Diffraction

Diffraction considers the wavelengths of light and when on small surfaces the reflected waves affect one another.

[](https://developer.nvidia.com/gpugems/GPUGems/elementLinks/fig08-07.jpg)

Figure Retrieved From: https://developer.nvidia.com/gpugems/GPUGems/gpugems\_ch08.html

How to Implement

In order to compute the diffraction pattern the halfway vector, between the light source and the view, is projected onto the tangent vector. From this value and the distance, we can compute the interfering wavelengths. The Red, Blue, and Green components are calculated by bump functions using C which controls the appearance of the color map and y which shows the wavelengths. The sum of the diffraction and the anisotropic highlight, computed by finding the hi-light color, roughness, and the normal direction, is the final output color.

Fresnel Effect

Power

Bias

Scale

Anisotropic Highlight:

Hi-light color

Roughness

RGB Values:

Color Map Appearance

Wavelengths

**Example 8-1. The Diffraction Shader Vertex Program**

**Retrieved from: https://developer.nvidia.com/gpugems/GPUGems/gpugems\_ch08.html**

***float3*** blend3 (***float3*** x)

{

***float3*** y = 1 - x \* x;

y = ***max***(y, ***float3*** (0, 0, 0));

***return*** (y);

}

***void*** vp\_Diffraction (

***in float4*** position : ***POSITION***,

***in float3*** normal : ***NORMAL***,

***in float3*** tangent : ***TEXCOORD0***,

***out float4*** positionO : ***POSITION***,

***out float4*** colorO : ***COLOR***,

***uniform float4x4*** ModelViewProjectionMatrix,

***uniform float4x4*** ModelViewMatrix,

***uniform float4x4*** ModelViewMatrixIT,

***uniform float*** r,

***uniform float*** d,

***uniform float4*** hiliteColor,

***uniform float3*** lightPosition,

***uniform float3*** eyePosition

)

{

***float3*** P = ***mul***(ModelViewMatrix, position).xyz;

***float3*** L = ***normalize***(lightPosition - P);

***float3*** V = ***normalize***(eyePosition - P);

***float3*** H = L + V;

***float3*** N = ***mul***((***float3x3***)ModelViewMatrixIT, normal);

***float3*** T = ***mul***((***float3x3***)ModelViewMatrixIT, tangent);

***float*** u = ***dot***(T, H) \* d;

***float*** w = ***dot***(N, H);

***float*** e = r \* u / w;

***float*** c = ***exp***(-e \* e);

***float4*** anis = hiliteColor \* ***float4***(c.x, c.y, c.z, 1);

***if*** (u < 0) u = -u;

***float4*** cdiff = ***float4***(0, 0, 0, 1);

***for*** (***int*** n = 1; n < 8; n++)

{

***float*** y = 2 \* u / n - 1;

cdiff.xyz += blend3(***float3***(4 \* (y - 0.75), 4 \* (y - 0.5),

4 \* (y - 0.25)));

}

positionO = ***mul***(ModelViewProjectionMatrix, position);

colorO = cdiff + anis;

}