**Unity内置及常用Shader整理**

# 一、Shader脚本结构:

Shader "Custom/zhezhao1"

{

Properties

{

}

SubShader

{

Tags {"Queue"="Transparent"}

Pass

{

}

}

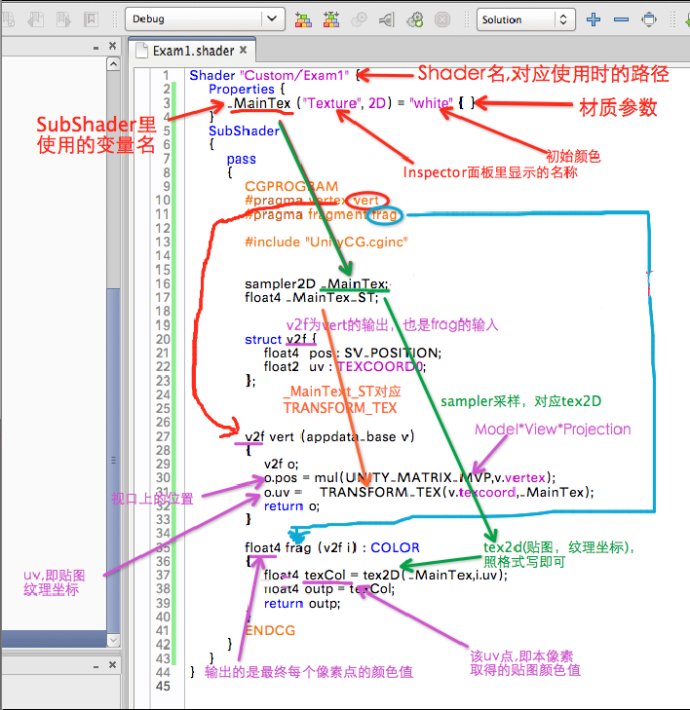
SubShader

{

//ENDCG

}

Fallback "Diffuse"

}

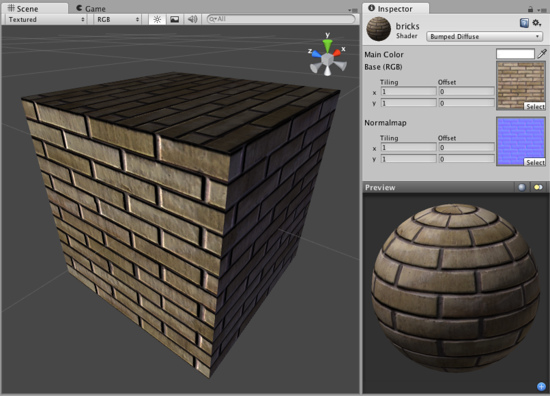
# 二、Unity自带的Shader描述

## 1、标准着色器系列（Normal Shader Family）

### （1）、凹凸漫反射（Bumped Diffuse）

**属性：**

NormalMap（法线贴图）属性，不改变对象形状，使用纹理来模拟较小表面的细节，而不是使用更多的多边形来实际雕刻细节。

漫反射属性：表面光照强度随表面与光之间夹角的减小而减小。光照强度仅取决于此角，且不随相机的移动或旋转而变化。

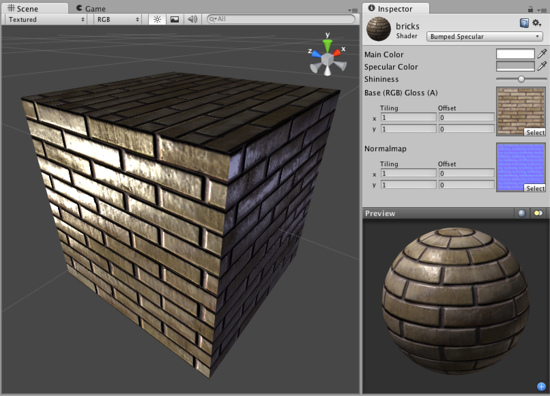
### （2）、凹凸高光 (Bumped Specular)

**属性**：

1、法线贴图属性(Normal Map)

2、高光属性，与漫反射 (Diffuse) 一样，高光也计算简单的 (Lambertian) 光照，并附加一个依赖于观察者的高光亮点。这个称为 Blinn-Phong 光照模型。其具有取决于面角、光角和观察角的高光亮点。该亮点实际上只是对光源模糊反射的实时适宜模拟。亮点的模糊层次由检视器 (Inspector) 中的反光 (Shininess) 滑块控制。

此外，主纹理的 alpha 通道用作高光贴图 (Specular Map)（有时称为“光泽贴图”），用于定义对象哪些区域的反射性更强。Alpha 的黑色区域是零高光反射区域，而白色区域是全高光反射区域。

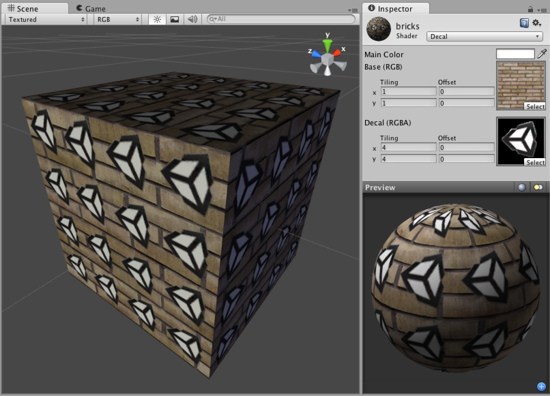


### （3）、印花 (Decal)

属性：

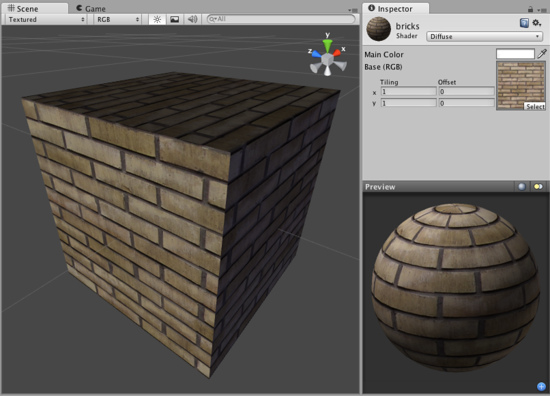
1、印花属性，该着色器是顶点光照 (VertexLit) 着色器的变型。该着色器上的所有光照都将被其渲染为顶点光照。除主纹理之外，该着色器还将第二个纹理用于其它细节。第二个“印花”(Decal) 纹理使用 alpha 通道来确定主纹理的可见区域。印花纹理应为主纹理的补充。

