Cha4-Unity3d 和 C#的双剑合璧 02

在上一课的内容中，我们逐行解释了一个新创建的 Unity 游戏脚本中每一行代码的作用。

而在这一课的内容中，我们将详细介绍 UnityEngine 所提供的重要事件函数。

还等什么呢？让我们开始吧~

个人微信号：iseedo

微信公众号：vrlife

02 Unity 和 C#的双剑合璧-Unity 游戏脚本剖析中

Unity 中的脚本和一般计算机程序的运行方式略有不同。一般的计算机程序会在某个循环内持续执行，直到完成任务。而 Unity 则会通过调用内部事先声明好的函数，把控制间隙性的交给某个脚本。每当某个函数执行完自己的任务后，控制权就会交回给 Unity。这些函数被称之为事件函数，当 Unity 响应游戏中的事件时，它们会被调用。

Unity 使用了特定的命名机制，以识别响应特定事件的函数。到目前为止，我们只用到了两个，也就是 Start 和 Update，实际上还有其它的事件函数，让我们依次来简单认识一下~

需要注意的是，对于初学者来说，没有必要一开始就陷入很深的细节中不可自拔。对于这一课的内容同样如此，你只需要对这些事件函数混个脸熟。至于每个事件函数在游戏开发中真正应该如何使用，这里即便都讲了相信大家也形成不了很深的印象，我们还是需要用实际的示例来说明。

需要再次强调的是，所有的这些事件函数都包含在 MonoBehaviour 这个类中，我们可以在官方文档中找到相关的详细介绍：

https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

以下列出了 Unity 中最常见同时也是最重要的一些事件，**大家先来混个脸熟吧**~

更新事件

在 Unity 中创建一个新脚本时，脚本中会默认添加 Update 事件函数。

游戏在某种程度上可以看作一个连续动画，里面的每一帧都是事先设置好的。在游戏中，在渲染每一帧之前，都需要改变游戏对象的位置、状态和行为。而 Update 函数正是放置此类代码的主要地方。

例如：

```
   void Update() {
        float distance = speed * Time.deltaTime * Input.GetAxis("Horizontal");
        transform.Translate(Vector3.right * speed);
   }
```

和帧的渲染类似，对物理系统的模拟也需要持续不断的更新。

对于跟物理系统相关的更新，Unity 提供了 FixedUpdate 事件函数，会在每次物理更新前调用。FixedUpdate **和 Update 函数的最根本区别是，FixedUpdate 用于固定频率的更新，而 Update 的更新频率取决于游戏的帧速。**

在 Unity 中依次单击菜单项 Edit→Project Settings→Time，可以打开 Time Manager 面板，其中 Fixed Timestep 选项用于设置 Fixed Update 的更新频率，默认为 0.02 秒，即每秒 50 次。

通常情况下，FixedUpdate()会比 Update()更频繁地调用。当帧率较低时，在某一帧的时间间隔内 FixedUpdate 可能会被调用多次；而当帧率很高时，在某一帧的时间间隔内 FixedUpdate 可能根本不会被调用。

还有一点，在 Unity 中，如果设置 Time Scale 值为 0，可以实现动力学特性的暂停，即所有在 FixedUpdate 函数中的代码都不会被执行。

例如：

```
void FixedUpdate() {
        Vector3 force = transform.forward * driveForce *
Input.GetAxis("Vertical");
        rigidbody.AddForce(force);
   }
```

除了 Update 和 FixedUpdate 函数，还有一个 LateUpdate 函数。

LateUpdate 用于延迟更新，此方法在 Update()之后执行，每一帧调用一次。

关于它们的具体用法和区别，这里先不展开讲，感兴趣的朋友可以查询官方文档了解其作用。

https://docs.unity3d.com/ScriptReference/MonoBehaviour.FixedUpdate.html
https://docs.unity3d.com/ScriptReference/MonoBehaviour.LateUpdate.html
https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html

2.初始化事件

在每个新创建的脚本中都有一个 Start()函数，其作用是在游戏的第一帧更新前调用。除了 Start()函数，还有一个 Awake()函数，用于脚本唤醒。此方法为系统执行的第一个方法，用于脚本的初始化，在脚本的生命周期中只执行一次。

