Unity3D协程介绍 以及 使用

转 2014年08月11日 15:09:05 huang9012 阅读数：93842 更多

作者ChevyRay，2013年9月28日，snaker7译 原文地址：http://unitypatterns.com/introduction-to-coroutines/
在Unity中，协程（Coroutines）的形式是我最喜欢的功能之一，几乎在所有的项目中，我都会使用它来控制运动，序列，以及对象的行为。在这个教程中，我将会说明协程是如何工作的，并且会附上一些例子来介绍它的用法。

协程介绍

Unity的协程系统是基于C#的一个简单而强大的接口，IEnumerator，它允许你为自己的集合类型编写枚举器。这一点你不必关注太多，我们直接进入一个简单的例子来看看协程到底能干什么。首先，我们来看一下这段简单的代码...

倒计时器

这是一个简单的脚本组件，只做了倒计时，并且在到达0的时候log一个信息。

```
using Unity Engine;
using System.Collections;

public class Countdown : MonoBehaviour
{
public float timer = 3;

void Update()
{
timer -= Time.deltaTime;
if(timer <= 0)
Debug.Log("Timer has finished!");
}
}
```

还不错，代码简短实用，但问题是，如果我们需要复杂的脚本组件（像一个角色或者敌人的类），拥有多个计时器呢？刚开始的时候，我们的代码也许会是这样的：

```
using UnityEngine;
using System.Collections;

public class MultiTimer : MonoBehaviour
{
public float firstTimer = 3;
public float secondTimer = 2;
public float thirdTimer = 1;

void Update()
{
firstTimer -= Time.deltaTime;
if(firstTimer <= 0)
Debug.Log("First timer has finished!");

secondTimer -= Time.deltaTime;
if(secondTimer <= 0)
Debug.Log("Second timer has finished!");

thirdTimer -= Time.deltaTime;
if(thirdTimer <= 0)
Debug.Log("Third timer has finished!");
}
}
```

尽管不是太糟糕，但是我个人不是很喜欢自己的代码中充斥着这些计时器变量，它们看上去很乱，而且当我需要重新开始计时的时候还得记得去重置它们（这活我经常忘记做）。

如果我只用一个for循环来做这些，看上去是否会好很多？

```
for(float timer = 3; timer >= 0; timer -= Time.deltaTime)
{
//Just do nothing...
}
Debug.Log("This happens after 5 seconds!");
```

现在每一个计时器变量都成为for循环的一部分了，这看上去好多了，而且我不需要去单独设置每一个跌倒变量。

好的，你可能现在明白我的意思：协程可以做的正是这一点！

码入你的协程！

现在，这里提供了上面例子运用协程的版本！我建议你从这里开始跟着我来写一个简单的脚本组件，这样你可以在你自己的程序中看到它是如何工作的。

```
using UnityEngine;
using System.Collections;

public class CoroutineCountdown : MonoBehaviour
{
void Start()
{
StartCoroutine(Countdown());
}

IEnumerator Countdown()
{
for(floattimer = 3; timer >= 0; timer -= Time.deltaTime)
Yield return 0;

Debug.Log("This message appears after 3 seconds!");
}
}
```

这看上去有点不一样，没关系，接下来我会解释这里到底发生了什么。

```
StartCoroutine(Countdown());
```

这一行用来开始我们的Countdown程序，注意，我并没有给它传入参数，但是这个方法调用了它自己（这是通过传递Countdown的return返回值来实现的）。

Yield

在Countdown方法中其他的都很好理解，除了两个部分：
l IEnumerator 的返回值
l For循环中的yield return

为了能在连续的多帧中（在这个例子中，3秒钟等同于很多帧）调用该方法，Unity必须通过某种方式来存储这个方法的状态，这是通过IEnumerator 中使用yield return语句得到的返回值，当你"yield"一个方法时，你相当于说了，"现在停止这个方法，然后在下一帧中从这里重新开始！"。