比如说你有一个砖砌的墙壁，你可以使用一个砖块的纹理作为主纹理，然后使用带有 alpha 通道的 Decal 纹理在墙壁的不同地方涂鸦。



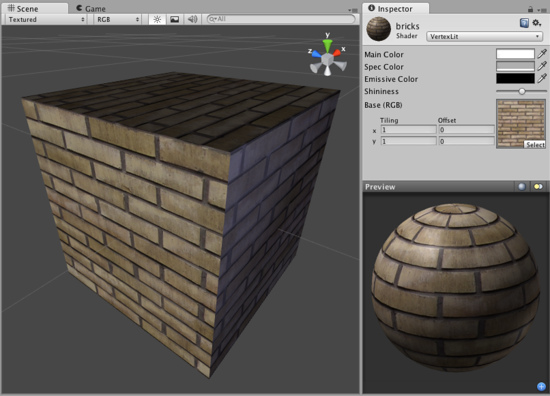
### （4）、漫反射（Diffuse）

Diffuse 基于一个简单的光照模型-Lambertian，光照强度随着物体表面和光入射角夹角的减小而减小(即光垂直于表面时强度最大)。光照的强度只和该角度有关系，和摄像机无关。

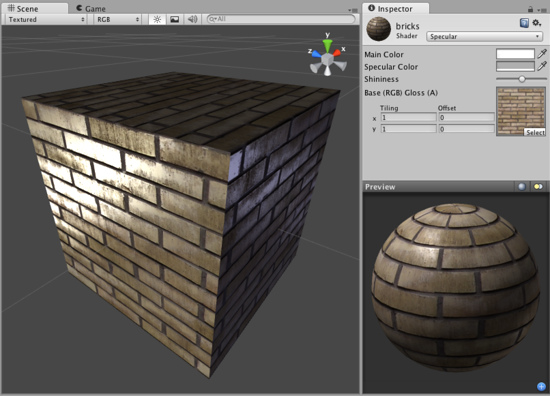


### （5）、顶点光照（Vertex-Lit）

属性：最简单的一种 Shader 之一，所有照射在该物体上的光在一个 Pass 里面渲染完，光源只在顶点计算。所以不会有任何基于像素渲染得效果，比如说：normal,light cookies 和 shadows.这个shader 对模型的剖分（将一个物体从几何描述变为多边形表示的过程）非常敏感，如果你将一个点光源放在很靠近一个立方体的一个顶点那里，并且对立方体使用这个 shader，光源只会在角落计算。所以，基于像素光照的 shader 对剖分没有要求，在表现圆形高光上效果也很好。



### （6）、高光（specular）

属性：与漫反射 (Diffuse) 一样，高光也计算简单的 (Lambertian) 光照，并附加一个依赖于观察者的高光亮点。这个称为 Blinn-Phong 光照模型。其具有取决于面角、光角和观察角的高光亮点。该亮点实际上只是对光源模糊反射的实时适宜模拟。亮点的模糊层次由检视器 (Inspector) 中的反光 (Shininess) 滑块控制。

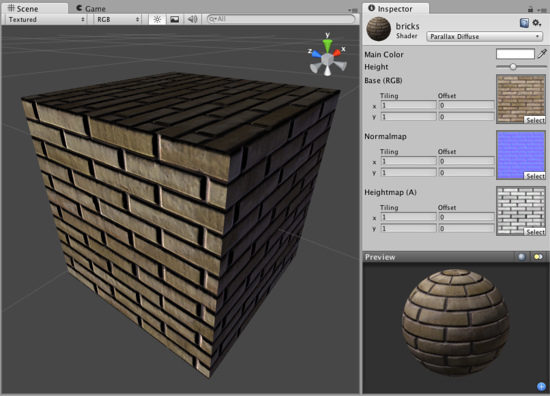
### （7）、视差法线贴图（Parallax Diffuse）

属性：

A、视差法线贴图属性，与传统的 normal mapped 一样，但是对“深度”的模拟更佳。额外的深度效果是通过 Height Map（高度图）来实现的。Height Map 在 Normal map 的alpha 通道里面保存。全黑表示么有高度，而白色表示有高度。通常这用来表现石头或者砖块间的裂缝。

B、漫反射属性

此着色器消耗的渲染资源较多。



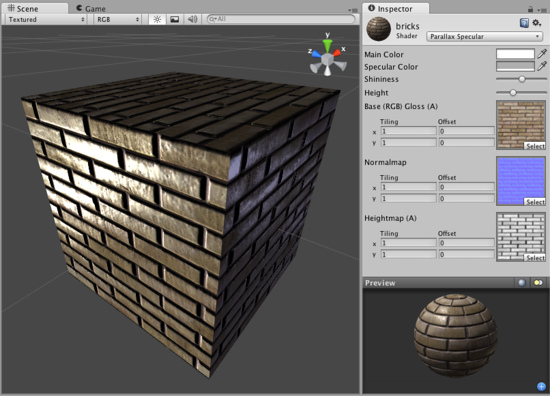
### （8）、视差凹凸高光（Parallax Specular）

属性：增加一张heightmap描述深度细节，此着色器消耗的渲染资源较多

A、视差法线贴图属性

B、高光属性

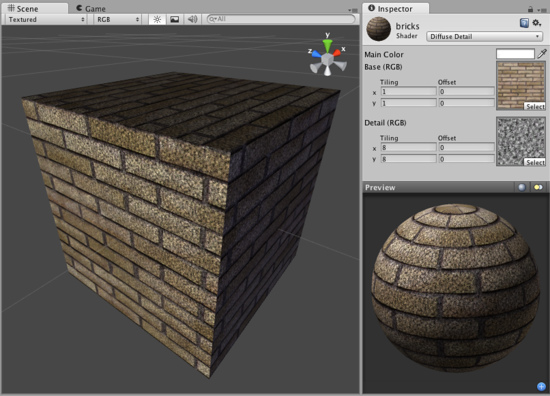
C、漫反射属性



### （9）、漫反射细节（Diffuse Detail）

属性：

漫反射细节属性：该着色器是带有额外数据的常规漫反射 (Diffuse) 着色器的版本。它使您能够定义第二个“细节”纹理，这种纹理会随着相机的靠近而逐渐显现。例如可以用于地形。您可以使用低分辨率的基础纹理来覆盖整个地形。当相机靠近低分辨率纹理时，纹理会变得模糊不清，这并不是我们想要的。为了避免这种效果，请创建平铺在地形上的通用细节 (Detail) 纹理。通过这种方法，额外的细节会在相机靠近时出现，从而避免模糊效果。细节 (Detail) 纹理放在基础纹理的“上面”。



## 2、透明着色器系列

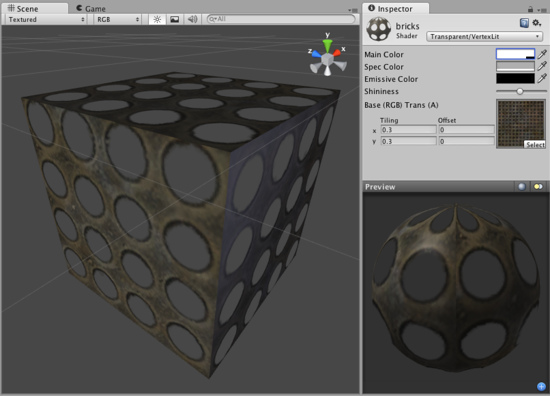
这个家族的 Shader 一共 7 种，原理和类型与 Normal 中的上差不多，只不过这些 Shader是用在半透明或者全透明的物体上面的。他们的主纹理接受 RGBA4 个通道。如果你的模型一部分是半透明，一部分是不透明的。请分开使用两张材质，半透明的材质这个家族的 Shader。这个家族的 Shader的内容和 Normal 中同名的几乎一样，只是添加了半透明效果。

