Lab Report
Lab 10 – Specular Shader
Darrick Hilburn

Introduction

When it comes to lighting in games, sometimes we want to show the source realistically hitting an object. This can be accomplished by using a specular shader. Specular shaders take the viewer's position into account when calculating light striking an object and considers that light is reflected at a certain angle when it strikes a surface, making the lighting of an object appear different depending on how the viewer looks at it.

Methods

The shader for this lab builds off of the basic flat color shader. From the flat color shader, a specular color property is added and recognized by the shader script. A shine factor is also added as a property and variable explicitly named _Shininess so that other functions can call this property. The Unity-defined light color variable is also declared at the start of the shader CG script. The input struct is declared containing only a vertex position and normal direction, and output struct is initially declared with only the pixel position and color.

In the vertex function, we pass an input structure as usual and expect an output structure to be returned. Inside, we first declare the return output struct and calculate the normal direction for the vertices. Next, we calculate the view direction based on where the camera is relative to the vertex position in world space, which is normalized. We calculate light direction by normalizing the world lighting position, and calculate a diffuse reflection based on attenuation, the light color Unity variable, and the maximum value between 0 and the dot product of the vertex normal direction and light direction. With all of this calculated, we can begin calculating the specular reflection property. We begin by calling the reflect function on the light direction and normal direction, taking the opposite of the light direction. We then calculate the dot product between this reflection value and the view direction. This value, if not less than 0, is raised to the power of the shininess property. Finally, the maximum between 0 and the dot product of the normal direction and light direction times the value is calculated. This value plus the diffuse reflection value and Unity's ambient lighting gives the final lighting, which we multiply by the color property and store in the output structure's color property.

However, we further expand upon the above by moving most of the calculating into the fragment function, keeping only the position calculation, normal calculation, and view calculation in the vertex function.

Conclusion

While the lab program did not appear to work for me, I still believe I have a good understanding of how specular shading operates. I would like to build a truly working model at some point to have a working visual to how specular shading operates, but otherwise, conceptually, it makes sense: the viewer will see lighting applied to an object differently dependent on their position.

Post Lab

1. The max(0.0, specularReflection) function softens the color bleed by “telling” the shader that light should not shine on the opposite side of what it's shining on to.
2. The The max(0.0, dot(normalDirection,lightDirection)) function removes the specular highlight from the back of an object by “telling” the shader that light simply isn't showing on that side of the object.