shader

深度： 物体垂直与摄像机视角正前方的垂直点到摄像机视角的距离。
什么是深度测试：将要渲染的物体的z值对比gbuffer里面的值。
ZWrite on/off
on 将 要渲染的物体深度写入深度缓存区。
off 不将 要渲染的物体深度写入深度缓存区。

ZTest
ZTest Less Greater LEqual GEqual Equal NotEqual Always
Less：将要渲染物体的深度值小于gbuffer的深度值
Always ： 总是通过

当ZWrite为On时，ZTest 通过时，该像素的深度才能成功写入深度缓存 否则，都不能通过。

1.当ZWrite为On时，ZTest 通过时，该像素的深度才能成功写入深度缓存，同时因为ZTest通过了，该像素的颜色值也会写入颜色缓存。.
2.当ZWrite.为On时，ZTest 不通过时，该像素的深度不能成功写入深度缓存，同时因为ZTest不通过，该像素的颜色值不会写入颜色缓存。.
3.当ZWrite为Off时，ZTest 通过时，该像素的深度不能成功写入深度缓存，同时因为ZTest通过了，该像素的颜色值会写入颜色缓存。。
4.当ZWrite为Off时，ZTest 不通过时，该像素的深度不能成功写入深度缓存，同时因为ZTest不通过，该像素的颜色值不会写入颜色缓存。。

Offset Factor , Units.
此语句用两个参数(Facto 和Units)来定义深度偏移。。
●Factor 参数表示Z缩放的最大斜率的值。值越 小越靠前。
●Units 参数表示可分辨的最小深度缓冲区的值。微调 buffer 值值越小越近相机。
使用条件:
两个物体在同-一个位置
做微调

Blend 要渲染的像素*factorA* *屏幕已经渲染的gbuffer里面的像素*factorB

factorA ：
one：1，Zero：0 ， SrcColor，要选人的像素， SrcAlpha：要渲染的阿尔法通道
DstColor：已经渲染在gbuffer里面的像素 ， DstAlpha：已经渲染在gbuffer里面的像素 α通道
OneMinusSrcColor：1-将要渲染的像素
OneMinusSrcAlpha：1-将要渲染的像素的α的通道
OneMinusDstColor：1-将要渲染在gbuffer的像素
OneMinusDstAlpha：1-将要渲染在gbuffer的像素的α通道

Blend SrcAlpha OneMinusScrAlpha

将要渲染的像素*将要渲染像素的α+（1-将要渲染的α）*gbuffer里面的像素值
BlendOp 指定将要渲染的像素和gbuffer像素进行逻辑运算

指定物体的渲染顺序:。
Tags { "Queue"=:“XXXX"}.

```
Background: 对应数值为1000 ,用于需要被最先渲染的对象
```

10
Geometry :对应数值为2000, 用于不透明的物体。这个是默认的选项(如果不指明Queue标签的值,自动给你指定为Geometry)。。（从前往后）
AlphaTest :对应的数值为2450,用于需要使用AlphaTest 的对象来提高性能。AlphaTest类似于裁剪(clip) 功能。
Transparent :对应的数值为3000 ， 用于需要使用alpha blending的对象, 比如粒子，玻璃等。。（从后往前）

Overlay :对应的数值为4000 ,用于最后被渲染的对象

Vertex Lit Rendering Path
灯光作用，只在顶点着色器作用 物体只会渲染一次
FotwardPath
找一个逐像素灯光和逐顶点和sh灯光渲染一次
逐像素灯光 1 directional 方向光 2 置成import的灯光

shader2.0对property变量需要引用
eg: float _Frenquncy;
float _Aranger;
float _Speed;

gudaizhangzheng

_Time 表示时间周期

颜色和vector

.xyz 或者 rgba

Clamp：表示UV坐标超过1，永远取1这个地方的位置，比0取的都是0这个地方的像素。
Repeat：表示UV坐标超过1，超过的地方从0开始取。（相当于图片的循环使用）

Shader1.0
设置纹理格式:
SetTexture [TextureName] {Texture Block}

TextureName: 表示纹理变量
Previous: 表示前面一个SetTexture出来以后的像素
Primary: 表示顶点计算出来的颜色
Texture: 等于SetTexture当前纹理变量
Constant: 示一个固定颜色如(1, 1, 1, 1)
Combine Primary* Texture 表示两个像素的乘法或加，亮度越乘越暗，越加越亮，因为像素值在0到1之间
Combine src1 lerp(src2) src3
插值运算Lerp:
让src1和src3进行混合
混合方式取决于src2的alpha值: (1-alpha)* src1+alpha* src2

Shader2.0
1.**计算顶点的位置变换**
Unity里面矩阵左乘
①将物体坐标系变换到世界坐标系
P(世界)= M (物体到世界的矩阵) *P (物体)
规律: 3D变换首先将物体坐标系变换到世界
②将世界坐标变换到相机坐标系
P(相机)= M (世界到相机的矩阵) *P (世界)

M:物体坐标系变换到世界坐标系
V:世界坐标系变换到相机坐标系
P:将3D坐标系转换成2D屏幕坐标系，包括正交投影与透视投影

Vertex :定义顶点着色器的入口函数.
Surf : Surface 入 口 可以看成片段着色器。
Lambert:
定义灯光作用的入口函数。
Finalcolor： 计算物 最后一次计算像素的入口函数。。

```
Shader "Hidden/NewImageEffectShader"
{
Properties//属性
{
  _MainTex ("Texture", 2D) = "white" {}
}
SubShader
```

{
// No culling or depth
Cull Off ZWrite Off ZTest Always

```
Pass
{
    CGPROGRAM
    #pragma vertex vert              //定义一个顶点着色器的入口函数 相当与main
    #pragma fragment frag            //定义一个片段着色器的入口函数 相当与main

    #include "UnityCG.cginc"

    struct appdata
    {
        float4 vertex : POSITION;          //POSITION;获取模型顶点的信息  NORMOL 获取模型的法线信息
        float2 uv : TEXCOORD0;          //TEXCOORD（n）可以高精度的传递多个值从定点着色器到片段着色器。。float1,float2orfloat4;
    };

    // COLOR表示低精度的 从定点着色器到片段着色器。float4;
    //TANGENT  获取切线信息。
    struct v2f
    {
        float2 uv : TEXCOORD0;
        float4 vertex : SV_POSITION;                //SV_POSITION 表示经过mvp矩阵转移到屏幕坐标位置
    };

    v2f vert (appdata v)                        //appdata值从meshrender里面来的  ' v2f顶点着色器的输出值，片段着色器的输入值
    {
        v2f o;
        o.vertex = UnityObjectToClipPos(v.vertex);
        o.uv = v.uv;
        return o;
    }



    //计算
    sampler2D _MainTex;
     //片段着色器的入口函数
    fixed4 frag (v2f i) : SV_Target                     //fixed4屏幕上的颜色输出  SV_Target输出到那个render target 上
    {
        fixed4 col = tex2D(_MainTex, i.uv);             //tex2D(_MainTex, i.uv);        tex2D:  表示一张图片  i.uv:   表示比例  UV比例范围是（0，1）
```

```
39            // just invert the colors
40            col.rgb = 1 - col.rgb;
41            return col;                //返回值就是当前要显示的区域
42        }
43        ENDCG
44    }
45 }
```

}