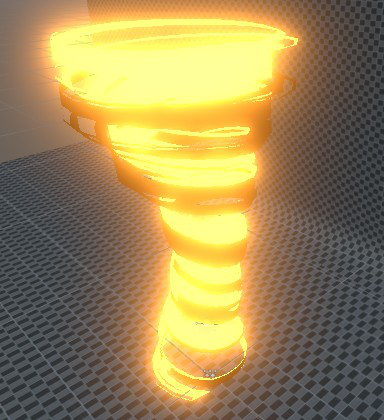
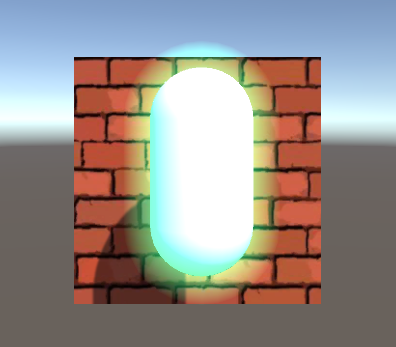
最近公司的美术同学用一个unity插件做了一个特性，是施放一个发光的龙卷风，可以看到，龙卷风泛着金色的光芒，把周围都照亮了，非常漂亮。特性本身是用布尔模糊做的。要做布尔模糊，就必须做后处理。但后处理在手机上是非常耗的，于是美术同学过来向我求助。



其原因就是因为Graphics.Blit函数调用的效率非常低。

于是我变换了一下思路，利用法线扩展的方式实现了一个近似的效果。

下面是我们这一章要展现的效果图。



如果要将光线扩展出物体的外部而不做后处理，我想到的第一个思路是法线扩张。

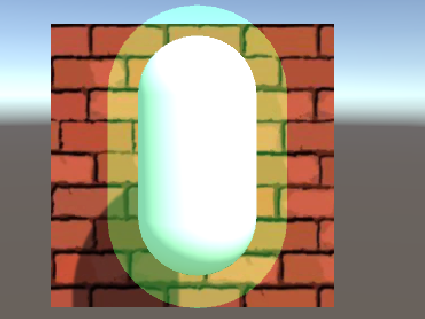
在vert渲染时用顶点=顶点\*法线\*缩放值对顶点做一个扩展。既最终的顶点位置为顶点向法相的方向扩展Radius个单位。只有将物体扩大，才能在超出物体的位置绘画像素。

|  |
| --- |
| float4 vert(float4 vertex : POSITION, float3 normal : NORMAL) : SV\_POSITION  {  float3 pos = vertex + normal\*\_Radius;  return mul(UNITY\_MATRIX\_MVP, float4(pos, 1.0));  } |

然后把alpha blend打开，混合模式为One One，即美术说的高光。

|  |
| --- |
| Blend One One |

出来的效果是这样的。



但是上图中，光的边缘明显不够平滑。

于是我们再做后续的处理，我简单用法线和相机的朝向来判断当前渲染的像素点是否模型的边缘，然后再用二次幂做一下曲线调节，再增加一个变量调整泛光的强度。

核心代码如下：

|  |
| --- |
| struct v2f {  half3 worldNormal : TEXCOORD0;  float rim : TEXCOORD1;  float4 pos : SV\_POSITION;  };  float \_Power;  float \_Radius;  v2f vert(float4 vertex : POSITION, float3 normal : NORMAL)  {  v2f o;  float3 pos = vertex + normal\*\_Radius;  o.pos = mul(UNITY\_MATRIX\_MVP, float4(pos, 1.0));  float3 viewDir = ObjSpaceViewDir(vertex);  viewDir = normalize(viewDir);  float f = max(0, dot(viewDir, normal));  o.rim = f\*f \*\_Power;  o.worldNormal = normal;  return o;  }  fixed4 \_Color;  fixed4 frag(v2f i) : SV\_Target  {  return \_Color \* i.rim;  }  ENDCG  } |

代码中ObjSpaceViewDir是unity的内置函数，功能是输入一个模型空间中的顶点位置，返回模型空间中从该点到摄像机的观察方向。float3 viewDir = ObjSpaceViewDir(vertex)是获取当前点到相机的朝向，而利用朝向与法线的点乘就可以知道法相与相机的夹角，当法线与相机的点乘接进1或者等于1时，便可简单认为这是物体的边缘。然后再把点乘结果为负值的范围处理为0，因为那些面是背向相机的。通过以上代码，本章开始展示的效果就出来了。