```

官网自助下单
免费得239元包鼠!



广告

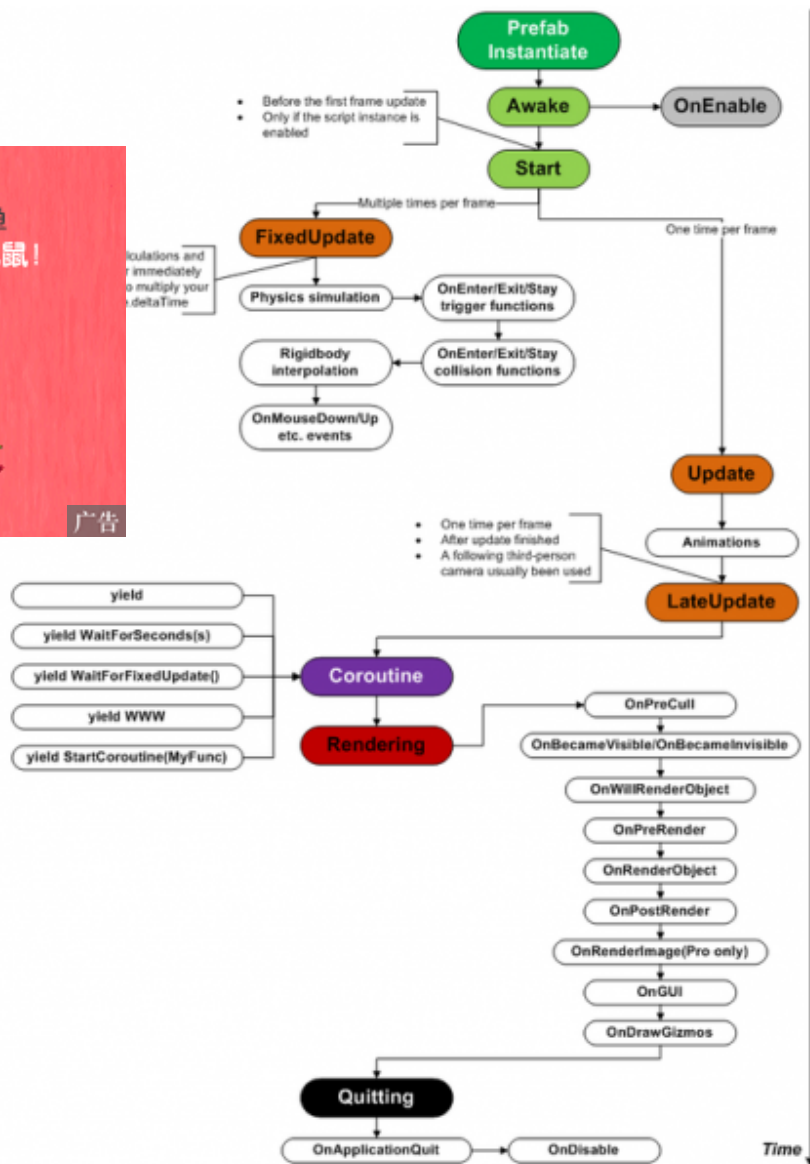
关闭

```

59. }
60. }

```



『今天意外发现MonoBehaviour的函数执行顺序图，发现协程的运行确实是在LateUpdate之后，下面附上：』



增补于：03/12/2014 22:14

前面在介绍[TaskManager工具](#)时，说到MonoBehaviour 没有针对特定的协程提供Stop方法，其实不然，可以通过MonoBehaviour.enabled = false 或者 gameObject.SetActive(false) 来关闭协程的执行『2』。

经过验证，『2』的结论也是错误的，正确的结论是，MonoBehaviour.enabled = false 协程会照常运行，但 gameObject.SetActive(false) 后协程却全部停止，即使在Inspector把 gameObject 激活还是没有继续执行：

C#代码  

1. using UnityEngine;
2. using System.Collections;
- 3.