注意：用0或者null来yield的意思是告诉协程等待下一帧，直到继续执行为止。当然，同样的你可以继续yield其他协程，我会在下一个教程中讲到这些。

一些例子

协程在刚开始接触的时候是非常难以理解的，无论是新手还是经验丰富的程序员我都见过他们对于协程语句一筹莫展的时候。因此我认为通过例子来理解它是最好的方法，这里有一些简单的协程例子：

多次输出"Hello"

记住，yield return是"停止执行方法，并且在下一帧从这里重新开始"，这意味着你可以这样做：

```
//This will say hello 5 times, once each frame for 5 frames
```

```
IEnumerator SayHelloFiveTimes()
{
Yield return 0;
Debug.Log("Hello");
Yield return 0;
Debug.Log("Hello");
Yield return 0;
Debug.Log("Hello");
Yield return 0;
Debug.Log("Hello");
Yield return 0;
Debug.Log("Hello");
}

//This will do the exact same thing as the above function!
IEnumerator SayHello5Times()
{
for(inti = 0; i < 5; i++)
{
Debug.Log("Hello");
Yield return 0;
}
}
```


每一帧输出“Hello”，无限循环。。。
通过在一个while循环中使用yield，你可以得到一个无限循环的协程，这几乎就跟一个Update（）循环等同。。。

```
//Once started, this will run until manually stopped or the object is destroyed
IEnumerator SayHelloEveryFrame()
{
while(true)
{
//1. Say hello
Debug.Log("Hello");

//2. Wait until next frame
Yield return 0;

}//3. This is a forever-loop, goto 1
}
```

计时
...不过跟Update（）不一样的是，你可以在协程中做一些更有趣的事：

```
IEnumerator CountSeconds()
{
int seconds = 0;

while(true)
{
for(float timer = 0; timer < 1; timer += Time.deltaTime)
Yield return 0;

seconds++;
Debug.Log(seconds +" seconds have passed since the Coroutine started.");
}
}
```

这个方法突出了协程一个非常酷的地方：方法的状态被存储了，这使得方法中定义的这些变量都会保存它们的值，即使是在不同的帧中。还记得这个教程开始时那些烦人的计时器变量吗？通过协程，我们再也不需要担心它们了，只需要把变量直接放到方法里面！

开始和终止协程

之前，我们已经学过了通过 StartCoroutine()方法来开始一个协程，就像这样：

```
StartCoroutine(Countdown());
```

如果我们想要终止所有的协程，可以通过StopAllCoroutines()方法来实现，它的所要做的就跟它的名字所表达的一样。注意，这只会终止在调用该方法的对象中（应该是指调用这个方法的类吧）开始的协程，对于其他的MonoBehavior类中运行的协程不起作用。

如果我们有以下这样两条协程语句：

```
StartCoroutine(FirstTimer());
StartCoroutine(SecondTimer());
```

。。。那我们怎么终止其中的一个协程呢？在这个例子里，这是不可能的，如果你想要终止某一个特定的协程，那么你必须得在开始协程的时候将它的方法名作为字符串，就像这样：

```
//If you start a Coroutine by name...
StartCoroutine("FirstTimer");
StartCoroutine("SecondTimer");

//You can stop it anytime by name!
StopCoroutine("FirstTimer");
```

更多关于协程的学习

即将为你带来：“Scripting with Coroutines”，一个更深入的介绍，关于如何使用协程以及如何通过协程编写对象行为。

扩展链接
l Coroutines – Unity Script Reference
如果你知道其他很棒的关于协程的Unity教程，或者相关的主题，请在回复中分享链接！当然，如果在教程有什么问题，比如链接无效或者其他一些问题，欢迎给我发邮件。

作者ChevyRay， 2013年9月28日，snaker7译 原文地址：http://unitypatterns.com/scripting-with-coroutines/

请注意：这个关于协程的教程共有两部分，这是第二部分，如果您未曾看过第一部分——协程介绍，那么在阅读这部分内容之前建议您先了解一下。

计时器例子

在第一个教程中，我们已经了解了协程如何让一个方法“暂停”下来，并且让它yield直到某些值到达我们给定的数值；并且利用它，我们还创建了一个很棒的计时器系统。协程一个很重要的内容是，它可以让普通的程序（比方说一个计时器）很容易地被抽象化并且被复用。

