Goldsmiths, University of London

CM3045 – 3D Graphics & Animation

Peer-Graded Assignment: Lit Shader

Report by Hristo Stantchev

This report covers my submission for the "Lit Shader" Peer-Graded Assignment. This submission implements Lambert lighting.

Disclaimer

No matter what I did, no toon shader worked, which was my idea of an extension task. I am honestly beyond frustrated so I'm just submitting what I have.

Code:

Vertex-to-fragment:

```
v2f vert (appdata v)
{
    v2f o;
    o.vertex = UnityObjectToClipPos(v.vertex);
    o.uv = TRANSFORM_TEX(v.uv, _MainTex);
    o.normal = v.normal;
    o.viewT = normalize(WorldSpaceViewDir(v.vertex));
    return o;
}
```

Fragment:

```
fixed4 frag (v2f i) : SV_Target
{
    // _WorldSpaceLightPos0 provided by Unity
    float3 lightDir = normalize(_WorldSpaceLightPos0.xyz);

    // get dot product between surface normal and light direction
    float lightDot = dot(i.normal, lightDir);
    float diffuse = _DiffuseCoeff * lightDot;
    // get direction to the camera
    float3 get_cam =  i.viewT;
    // get bisection: h = (e + l) / ||e + l||
    float3 bisect = (-1*lightDir + get_cam)/(normalize(lightDir*-1 + get_cam));
    float specDot = pow(dot(bisect, i.normal), _Shine);
    float specular = _SpecularCoeff * specDot;
    // sample texture for color
    float3 color = tex2D(_MainTex, i.uv.xy).rgb;

    // add ambient light
    //color += ShadeSH9(half4(i.normal, 1));
    float mod = _LightColor0 * (diffuse + specular);
    // multiply albedo and lighting
    float3 rgb = color * mod;
    //my code
    //end my code
    float4 finalColor = float4(rgb, 1.0);

    finalColor *= _DiffuseColor;
    finalColor *= _SpecularColor;
    return finalColor;
}
```

Task 1: Lambert Diffusion

The formula for diffusion is Kd * Ii (n . l). Where Kd is the coefficient of diffusion (0,1). Ii is the light intensity and n . l is the dot product of the normal surface direction vector and the light direction vector.

In this code, heavily borrowed from Linden Reid [1], light direction is received in the fragment shader from the world space using a built in function, and the dot product is gotten from the vertex normal. Then, that is added as part of the final color for the fragment. I have added it being multiplied by a diffuse coefficient in a (0,1) range. Then, the final color is given from the main texture (if any), and multiplied by the diffuse colour and diffuse variable.

Task 2: Lambert Specular

The formula for the Lambert specular is Ks Ii $(h . n)^{m}$. where Ks and Ii similarly are coefficient (i.e. strength of effect) and light intensity. “n” is once again, the normal face, however h is interesting as it is the result of the direction to the light (l) + (e) the direction to the eye (view), divided by the normal of their sum, better explained in my code comments. “m” is the amount of shine, which must be > 0.

The full formula becomes light intensity * (diffuse + specular). Funnily enough, Unity has no CG/HLSL function in to directly get light intensity, only light color, which is why I could not use intensity in my formula.

My code for this is original (partly why half of my sphere is black which I have been unable to fix). The view direction is specified in v2f, whilst the light direction is reused and multiplied by -1 (as it is direction “to” the light, not “from”). H is then obtained in the form of the “bisect” variable. Then the specular dot product is obtained to the power of the shine. It is then finally multiplied by shine.

The way this ties together with the diffuse shader is by getting their sum and multiplying the final color of the fragment by both the diffuse and specular color.

References:

1. **Reid, Linden, “Custom Diffuse Shader In Unity”, date unavailable:** **https://lindenreidblog.com/2018/02/20/custom-diffuse-shader-in-unity/**