```

4. public class TestCoroutine : MonoBehaviour {
5.
6.     private bool isStartCall = false; //Makesure Update() and LateUpdate() Log only once
7.     private bool isUpdateCall = false;
8.     private bool isLateUpdateCall = false;
9.     // Use this for initialization
10.    void Start () {
11.        if (!isStartCall)
12.        {
13.            Debug.Log("Start Call Begin");
14.            StartCoroutine(StartCoutine());
15.            Debug.Log("Start Call End");
16.            isStartCall = true;
17.        }
18.    }
19.
20.    void Update()
21.    {
22.
23.        Debug.Log("This is Start Coroutine Call Before");
24.        yield return new WaitForSeconds(1f);
25.        Debug.Log("This is Start Coroutine Call After");
26.
27.    }
28.    // Update is called once per frame
29.    void Update () {
30.        if (!isUpdateCall)
31.        {
32.            Debug.Log("Update Call Begin");
33.            StartCoroutine(UpdateCoutine());
34.            Debug.Log("Update Call End");
35.            isUpdateCall = true;
36.            this.enabled = false;
37.            //this.gameObject.SetActive(false);
38.        }
39.    }
40.    IEnumerator UpdateCoutine()
41.    {
42.        Debug.Log("This is Update Coroutine Call Bef");
43.        yield return new WaitForSeconds(1f);
44.        Debug.Log("This is Update Coroutine Call After");
45.        yield return new WaitForSeconds(1f);
46.        Debug.Log("This is Update Coroutine Call Second");
47.    }
48.    void LateUpdate()
49.    {
50.        if (!isLateUpdateCall)
51.        {
52.            Debug.Log("LateUpdate Call Begin");
53.            StartCoroutine(LateCoutine());
54.            Debug.Log("LateUpdate Call End");

```



关闭

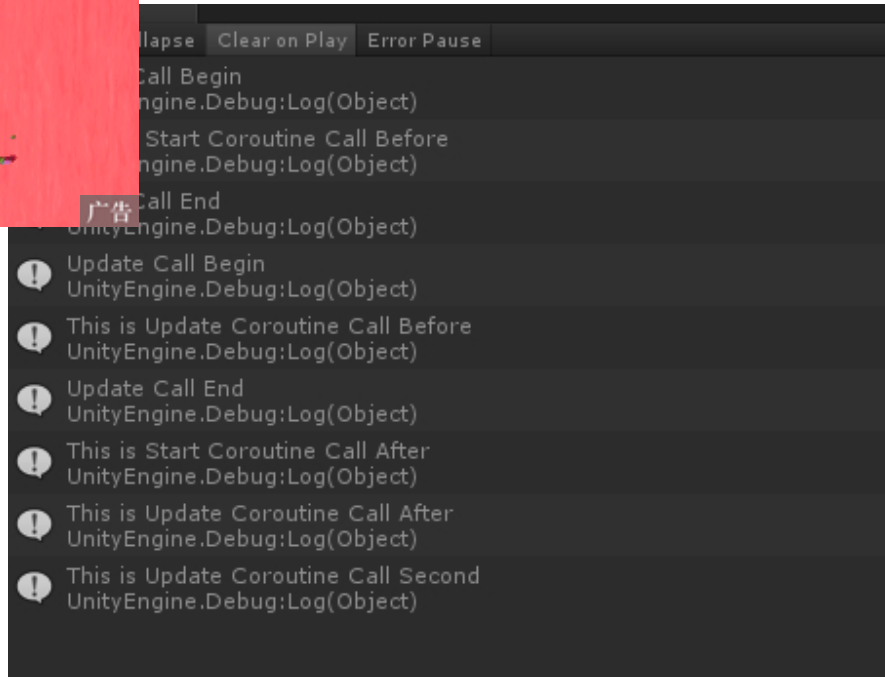
```

55.         isLateUpdateCall = true;
56.
57.     }
58. }
59. IEnumerator LateCoutine()
60. {
61.     Debug.Log("This is Late Coroutine Call Before");
62.     yield return null;
63.     Debug.Log("This is Late Coroutine Call After");

