解决--多地形场景带来的大量内存占用问题

涉及到几个U3D函数：

Application.LoadLevel(lv) 　　场景读取(记着用多场景前要现在File-Build Setting里登记下场景，才能在脚本里读取到) DontDestroyOnLoad(object)　 保持物体在场景切换的时候不被卸载(能保持他的所有属性哦)

\*AssetBundle 类　　　　　　 预读资源（主要用于web3d，运行时实时从服务器下载需要的场景资源）

涉及到几个基础知识：

static 静态类、静态变量：在整个游戏中都不会被重新加载，所以可以当全局全场景变量使用，主要用于记录场景数组。

Collider的Is Trigger属性：设置成True，他是可以穿越不会产生能量传递的，但是，他是可以接受碰撞侦测的。配合主角的OnTriggerEnter事件，就可以知道你是否正在穿越一个Trigger了。在这里，我们用在判断何时加载新场景上。

2.制作场景边界

使用Cube + IsTrigger=True属性是最好的办法。

1）如果是双场景切换，注意两个场景边界坐标别重在一起，不然你走到边界会发现两个边界不停的切换=.=，要让2个场景边界互相交错一起。有必要的话，边界可以往里面缩一点，避免用户看到边界。然后建议是把Cube

                                 3.编写脚本

先理清楚逻辑关系以及一些常识：

1.是主摄像机走到边界才会做场景加载或卸载动作。所以代码是放在主摄像机上或者主角上。

2.场景可以加载，但是没有卸载场景这个东西（也没必要卸载，因为同一时间只会有一个场景为当前场景），所以如果你用九宫格方式做无缝连接，你需要把场景读取，然后让场景里所有物体DontDestroyOnLoad（当然包括主角，也就是this）,然后其他场景也这么操作，当需要卸载场景时，只要把所有那个场景Object给Destroy掉就可以了。而做双场景连接则不需要这样，也简单的多。

3.如果是九宫格，你需要一个静态二维数组去记录每个位置场景的名字。这里也可以不需要这么做，有个技巧，你可以格式化场景名字规则来推算下一个需要加载的场景名字，比如M1N1表示(1,1)场景，那你就可以用字符串拆分的方法知道需要读取M0N0,M1N0,M2N0等等的场景。

4.接下来就是处理碰撞，获取下一个场景(双场景方式)或者当前场景(九宫格方式)的名字，这里你就可以看到一个被格式化过的场景名有多么重要。

　　　　function OnTriggerEnter(other:Collider){

　　　　　　Application.LoadLevel(other.name);  //这是双场景方式直接把边界Cube名字设为了下一个场景名

　　　　}

     上面的介绍是大概的描述-具体实现方法如下

我们在使用这个技术之前要将你的地形--在U3D里的File-Build Setting里登记下场景，才能在脚本里读取到-Add Current(这个是登记地形）

      之后我们来制作一个简单的-2个地形的切换方法

如图：

           这个图是做好了的2张地形--中间的Cube是用来接受角色的碰撞的--这样我们就知道在何时去载入我们的下张地形场景了（不包括其他-数据。。列入--坐骑这类的-这个需要另一段代码单独给坐骑-马或者车子这类的--还有很多）

           Cube---把Inspector--Is Trigger--划勾--（这个作用是接受碰撞但不产生能量传递-也就是说他接受碰撞，但可以让同样具有 Is Trigger-划勾的属性物体通过--列入-我们的角色或坐骑）（补充--我们的角色也需要去勾选-Is Trigger-这些才可以通过，但官方自带的FPS--没有Is Trigger-这个勾选像-可以用如下方法解决-创建一个新的Cube为他重新命名-把他作为FPS-父物体--然后勾选-Cube的 Is Trigger-子物体就会有这项属性了）

           这些工作建立好了以后-开始我们的脚本工作。

脚本如下：

function OnTriggerEnter(other:Collider)

{

if (other.gameObject.name=="Cube1")

Application.LoadLevel("Terrain 1");

};

为了方便大家理解--我没有重新命名--Cube1--就是我们接受碰撞体，Terrain 1是我们的要载入的下张地形。（大家可以用很多方法去优化他--有优化的方法请发在这个帖子内-方便大家查看-非常感谢）

     这段代码要放在角色上或角色摄像机上或FPS的父物体Cube上。

以上的操作就完成了-一个重地形0到地形1的切换--（可以解决100M或跟高的内存占用问题-这个要看你的地形场景而定了）

      但这并不完整--我们也可能要重地形1回到地形0---这个要你们自己解决了-以上的内容已经把这个解决的方法说出来了--大家自己学习发挥下--这样才有进步。

      下面我们来说明4张地形场景的载入--逻辑（这里只说明逻辑，具体代码和上方一样，需要大家自己发挥下）

      如图 ：   这个图中可以看到4张地形场景--我重点讲解--中间的2个大的Cube逻辑--那4个长方形的大家应该都清楚了。除非你没认真看。

      中间最大的Cube是来判断--角色走的这个范围内的时候他要去载入那张地形场景-如果在这个大的Cube的范围内折载入其他的2张地形场景。（这个大的Cube可以根据自己的地形规格-做出调整这里给出的并不准确）

      中间最小的Cube是来判断--角色走的这个范围内的时候他要去载入那张地形场景-如果角色走入小的Cube中后-载入其他3地形。

上面的方法只是--引导--实际制作中可以跟据这些方法推敲-这个4张地形场景同样少了-往返的-制作介绍--给大家学习的空间去发挥吧。