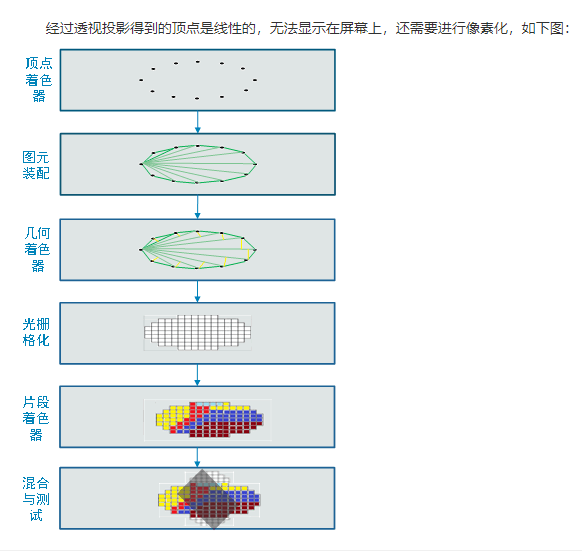
[**编写shader之前必须的基本认识**](https://www.cnblogs.com/chenggg/p/11204708.html)

***Vertex shader*** (顶点着色器):通过编程的方式实现对顶点的操作,如空间转换,颜色,纹理坐标,灯光 得到最终的顶点数据后,输出到片元着色器

**顶点着色器可以用来指定一系列完全通用的(如坐标,灯光,颜色,纹理)、并将应用于各个顶点及相关数据上的操作。**

***Fragment shader***(片元着色器):在像素没显示到屏幕之前,将要执行一些列操作,这些操作可能会修改甚至丢弃这些片断。所有这些操作都可以被启用或禁用。第一个可能执行的操作时纹理处理。在纹理内存中为每个片断生成一个纹理单元(texel，也就是纹理元素,也就是uv)，并应用到这个片断上。接着可能进行的是雾计算，然后是剪裁测试，alpha测试，模板测试和深度缓冲区测试(深度缓冲区用于消除被隐藏的表面)。如果一个片断无法通过一个启用的测试，它的连续处理过程可能会被中断。随后，将要执行的可能是混合，抖动，逻辑操作以及根据一个位掩码的屏蔽操作。最后，经过完整处理的片断就被绘制到适当的缓冲区，最终成为一个像素并到达它的最终栖息地。

物体的坐标抓换:本地坐标(原点为自身轴向为自身)->惯性坐标-(原点为自身轴向与世界坐标系重合)->世界坐标(原点与世界坐标重合 然后旋转到本地的轴向)->相机坐标(我个人比较喜欢叫相机坐标,书面化说法又称视锥体坐标,其实就是相机的视角是个四棱锥或长方体)->在相机视角中对三维的坐标进行降维操作(<https://www.bilibili.com/video/av37614073?from=search&seid=2402871598313506313>   1:09了解一下)->



**接下来就介绍下Shader的三大测试:**

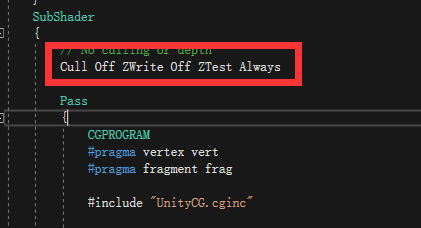
**深度测试:**

　　深度:指该像素距离摄像机的距离。离摄像机越远，深度越大。

 　  深度缓冲:存储着每个像素对应的深度值。

       颜色缓冲:存储着每个像素对应的颜色值。

       深度测试:将像素的深度与深度缓冲的深度进行比较。如果通过了深度测试，就有机会去改写深度缓冲的对应值；否则该像素会被舍弃



　　 Zwrite

  　   参数 On/Off    开启/关闭深度写入

　　 ZText

 　　Greater  GEqual  Equal Less LEqual  NotEqual  Always  Never

　　　　　　　　　　　　分别对应

　　   >              >=       ==       <        <=         !=            总是     从不

　　 像素对比GBuffer中的像素，满足条件则替换。否则舍弃，物体的像素渲染并不是直接就到屏幕上，需要计算，符合条件则通过测试，最终到屏幕上。

**Alpha测试:**

　　Alpha Test ,中文就是透明度测试。简而言之就是V&F shader中最后fragment函数输出的该点颜色值（即上一讲frag的输出half4）的alpha值与固定值进行比较。 AlphaTest语句通常位于Pass{}中的起始位置。

