Technical Report

Anisotropic Lighting using HLSL

Anisotropic Lighting Demo

Anisotropic lighting is a lighting technique that does not require that the surface behave the same from different angles. That is to say, over the surface of the object, the lighting is not isotropic. However, calculating anisotropic lighting is often expensive, and difficult to accomplish in real-time.

This demo will show a technique of baking anisotropic lighting into a texture and using a texture load in place of lighting calculations. This technique allows for a palette of custom lighting effects without having a slew of long and complicated shaders.

This technique can be used to create many and varied lighting models, and some fairly interesting effects.

Bryan Dudash
bdudash@nvidia.com

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050

1/23/2004

Introduction

This technique uses ideas from the Blinn lighting equation to calculate and use a half angle H. It generates texture coordinate U from the dot of the light vector and the surface normal. The V texture coordinate is calculated from the dot of the half angle H and the surface normal.   The texture contains both a horizontal color ramp to implement a velvet-like diffuse lighting as well as a bright diagonal in the alpha channel where H dot N and L dot N are equal to simulate a specular highlight. Figure 1 shows the single input texture. The color channel is displayed on the left and the alpha on the right. It is very easy to tweak the ramps to change the overall look of the object. The texture we have is a custom ramp, and so gives us the nice soft velvet effect.
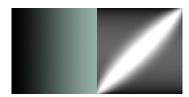

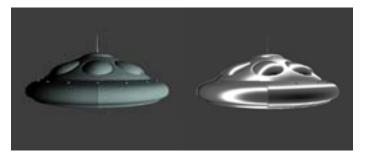*Figure 1: sampled texture. Color and Alpha*

Figure 2: On the left we have the object shaded with the color channel lookup only. It shows the velvet-like lighting. On the right we have the alpha channel only showing the custom specular highlight

The texture coordinates are calculated using a vertex shader and interpolated across each pixel. Then using the standard fixed function pixel pipeline, we combine the color channel lookup and the alpha channel lookup with a *Modulate4x*, and using *AlphaReplicate.*. We multiply by 4 to enhance the specular highlight. Figure 4 shows the final rendering. The resulting color equation is:
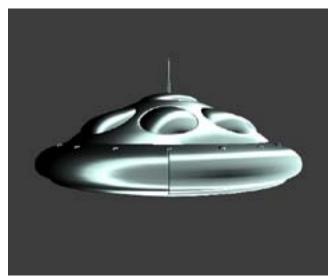
```
Final Color = (Color.rgb * Color.aaa)*4.0
```


Figure 4: Final Scene, steel velvet

Final Thoughts

This technique is interesting because it allows the graphics programmer to implement potentially complex lighting equations by "baking" pre-calculated values into a texture and then simply looking them up by calculating the UV cords. This technique also has the potential to be a performance boost on hardware where texture lookups are fast. If the time to do the texture fetch is less than the math you save, then it's a win.

So the next time your artists come to you with a "silly" lighting idea, you might just be able to easily and efficiently implement it.

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com