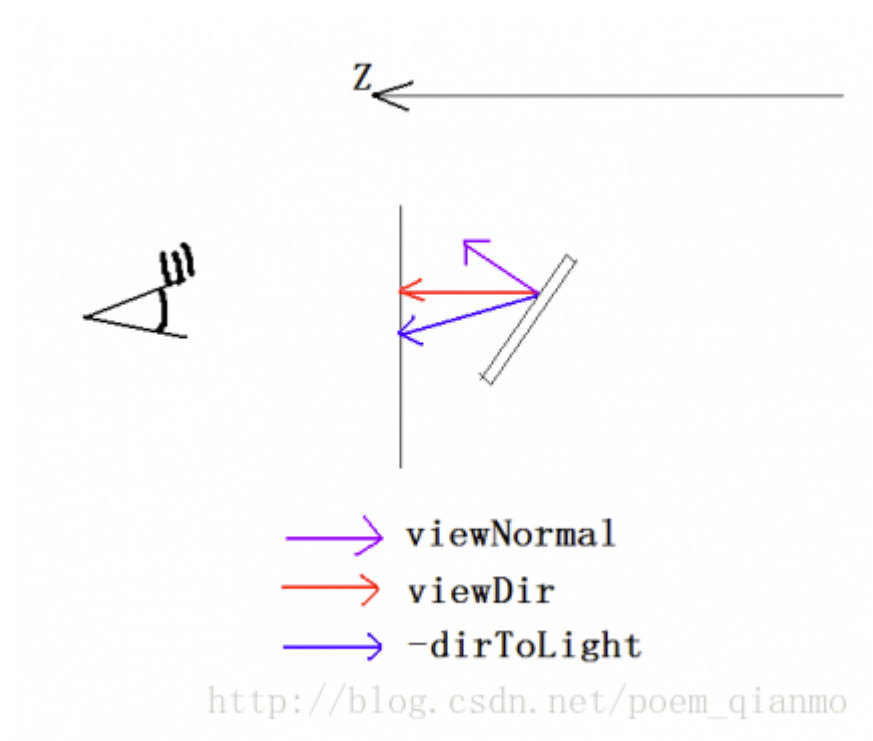
Unity Shader_7---------边缘光照的实现（顶点片元与表面着色器的