### （1）、透明顶点光照（transparent VertexLit）

属性：

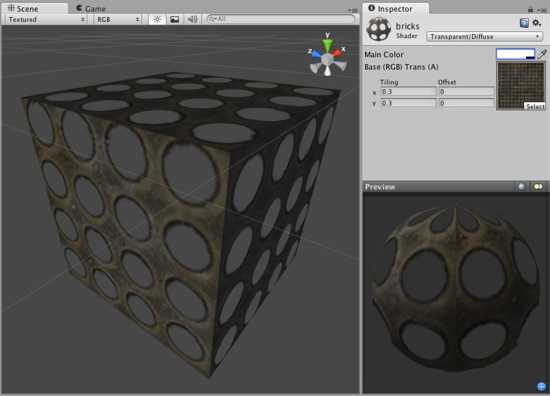
A、透明属性，通过读取主纹理的 alpha 通道使网格几何结构部分透明或全部透明。在 alpha 中，0（黑）表示完全透明，而 255（白）则表示完全不透明。如果主纹理没有 alpha 通道，则该物体看起来完全不透明。

B、顶点光照属性，不会显示基于像素的任何渲染效果



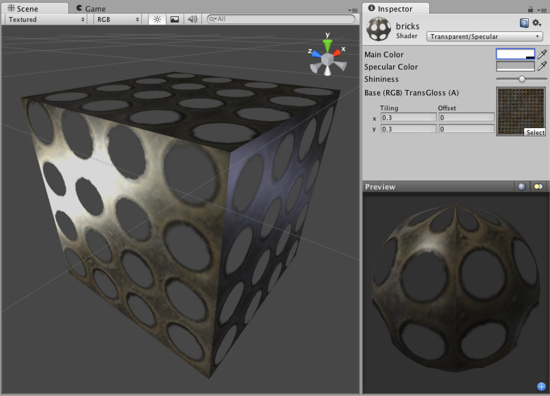
### （2）、透明漫反射（Transparent Diffuse）

属性：透明（apha）+漫反射属性



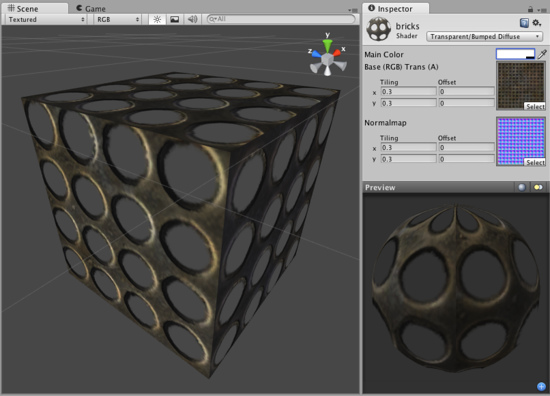
### （3）、透明高光

属性：透明（apha）+高光属性、同 Normal Specular，支持了半透明



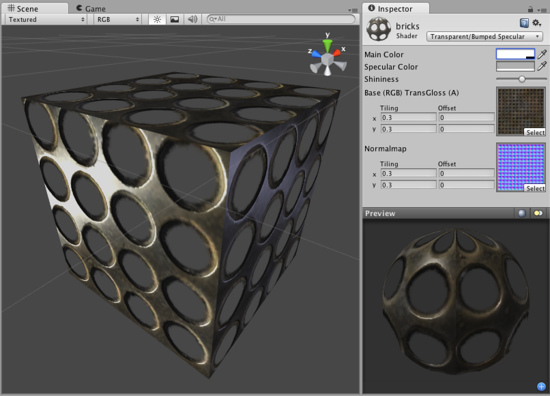
### （4）、透明凹凸漫反射

属性：透明（apha）+法线贴图+漫反射属性



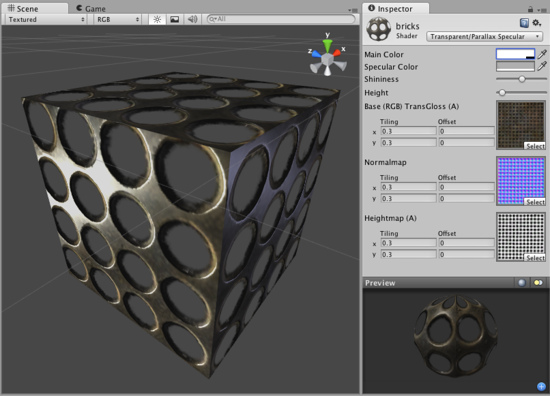
### （5）、透明凹凸高光

属性：透明（apha）+法线贴图+高光，基础 (Base) 纹理的 alpha 通道定义了透明 (Transparent) 区域和高光贴图 (Specular Map)。



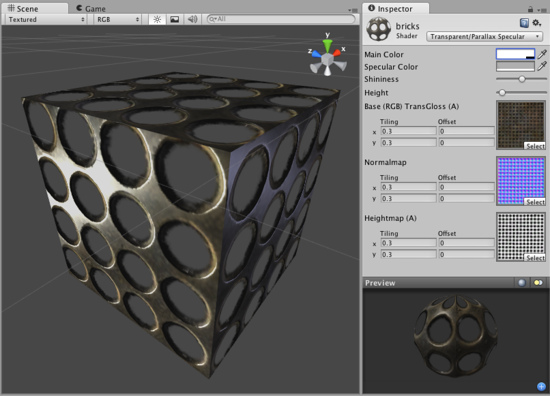
### （6）、透明视差漫反射

属性：透明（apha）+视差法线贴图+漫反射属性，



### （7）、透明视差高光

属性：透明（apha）+视差法线贴图+高光属性



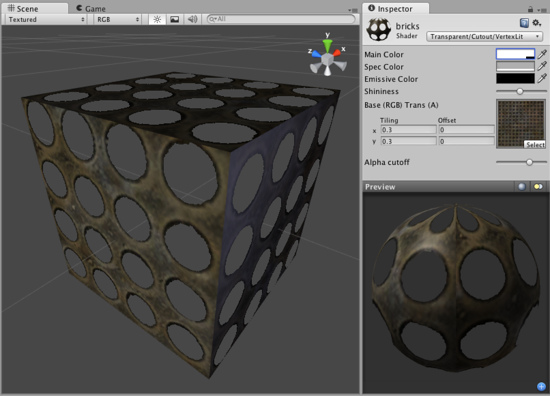
## 3、透明镂空着色器系列（Transparent Cutout Shader Family）

