**Unity3D教程：隐藏3D模型及GetComponentsInChildren的使用**

Posted on 2013年04月24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 209 次

有时游戏设计的需求，在场景中的3D模型物件未必都是一开始就一直是显示的状态，有可能会因为某些事件的发生而必须将物件隐藏，或因为某些事件的发生必须使该物件再次显示出来，通常在大部份的状况，我们可能会先制作一个Prefabs，在需要该物件时利用Instantiate()建立一个实例物件，使物件在场景中运作，在不需要该物件时利用Destroy()将此物件销毁，可是如果我们的需求是必须持续使用该物件，物件本身带有某些资料值会持续使用到，那么就不能随意将其销毁，但此时需有一段时间不能使该物件出现在画面上，那么我们应该怎么做呢？

物件的隐藏与显示，这个定义上很广泛，主要就是让镜头里看不到此物件，以下就来讨论几种方式：

**将物件移出镜头外**

通常游戏场景都会有地板、背景等，我们如果暂时需要将物件隐藏掉，可暂时改变物件在3D空间的座标位置，将它藏在地板下、背景后，甚至是镜头后方，这样就可以很轻易的隐藏画面中的物件，不过，有些缺点，就是当需要再次将物件显示在画面之中时，必须将物件放回正确的位置，如果游戏场景及镜头都是定点固定不动的，那么将物件放回原来位置并不难，但如果游戏场景、镜头都是依照游戏进行在不断运作改变方向及内容的话，此时在处理物件座标定位上可能就要多费点功夫了。

**统一储存相关资讯**

如上面第一段所提到的，在需要时产生物件，在不需要显示时销毁物件，需要延续使用的相关数据，统一储存在某个记忆位置、档案、资料库...中，如此做法其实相当直接，不需要的东西就丢，但也有同样的问题，当该物件于下次需要显示时，我们必须要很清楚知道他应该出现的位置，然后在这个位置产生物件，使游戏进行不会错乱掉，但对于有使用物理碰撞反弹等等的游戏，那就很难去预测到该出现的正确位置，而且如果这些位置数据还要自己写程式计算的话，Unity的物理引擎反倒显得无用武之地了。

**关闭物件，使物件无法运作**

当游戏进行到某事件发生时，我们不想将物件销毁，而只想让物件在目前位置消失并停止运作，等待之后事件发生时再显示出来，此时我们就可以利用GameObject.active将该物件关闭，或使用GameObject.SetActiveRecursively()直接将该物件及他的子物件全部关闭，使他停止运作，因为物件被停止了，所以此时他将不会显示在画面上，而物件本身带有的Component也都会停止运作，待之后事件发生需要显示该物件时再将其开启，不过此时很容易发生一种情形是，GameObject.Find()要找回此物件并将其开启时，却发现找不到，因为Find()只会帮你找出正在活动中的物件，所以在将物件关闭前，我们必须将此物件放至预先定义好的变数成员中，使我们的程式保有他的参照，在需要开启时才能利用此参照找回这个已被关闭的物件；因为已持有他的参照，所以此物件虽然是在关闭的状态，但我们仍然能对他控制，使他改变位置或变更其他数据，但也因为是关闭的状态，所以我们付予他的任何主动功能动作将不会运作。

**只控制renderer的开关**

这是最直接的方式，利用renderer.enabled来控制模型渲染是否启用，关闭了就只是看不到该物件，但全部的Component仍然正常运作中，待需要显示时再将它开启，不过通常一个模型并不单单只有一个物件，例如一个人物可能分为上半身、下半身、身体、手、脚、头等等，甚至是更细部，所以我们可能就会想利用类似这样的方式将子物件的模型渲染都关闭..

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

|  |  |
| --- | --- |
| 1 | **for**(var\_child:TransforminparentObject){ |
| 2 |  |
| 3 | \_child.renderer.enabled=**false**; |
| 4 |  |
| 5 | } |
| 6 |  |

但是如果整个模型构造不只一层，而是多层次的树状结构，那么我们可能就需要用巢状式的forin回圈一层一层的执行renderer.enabled=false，此时又必须考虑到每个模型的树状结构都不同，总不能为每个模型都客制化写一个巢状回圈让他们去做隐藏，这样子写程式就太笨了，于是我们可能会写一个script专门做这个动作，不过我们并不知道模型的结构会有几层，这么做还真是不好处理；幸好Unity有个GameObject.GetComponentsInChildren()帮我们解决了这个问题，在这部份，我们不再去考虑模型本身是否有子物件或者是有几层的结构，一句话就能取得全部并控制他们；官方说明页面的范例，我照样的改到我的程式中并无法取得我需要的东西，所以在此提供我的写法以供参考...

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

|  |  |
| --- | --- |
| 1 | varmeshObject:Transform; |
| 2 |  |
| 3 | \_components:Component[]=meshObject.GetComponentsInChildren(Renderer); |
| 4 |  |
| 5 | **for**(var\_comp:Componentin\_component){ |
| 6 |  |
| 7 | \_comp.renderer.enabled=**false**; |
| 8 |  |
| 9 | } |

如此，我们很方便的就能为物件隐藏或显示，但物件在游戏场景中仍正常动作中，如果设计了一些物理运作或碰撞事件，那么物件碰到某物件或移动到某个区域就自己隐藏起来，隐藏后仍继续活动，可能又碰到了某物件或到了某个特定区域又再自己显示出来，如此下去，我们可以省掉许多程式运算及控制，只要在适当的时候告诉物件应该隐藏或显示就行了。

GetComponentsInChildren()不只是用在物件的renderer的控制上，还可用在任何Component的控制上，所以我们可以将数个相关的物件都放到一个空物件中，以群组的方式利用GetComponentsInChildren()来统一管理并控制其中的Component及script中的变数成员及函式。