Unity3D教程：协同程序和让步

在编写游戏代码时，通常需要在脚本中结束一系列的事件。例如：

private var state = 0; 　　function Update() 　　{ 　　if(state==0) 　　{ 　　//do step 0 　　state = 1; 　　return; 　　} 　　if(state==1) 　　{ 　　//do step 1 　　state = 2; 　　return; 　　} 　　... 　　}

  这时，我们使用让步（yield）语句是非常方便的，yield语句是一种特殊的 return语句。它确保函数在下次被调用时，能够从 yield语句开始继续执行。[Unity3D](http://www.unitymanual.com/)教程手册

while(true) 　　{ 　　//do step 0 　　//wait for one frame 　　yield; 　　//do step 1 　　//wait for one frame 　　yield; 　　... 　　}

  你也可以传递一个特定的值给yield语句来推迟Update函数的执行， 直到某个事件发生：

//do something //wait for 5 seconds yield WaitForSeconds(5.0); //do something more...

  你可以堆积并连接一些协同程序（Coroutine） 。例如，下面的示例在执行 Do 函数时，立即执行随后的代码：

Do(); print(“This is printed immediately”); function Do() { print(“Do now”); yield WaitForSeconds(2); print(“Do 2 seconds later”); }

下面的示例在执行完Do函数后，才执行随后的代码：

// chain the coroutine yield StartCoroutine(“Do”); print(“Also after 2 seconds”); print(“This is after the Do coroutine has finished execution”); function Do() { print(“Do now”); yield WaitForSeconds(2); print(“Do 2 seconds later”); }

注：不能在 Update 或 FixedUpdate 函数中使用 yield 语句，但可以在它们中使用  
  
    StartCoroutine函数来调用一个函数。

http://www.unitymanual.com/thread-1839-1-1.html