**语法：**

　　第一种: AlphaTest Off： 不测试，全渲染

　　第二种：Alpha 比较符 目标alpha值

　　其中目标alpha值取值范围是 0至1， 也可以用参数 ，如 AlphaTest [varName]。

**比较符:**(目标alpha值下面记作x)

　　Always  全渲染（任意x值）

　　Never   全不渲染

　　Greater  点的alpha值大于x时渲染

　　GEuqal   点的alpha值大于等于x时渲染

　　Less       点的alpha值小于x时渲染

　　LEqual    点的alpha值小于等于x时渲染

　　Equal  点的alpha值等于x时渲染

　　NotEqual  点的alpha值不等于x时渲染

　　例：

　　AlphaTest never  0      全不渲染（这个0可以是任意值，但必须填）

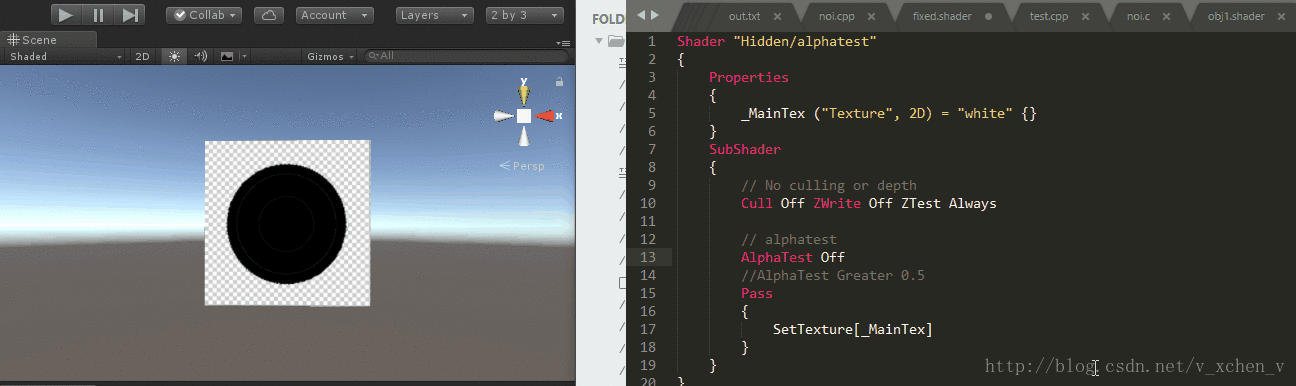
　　AlphaTest always 0      全渲染（这个0可以是任意值，但必须填）

