**关于Unity中Camera的Aspect**

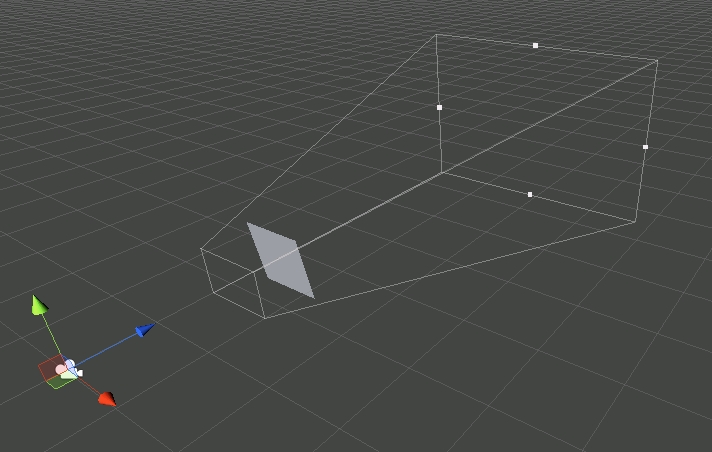
2016-05-01 16:47:18 [cbbbc](https://me.csdn.net/cbbbc) 阅读数 3490更多

分类专栏： [Unity3D](https://blog.csdn.net/cbbbc/article/category/5758469) [3D渲染管线](https://blog.csdn.net/cbbbc/article/category/6209896)

　一直以来对Camera的Aspect和Game窗口的Aspect都是一知半解,某天从一本书中看到了对Camera的API讲解,但是总觉得对Aspect讲解的有问题。于是就认真的思考起了这个问题，还发现设置完Cmera.aspect之后,Scene窗口的视椎体竟然不同步, 也不知其原因。苦恼了很久。经过一番研究并与同事讨论有所收获,便写下此文。一方面为了强化自己的理解,一方面也为了分享给更多人。

　　言归正传,大家都知道我们在场景中放置的物体最终渲染到屏幕上都是离不开我们的摄像机。对于透视摄像机(Perspective Projection)来说,在Unity中你会在Scene场景中看到一个白色边框的锥形体,也就是常说的视椎体,在整个渲染流程中这个视椎体是非常重要的,它会参与到透视投影矩阵的计算以及裁剪等处。

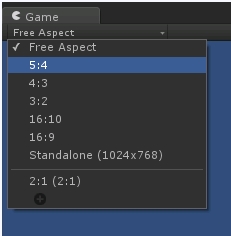
　　如下图：



　　视椎体有4个非常重要的参数,一个是Field of View,他是表示的相机在Y方向上上顶面和下底面的夹角的一半；另外一个是相机的Aspect,他代表的是相机视椎的宽高比.在最终的投影矩阵中,还有两个是近平面和远平面。这4个参数是很重要的。

　　Unity中的摄像机有个Camera组件,上面是直接可以调节FieldofView,单位是度。但你在上面是找不到Aspect属性的。可能有人会去拖动Scene中相机远平面上的四个白点去改变视椎体的形状,你可以观察一下Camera组件的变化,你发现变化的启示是Fieldofview,无论你去调整宽还是高,Unity都会按照Aspect去相应的调整高或者宽。以保证你的调整不会影响相机的Aspect.

　　那么相机的Aspect是怎么设置的呢？你当然可以再代码里用camera.aspect = x这种方式去改变他的值.但是为什么我们每次即使没有去自己写代码为aspect赋值也没感觉有什么不便呢？那是因为Unity会根据Game窗口的设置去自动设置它。(如下图)



　　可以看到Game窗口也有个Aspect的东西,但是这个Aspect可并不是指相机的Aspect,而是指最终游戏屏幕的宽高比.可以看到大体上这些可选的设置中可以分为三大类,一类是Free Aspect,一类是给出宽高比,另一类是指定分辨率.分别说明一下。

　　1.选择Free Aspect时候(默认选项)：

　　　　屏幕的宽高比实际上就是你Game窗口的宽高比,你可以手动去拖动来调节他,而这时候Unity也会实时的去修改场景中对应摄像机的Aspect值,使它和Game窗口的Aspect保持一致。你可以一遍改变Game窗口大小,一边看Scene窗口中相机视椎体的变化。

　　2.选择X:Y的时候:

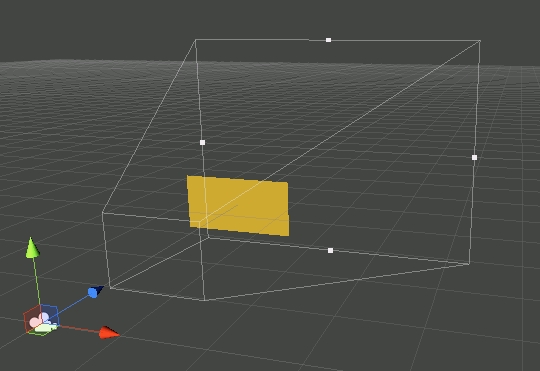
　　　　屏幕的宽高比被固定在X:Y这个大小上,而Camera的Aspect也会被设置成X:Y,这时候无论你怎么去改变Game窗口的大小,Camera的Aspect都是不会改变的.Scene中视椎体不会变。