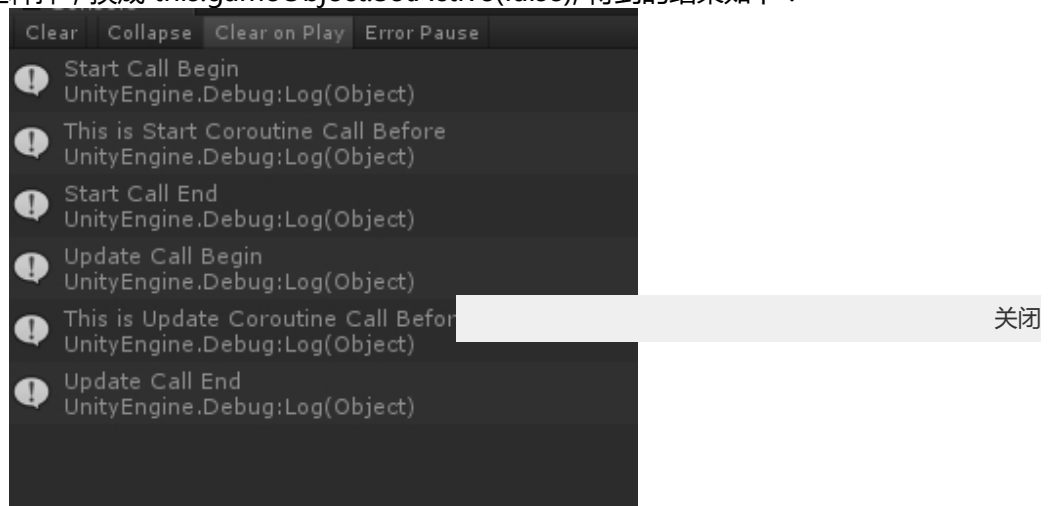
```



nd = false; 得到的结果：



然后把 this.enabled = false; 注释掉，换成 this.gameObject.SetActive(false); 得到的结果如下：



整理得到：通过设置MonoBehaviour脚本的enabled对协程是没有影响的，但如果gameObject.SetActive(false) 则已经启动的协程则完全停止了，即使在Inspector把gameObject 激活还是没有继续执行。也就说协程虽然是在MonoBehaviour启动的（StartCoroutine）但是协程函数的地位完全是跟MonoBehaviour是一个层次的，不受MonoBehaviour的状态影响，但跟MonoBehaviour脚本一样受gameObject控制，也应该是和MonoBehaviour脚本一样每帧“轮询” yield 的条件是否满足。

yield 后面可以有的表达式：

- a) null - the coroutine executes the next time that it is eligible
- b) WaitForEndOfFrame - the coroutine executes on the frame, after all of the rendering and GUI is complete
- c) WaitForFixedUpdate - causes this coroutine to execute at the next physics step, after all physics is calculated



...es the coroutine not to execute for a given game time period

...o request to complete (resumes as if WaitForSeconds or null)

...which case the new coroutine will run to completion before the yielder is

...)受Time.timeScale影响，当Time.timeScale = 0f 时，yield return new

IEnumerator & Coroutine

协程其实就是一个IEnumerator（迭代器），IEnumerator 接口有两个方法 Current 和 MoveNext()，前面介绍的 [TaskManager](#) 就是利用者两个方法对协程进行了管理，只有当MoveNext() 返回 true时才可以访问 Current，否则会报错。迭代器方法运行到 yield return 语句时，会返回一个 expression表达式并保留当前在代码中的位置。当下次调用迭代器函数时执行从该位置重新启动。

Unity在每帧做的工作就是：调用 协程（迭代器）MoveNext() 方法，如果返回 true，就从当前位置继续往下执行。

Hijack

这里在介绍一个协程的交叉调用类 Hijack（参见附件）：

C#代码 ☆

```

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using UnityEngine;
5. using System.Collections;
6.
7. [RequireComponent(typeof(GUIText))]
8. public class Hijack : MonoBehaviour {
9.
10.     //This will hold the counting up coroutine
11.     IEnumerator _countUp;
12.     //This will hold the counting down coroutine
13.     IEnumerator _countDown;
14.     //This is the coroutine we are currently
15.     //hijacking

```

关闭

```
16. IEnumerator _current;
17.
18. //A value that will be updated by the coroutine
19. //that is currently running
20. int value = 0;
21.
22. void Start()
23. {
24.     //Create our count up coroutine
25.     countUp();
26.     //Create our count down coroutine
27.     countDown();
28.     //Create our coroutine for the hijack
29.     DoHijack();
30.
31.     //Show the current value on the screen
32.     guiText.text = value.ToString();
33. }
34.
35. void OnGUI()
36. {
37.     //Switch between the different functions
38.     if(GUILayout.Button("Switch functions"))
39.     {
40.         if(_current == _countUp)
41.             _current = _countDown;
42.         else
43.             _current = _countUp;
44.     }
45. }
46.
47. IEnumerator DoHijack()
48. {
49.     while(true)
50.     {
51.         //Check if we have a current coroutine and
52.         if(_current != null && _current.MoveNext())
53.         {
54.             //Return whatever the coroutine yielded, so we will yield the
55.             //same thing
56.             yield return _current.Current;
57.         }
58.         else
59.             //Otherwise wait for the next frame
60.             yield return null;
61.     }
62. }
63.
64. }
```

官网自助下单
免费得239元包鼠!



广告

关闭