也就是说，当某个游戏场景开启的时候，会首先调用 Awake()等函数。

Start 方法在 Awake 之后执行，在脚本的生命周期中只执行一次。

由于 Awake 和 Start 函数的特性与 C#中的构造函数类似（一般情况下，在整个脚本生命周期中只执行一次），所以在 Unity 中也常用来初始化类的成员变量。

3.游戏界面响应事件

Unity 有个系统用来渲染并响应 GUI 控件，不过在 UGUI 系统推出后，它的使用频度和作用相对减少了。

以下代码可以用来在屏幕中显示一个 Game Over 的标签：

```
void OnGUI() {
      GUI.Label(labelRect, "Game Over");
}
```

关于 OnGUI 函数的详细用法，请参考：

https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnGUI.html

4.物理检测事件

物理引擎通过特定的事件函数来判断碰撞、触发等事件，比如 OnCollisonEnter、OnCollisonStay、OnCollisionExit 分别在碰撞开始、持续与结束时调用。

如果碰撞体设置成了触发器，则可以调用对应的 OnTriggerEnter、OnTriggerStay、OnTriggerExit 函数。

此外，**还有对关节的检测事件，如 OnJointBreak、OnJointBreak2D 等函数。**

例如：

```
void OnCollisionEnter(otherObj: Collision) {
      if (otherObj.tag == "Arrow") {
            ApplyDamage(10);
      }
}
```

5.动画控制事件

Unity 提供了动画控制事件函数来处理动画，主要是 OnAnimatorIK 和 OnAnimatorMove。

具体请参考：

https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnAnimatorIK.html
https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnAnimatorMove.html

6.程序控制事件

Unity 提供了程序控制事件函数来处理程序的启动、暂停、退出等，主要是 OnApplicationFocus、OnApplicationPause、OnApplciationQuit。

具体请参考：

https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnApplicationFocus.html
https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnApplicationPause.html
https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnApplicationQuit.html

7.游戏对象控制事件

Unity 提供了一系列的游戏对象控制事件函数来处理和游戏对象状态相关的事件，主要是 OnEnable、OnDisable、OnTransformChildrenChanged、OnTransformParentChanged 等。

感兴趣的童鞋可以自行到这里搜索查看具体的使用细节~

https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

8.鼠标控制事件

Unity 提供了一系列的鼠标控制事件函数来处理和鼠标状态相关的事件，主要是 OnMouseDown、OnMouseEnter、OnMouseExit、OnMouseOver、OnMouseUp、OnMouseUpAsButton 等。

感兴趣的童鞋可以自行到这里搜索查看具体的使用细节~

https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

9.网络系统事件

Unity 提供了一系列的网络系统事件函数来处理和网络连接相关的事件，主要是 OnDisconnectedFromServer、OnFailedtoConnect、OnFailedToConnectToMasterServer、OnPlayerConnected、OnPlayerDisconnected、OnMasterServerEvent、OnServerInitialized 等。

感兴趣的童鞋可以自行到这里搜索查看具体的使用细节~

https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

10.视觉渲染事件

Unity 提供了一系列的视觉渲染事件函数来处理和视觉渲染相关的事件，主要是 OnDrawGizmos、OnDrawGizmosSelected、OnBecameInvisible、OnBecameVisible、OnPostRender、OnPreCull、OnPreRender、OnRenderImage、OnRenderObject、OnWillRenderObject 等。

感兴趣的童鞋可以自行到这里搜索查看具体的使用细节~

https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

11.粒子系统事件

Unity 提供了一系列的粒子系统事件函数来处理和粒子系统相关的事件，主要是 OnParticleCollision、OnParticleSystemStopped、OnParticleTrigger 等。

