**脚本中Start()和Awake()的区别？**

Posted on 2013年06月13日 by U3d / [Unity3D 基础教程](http://www.unitymanual.com/category/manual/unity3d-%e5%9f%ba%e7%a1%80%e6%95%99%e7%a8%8b)/被围观 62 次

Unity3D初学者经常把Awake和Start混淆。官方文件所描述：

Awake(): Awake is called when the script instance is being loaded.

Start(): Start is called just before any of the Update methods is called the first time.

Awake在MonoBehavior创建后就立刻调用，Start将在MonoBehavior创建后在该帧Update之前，在该Monobehavior.enabled == true的情况下执行。

Awake()：当一个脚本实例被载入时Awake被调用。Awake用于在游戏开始之前初始化变量或游戏状态。在脚本整个生命周期内它仅被调用一 次.Awake在所有对象被初始化之后调用，所以你可以安全的与其他对象对话或用诸如 GameObject.FindWithTag 这样的函数搜索它们。每个游戏物体上的Awke以随机的顺序被调用。因此，你应该用Awake来设置脚本间的引用，并用Start来传递信息。Awake 总是在Start之前被调用。它不能用来执行协同程序。Unity3D教程手册

Start()：Start仅在Update函数第一次被调用前调用。Start在behaviour的生命周期中只被调用一次。它和Awake的不同是Start只在脚本实例被启用时调用。

你可以按需调整延迟初始化代码。Awake总是在Start之前执行。这允许你协调初始化顺序。

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 1 | **void** Awake (){ |
| 2 | } |
| 3 | *//初始化函数，在游戏开始时系统自动调用。一般用来创建变量之类的东西。* |
| 4 |  |
| 5 | **void** Start(){ |
| 6 | } |
| 7 | *//初始化函数，在所有Awake函数运行完之后（一般是这样，但不一定），在所有Update函数前系统自动条用。一般用来给变量赋值。* |

通常书写的脚本，并不会定义[ExecuteInEditMode]这个Attribute，所以Awake和Start都只有在Runtime中才会执行。

例1.

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 01 | **public** **class** Test : MonoBehaviour { |
| 02 | **void** Awake () { |
| 03 | Debug.Log("Awake"); |
| 04 | enabled = **false**; |
| 05 | } |
| 06 |  |
| 07 | **void** Start () { |
| 08 | Debug.Log("Start"); |
| 09 | } |
| 10 | } |

以上代码，在Awake中我们调用了enabled = false; 禁止了这个MonoBehavior的update。由于Start, Update, PostUpdate等属于runtime行为的一部分，这段代码将使Start不会被调用到。

在游戏过程中，若有另外一组代码有如下调用:

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 1 | Test test = go.GetComponent<Test>(); |
| 2 | test.enabled = **true**; |

这个时候，若该MonoBehavior之前并没有触发过Start函数，将会在这段代码执行后触发。

例2.

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 01 | **private** Transform handAnchor = **null**; |
| 02 | **void** Awake () { handAnchor = transform.Find("hand\_anchor"); } |
| 03 | *// void Start () { handAnchor = transform.Find("hand\_anchor"); }* |
| 04 | **void** GetWeapon ( GameObject go ) { |
| 05 | **if** ( handAnchor == **null** ) { |
| 06 | Debug.LogError("handAnchor is null"); |
| 07 | **return**; |
| 08 | } |
| 09 | go.transform.parent = handAnchor; |
| 10 | } |

other.cs

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 1 | ... |
| 2 | GameObject go = new GameObject("player"); |
| 3 | player pl = go.AddComponent<player>(); *// Awake invoke right after this!* |
| 4 | pl.GetWeapon(weaponGO); |
| 5 | ... |

以上代码中，我们在player Awake的时候去为handAnchor赋值。如果我们将这步操作放在Start里，那么在other.cs中，当执行GetWeapon的时候就会出现handAnchor是null reference.Unity3D教程手册

总结：我们尽量将其他Object的reference设置等事情放在Awake处理。然后将这些reference的Object的赋值设置放在Start()中来完成。

当我们为MonoBehavior定义了[ExecuteInEditMode]后，我们还需要关心Awake和Start在编辑器中的执行状况。

当该MonoBehavior在编辑器中被赋于给GameObject的时候，Awake, Start 将被执行。

当Play按钮被按下游戏开始以后，Awake, Start 将被执行。

当Play按钮停止后，Awake, Start将再次被执行。

当在编辑器中打开包含有该MonoBehavior的场景的时候，Awake, Start将被执行。

值得注意的是，不要用这种方式来设定一些临时变量的存储(private, protected)。因为一旦我们触发Unity3D的代码编译，这些变量所存储的内容将被清为默认值。

相关文章：[Awake()跟Start()差在哪？](http://www.unitymanual.com/5264.html)