```

67. IEnumerator CountUp()
68. {
69.     //We have a local increment so the routines
70.     //get independently faster depending on how
71.     //long they have been active
72.     float increment = 0;
73.     while(true)
74.     {
75.         //Exit if the Q button is pressed
//Input.GetKeyDown(KeyCode.Q))
//Time.deltaTime;
//Mathf.RoundToInt(increment);
//null;
//tDown()
85. {
86.     float increment = 0f;
87.     while(true)
88.     {
89.         if(Input.GetKey(KeyCode.Q))
90.             break;
91.         increment += Time.deltaTime;
92.         value -= Mathf.RoundToInt(increment);
93.         //This coroutine returns a yield instruction
94.         yield return new WaitForSeconds(0.1f);
95.     }
96. }
97.
98. }

```

上面的代码实现是两个协程交替调用，对有这种需求来说实在太精妙了。

小结：

关闭

今天仔细看了下UnityGems.com 有关Coroutine的两篇文章，虽然第一篇（参考①）现在验证的结果有很多错误，但对于理解协程还是不错的，尤其是当我发现Hijack这个脚本时，就迫不及待分享给大家。

本来没觉得会有UnityGems.com上的文章会有错误的，无意测试了发现还是有很大的出入，当然这也不是说原来作者没有经过验证就妄加揣测，D.S.Qiu觉得很有可能是Unity内部的实现机制改变了，这种东西完全可以改动，Unity虽然开发了很多年了，但是其实在实际开发中还是有很多坑，越发觉得Unity的无力，虽说容易上手，但是填坑的功夫也是必不可少的。

看来很多结论还是要通过自己的验证才行，贸然复制粘贴很难出真知，切记！

如果您对D.S.Qiu有任何建议或意见可以在文章后面评论，或者发邮件（gd.s.qiu@gmail.com）交流，您的鼓励和支持是我前进的动力，希望能有更多更好的分享。

转载请在文首注明出处：<http://dsqiu.iteye.com/blog/2029701>

更多精彩请关注D.S.Qiu的博客和微博（ID：静水逐风）



tygems.com/coroutines/



tygems.com/advanced-coroutines/

sina.com.cn/s/blog_5b6cb9500100xgmp.html

广告

- [coroutines.zip](#) (37.3 KB)
- 下载次数: 29
- [查看图片附件](#)

2
顶
2
踩

分享到： 

[Unity 动画（UITweener）、协程（Coroutin ... | Unity3D Shader编程实践——“Hello Shader ...](#)

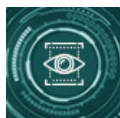
- 2014-03-12 00:02
- 浏览 61368
- [评论\(12\)](#)
- 分类: [移动开发](#)
- [相关推荐](#)

关闭

参考知识库



[语音识别与合成知识库](#) 291 关注 / 256 收录



[计算机视觉知识库](#) 775 关注 / 203 收录



[自然语言理解和处理知识库](#) 294 关注 / 87 收录



[知识工程知识库](#) 89 关注 / 69 收录



5-09

不是线程，也不是异步执行的。协程和 MonoBehaviour 的 Update函数
执行的。使用协程你不用考虑同步和锁的问题。++
企图用携程替代update后台跑，结果failed了

5

继续执行呢？yield return null是下一帧再接着执行，那 yield return new
就是同一帧执行呢，那yield return new WaitForFixedUpdate()是下一

帧？

10 楼 [fgfd0](#) 2015-05-08

fgfd0 写道

自己又分析了一下，

你这里上面这句话是错的：

“经过测试验证，协程至少是每帧的LateUpdate()后去运行。”

我这边的验证结果是：

Update

yield null

yield WaitForSeconds

yield WWW

yield StartCoroutine

LateUpdate

协程Update之后执行，在LateUpdate之前执行完成，当然还有其它的协程方法，是在FixedUpdate里
和WaitForEndOfFrame.

我用的是4.5.1版本的Unity

9 楼 [fgfd0](#) 2015-05-08

自己又分析了一下，

你这里上面这句话是错的：

“经过测试验证，协程至少是每帧的LateUpdate()后去运行。”

我这边的验证结果是：

Update

yield null

yield WaitForSeconds

yield WWW

关闭

yield StartCoroutine
LateUpdate

协程Update之后执行，在LateUpdate之前执行完成，当然还有其它的协程方法，是在FixedUpdate里和WaitForEndOfFrame.

8 楼 [fuyunzzy](#) 2015-02-27

第一张图不显示了，能重新上传一下吗

7 楼 [DSQiu](#) 2014-11-20

silverys 写道



网上很多地方都说是多线程什么的，真是误人子弟。。

在update中要延迟执行一些代码，或者满足一定条件后执行一些代码。需要在当前时间来减去前面记录的时间来判断执行。当这种情况越来越多的时候，代码就越来越乱。。

这时候一般会抽象一个框架出来处理这个问题。

这个框架，为解决这个问题而设计的一个设计模式。。套着这个概念来写，让和C#的多线程是两码事。。

6 楼 [silverys](#) 2014-11-19

其实协程并没有那么复杂，网上很多地方都说是多线程什么的，真是误人子弟。。

在传统实时游戏中，在update中要延迟执行一些代码，或者满足一定条件后执行一些代码。需要在update添加一个计时器，用当前时间来减去前面记录的时间来判断执行。当这种情况越来越多的时候，会添加很多变量和代码，代码就越来越乱。。

实时游戏写多了就知道。。这时候一般会抽象一个框架出来处理这个问题。

而协程可以说是unity的一个框架，为解决这个问题而设计的一个设计模式。。套着这个概念来写，让这种代码非常整洁易维护。。和C#的多线程是两码事。。

5 楼 [奥东here](#) 2014-10-30

好吧，我找到了，文章下面有引用，还有就是文章中的Start()方法本来就是一次的哦，

