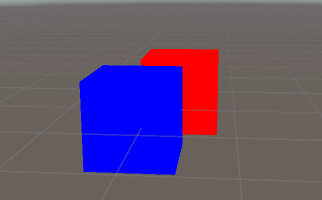
**[Unity ZTest 深度测试 & ZWrite 深度写入](http://www.cnblogs.com/ljx12138/p/5341381.html)**

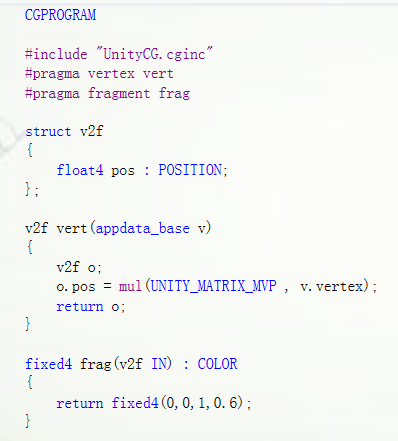
    初学Shader，一开始对于渲染队列，ZTest 和 ZWrite一头雾水，经过多方查阅和实验，有了一些自己的理解。发此文与初学Shader的朋友分享，也算是为自己做个笔记。不对或不足之处欢迎指正。

    不说废话，直接进入正题。

    首先是实验场景，一蓝一红两个Cube。蓝在红前。

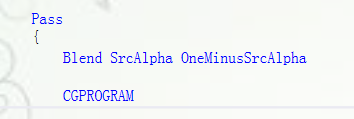


    两个方块所使用的Shader都是最简单的 V&F 着色程序，不同的是蓝色方块alpha返回值为0.6，红色为1。



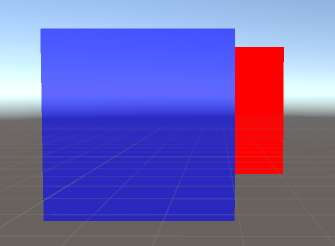
     但是蓝色方块并没有丝毫透明的效果，这时我们在蓝色方块的Shader内加上这样一行代码 :

     Blend SrcAlpha OneMinusSrcAlpha 。



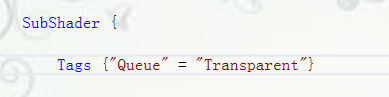
     稍微解释一下，这行指令意思就是将本 Shader 计算出的颜色值(源颜色值，即蓝色) \* 源Alpha值(0.6) + 目标颜色值(可以理解为背景色) \* (1-0.6)，从而让蓝色方块展示出了40%的透明度。

    然后我们看到了这样的效果：



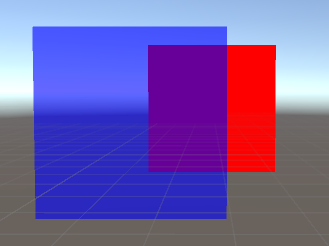
    似乎有些透明的样子了，但是红色方块还是显示不出来。我们再加上这样一行代码：

     Tags {"Queue" = "Transparent"}



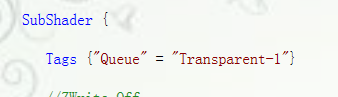
    意思是设置它在渲染队列中的值为 Transparent (透明) = 3000，值越小越先渲染，而后渲染( Queue 值大)的物体会覆盖先渲染的物体（红块未设置 Queue 值，默认为 Geometry(几何体) = 2000）。在理想的世界中，我们应该让远处的物体先渲染，近处的物体后渲染，这样远处的物体就不会遮挡住近的物体。

    接下来我们看到了正确的结果：

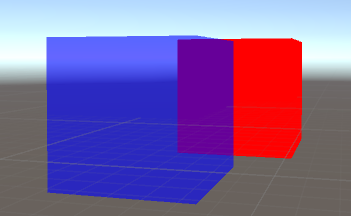


    Queue 其他预定义的值为：Background = 1000 , AlphaTest = 2450，Overlay = 4000。默认值是Geometry 。

