

Surface Reflection Models

Frank Losasso (flosasso@nvidia.com)

Introduction

One of the fundamental topics in lighting is how the **light interacts** with the **environment**. The academic community has researched several models over the last century, but for the most part, we are still stuck with the most basic of models. The reason for this is for the most part the **lack** of **computational power**, making it difficult to **use** more **truthful models in real-time**. The goal of this paper is to investigate the viability of some of the more sophisticated surface reflection models that have been created, giving developers a sense as to how much computational power needs to be used to create **better looking surfaces**.

Traditionally, the most used method of finding the color of a pixel has been through **Gouraud interpolation** of Lambert's Cosine Law at the vertices. This lighting model is extremely cheap to compute and is **amenable** to fixed function pipelines. With the advent of GPUs, Phong interpolation has become a viable option. This model has allowed for innovations like **bump mapping** giving more realism to modern virtual worlds. The underlying assumptions however are still the same; the surfaces are Lambertian. Unfortunately, this has the side effect of making most of the computer generated images look very much alike, and very different from the real world. The Lambertian surface assumption works great for some materials (like plastic), which means that different models must be used if other materials are to be represented faithfully. And this is what we will investigate over the next few pages.

The Lambertian Lighting Models

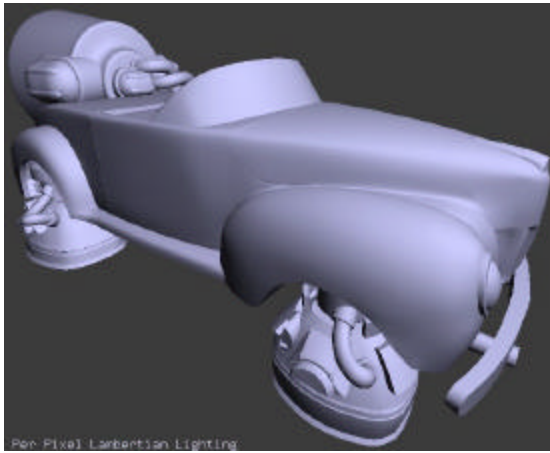
Lambert created this model over a century ago, and as mentioned previously, it is still the most used model (**by a huge margin**) in real time graphics today. Lambert assumed that when light strikes a surface, the light is reflected equally in all directions. The light that strikes any given part of a surface is proportional to the cosine of the incident angle¹.

The Lambertian Diffuse Lighting Equation:

$$I = k_a + k_d f_{att} (N \cdot L)$$

¹ Please see Appendix A for explanation of terms and symbols

The cost of this equation is extremely small, and the results are usable, but it does give the appearance that the lit object is made of rough plastic (no specular component).



Screenshot of Lambertian lighting

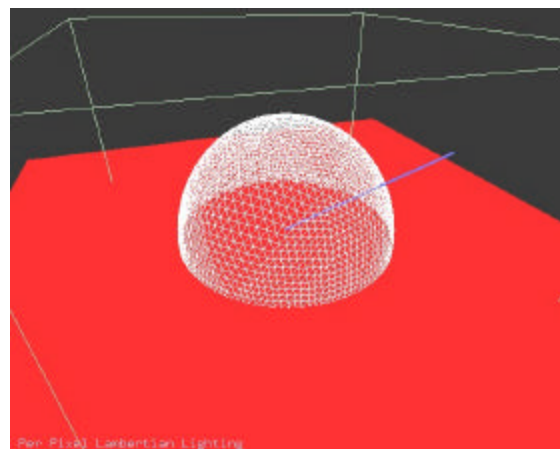
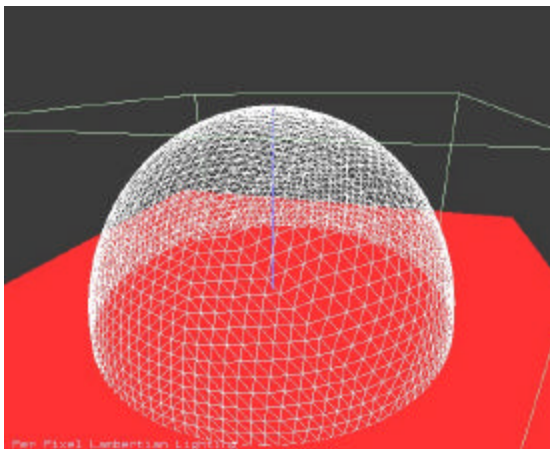


