Lab 10 Report - Gipson

# Introduction

Object in the real world have varying specularity, such as metals which are highly reflective and rough surfaces like canvas, which are not. To mimic these a shader is going to need to use light information and the direction the camera is viewing it in order to replicate these properties.

# Methods

Start out with a basic shader. Then you will need to add a \_Shininess property of type float. This needs to be named exact for specific shaderlab functions. Then you will need to get normal and camera directions (code at end of report). This is done by normalizing the vertex position multiplied by \_World2object for normal, and \_WorldSpaceCameraPos needs to be multiplied by the normal direction for camera direction. Afterwards you need to get the diffuse reflection which uses the dot product of the normal and lightDirections, applied to attenuation and \_LightColor, again code is at the end. Next is the big part, specular reflection. Specular reflection uses a lot of math I don’t quite understand, but the gist is that you see more of the reflection the smaller the angle is between the camera direction and the normal direction. CG has a built in function for reflections called reflect(what to reflect, axis to reflect on). First get the reflection of –lightDirection over the normalDirection, then get the dot product of that value and the viewDirection. Then to soften the reflection you need to apply the max of specularReflection and 0.0 to specularReflection and then the max of 0.0 and the dot product of normalDirection and lightDirection, multiplied by the specularReflection itself. Add this to the diffuse reflection and unities ambient light to get your final color, not including specular color. You can do this in the vertex function or the fragment function. Fragment will be heavier, as its calculating per fragment, but it will look better and smoother.

# Conclusion

Specularity is easy to do with the right math. I can see using this shader and improving on it to add things like spec maps and special effects.

# Post-Lab

1. How does max(0.0, specularReflection) soften the colour bleed onto the backside of the object when using a specular shader?
   1. It allows you to display values from the specular reflection, instead of the lower 0 values.
2. How does max(0.0, dot(normalDirection, lightDirection)) remove the specular highlight from appearing on the back of the object?
   1. The dot product of the lightdirection and normaldirection will return negative numbers if the normal is facing away from the camera, and the max will return 0.0 if the value is negative

# Code

Shader "Custom/Specular" {

Properties {

\_Color ("Color", Color) = (1,1,1,1)

\_SpecColor("Specular color", Color) = (1,1,1,1)

\_Shininess("Shininess", float) = 10

\_Attenuation("Fall off", Range(0,5)) = 1

}

SubShader {

Tags{"LightMode"="ForwardBase"}

Pass{

CGPROGRAM

#pragma vertex vert

#pragma fragment frag

//user defined

uniform float4 \_Color;

uniform float4 \_SpecColor;

uniform float \_Shininess;

uniform float \_Attenuation;

//unity

uniform float4 \_LightColor0;

//input

struct input{

float4 vertexPos : POSITION;

float3 vertexNormal : NORMAL;

};

struct v2f{

float4 pixelPos : SV\_POSITION;

float4 pixelCol : COLOR;

float3 normalDirection : TEXCOORD0;

float4 pixelWorldPos : TEXCOORD1;

};

v2f vert(input i){

v2f toReturn;

//normal, view and light directions

toReturn.normalDirection = normalize(mul(float4(i.vertexNormal, 0.0), \_World2Object).xyz);

toReturn.pixelWorldPos = mul(\_Object2World, i.vertexPos);

toReturn.pixelPos = mul(UNITY\_MATRIX\_MVP, i.vertexPos);

return toReturn;

}

float4 frag(v2f i) : COLOR {

//direction declarations

float3 viewDirection = normalize(float3(float4(\_WorldSpaceCameraPos.xyz, 1.0) - i.pixelWorldPos.xyz));

float3 lightDirection = normalize(\_WorldSpaceLightPos0.xyz);

//reflections

float3 diffuseReflection = \_SpecColor \* \_Attenuation \* \_LightColor0.xyz \* max(0.0, dot(i.normalDirection, lightDirection));

float3 specularReflection = reflect(-lightDirection, i.normalDirection);

specularReflection = dot(specularReflection, viewDirection);

specularReflection = pow(max(0.0, specularReflection), \_Shininess);

specularReflection = max(0.0, dot(i.normalDirection, lightDirection)) \* specularReflection;

//final

float3 finalLight = specularReflection + diffuseReflection + UNITY\_LIGHTMODEL\_AMBIENT;

i.pixelCol = float4(finalLight \* \_Color, 1.0);

return i.pixelCol;

}

ENDCG

}

}

//FallBack "Diffuse"

}