透明镂空 (Transparent Cutout) 着色器用于有完全不透明部分和完全透明部分（无部分透明）的物体。如链栅栏、树、草等。

### （1)、透明镂空顶点光照（transparent cutout vertexlit）

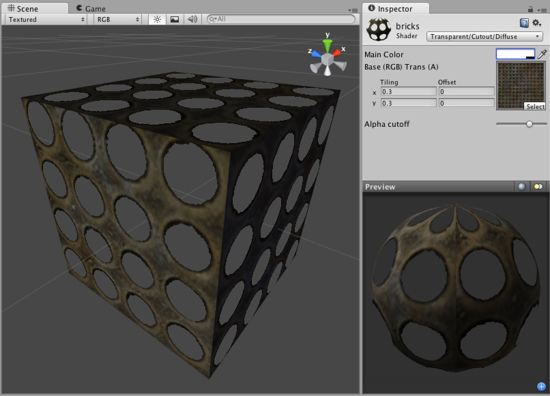
属性：

* 1、透明镂空属性，此着色器不能有部分透明的区域。所有区域都必须是完全不透明或完全透明的。使用此着色器的物体可投射和接收阴影！使用此着色器时，不会产生通常与透明 (Transparent) 着色器相关的图形排序问题。此着色器使用**基础 (Base)** 纹理中包含的 alpha 通道确定透明区域。如果 alpha 包含透明和不透明区域的混合，则可手动确定显示区域的分界点。调整**Alpha 分界 (Alpha Cutoff)** 滑块可更改此分界点。
* 2、顶点光照属性。



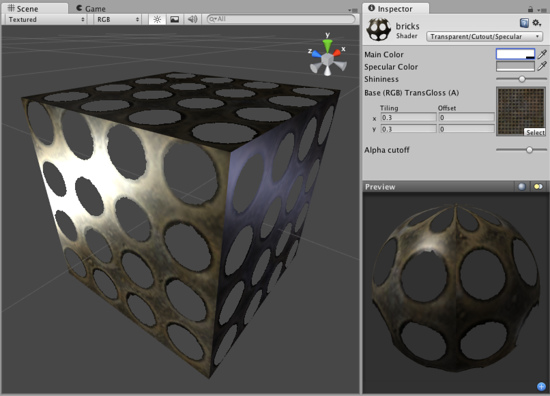
### （2）、透明镂空漫反射（transparent cutout diffuse）

属性：透明镂空属性+漫反射属性（表面光照强度随表面与光之间夹角的减小而减小。光照强度仅取决于此角，且不随相机的移动或旋转而变化。）



### （3）、透明镂空高光（transparent cutout Specular）

属性：透明镂空+高光属性，基础 (Base) 纹理的 alpha 通道定义了透明 (Transparent) 区域和高光贴图 (Specular Map)。主纹理的 alpha 通道用作高光贴图 (Specular Map)（有时称为“光泽贴图”） 用于定义对象哪些区域的反射性更强。Alpha 的黑色区域是零高光反射区域，而白色区域是全高光反射区域。这在您希望对象的不同区域反射不同级别的高光时非常有用。例如，生锈的金属使用弱高光，而抛光金属使用强高光。唇膏的高光强于皮肤，而皮肤的高光又强于棉质衣物。



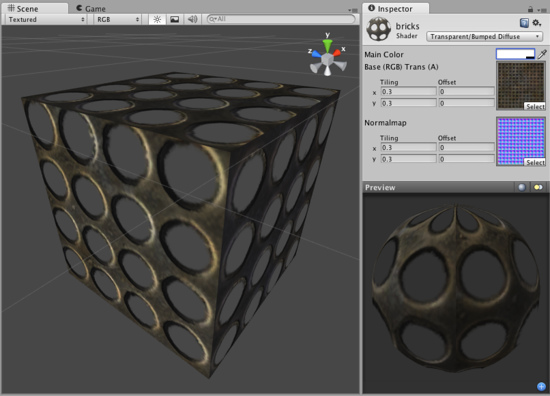
### （4）、透明镂空凹凸漫反射（~cutout Bumped Diffuse）

属性：透明镂空+法线贴图+漫反射，

法线贴图 (Normal Map) 是法线贴图的一种切线空间类型。切线空间是“跟随”模型几何结构表面的空间。在这个空间里，Z 轴始终远离表面。相对于另一种“对象空间”类型的法线贴图 (Normal Map)，切线空间法线贴图 (Normal Map) 稍微昂贵一些，但也有一些优势：

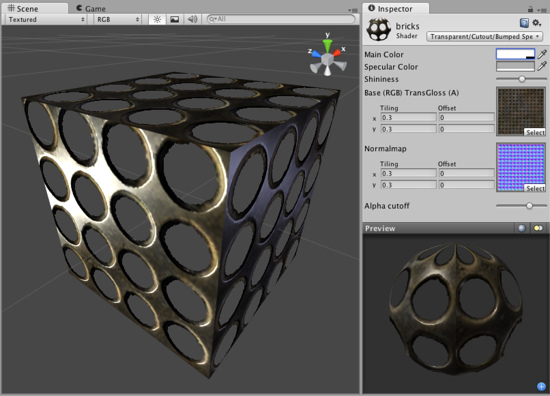
1. 可用于奇形怪状的模型 - 恰好表现表面凹凸不平的效果。
2. 可将部分法线贴图重复用于模型的不同区域或者不同的模型

透明镂空属性中，在 alpha 中，0（黑）表示完全透明，而 255（白）则表示完全不透明。如果主纹理没有 alpha 通道，则该物体看起来完全不透明。



### （5）、透明镂空凹凸高光（~cutout Bumped specular）

属性：透明镂空+法线贴图+高光属性，与漫反射 (Diffuse) 一样，高光也计算简单的 (Lambertian) 光照，并附加一个依赖于观察者的高光亮点。这个称为 Blinn-Phong 光照模型。其具有取决于面角、光角和观察角的高光亮点。该亮点实际上只是对光源模糊反射的实时适宜模拟。亮点的模糊层次由**检视器 (Inspector)** 中的**反光 (Shininess)** 滑块控制。



### （6）、透明镂空transparent Cutout Soft Edge lit

这个是官方文档里面没有记录的，估计是后面新加入的。 这个 Shader 绘制物体的正反两面，不受光照影响，常常用来绘制花草，树木，叶子。

他包含了两个 Pass，

第一次将对象中 Alpha 大于 alpha cutoff 的部分以不透明的方式绘制。

第二次关闭写 zbuffer 后，以半透明的方式绘制 Alpha 小于 alpha cutoff 的部分。这样做可以将绿草这种半透明的对象不需要排序就能绘制正确。