```
void Start ()
```

```
{
if (!isStartCall)
{
```

4 楼 [奥东here](#) 2014-10-30

Monobehaviour的函数执行顺序图,楼主在哪里得到的？

3 楼 [DSQiu](#) 2014-09-26

mhj 写道

请问楼主 协程是否能够等待另外一个函数执行完毕后 执行后面的代码

能

关闭

2 楼 [mhj](#) 2014-09-25

请问楼主 协程是否能够等待另外一个函数执行完毕后 执行后面的代码

1 楼 [dusthand](#) 2014-07-19

hi,关于文章里面说的协程的执行时刻，我用你的代码做了测试，同时加了两行输出

最后输出是这样的，按照这个，上面第一张图片的说法应该是对的，我是在4.6.0b7下做的测试，会不会是4.6.0机制又改了呢

```
UPDATE>>>>>
```

```
UnityEngine.Debug.Log(Object)
```

```
This is Start Coroutine Call After
```

```
UnityEngine.Debug.Log(Object)
```


This is Update Coroutine Call After
UnityEngine.Debug:Log(Object)

This is Late Coroutine Call After
UnityEngine.Debug:Log(Object)

LATEUPDATE<<<<<<<
UnityEngine.Debug:Log(Object)



[您还没有登录,请您登录后再发表评论](#)

DSQiu

- 浏览: 816176 次
- 性别:
- 来自: 广州
- 我现在离线

最近访客 [更多访客>>](#)



[edisongz123](#)



[beidou566](#)



[jizhonglee](#)



[天涯还是亡徒](#)

博客专栏



[NGUI所见即所得](#)
浏览量：142963

关闭



[Effective C# ...](#)

浏览量：0

文章分类

- [全部博客 \(106\)](#)



- [高级数据结构 \(8\)](#)
- [数组和字符串问题 \(7\)](#)
- [C/C++学习 \(12\)](#)
- [C语言名题精选百则 \(4\)](#)
- [Java语言学习一篇足以 \(1\)](#)
- [工作进度 \(7\)](#)
- [Unity \(20\)](#)
- [NGUI \(13\)](#)
- [C# \(4\)](#)
- [Unity3D插件学习，工具分享 \(9\)](#)
- [Unity3D Shader & Effect 编程实践 \(1\)](#)
- [读书笔记 \(1\)](#)

社区版块

- [我的资讯 \(0\)](#)
- [我的论坛 \(0\)](#)
- [我的问答 \(0\)](#)

存档分类

- [2016-11 \(1\)](#)
- [2015-12 \(1\)](#)
- [2014-06 \(1\)](#)
- [更多存档...](#)

关闭

最新评论

- [sdgxxtc](#)：[quo[color=red]te][color]C#使用OleDb读取Excel，生成SQL语句
- [zcs302567601](#)：博主，你好，一直都有个问题没有搞明白，就是 2.x的版本是通过 ...
[NGUI所见即所得之UIPanel](#)

- [一样的追寻](#)：感谢楼主！
[有向强连通和网络流大讲堂——史无前例求解最大流（最小割）、最小费用最大流](#)
- [cp1993518](#)：感谢！从你的博客里学到了很多
[Unity日志工具——封装，跳转](#)
- [cp1993518](#)：学习了~，话说现在的版本custom还真的变委托了
[NGUI所见即所得之UGrid & Uitable](#)

声明：ITeye文章版权属于作者，受法律保护。没有作者书面许可不得转载。若作者同意转载，必须以超链接形式标明文章原始出处和作者。

© 2003-2017 ITeye.com. All rights reserved. [京ICP证110151号 京公网安备110105010620]



关闭