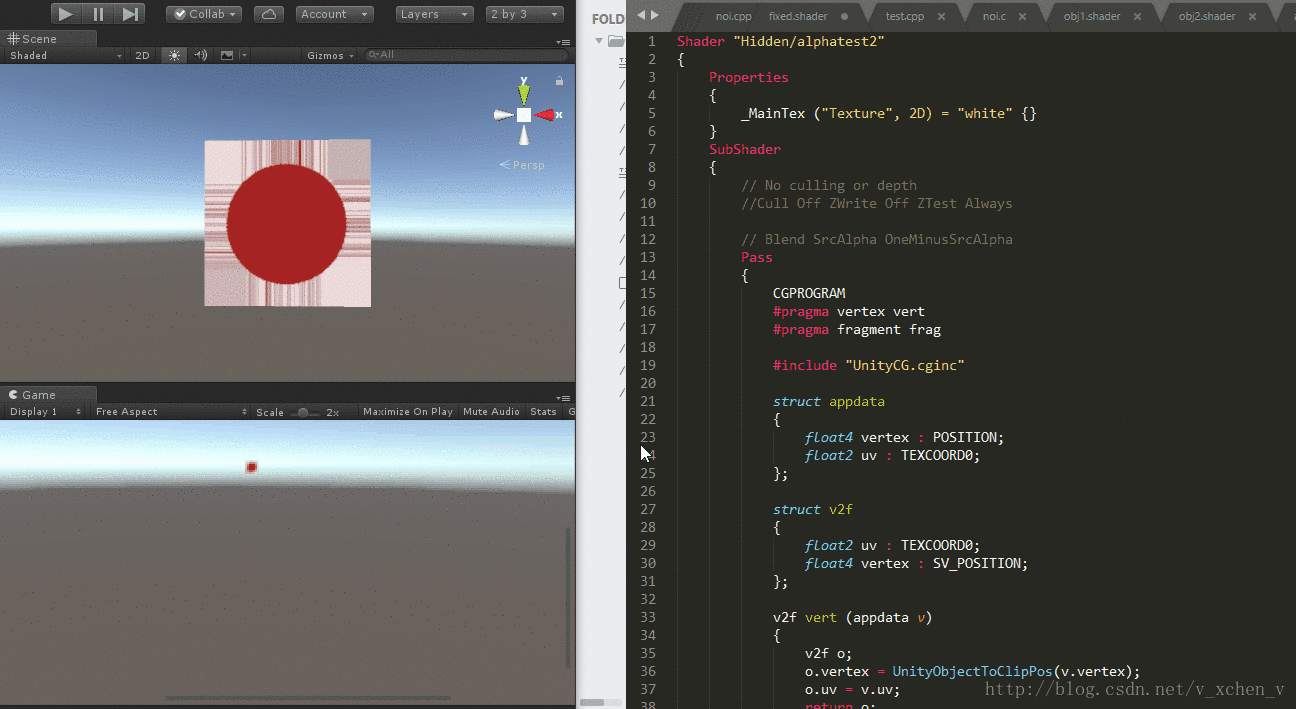
　　AlphaTest off               全渲染

　　AlphaTest Greater 0.4  当alpha大于0.4渲染，   缺点是边缘过于锋利并且显示不完整，alpha小于0.4的部分被忽略掉了

　　Alpha Blend                 边缘柔和，但是中心也太柔和，没有实体的感觉

　　AlphaTest Greater 0.4 / AlphaTest LEqual 0.4 两个部分，大于0.4不混合，小于0.4混合。效果是既有实体感觉，边缘也柔和。





**模板测试:**

　　模版测试较为复杂，它不像Alpha测试一样，单单通过简单的大于小于等运算符来比较，但是和Alpha测试一样，都是测试像素点，符合要求的就渲染，不符合要求的就不渲染。了解模版测试，需要理解较多术语:

　　1.GBuffer：几何缓冲,相当于一个容器

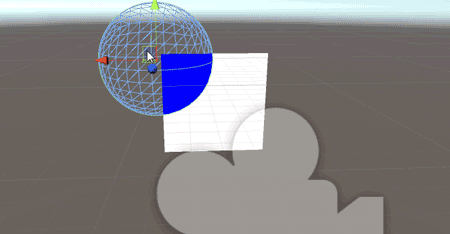
　　2.RefenceValue：模版值，Guffer中的像素点就是与这个值按位与后进行比较

　　3.stencilOperation：比较后进行的操作

　　4.ReadMask|WriteMask：读遮罩|写遮罩

　　理解了以上的术语，那么模版测试的公式就差不多可以理解了：其公式为:

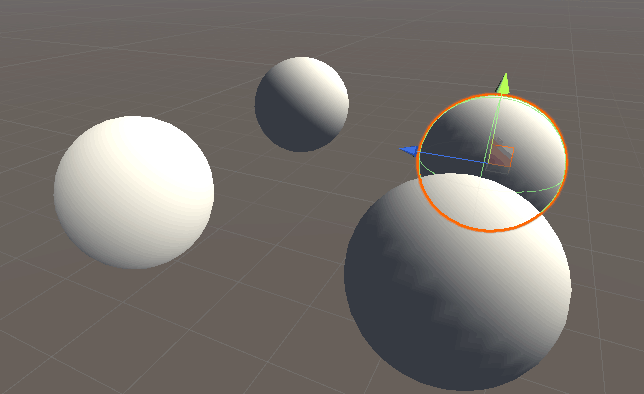
　　referenceValue&readMask comparisonFunction stencilBufferValue&readMask



转:<https://blog.csdn.net/coolbeliever/article/details/81979089>

**关于逐顶点和逐像素:**

　　per-vertex是先计算再插值，per-pixel是先插值后计算。后者的控制粒度是pixel(像素)，前者是triangle(三角形)。



左边逐个像素 右边逐个顶点 可以明显的看到分割处的三角面