## 4、自发光着色器系列（Self-Illuminated）

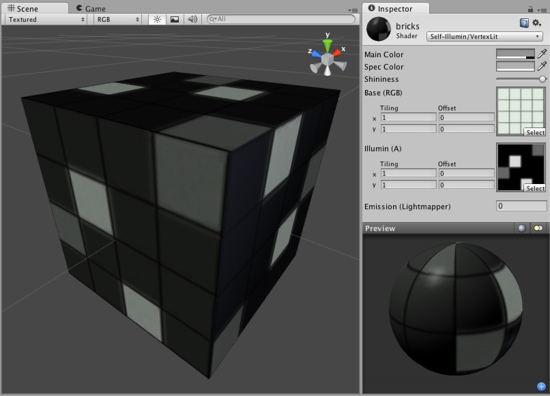
此着色器使您能够定义物体的发光部分和黑暗部分。辅纹理的 alpha 通道将定义在即使没有光线照耀时，物体自身的“发光”区域。在 alpha 通道中，黑色表示零光照，而白色表示物体发出的最大光照。任何场景光都会增加着色器光照顶部的亮度。因此，即使您的物体自身不发光，其仍将被场景中的光照亮。

### （1）、自发光顶点光照（Self-Illumin VertexLit）

属性：

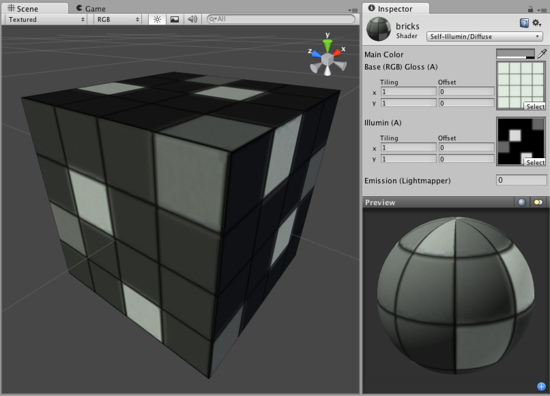
自发光属性+顶点光照属性。即此着色器使您能够定义物体的发光部分和黑暗部分。辅纹理的 alpha 通道将定义在即使没有光线照耀时，物体自身的“发光”区域。在 alpha 通道中，黑色表示零光照，而白色表示物体发出的最大光照。任何场景光都会增加着色器光照顶部的亮度。因此，即使您的物体自身不发光，其仍将被场景中的光照亮。

由于是顶点光照，因此不会显示基于像素的任何渲染效果，如灯光 cookies、法线贴图或阴影。此着色器还对模型的密铺 (tesselation) 敏感得多。如果使用此着色器将点光灯放在离立方体非常近的地方，则光只会在角位被计算。像素光照着色器能更有效地创建漂亮的圆形亮点，且不受密铺 (tesselation) 影响。



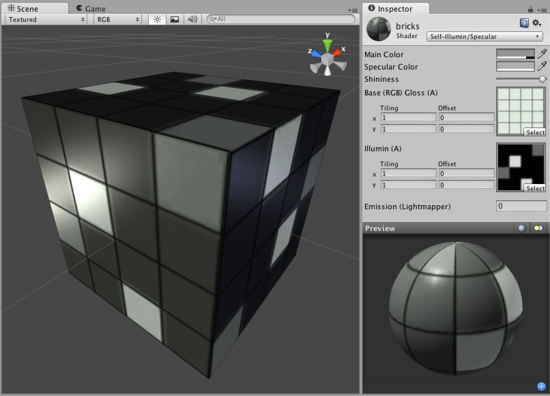
### （2）、自发光漫反射（Self-Illumin Diffuse）

属性：自发光属性+漫反射，漫反射计算简单的 (Lambertian) 光照模型。表面光照强度随表面与光之间夹角的减小而减小。光照强度仅取决于此角，且不随相机的移动或旋转而变化。



### （3）、自发光高光（Self-Illumin Specular）

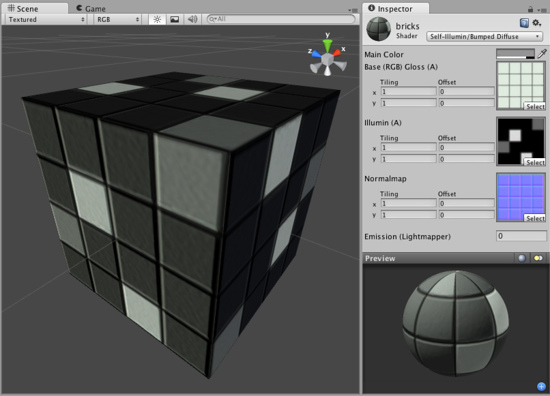
属性：自发光+高光属性，主纹理的 alpha 通道用作高光贴图 (Specular Map)（有时称为“光泽贴图”），用于定义对象哪些区域的反射性更强。Alpha 的黑色区域是零高光反射区域，而白色区域是全高光反射区域。这在您希望对象的不同区域反射不同级别的高光时非常有用。例如，生锈的金属使用弱高光，而抛光金属使用强高光。唇膏的高光强于皮肤，而皮肤的高光又强于棉质衣物。



### （4）、自发光法线贴图漫反射（~Bumped Diffuse）

属性：

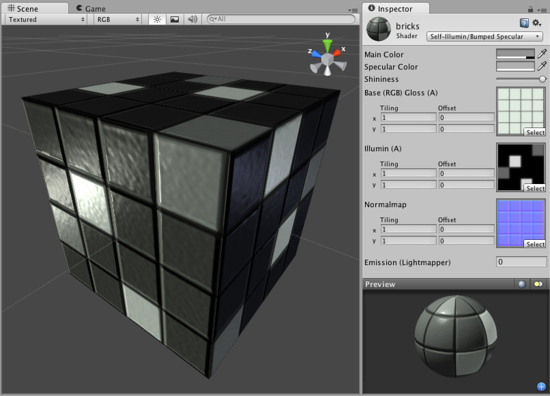
自发光+凹凸法线贴图+漫反射属性，即不仅自发光，表面光根据光照与之夹角来计算，不随着相机改变而改变。同时，注重细节纹理，可用于奇形怪状的模型 - 恰好表现表面凹凸不平的效果。可将部分法线贴图重复用于模型的不同区域或者不同的模型。



### （5）、自发光法线贴图高光（~Bumped Specular）

属性：

自发光+凹凸法线贴图+高光，即除了自发光以及注重凹凸细节纹理之外，增加高光效果。一般而言，此着色器消耗的渲染资源较多。



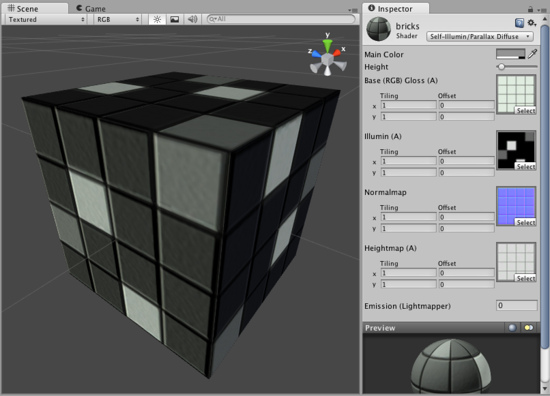
### （6）、自发光视差漫反射（~Parallax Specular）

属性：

自发光+视差法线贴图+漫发射属性，

视差法线 (Parallax Normal) 贴图 与与普通的法线 (Normal) 贴图一样，但对“深度”的模拟更佳。使用高度贴图 (Height Map) 可获得额外的深度效果。高度贴图 (Height Map) 包含在法线 (Normal) 贴图的 alpha 通道中。在 alpha 通道中，黑色表示零深度，而白色表示最大深度。这常用于砖块/石头，以便更好地显示它们之间的缝隙。