协程的参数

抽象化一个协程的第一个方法是给它传递参数，协程作为一个函数方法来说，它自然能够传递参数。这里有一个协程的例子，它在特定的地方输出了特定的信息。

```
[csharp]
Using UnityEngine;
Using System.Collections;

Public class TimerExample : MonoBehaviour
{
    Void Start()
    {
        //Log "Hello!" 5 times with 1 second between each log
        StartCoroutine(RepeatMessage(5, 1.0f,"Hello!"));
    }

    IEnumerator RepeatMessage(int count,float frequency,string message)
    {
        for(int i = 0; i < count; i++)
        {
            Debug.Log(message);
            for(float timer = 0; timer < frequency; timer += Time.deltaTime)
                Yield return 0;

        }
    }
}
```

嵌套的协程

在此之前，我们yield的时候总是用0（或者null），仅仅告诉程序在继续执行前等待下一帧。协程最强大的一个功能就是它们可以通过使用yield语句来相互嵌套。

眼见为实，我们先来创建一个简单的Wait()程序，不需要它做任何事，只需要在运行的时候等待一段时间就结束。

```csharp
[csharp]
IEnumerator Wait(float duration)
{
    for(float timer = 0; timer < duration; timer += Time.deltaTime)
        Yield return 0;
}
```

接下来我们要编写另一个协程，如下：

```csharp
[csharp]
Using UnityEngine;
Using System.Collections;

Public class TimerExample : MonoBehaviour
{
    voidStart()
    {
        StartCoroutine(SaySomeThings());
    }

    //Say some messages separated by time
    IEnumerator SaySomeThings()
    {
        Debug.Log("The routine has started");
        Yield return StartCoroutine(Wait(1.0f));
        Debug.Log("1 second has passed since the last message");
        Yield return StartCoroutine(Wait(2.5f));
        Debug.Log("2.5 seconds have passed since the last message");
    }

    //Our wait function
    IEnumerator Wait(float duration)
    {
        for(float timer = 0; timer < duration; timer += Time.deltaTime)
            Yield return 0;
    }
}
```

第二个方法用了yield，但它并没有用0或者null，而是用了Wait()来yield，这相当于是说，“不再继续执行本程序，直到Wait程序结束”。

现在，协程在程序设计方面的能力要开始展现了。

控制对象行为的例子

在最后一个例子中，我们就来看看协程如何像创建方便的计时器一样来控制对象行为。协程不仅仅可以使用可计数的时间来yield，它还能很巧妙地利用任何条件。将它与嵌套结合使用，你会得到控制游戏对象状态的最强大工具。

运动到某一位置

对于下面这个简单脚本组件，我们可以在Inspector面板中给targetPosition和moveSpeed变量赋值，程序运行的时候，该对象就会在协程的作用下，以我们给定的速度运动到给定的位置。

```csharp
[csharp]
usingUnityEngine;
Using System.Collections;

Public class MoveExample : MonoBehaviour
{
    ublic Vector3 targetPosition;
    ublic float moveSpeed;

    Void Start()
    {
        StartCoroutine(MoveToPosition(targetPosition));
    }

    IEnumerator MoveToPosition(Vector3 target)
    {
        while(transform.position != target)
        {
            transform.position = Vector3.MoveTowards(transform.position, target, moveSpeed * Time.deltaTime);
            Yield return 0;
        }
    }
}
```

这样，这个程序并没有通过一个计时器或者无限循环，而是根据对象是否到达指定位置来yield。

按指定路径前进

我们可以让运动到某一位置的程序做更多，不仅仅是一个指定位置，我们还可以通过数组来给它赋值更多的位置，通过MoveToPosition()，我们可以让它在这些点之间持续运动。

```csharp
[csharp]
Using UnityEngine;
Using System.Collections;

Public class MoveExample : MonoBehaviour
{
    ublic Vector3[] path;
    ublic float moveSpeed;

    Void Start()
    {
        StartCoroutine(MoveOnPath(true));
    }

    IEnumerator MoveOnPath(bool loop)
    {
        do
        {
            foreach(var point in path)
                Yield return StartCoroutine(MoveToPosition(point));
        }
        while(loop);
    }

    IEnumerator MoveToPosition(Vector3 target)
    {
        while(transform.position != target)
        {
            transform.position = Vector3.MoveTowards(transform.position, target, moveSpeed * Time.deltaTime);
            Yield return 0;
        }
    }
}
```

