有些知识是被遗漏的，一知半解的，一下是新的发现：

深度写入和深度测试是不一样的。

不透明的物体既开启深度写入，又开启深度测试。

半透明的物体开启深度测试，关闭深度写入。

什么是深度写入？

当渲染某一物体时，把该物体的深度写入深度缓冲，后来的物体如果离摄像机更近，则更新深度缓冲，并把颜色直接写入颜色缓冲。

什么是深度检测？

渲染物体时，自身的深度与深度缓冲中的数据进行对比，如果自身的深度大，则抛弃。

为什么半透明的物体要关闭深度写入？

例如： 相机 半透明A 不透明B

正常现象，我们应透过A看到B。

如果A开启深度写入：

先渲染B，再渲染A，A的深度小于B，直接写入，颜色直接替换，于是看不到B。

先渲染A，再渲染B，B的深度大于A，所以B不渲染。

如果A关闭深度写入：

渲染顺序BA，A开启深度检测，发现比B近，于是A的颜色和B进行混合写入颜色缓冲，深度缓冲不变。

渲染顺序AB，渲染B时，发现深度缓冲没被写入，所以写深度缓冲，替换颜色缓冲，因此没有了A的颜色。

深度缓冲的写入，必然带着颜色缓冲的替换

问什么开启深度检测呢？

例如：相机 不透明A 半透明B 不透明C

透明物体的渲染规则，先渲染不透明，再渲染半透明。也就是先渲染AC，再渲染B，当渲染B时，进行深度测试时，发现B没有通过，因为A比B更靠近相机，所以B的混合颜色被舍弃了，于是我们才能正常的看到A而看不到B。

透明度测试：

Tags {

“Queue”=”AlphaTest”

“IgnoreProjector”=”True”

“RenderType”=”TransparentCutout”

}

透明度混合

Tags {

“Queue”=” Transparent”

“IgnoreProjector”=”True”

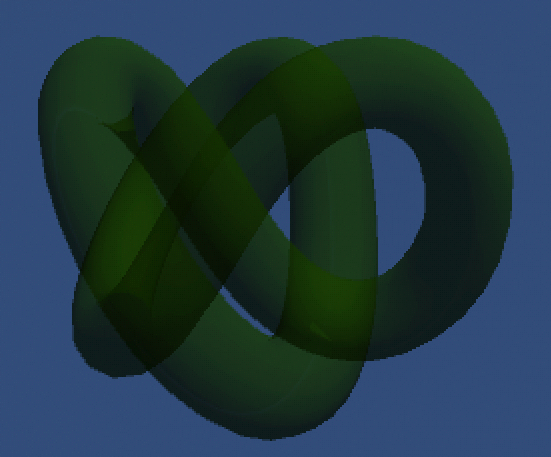
“RenderType”=”Transparent”

}

处理透明度的基本方法：

1 先渲染不透明的物体，开启深度检测与深度写入

2 将半透明的物体进行排序，距离摄像机远的先渲染，开启深度检测，关闭深度写入。

透明度遇到的问题一：物体本身是相互遮挡的，透明度渲染会有问题。

例如

后边的也显示出来了，如果想屏蔽掉后边的内容。可以对自身开启深度写入，但不屏蔽所有颜色。

添加Pass

Pass

{

ZWrite On

ColorMask 0

}

透明度遇到的问题二：物体是透明的，可能需要看到物体内部东西。

看不见后边是因为 背面剔除了，所以我们需要先渲染背面，再渲染正面。

Pass

{

Tags {"LightModel"="Forwardbase"}

Blend SrcAlpha OneMinusSrcAlpha

ZWrite Off

Cull Front

CGPROGRAM

#pragma vertex vert

#pragma fragment frag

ENDCG

}

Pass

{

Tags{ "LightModel" = "Forwardbase" }

Blend SrcAlpha OneMinusSrcAlpha

ZWrite Off

Cull Back

CGPROGRAM

#pragma vertex vert

#pragma fragment frag

ENDCG

}

渲染顺序，由远及近。

如果不想看见自身的背面，开启深度写入。

如果想看见自身的背面，先渲染背面，再渲染正面。