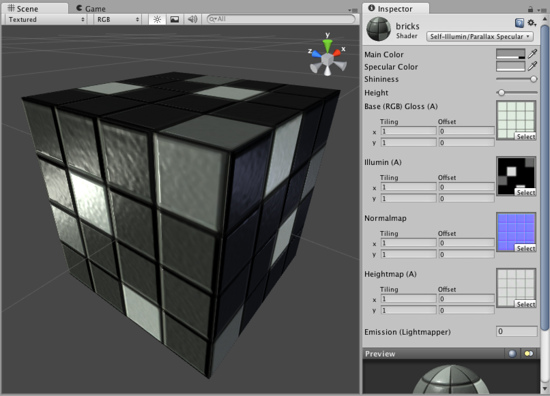
视差 (Parallax) 贴图技术非常简单，通过它可以获得不寻常的艺术级别效果。具体来说，高度贴图 (Height Map) 中应避免非常陡峭的高度转换。在检视器 (Inspector) 中调节高度 (Height) 值还可能导致物体以奇怪、不真实的方式扭曲。因此，建议使用渐变的高度贴图 (Height Map) 转换或者保持高度 (Height) 滑块朝向浅端。（主要随着摄像机改变，细节改变）



### （7）、自发光视差高光（~ Parallax Specular）

属性：

自发光+视差法线贴图+高光，在自发光和视差法线贴图基础上引入高光属性，调节Height可以改变高度图，而调节shininess可以改变光照强度。

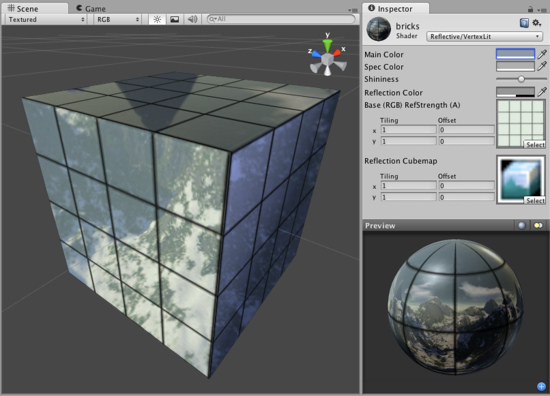


## 5、反射着色器系列（Reflective family）

### （1）、反射顶点光照（~VertexLit）

属性：

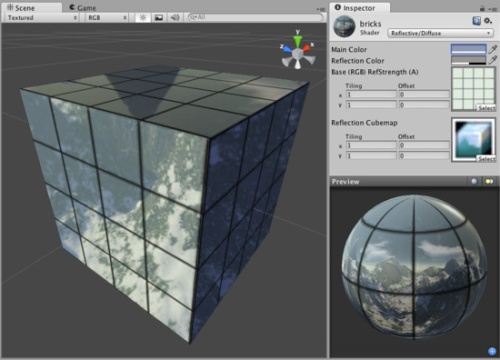
反射+顶点光照，该着色器将模拟反射性表面，如汽车、金属物体等。它需要定义具体反射区域的外界立方体贴图 (Cubemap)。主纹理的 alpha 通道定义物体表面的反射强度。任何场景光都会增加反射物顶部的亮度。照射它的所有光照在一个通道中被渲染并仅在顶点被计算。



### （2）、反射性漫反射（~Diffuse）

属性：

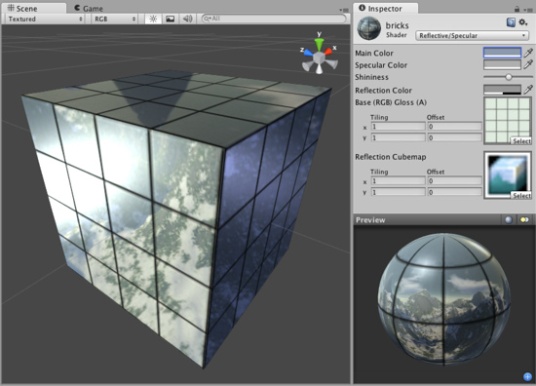
反射+漫反射，该着色器将模拟反射性表面，如汽车、金属物体等。它需要定义具体反射区域的外界立方体贴图 (Cubemap)。主纹理的 alpha 通道定义物体表面的反射强度。任何场景光都会增加反射物顶部的亮度。同时，表面光照强度随表面与光之间夹角的减小而减小。光照强度仅取决于此角，且不随相机的移动或旋转而变化。



### （3）、反射性高光（~Specular）

属性：

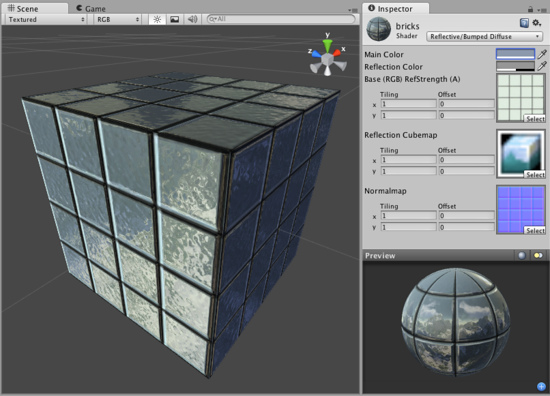
反射+高光，与漫反射 (Diffuse) 一样，高光也计算简单的 (Lambertian) 光照，并附加一个依赖于观察者的高光亮点。这个称为 Blinn-Phong 光照模型。其具有取决于面角、光角和观察角的高光亮点。该亮点实际上只是对光源模糊反射的实时适宜模拟。亮点的模糊层次由**检视器 (Inspector)** 中的**反光 (Shininess)** 滑块控制。



### （4）、反射性凹凸漫反射（~Bumped Diffuse）

属性：

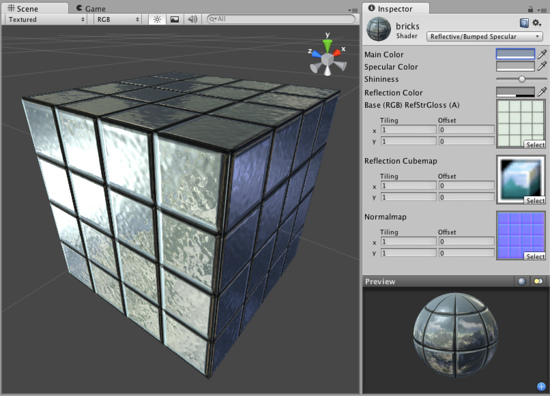
反射属性+法线贴图+漫反射，恰好表现表面凹凸不平的效果。可将部分法线贴图重复用于模型的不同区域或者不同的模型。同时通过放射方式展示出来。