边缘光照的实现

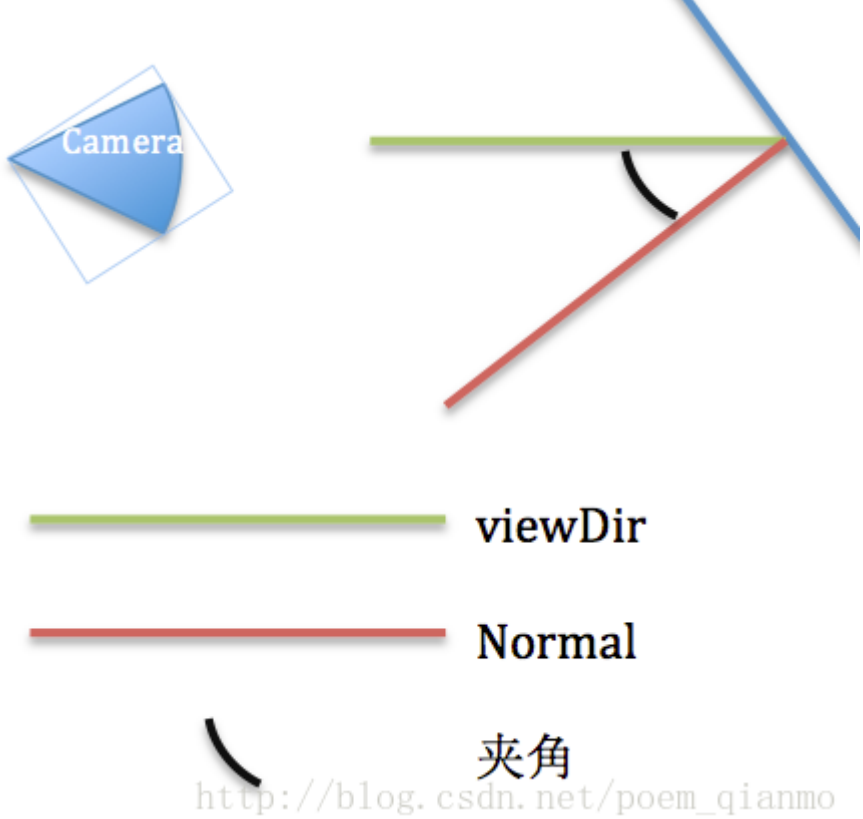
记录一下***表面着色器实现的边缘光照*** 原理

如下图所示：

其中的viewDir 意为WorldSpace View Direction，也就是当前坐标的视角方向：



使用Normalize函数，用于获取到的viewDir坐标转成一个单位向量且方向不变，外面再与点的法线做点积。最外层再用 saturate算出一[0,1]之间的最靠近的值。这样算出一个rim边界。原理可以这样解释：



这里o.Normal就是单位向量。外加Normalize了viewDir。因此求得的点积就是夹角的cos值。因为cos值越大，夹角越小，所以，这时取反操作。这样，夹角越大，所反射上的颜色就越多。于是就得到的两边发光的效果：

```
Shader "学习Shader/Surface/Surf_RimLighting" {
    Properties {
        _MainTex ("Albedo (RGB)", 2D) = "white" {}
        _BumpMap("BumpMap Normal",2D) = "bump"{}  //使用了凹凸纹理
        _Detail("Texture Detail", 2D) = "gray"{}  //使用了灰度图

        //用于自定义表面色泽时使用
        _ColorTint("ColorTint",Color) = (0.6,0.3,0.6,0.3)

        //边缘光照 颜色
        _RimColor("Rim Color", Color) = (0.26,0.19,0.16,0.0)
        //边缘光照 宽度
        _RimPower("Rim Power",Range(0.5,8.0)) = 3.0
    }
    SubShader
    {
        //指明着色器类型， 当渲染非透明物体时调用
        Tags { "RenderType"="Opaque" }
        LOD 200

        CGPROGRAM
```

```
        //【1】声明光照模式：使用Lambert光照 + 自定义颜色函数
#pragma surface surf Lambert finalcolor:setcolor
        //【2】输入结构
        struct Input
        {
            //主纹理的uv坐标
            float2 uv_MainTex;
            //凹凸纹理的uv坐标
            float2 uv_BumpMap;
            //细节纹理的uv坐标
            float2 uv_Detail;   /*  注意：此处名称需要保证为uv_面板属性名 否则 最终显示的效果会不一致 */
            //当前坐标的视角方向
            float3 viewDir;
        };

        //【3】 变量声明
        sampler2D _MainTex;
        sampler2D _BumpMap;
        sampler2D _Detail;
        fixed4 _ColorTint;
        float4 _RimColor;
        float _RimPower;

        //【4】自定义颜色函数 用于编译指令的自定义属性
        void setcolor(Input IN, SurfaceOutput o, inout fixed4 color)//***需要三个参数***
        {
            color *= _ColorTint;
        }
        //【5】表面着色函数
        void surf(Input IN, inout SurfaceOutput o)
        {
            //从主纹理中获取RGB颜色值
            o.Albedo = tex2D(_MainTex, IN.uv_MainTex).rgb;
            //设置细节纹理
            o.Albedo *= tex2D(_Detail, IN.uv_Detail).rgb * 2;
            //从凹凸纹理获取法线值
            o.Normal = UnpackNormal(tex2D(_BumpMap, IN.uv_BumpMap));
            //根据   视线 和 表面法线  获取 不同区域的表面的  自发光  颜色及强度
            half rim = 1.0 - saturate(dot(normalize(IN.viewDir), o.Normal));
           // saturate : x = Max(0,Min(1,x))   dot 点积
            o.Emission = _RimColor.rgb * pow(rim, _RimPower); //指数
        }
```

```
            ENDCG
        }
        //【6】回滚 普通漫反射
        FallBack "Diffuse"
}
```

记录一下***顶点片元着色器实现的边缘光照*** 原理