**>>Shader1.0**

##大部分关键字不区分大小写

1. **名称**

Name”名称” //放于Pass上以起名

UsePass”路径/名称” //引用Pass

1. **混合透明**

Tags{"Queue"="Transparent"} //半透明通道标签

BlendOp [] () //混合(生成颜色(Alpha)[运算]屏幕颜色(Alpha))

Blend [] [] //混合([]\*生成颜色+[]\*屏幕颜色)

OneMinus...... //一减去

Src...... //此阶段

Dst...... //缓冲区

+Alpha //透明度

+Color //颜色

1. **传统照明**

Color {值}/[变量] //设置纯色

Lighting //照明系统开关

SeparateSpecular //高光镜射开关

Material //材质设置

{

Diffuse //漫反射

Ambient //环境光

Specular //高光镜射

Shininess //亮度

Emission //自发光

}

1. **纹理组合**

SetTexture[纹理变量\*] //纹理设置

{

Combine //结合

texture //当前处理块纹理颜色

Previous //前一个处理块纹理颜色

Primary //光照颜色

ConstantColor //定义颜色常量

Constant //颜色常量

}

+Double //\*2

+Quad //\*4

1. **深度测试**

Cull //表面绘制模式

ZWrite //深度写入开关

ZTEST(比较) //深度写入设置

Offset //重叠时的修正

1. **Alpha测试(2.0不可用)**

AlphaTest （比较） [数值] //比较Alpha数值决定运行

Greater //大于

Gequal //大于等于

Equal //等于

Lequal //小于等于

Less //小于

NotEqual //不等于

1. **模板测试**

Stencil //公式

{

Ref //测试值

(ReadMask)

(WriteMask)

Comp //比较

Pass //通过

Fail //失败

Zfail //深度测试失败

}

Replace //将参考值写入缓冲区

**>> Shader2.0**

CGPROGRAM…ENDCG //编写程序块

Sampler2D //图片变量声明(所有变量都需声明引用)

#pragma vertex XXX //引入定点着色器入口函数

#pragma fragment XXX //引入片段着色器入口函数

**输入**

POSITION //模型空间顶点坐标

TEXCOORDO //纹理空间顶点坐标

**输出（剪辑空间）**

SV\_POSITION //顶点坐标

SV\_Target //片段渲染

tex2D (纹理,坐标) //纹理采样

\_Time //相当于Time.deltaTime

**顶点波动公式：**

Y += 波动幅度 \* sin(\_Time\* 波动频率 + X(Y) \* 波动曲线);

**UV旋转公式：**

float sleep = \_Time \* \_float; //旋转速度

uv.xy -= float2(0.5, 0.5); //UV平移至中心的

if(length(uv)>0.5{return float4(0,0,0,0) //判断与中心距离(长度不应超过半径)

float2 olduv = uv; //记录原点UV位置

uv.x = olduv.x\*cos(sleep) + olduv.y\*sin(sleep); //新的x坐标

uv.y = olduv.y\*cos(sleep) - olduv.x\*sin(sleep); //新的y坐标

uv.xy += float2(0.5, 0.5); //新UV平移至原点

**>>Surface Shader**

##直接放在SubShader块，而不是Pass

#pragma target 3.0 声明为表面着色器

#pragma surface (surf) (Lighting) [vertex:xxx] [finalcolor:xxx]

Vertex 顶点着色器

void vert(inout appdata\_full o)

Surf 片段着色器

void surf (Input IN, inout SurfaceOutput[Standard] o)

Lighting 灯光着色器(表面着色器)

half4 Lightingb(SurfaceOutput o,half3 lightDir,half3 viewDir, half atten)

Lambert 漫射照明(内置)

BlinnPhong 镜面照明(内置)

Finalcolor 雾，最后一次计算像素

void final (Input IN, SurfaceOutput o, inout fixed4 color)

**输入：**

struct Input //输入结构

{

float2 uv\_贴图名称 //贴图坐标

float4 screenPos //屏幕坐标

half3 lightDir //入射光角度

half3 viewDir //相机角度

float3 worldRefl //反射向量

half atten //衰减

\_LightColor0.rgb //灯光颜色

}

inout SurfaceOutput o //片段着色器输入(输出)

inout appdata\_full v //顶点着色器输入(输出)

half4 Lighting+名称(…) //表面着色器

**公式：**

half dot(lightDir,s.Normal) //点积

UnpackNormal(tex2D(\_NorTex,IN.uv\_NorTex)) //RGB转XYZ

clamp(dot(s.Normal,viewDir),0,1) //限制取值范围

max (0, dot (o.Normal, gv)) //限制最小值

min (1, dot (hpos.xy, hpos.xy)\*0.5) //限制最大值

normalize (lightDir + viewDir) //向量单位化

frac((IN.worldPos.y+IN.worldPos.z\*0.1) //截取小数位

clip() //低于0不渲染

WorldReflectionVector (IN, o.Normal) //计算法线后反射

UnityObjectToClipPos(v.vertex) //从模型空间转换为屏幕空间

UnityObjectToViewPos(v.vertex) //相机的向量

**属性：**

INTERNAL\_DATA

//计算法线后的反射需加，放于Input结构中

UNITY\_INITIALIZE\_OUTPUT(结构名,参数名);

//实例化结构，以用来为自定义变量赋值,需配合out返回参数