我还加了一个布尔变量，你可以控制在对象运动到最后一个点时是否要进行循环。

把Wait()程序加进来，这样就能让我们的对象在某个点就可以选择是否暂停下来，就像一个正在巡逻的AI守卫一样，这真是锦上添花啊！

注意：

如果你刚接触协程，我希望这两个教程能帮助你了解它们是如何工作的，以及如何来使用它们。以下是一些在使用协程时须谨记的其他注意事项：

- l 在程序中调用StopCoroutine()方法只能终止以字符串形式启动（开始）的协程；
- l 多个协程可以同时运行，它们会根据各自的启动顺序来更新；
- l 协程可以嵌套任意多层（在这个例子中我们只嵌套了一层）；
- l 如果你想让多个脚本访问一个协程，那么你可以定义静态的协程；
- l 协程不是多线程（尽管它们看上去是这样的），它们运行在同一线程中，跟普通的脚本一样；
- l 如果你的程序需要进行大量的计算，那么可以考虑在一个随时间进行的协程中处理它们；
- l IEnumerator类型的方法不能带ref或者out型的参数，但可以带被传递的引用；

- I 目前在Unity中没有简便的方法来检测作用于对象的协程数量以及具体是哪些协程作用在对象上。

如果您发现教程中存在问题和错误的信息，或者有任何建议又或者您想要在这里看到其他需要的教程，可以发邮件或者在评论中留言。

想对作者说点什么？ 我来说一句

守护擎天柱：good！ (4个月前 #28楼)

qq_42259066：博主，协程介绍链接失效了，麻烦解决一下。 (4个月前 #27楼)

qq_34961627：讲的很好受教了 (9个月前 #26楼)

qq784864003：在程序中调用StopCoroutine()方法只能终止以字符串形式启动（开始）的协程； 这字段放到上面讲怎么停止协程那里更好。免得看到上面时候蒙圈。 (10个月前 #25楼)

OrangeAHao：那如果想对值类型进行传递怎么办 哭了 (10个月前 #24楼)

秩亦：见过讲协程最好的！！！ (11个月前 #23楼)

SuperFierceBigKK：受教了，讲的很清晰易懂 (1年前 #22楼)

qq_26902089：很棒的携程讲解，谢谢up主 (1年前 #21楼)

aloneSteven：十分精彩，万分感谢 (1年前 #20楼)

TengWan_AlunI：深入浅出，非常感谢！ (1年前 #19楼)

tiramisu850326：酷，通过几个简单的例子就把协程的实际使用案例都列举了，简单易懂。不仅说明了协程的一些原理，还说明了我们大概会在什么情况下使用它。 (1年前 #18楼)

FirsD：哇，非常的感谢，对初学者帮助很大很大 (1年前 #17楼)

LPL2333：很棒棒 (1年前 #16楼)

super_mage_J：恍然大悟，感谢 (1年前 #15楼)

wangruixu：写的很好 终于对协程明白了些 (1年前 #14楼)

上一页 1 2 下一页

Unity中协程(IEnumerator)的使用方法介绍 - 悲欢离合的博客(good good… 2.5万
在Unity中，一般的方法都是顺序执行的，一般的方法也都是在一帧中执行完毕的，当我们… 来自： 悲欢离合…

Unity携程中做一个双层while循环 - 海涛高软(徐海涛（hunk xu)) 211
执行结果：鼠标左键点击，不停的打印xxxx鼠标右键点击，不停的打印yyyy-------------------… 来自： 海涛高软

unity 携程 - a498506133的专栏 1536
http://blog.csdn.net/huang9012/article/details/29595747 unity3d官方对于协程的解释是：… 来自： a498506…

