**Unity3D教程：后台资源加载与yield用法**

Posted on 2013年06月17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 65 次

项目需求是一个应用在启动时需要读取几个配置文件，原始文档自然是excel格式，为了方便，把它们转存为csv格式，即用逗号分割的文本格式，然后在js脚本中自己解析，倒也轻松，但是问题就是解析得比较慢。 一个100行80多列的文档解析需要几秒钟，这个有点难受了。更可怕的是Unity发布到ios上的程序启动有个限制，超过一定的时间没有加载完成会被系统强制终止。

控制台错误输入如下：

Aug 20 17:00:45 xxx-iPhone SpringBoard[30] : xxx failed to launch in time

Aug 20 17:00:45 xxx-iPhone SpringBoard[30] : Forcing crash report of xxx[1067]...

Aug 20 17:00:46 xxx-iPhone SpringBoard[30] : Finished crash reporting.

Aug 20 17:00:46 xxx-iPhone com.apple.launchd[1] (UIKitApplication:com.xxx.xxx[0xd288][1067]) : (UIKitApplication:com.xxx.xxx[0xd288]) Exited: Killed: 9

其中的xxx是代替了个人的一些信息，注意几个关键字：

failed to launch in time (没能及时启动成功)

Exited : Killed : 9 (以kill 9的方式退出， kill 9 是被强制终止)

yield可以解决此问题。yield的核心功能是让代码在此返回，下一循环（update）继续执行，看Unity自己的例子：

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

|  |  |
| --- | --- |
| 01 | function Start () { |
| 02 |  |
| 03 | StartCoroutine(“DoSomething”, 2.0); |
| 04 |  |
| 05 | **yield** WaitForSeconds(1); |
| 06 |  |
| 07 | StopCoroutine(“DoSomething”); |
| 08 |  |
| 09 | } |
| 10 |  |
| 11 | function DoSomething (someParameter : **float**) { |
| 12 |  |
| 13 | **while** (**true**) {&nbsp; *//Unity3D教程手册* |
| 14 |  |
| 15 | print(“DoSomething Loop”); |
| 16 |  |
| 17 | *// Yield execution of this coroutine and return to the main loop until next frame* |
| 18 |  |
| 19 | **yield**; |
| 20 |  |
| 21 | } |
| 22 |  |
| 23 | } |

上面的一段先不管，先看下面一段，循环中的yield会让函数暂时返回，下一帧继续，这个在c++里面是没有的，一开始难以理解，以为类似于MFC里面的Dialog.DoModal()，于是就这么用了：

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

|  |  |
| --- | --- |
| 1 | function Start(){ |
| 2 |  |
| 3 | LoadA(); |
| 4 |  |
| 5 | LoadB(); |
| 6 |  |
| 7 | LoadC(); |
| 8 |  |
| 9 | } |

LoadA(), LoadB(), LoadC()分别是一个类似于上面DoSomething的函数，但是我的LoadB是依赖LoadA完成的，就是说如果LoadA没有完成时是不能开始LoadB的，这样用就出错了。

因为LoadA并没有想Dialog.DoModal()一样，而是直接返回了，就是说第一帧，LoadA，LoadB,LoadC都执行了，这非我预想。经过加入各种输出信息，终于明白过来，函数是立即返回的，但是里面的循环工作（yield后面的代码）会在后续帧里面得以继续执行。Unity3D教程手册

为此，需要重新找到LoadA完成的切入点，我是这样做的：

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

|  |  |
| --- | --- |
| 01 | function LoadA() |
| 02 |  |
| 03 | { |
| 04 |  |
| 05 | **for**( **var** i : **int** = 0; i < 99; ++i) |
| 06 |  |
| 07 | { |
| 08 |  |
| 09 | *// do something ......* |
| 10 |  |
| 11 | **yield**; |
| 12 |  |
| 13 | } |
| 14 |  |
| 15 | CallFunctionFinished(); |
| 16 |  |
| 17 | } |

然后在CallFunctionFinished()这个函数里面调用LoadB,当然也可以用SendMessage把调用做的低耦合一点。还有个StartCoroutine，在js里面是可无的，Unity的文档中明确说了：如果要做一个loading的进度条，就只需要在我们读取数据的循环中加入计数，然后在update函数中读取计数显示进度条即可。