感兴趣的童鞋可以自行到这里搜索查看具体的使用细节~

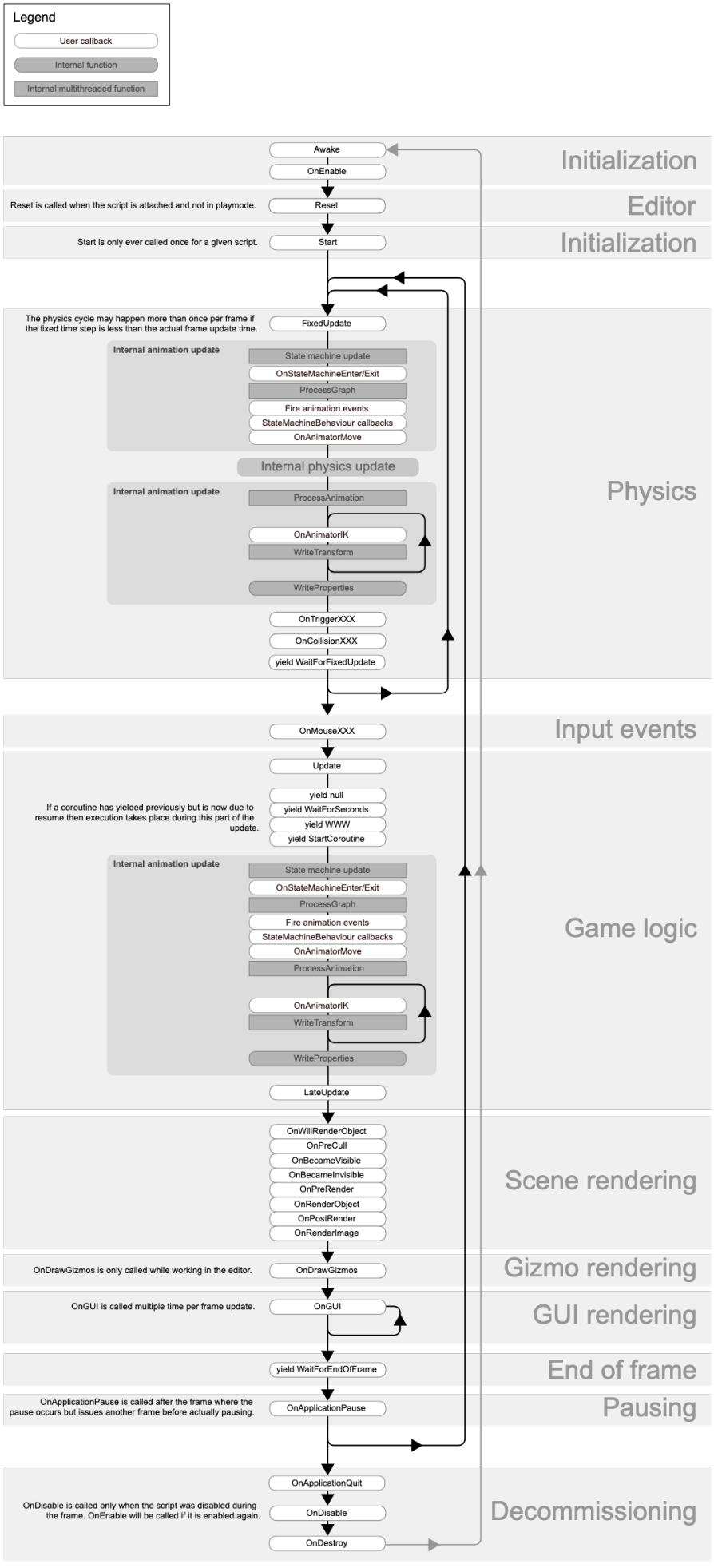
https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

