技术专题作业 4：自定义特效（Shader）

516030910435　　汤志彪

一、基本要求（在 sence ShaderSence 中）

根据说明文档完成了三个函数的编写

Distribution term：

$$D_{GGX} = \frac{\alpha^2}{\pi((n \cdot h)^2(\alpha^2 - 1) + 1)^2}$$

```
float GGX_D(float roughness, float NdotH)
{
    //// TODO: your implementation
    float D;
    float a2 = pow(roughness, 2);
    float temp = (pow(NdotH, 2)*(a2 - 1) + 1);

    D = a2 / (UNITY_PI*pow(temp, 2));

    return float4(float3(1, 1, 1)* D, 1);;
}
```

Fresnel term：

$$F_{Schlick} = C_{spec} + (1 - C_{spec})(1 - l \cdot h)^5$$

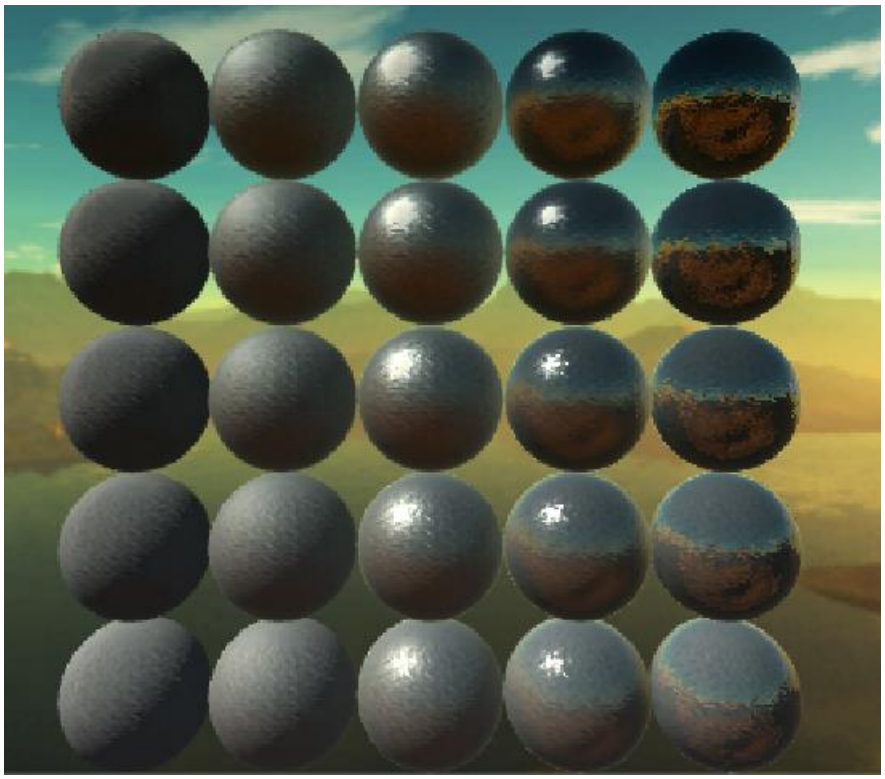
```
float3 Schlick_F(half3 R, half cosA)
{
    //// TODO: your implementation
    float3 F = R + (1 - R)*pow((1 - cosA), 5);
    return F;
}
```

Geometry term：

$$G_{Cook-Torrance} = \min\left(1, \frac{2(n \cdot h)(n \cdot v)}{v \cdot h}, \frac{2(n \cdot h)(n \cdot l)}{v \cdot h}\right)$$

```
float CookTorrence_G (float NdotL, float NdotV, float VdotH, float NdotH)
{
    //// TODO: your implementation
    float G = min(1, 2 * NdotH*NdotV / VdotH);
    G = min(G, 2 * NdotH*NdotL / VdotH);
    return G;
}
```

最终效果：



二、非真实感渲染（在场景 ShaderCartoon 中）

卡通风格渲染

根据论文

（实现时只实现了单光源）

1. 非依赖视角的光照部分

   非依赖视角的部分由一个空间变化的平行环境光部分加上一个改动后的兰伯特光照部分。

$$k_d\left[a(\hat{n})+\sum_{i=1}^{L}c_i w\left(\left(\alpha\left(\hat{n}\cdot\hat{l}_i\right)+\beta\right)^{\gamma}\right)\right] \qquad (1)$$