　　3.选择指定分辨率(XXXX:YYYY)：

　　　　第二钟情况的另一种表现方式类似,Camera的Aspect也会被设置成XXXX:YYYY,同样无论你怎么改变Game窗口大小,Camera的Aspect都是不会变的。Scene中视椎体也不会变。

　　很多人也许就奇怪了为什么我无论怎么去改变Game窗口的Aspect选项,我在屏幕上看到的渲染后的画面并没有发生太大变化(比如被拉伸或者缩小).这就是因为Game的Aspect一直和Camera的Aspect保持一直所导致的。这其中涉及到投影变换,透视除法以及最终投影到屏幕空间。说起来很乱，我也怕自己说错。大家如果发现不对的地方欢迎指正,首先为了裁剪的方便,最终需要把物体的坐标从相机空间转换到一个长方体裁剪空间,这一步是通过投影变换来完成的,一般会引擎会在这一步之后进行空间裁剪(也有的会选择在透视除法之后进行裁剪)。然后再经过透视除法,对坐标进行归一,变换到所谓的规范化设备空间,坐标的大小被限定在一个[-1~1]的空间内,可以看成一个长方体被压缩成了一个正方体(也可能不是正方体,DX和OpenGL是不同的)。想要详细了解的同学不妨参看[Twinsen](http://my.csdn.net/popy007)前辈blog中相关文章。

　　那我们就可以想象,加入我在摄像机的视椎体内放置了了一个宽高比2:1的平面(如下图).在透视变换后的裁剪空间中我们看到的应该依然还是一个(2:1)的平面,但是经过透视除法的规范化之后,大家想象一下视椎体从一个2:1(相机的Aspect)的长方体被压缩成(1:1)正方体的过程.在规范化设备空间中我们的长方形平面应该是被压成了1:1的正方形平面了(实际上不是图形在变,只是顶点的坐标变了而已)。如果直接把这样的结果投影到平面显然不是我们想要的结果。这时候就需要Game窗口的Aspect出马了。



　　在透视除法把坐标归一化之后,还要在规范化设备空间上进行一次视口变换才把物体真正的映射到屏幕空间,而这一步中实际上主要是解决之前投影变换和透视除法造成的失真(就像我们的长方行被压扁了),具体做法就是把规范化后的x,y坐标按照屏幕的Aspect来进行一次调节,那么如果屏幕的Aspect和摄像机的Aspect保持一致也是2:1,我们被压扁的长方形就又被拉回到了原来的2:1了。最后再经过光栅化处理,也就得到了我们在屏幕中最终看到的结果了。

　　所以说之所以屏幕的Aspect和Camera的Aspect始终保持一致,就是为了保证透视投影的正确。那么如果我们强行让两者不一致的话,那么在透视变换和透视除法造成的图像失真就无法得到修正,也就会使我们最终渲染的屏幕的图像是错的。比如说你在Camera上挂个脚本在它的Start方法中写上camera.aspect = 2(也就是2:1).而设置Game视口的Aspect是1:1那么你会看到屏幕上是个正方形.如果设置Game视口的Aspect是1:2,你会发现原来屏幕上的长方形宽高比从原来的2:1变成了1:2了。这正验证了我们刚才所说的。

　　我在做这个例子的时候还发现了一个问题,就是说你在代码里去修改相机的Aspect之后,你会发现Scene窗口里的相机视椎体并不会同步到你设置的值.而且如果这时候去修改Game窗口的Aspect,Scene中的视椎体竟然会跟着变化,但是你在Camera刚才的脚本里Update输出camera.aspect发现还是我们刚才设置的值。并且从渲染效果上观察也确实使用的是我们设置的值。所以我觉得一旦你在代码里设置了camera.aspect之后,Scene窗口中的相机视椎体不会同步到新值,而且也失去了参考价值。不知道这是不是Unity的一个Bug.或者也许这个白色椎体并不是代表的视椎体而是有着其它的含义？如果有知道的同学,请一定告诉我。

　　在最后还有两个地方需要提及：

　　1.当我们把摄像机的内容渲染到的是RenderTexture上而不是屏幕上时,那么相机的Aspect默认会设置成和RenderTexture的分辨率一样.不过最终如果把RenderTexture作为贴图贴到模型上去的时候还是会被由于被UV拉伸和缩小的。

　　2.对于Camera组件的的Viewport Rect属性也就是所谓的视口,他会影响实际上最终渲染的屏幕窗口大小,最终渲染窗口的Aspect实际上是由Game窗口的Aspect和Viewport Rect的Aspect相乘得到的结果。这点要注意。