### （5）、反射性凹凸高光（~Bumped Specular）

属性：

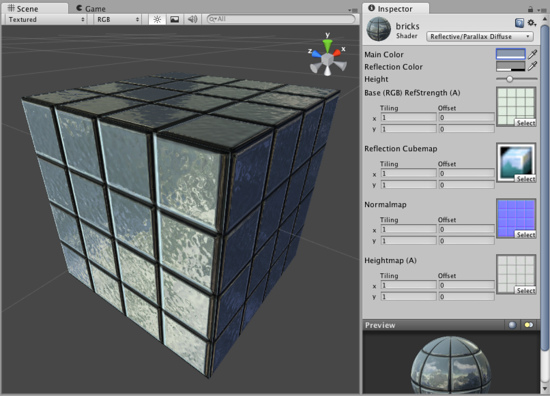
反射性+凹凸法线+高光，即在反射出凹凸不平的纹理细节效果上加大光强属性。达到高光效果。



### （6）、反射性视差漫反射（~Parallax Diffuse）

属性：

反射性+视差+漫反射，视差 (Parallax) 贴图技术非常简单，通过它可以获得不寻常的艺术级别效果。具体来说，高度贴图 (Height Map) 中应避免非常陡峭的高度转换。在检视器 (Inspector) 中调节高度 (Height) 值还可能导致物体以奇怪、不真实的方式扭曲。因此，建议使用渐变的高度贴图 (Height Map) 转换或者保持高度 Height 滑块朝向浅端。



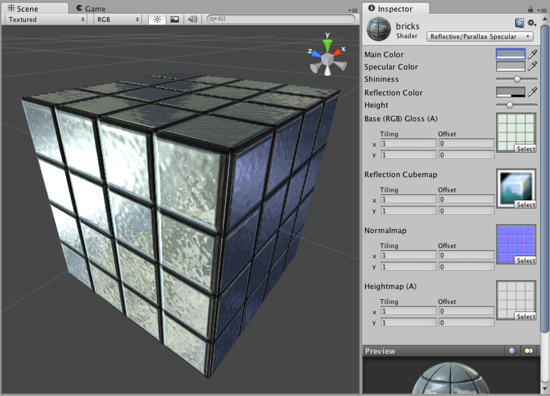
### （7）、反射性视差高光（~Parallax Specular）

属性：

反射+视差法线贴图+高光。

视差法线 (Parallax Normal) 贴图 与与普通的法线 (Normal) 贴图一样，但对“深度”的模拟更佳。使用高度贴图 (Height Map) 可获得额外的深度效果。高度贴图 (Height Map) 包含在法线 (Normal) 贴图的 alpha 通道中。在 alpha 通道中，黑色表示零深度，而白色表示最大深度。这常用于砖块/石头，以便更好地显示它们之间的缝隙。

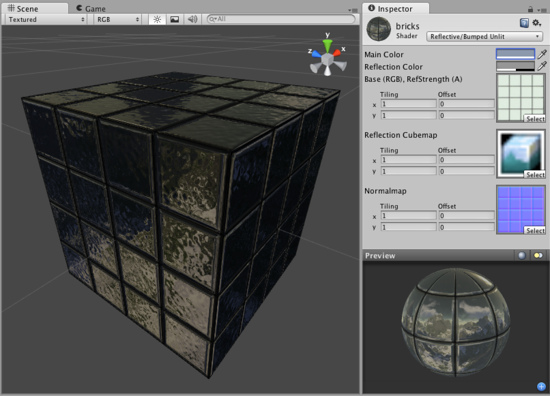
视差 (Parallax) 贴图技术非常简单，通过它可以获得不寻常的艺术级别效果。具体来说，高度贴图 (Height Map) 中应避免非常陡峭的高度转换。在检视器 (Inspector) 中调节高度 (Height) 值还可能导致物体以奇怪、不真实的方式扭曲。因此，建议使用渐变的高度贴图 (Height Map) 转换或者保持高度 (Height) 滑块朝向浅端。



### （8）、反射性法线无光照（~Bumped Unlit）

属性：

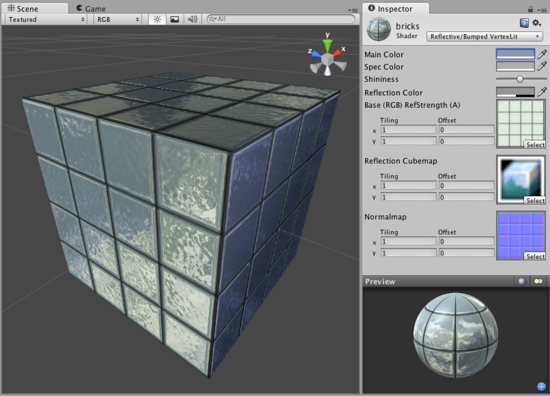
该着色器之所以特别，是因为它对光毫无响应，所以您不必担心性能会因多个光照的使用而降低。它只在模型上显示反射立方体贴图。由于反射被法线贴图扭曲，所以您可以从详细反射获益。因其不响应光，所以该着色器十分便宜。这是一个有些专业的用例，但在这些案例中，它确实以尽可能便宜的价格满足您的要求。



### （9）、反射性法线顶点（~Bumped Vertex）

属性：

反射+凹凸法线+顶点属性，该着色器是反射法线 (Reflective Normal) 贴图的较好选择。如果不想让物体本身受到像素光照影响，但又确实希望反射受到法线贴图影响，那么这款着色器是适合您的。该着色器是顶点光照，所以和其反射法线 (Reflective Normal) 贴图副本相比，它渲染地更快。



## 6其它典型常见Shader

### （1）、遮罩功能

### A、技能冷却

NGUI中UISprite中filedsprite 🡪Radio360🡪调节FiledMount

### B、通过透明shader，绘制网格,插件MaskEffect

Shader "MaskEffect"

{

SubShader

{

Tags { "Queue" = "Transparent" }

Pass

{

Fog { Mode Off }

Blend SrcAlpha OneMinusSrcAlpha

CGPROGRAM

#pragma vertex vert

#pragma fragment frag

// 顶点输入结构

struct appdata

{

float4 vertex : POSITION0;

float4 color : COLOR;

};

// 顶点输出结构

struct v2f

{

float4 pos : SV\_POSITION;

fixed4 color : COLOR;

};

// vertex

v2f vert( appdata IN )

{

v2f OUT;

OUT.pos = mul( UNITY\_MATRIX\_MVP, IN.vertex );