Unity 携程简易使用 - l_bacu的博客(懒人发展科技) 2535
本人Unity菜鸟，此博客只为笔记和分享用，不对的地方请大伽指正 携程： IEnumerator H… 来自： L_Bacu…

unity 携程倒计时 - huhudeni的博客 932
大家可以去看看 这个博主写的倒计时http://blog.csdn.net/pangpangxiong0309/article/detai… 来自： huhudeni…

Unity中携程与线程指南 - fb791364的专栏 1076
欢迎使用Markdown编辑器写博客本Markdown编辑器使用StackEdit修改而来，用它写博客… 来自： fb791364…

Unity 协程使用指南 - 武龙飞的空中楼阁(always be coding.) 6861
使用Unity的过程中，对协程只知道如何使用，但并不知道协程的内部机理，对于自己不清… 来自： 武龙飞的…

[Unity3D]-协程的介绍和使用 - 瞎蹦跶的博客(好好学习,天天向上) 1.1万
本文是个人对Unity协程的一些理解和总结.Unity协程长的有点像线程,但却不是线程.因为协… 来自： 瞎蹦跶的…

Unity3D之协程(Coroutines & Yield) - 道心惟纯(“哦，那你肯定能做到，… 6.5万
写游戏代码,往往最终需要代码为连续的事件.结果会像这样: [它可以实现将一段程序延迟执… 来自： 道心惟纯

相关热词 Unity3d unity3D Unity3D 玩unity3d unity3d吧

unity 协程原理与线程的区别 - it你我 1.6万
说到协程，我们首先回顾以下线程与进程这两个概念。在操作系统（os）级别，有进程（p… 来自： IT你我

jhl227 关注 3篇文章 | linkfly1 关注 49篇文章 | _autism 关注 28篇文章 | 换一批

unity中的简单的协程用法。 - zscjob的专栏 583
public void Init() { StartCoroutine(Next());//调用协程 } IEnumerator Next()//协程 ... 来自： zscjob的…

unity3d倒计时c#代码 - lvmengmeng(每天坚持进一步一点点) 3552
c#倒计时/计时代码 using UnityEngine; using System.Collections; using System; public cl… 来自： lvmengm…

【UNITY3D 游戏开发之六】UNITY 协程COROUTINE与INVOKE - 内推君… 632
这里Himi强调一点：Unity里面的协程并不是线程，协程是在unity主线程中运行的，每一帧… 来自： 内推君（…

unity3D 协程如何模拟Update - life_demo的博客 2057
协程介绍 Unity的协程系统是基于C#的一个简单而强大的接口 ，IEnumerator，它允许你… 来自： life_dem…

我在携程的这十年：一个老运维的成长往事 - garfielder007的专栏(勤奋治… 2490
作者简介 雷兵 携程网 安全中心信息安全专家 2007年1月加入携程，曾任安全经理、高级… 来自： GarfieldE…

快速理解多进程与多线程以及协程的使用场合和特点 - zlx_csdn的博客 844
这篇文章是我从也是同网站的一个大神里面拷贝过来的，有什么不对的愿接受批评首先我… 来自： zlx_csdn…

C#学习 【HttpClient学习】 - 墙角无名氏 603
简单的说几句：由于之前一直做的是PC端的APP，没有做过WEB方面的项目，一直对类…

Unity入门--Unity的下载安装及基本使用 - nicolelili1的专栏 4022
一、Unity的下载安装 1.打开网址：https://unity3d.com/cn/,注册并登陆，点击“获取Unity” 2… 来自： nicolelili1…

初学VR（一）： Unity的使用 - ai_xao的博客(编程之路) 2771
我是很有幸能够在接下来的两周学习到VR，特整理以下的笔记，供以后学习使用。 工欲… 来自： ai_xao的…

U3d引擎与资源管理 - qq_40219038的博客 451
U3d引擎界面介绍五个视图区：Scene视图：场景视图 用来摆放Hierarchy中的游戏对象。… 来自： qq_4021…