Diagram showing the amount of light emitted in all directions given a light direction (purple line)

The bottom two screenshots give a pictorial description of the behavior of the function. The white wireframe contour indicates how much light is given off in each direction, and the purple line indicates the light vector (the direction from which the light is coming). Notice that when the light is coming from an angle, the amount of light reflected in all directions is less than when the incident angle is close to zero (the light vector is perpendicular to the surface)

The Phong Lighting Model

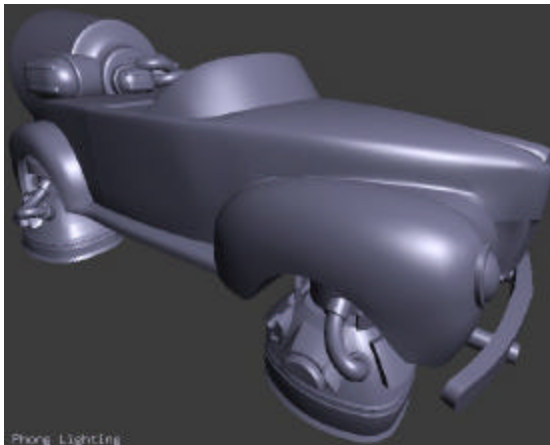
Phong lighting is the other major lighting model that is used in real time rendering, especially after the advent of programmable pipelines. Phong did not change any of Lambert's assumptions, and hence the cosine of the angle between the incident light vector and the surface normal is still used to calculate the diffuse component of the surface. Phong did however create Phong interpolation that involves the interpolation of

the vectors across the faces of the polygons as opposed to colors. The largest benefit of interpolating vectors across the polygon is that accurate specular highlights can be reproduced. The major drawback of calculating the lighting equation at every pixel using these interpolated vectors is of course that a lot more computational power is used.

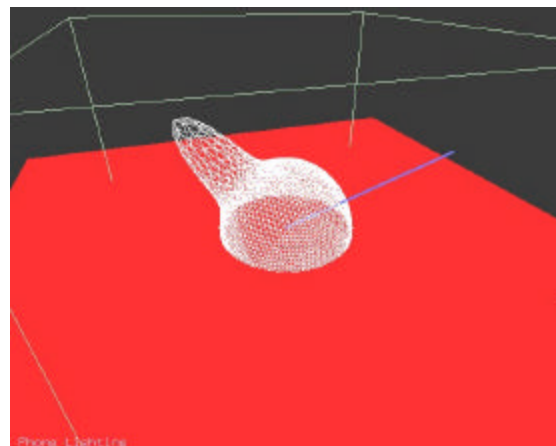
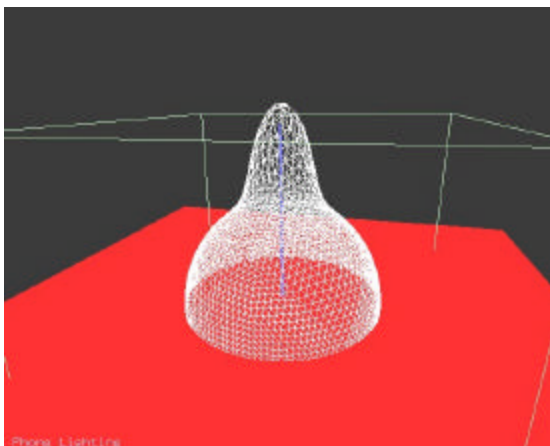
The Phong Lighting Equation:
$I = k_a + k_d f_{att}(N \cdot L) + k_s f_{att}(E \cdot R)^n$

Two other advantages of Phong interpolation are that it is easy to add bump mapping to surfaces, and the visual appearance of the surface is significantly better than per-vertex interpolated lighting.

Objects that are lit by Phong lighting tend to look like various types of plastic (depending on the specular exponent).



Screenshot of Phong lighting



Figures clearly showing that the specular highlight is in the direction of the reflection vector. The diffuse component is the same as the Lambertian diffuse component.

The Cook-Torrance Lighting Model

The model presented by Cook and Torrance, strives for as much **physical realism** as possible, as opposed to the models above that are made entirely **ad-hoc** for computer graphics only, the Cook-Torrance model uses **physical properties**, and has all the nice properties one would expect a BRDF to have (such as energy conservation, complex substructures, and a predictive result).