OUT.color = IN.color;

return OUT;

}

// fragment

fixed4 frag( v2f IN ) : COLOR

{

fixed4 c = fixed4( IN.color.r, IN.color.g, IN.color.b, IN.color.a );

return c;

}

ENDCG

}

}

Fallback "Diffuse"

}

或者

Shader "mask shader"  
{  
Properties  
{  
\_MainTex ("Base (RGB)", 2D) = "white" {}  
\_Mask ("Culling Mask", 2D) = "white" {}  
\_Cutoff ("Alpha cutoff", Range (0,1)) = 0.1  
}  
SubShader  
{  
Tags {"Queue"="Transparent"}  
Lighting Off  
ZWrite Off  
Blend SrcAlpha OneMinusSrcAlpha  
AlphaTest GEqual [\_Cutoff]  
Pass  
{  
SetTexture [\_Mask] {combine texture}  
SetTexture [\_MainTex] {combine texture, previous}  
}  
}  
}

//-------------------------------------------------------------------

Shader "Depth Mask Complex"  
 {  
     SubShader  
     {  
         Tags {"Queue" = "Background"}  
         Blend SrcAlpha OneMinusSrcAlpha  
         Lighting Off  
         ZWrite On  
         ZTest Always  
         Pass  
         {  
   Color(0,0,0,0)  
         }  
     }  
 }

//-------------------------------------------------------------------

Shader "Separate Alpha Mask" {

Properties {

\_MainTex ("Base (RGB)", 2D) = "white" {}

 \_Alpha ("Alpha (A)", 2D) = "white" {}

}

SubShader {

Tags { "RenderType" = "Transparent" "Queue" = "Transparent"}

ZWrite Off

Blend SrcAlpha OneMinusSrcAlpha

ColorMask RGB

Pass {

SetTexture[\_MainTex] {

Combine texture

}

SetTexture[\_Alpha] {

Combine previous, texture

}

}

}

}

//-------------------------------------------------------------------

Shader "Depth Mask" {  
    SubShader {  
        Tags {"Queue" = "Geometry-10" }         
        Lighting Off  
        ZTest LEqual  
        ZWrite On  
        ColorMask 0  
        Pass {}  
    }  
}

### C、通过调节Shader

例子一、

Shader "Custom/Mask"

{

Properties

{

\_MainTex ("Main Texture", 2D) = "white" {}

\_Mask ("Mask Texture", 2D) = "white" {}

}

SubShader

{

Tags { "Queue"="Transparent" }

Lighting On

ZWrite Off

Blend SrcAlpha OneMinusSrcAlpha

Pass

{

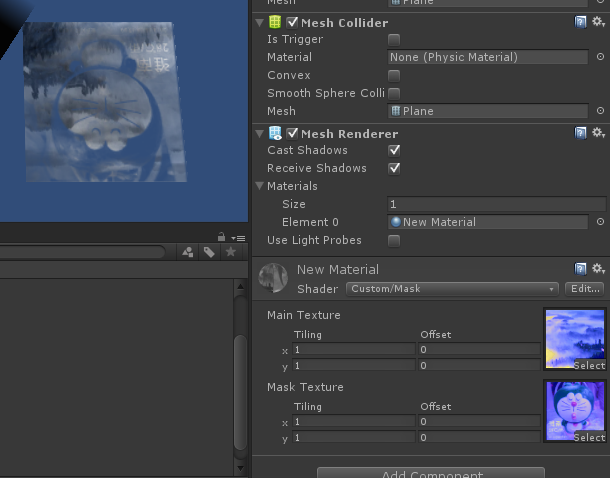
SetTexture [\_Mask] {combine texture}

SetTexture [\_MainTex] {combine texture, previous}

}

}

}



例子二、

//针对模型的一种渐隐效果展示

Shader "Custom/jianying" {

Properties {

\_MainTex ("Base (RGB)", 2D) = "white" {}

\_zhengfu("zhengfu",vector)=(0.2,4,0,0)

\_Offset("Offset",float)=0

\_dept("Dept",float) = 0.5

}

SubShader {

Tags { "RenderType"="Opaque" }

LOD 200

Blend SrcAlpha OneMinusSrcAlpha

CGPROGRAM

#pragma surface surf Lambert

sampler2D \_MainTex;

float2 \_zhengfu;

float \_dept;

float \_Offset;

struct Input {

float2 uv\_MainTex;

};

void surf (Input IN, inout SurfaceOutput o) {

half4 c = tex2D (\_MainTex, IN.uv\_MainTex);

o.Albedo = c.rgb;

o.Alpha = c.a;

o.Alpha = IN.uv\_MainTex.x\*4+sin(IN.uv\_MainTex.y\*2\*3.14\*\_zhengfu.y)\*\_zhengfu.x- \_dept ;//波浪形式

//o.Alpha = IN.uv\_MainTex.x\*\_Offset;

}

ENDCG

}

FallBack "Diffuse"

}



### （2）.雾化

（1）、全局雾化

在Edit->RenderSettings里设置，具体如下

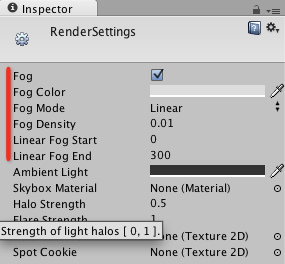
Fog： 激活或不激活  
Fog Color:雾的颜色，仅RGB起作用。

Fog Mode: Linear线性，Exp指数，Exp2 指数2

Fog Density:雾化的强度

Linear Fog Start:开始距离

Linear Fog End:结束距离



（2）部分雾化（Shader里面调）

在需要雾化的Shader里加入代码Fog{Mode Linear Color (0.87,0.87,0.87,1) Density 0.1  Range 0,300}

测试代码：

Shader "Custom/Fog" {

Properties {

\_MainTex ("Texture", 2D) = "white" {}

\_FogColor ("Fog Color", Color) = (0.3, 0.4, 0.7, 1.0)

}

SubShader {

Tags { "RenderType" = "Opaque" }

CGPROGRAM

#pragma surface surf Lambert finalcolor:mycolor vertex:myvert

struct Input {

float2 uv\_MainTex;

half fog;

};

void myvert (inout appdata\_full v, out Input data)

{

float4 hpos = mul (UNITY\_MATRIX\_MVP, v.vertex);

data.fog = min (1, dot (hpos.xy, hpos.xy) \* 0.1);

}

fixed4 \_FogColor;

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

{

fixed3 fogColor = \_FogColor.rgb;

#ifdef UNITY\_PASS\_FORWARDADD

fogColor = 0;

#endif

color.rgb = lerp (color.rgb, fogColor, IN.fog);

}

sampler2D \_MainTex;

void surf (Input IN, inout SurfaceOutput o) {

o.Albedo = tex2D (\_MainTex, IN.uv\_MainTex).rgb;

}

ENDCG

}

Fallback "Diffuse"

}