其中半兰伯特部分$\left(\alpha\left(\hat{n}\cdot\hat{l}_i\right)+\beta\right)^{\gamma}$令 α = 0.5、β = 0.5、γ = 2，将让点乘的结果（范围是-1 到+1）转换到 0 到 1。W（）是个变形函数，该变形函数使

用一个 1 维贴图进行查找映射，例如以下图所看到的。这样的方式间接地创建了一个“硬阴影”，在保留全部法线对光照的变化的同一时候，收紧了明暗交界处从亮到暗的跳变。

Figure 7: *Typical diffuse light warping function*

再乘上光源颜色。

平行环境光部分（Directional Ambient Term）：$a(\hat{n})$.

由于这部分我不是很懂“环境光盒子”，所以实现的时候都用（1，1，1）代替该部分。

2. 依赖视角的光照部分

$$\sum_{i=1}^{L}\left[c_i k_s max\left(f_s(\hat{v}\cdot\hat{r}_i)^{k_{spec}}, f_r k_r(\hat{v}\cdot\hat{r}_i)^{k_{rim}}\right)\right]+(\hat{n}\cdot\hat{u})\,f_r k_r a(\hat{v}) \quad (2)$$

多重 Phong 部分（Multiple Phong Term）：

$$\sum_{i=1}^{L}\left[c_i k_s max\left(f_s(\hat{v}\cdot\hat{r}_i)^{k_{spec}}, f_r k_r(\hat{v}\cdot\hat{r}_i)^{k_{rim}}\right)\right].$$

包括了使用常见的表达式来计算 Phong 高光。而且使用了适当的常量以及一个菲涅耳因子对它进行调整。然而，在求和的内部还使用了 max()函数将 Phong 高光和额外的 Phong lobes（使用了不同的指数、菲涅耳系数以及遮罩）结合在一起。

专用的边缘光照（Dedicated Rim Lighting）：

$$(\hat{n}\cdot\hat{u})\,f_r k_r a(\hat{v})$$

当角色逐渐移动远离光源时。只基于 Phong 部分的边缘高光可能不会向我们想要的那么明显了。为此，我们还加入了一个专用的边缘高光部分（等式 2 的右半部分）。这个部分使用了观察方向对环境盒子（ambient cube）进行了求值。而且使用了一个由美术控制的遮罩纹理、菲涅耳因子以及表达式进行调整。

这个最后的表达式不过逐像素的法线和空间向上向量（up vector）点乘的结果，再约束到正数范围（clamped to be positive）。这使得这样的专用的边缘高光看起来包括了环境中的间接光源。但只适用于朝上的法线。这样的方法是一种基于美学和感性的选择。我们希望这样能够让人感觉这些光照好像是从上方照下来的一样。

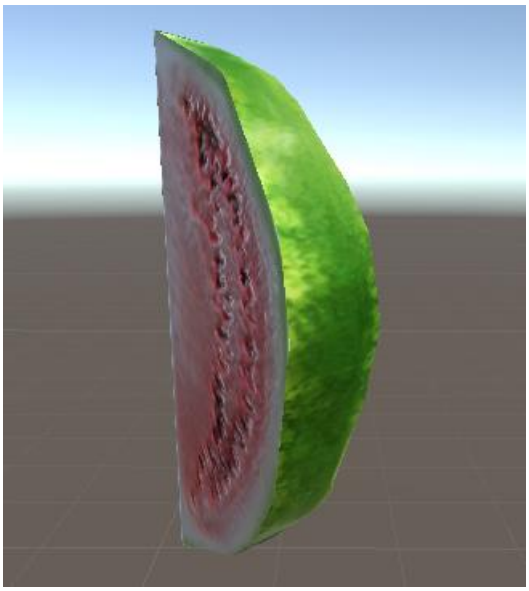
这里的光环境盒子我还是用（1，1，1）代替

（1），（2）两式相加就是我们所需要的结果

效果显示：

Sence ->ShaderCartoon

使用前



使用后：