12.其它事件

以上没有提到过的事件函数可以归到这一类。

在 Unity 脚本中有如此多的事件函数，那么在实际程序运行的时候有没有一个指定的先后顺序呢？~

答案是：有的，而且这个执行顺序是 Unity 官方预先确定的，其执行顺序可以参考下图。


Legend
User callback
Internal function
Internal multithreaded function
Initialization
Awake
OnEnable
Editor
Reset is called when the script is attached and not in playmode.
Reset
Initialization
Start is only ever called once for a given script.
Start
Physics
The physics cycle may happen more than once per frame if the fixed time step is less than the actual frame update time.
FixedUpdate
Internal animation update
State machine update
OnStateMachineEnter/Exit
ProcessGraph
Fire animation events
StateMachineBehaviour callbacks
OnAnimatorMove
Internal physics update
Internal animation update
ProcessAnimation
OnAnimatorIK
WriteTransform
WriteProperties
OnTriggerXXX
OnCollisionXXX
yield WaitForFixedUpdate
Input events
OnMouseXXX
Game logic
Update
If a coroutine has yielded previously but is now due to resume then execution takes place during this part of the update.
yield null
yield WaitForSeconds
yield WWW
yield StartCoroutine
Internal animation update
State machine update
OnStateMachineEnter/Exit
ProcessGraph
Fire animation events
StateMachineBehaviour callbacks
OnAnimatorMove
ProcessAnimation
OnAnimatorIK
WriteTransform
WriteProperties
LateUpdate
Scene rendering
OnWillRenderObject
OnPreCull
OnBecameVisible
OnBecameInvisible
OnPreRender
OnRenderObject
OnPostRender
OnRenderImage
Gizmo rendering
OnDrawGizmos is only called while working in the editor.
OnDrawGizmos
GUI rendering
OnGUI is called multiple time per frame update.
OnGUI
End of frame
yield WaitForEndOfFrame
Pausing
OnApplicationPause is called after the frame where the pause occurs but issues another frame before actually pausing.
OnApplicationPause
Decommissioning
OnApplicationQuit
OnDisable is called only when the script was disabled during the frame. OnEnable will be called if it is enabled again.
OnDisable
OnDestroy

Unity 中的脚本生命周期

看不清楚的 童鞋可以直接在官网看大图~

https://docs.unity3d.com/Manual/ExecutionOrder.html

如果这些执行顺序用文字来说明的话，估计大家直接就懵逼掉了。所以，我们用示例来说明，测试事件函数的执行顺序。

打开上一课创建的 FairyLand 项目，双击在 Visual Studio 中打开 ExampleScrip.cs 文件（在 Project 视图中是看不到后缀的~处女座童鞋们要注意了）

更改其中的代码如下：

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ExampleScript : UnityEngine.MonoBehaviour
{
    //OnApplicationQuit

    private void OnApplicationQuit()
    {
        print("OnApplicationQuit");
    }


    //OnDisable
    private void OnDisable()
    {
        print("OnDisable");
    }


    //OnDestroy

    private void OnDestroy()
    {
        print("OnDestroy");
    }
```

```
//OnEnable method

private void OnEnable()
{
    print("OnEnable");
}

//Awake method
private void Awake()
{
    print("Awake");
}




// Start is called before the first frame update
void Start()
{
    print("Start");
}

// Update is called once per frame
void Update()
{
    print("Update");
}

//LateUpdate
private void LateUpdate()
{
    print("LateUpdate");
}
//FixedUpdate
private void FixedUpdate()
{
    print("FixedUpdte")
}