    例如：如果我们想让远处的红色方块遮挡住近处的蓝色方块，即让蓝色的先渲染，红色的后渲染，只需将红块的Queue 也设置为 Transparent ，蓝块的Queue值-1。

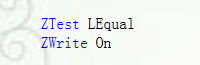


    然后我们看看效果：



    好像并没有什么卵用。。。这是为什么呢？

    原因在于这样两条指令：



    虽然我们的代码里并没有这两行，但它们是默认存在的。

     ZTest ，深度测试；LEqual ，小于等于。

     ZWrite ，深度写入，On ，打开。

     ZTest 可取值为：Greater , GEqual , Less , LEqual , Equal , NotEqual , Always , Never , Off，默认是 LEqual，ZTest Off 等同于 ZTest Always。

     ZWrite 可取值为：On , Off，默认是 On。

    系统中存在一个颜色缓冲区和一个深度缓冲区，分别存储颜色值和深度值，来决定画面上应该显示什么颜色。

    深度值是物体在世界空间中距离摄像机的远近。距离越近，深度值越小；距离越远，深度值越大。

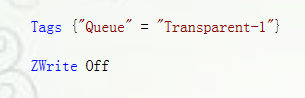
    例如在我们的场景中，蓝色方块比红色更靠近相机，蓝块的深度值就比红块小。

    假设蓝块的深度值为 0.5，红块为 0.7。还记得在上面的例子中，我们让蓝块在渲染队列中排在红块前面，系统就先将蓝色值存入了颜色缓冲区中对应的区域，将深度值 0.5 存入了深度缓冲区中对应的区域。接下来渲染红块，系统会将红块的深度值与深度缓冲区中的值进行比较（这个过程就是深度测试），由于默认的 ZTest 深度测试的方式是 LEqual 小于等于，即深度值小于等于 0.5 的颜色才会通过测试。如果通过了测试，且 ZWirte 处于 On 的状态，该颜色的深度值就会替代深度缓冲区中的值，颜色值也会替代颜色缓冲区中的值，从而显示出新颜色。

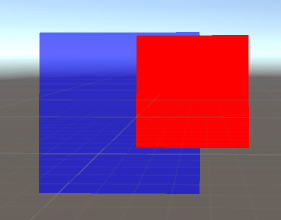
    很显然，0.7 > 0.5，所以红色并不能通过测试，红块也就不能显示在蓝块前面。

    如果我们硬要远处的红块遮挡住近处的蓝块，很显然，我们应该改变或关闭深度测试，或者关闭深度写入（关闭了深度测试或者深度写入之后，物体颜色的遮挡关系就会和渲染队列一致，即排在后面的会挡住前面的）。

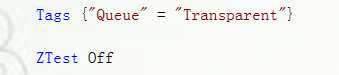
    接下来我们试试关闭蓝块的深度写入：

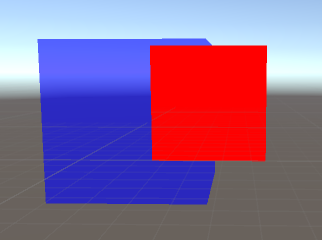


    得到了想要的结果：

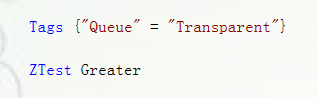


    再试试单独关闭红块的深度测试，注释蓝块的深度写入：

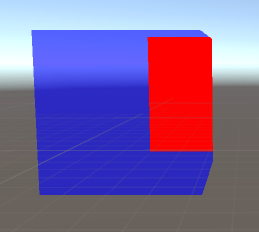




    结果还是一样。但如果我们改变红块深度测试的方式呢：



    即深度值大于深度缓冲区中的值就能通过测试，还记得我们假设红块深度值为 0.7，蓝块为 0.5。理论上会得到我们想要的结果：



    奇怪的是，红块的另一半去哪了？

    答案是被背景挡住了。

    按照距离相机的远近，我们可以假设背景的深度值为 1。在消失的另一半的深度缓冲区中的深度值应该是背景的深度值 1。而我们设置了 ZTest Greater，0.7 < 1，所以红块另一半没有通过深度测试。