//Reset
private void Reset()
```

```
    {
        print("Reset");
    }
}
```

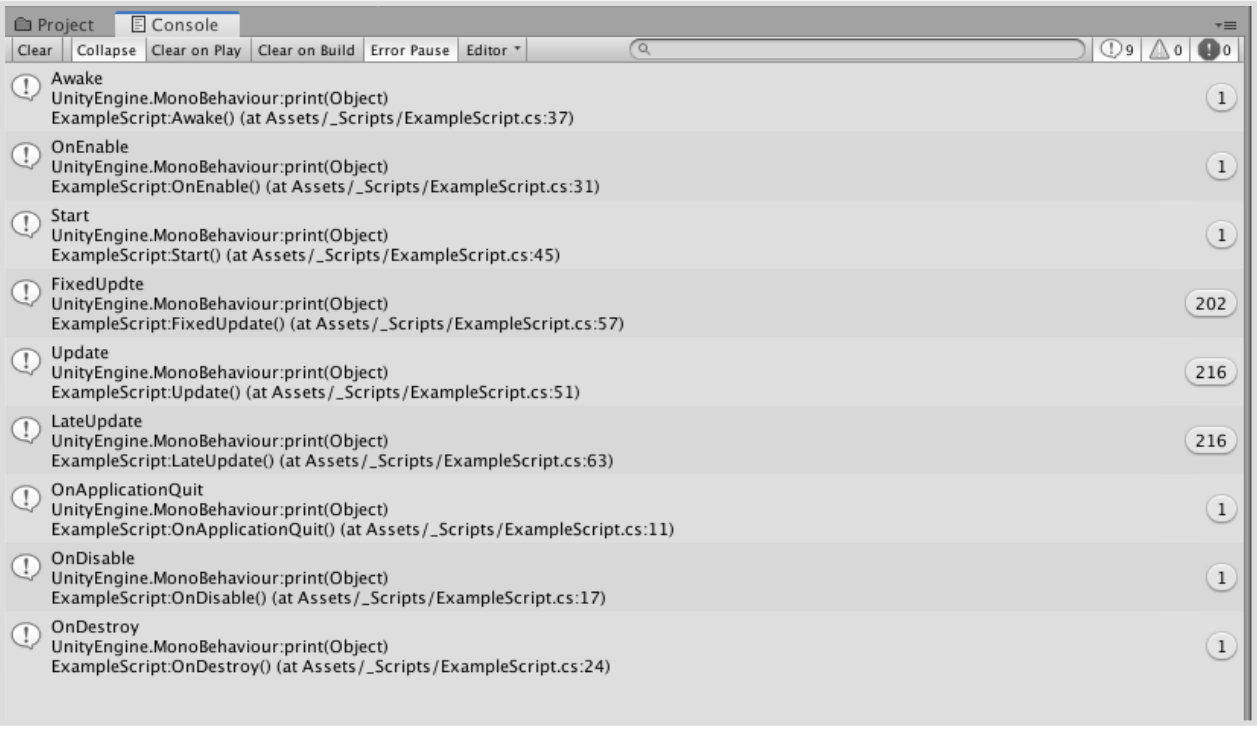
在脚本中实现了上述几个主要生命周期方法。

注意：

1. 当我们手动敲代码的时候，在创建这些事件函数的时候会自动添加一个 private 在前面。其实默认情况下，不添加 public 就等于添加了 private~

2. 这里的 print 和之前我们用的 Debug.Log 是一回事，就看大家习惯用哪个了，都是在 Unity 编辑器的 Console 视图中输出信息。

在运行场景之前，请确保勾选了 Console 窗口的 Collapse 选项，如图所示。



之所以**要勾选这个选项，是因为有些事件函数（如 Update 会反复输出~）**

运行场景，会输出各个生命周期方法的执行次数，如图所示：

结束场景后，可以清楚看到：**最先执行的是 Awake()方法，然后是 OnEnable()方法，随后是 Start()方法，这三个方法都执行了一次**。

接下来调用的是 FixedUpdate()方法。

而 Update()和 LateUpdate()方法执行频率一样，都是每帧执行一次。

最后结束场景的时候，调用到了 OnApplicationQuit()、OnDisable()**和** OnDestroy()方法。

当然，有些事件函数并没有展示其作用。而且这些只是 Unity 生命周期的一部分，其他部分会在后续章节中详细讲解。

Unity Events 不仅仅包含脚本的生命周期，还包含一些由系统自动调用的方法，也就是“回调”。大家需要明白的是，这些方法不由开发者自己调用，而是由 Unity 在适当的时候自行调用，开发者只要实现这些方法即可。

这一课的内容相对比较枯燥，但是再次强调，大家无需完全看懂，只需要留下个大概的印象就好。

关于事件函数的具体使用，我们会在后续的示例游戏开发过程中让大家更全面深入的了解~

好了，那么下一课是不是就要开始动手开发游戏了？

且慢，我们还有最后一堂小小的理论课要上，那就是关于 Unity 游戏脚本中的系统类和方法。和本课的内容一样，下一课的内容仍然只是让大家构建一个基本的概念，混个脸熟而已~

好了，让我们下一课再见。