BRDFs like Cook-Torrance also has the advantage of using parameters with physical **analogies**, meaning that artists can **tweak** the properties of the surface easily to create realistic looking objects.

The Cook-Torrance model assumes that the surface is made up of microscopic perfect Lambertian reflectors called microfacets. Several of the terms in the lighting equation deal with how these microfacets are oriented, masked and shadowed.

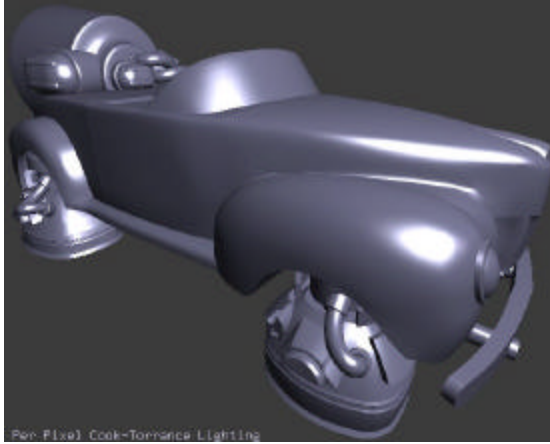
The Cook-Torrance Lighting Equation

$$I = k_a + k_d \mathbf{r}(\mathbf{L} \cdot \mathbf{N}) + k_s \frac{DGF_1(\mathbf{q}_i)}{\mathbf{p}(\mathbf{E} \cdot \mathbf{N})}$$

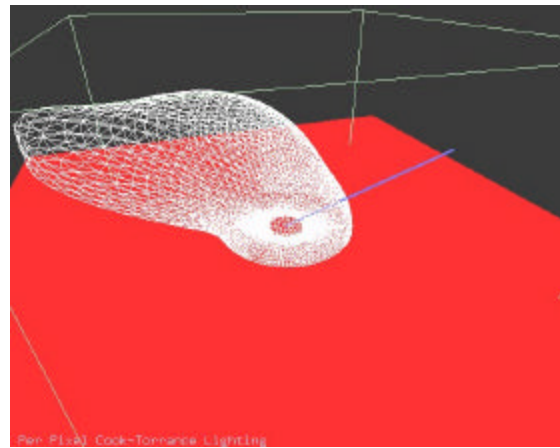
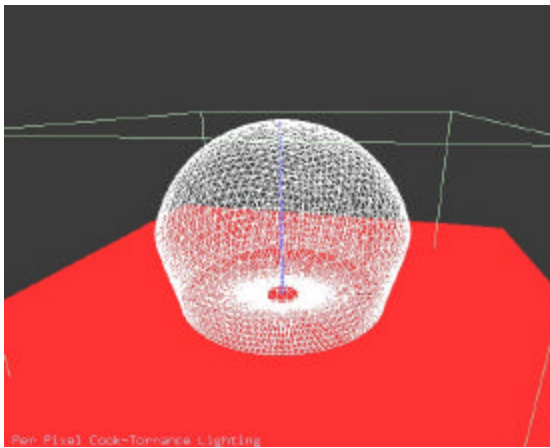
$$\mathbf{r} = k_s f_{att} \frac{DGF_1(\mathbf{q}_i)}{\mathbf{p}(\mathbf{L} \cdot \mathbf{N})(\mathbf{E} \cdot \mathbf{N})}$$

$$D = \frac{e^{-\left(\frac{\tan(\mathbf{b})}{m}\right)^2}}{4m^2 \cos^4(\mathbf{b})}$$

The D term in the equation is the distribution function of the **microfacets** based on the Beckman distribution function. The m parameter is the root-mean-square (RMS) of the slope of the microfacets. This means that a large m makes the reflections spread out (since the average slope of the microfacets is larger). The G term is the geometric attenuation term, which deals with how the individual microfacets shadow and mask each other. The F term is the Fresnel Conductance term that is wavelength dependent (notice the λ), but for computational simplicity, we will assume that it is wavelength independent (and hence only one calculation is needed). The Fresnel Conductance term deals with the amount of light that is reflected versus absorbed as the incident angle changes (an example of this is often seen when driving on a straight road, and the road appearing mirror-like when viewed from grazing angles).



Screenshot of Cook-Torrance lighting



From these diagrams, it is clear that this BRDF is clearly different. Notice the complete change in behavior when the incident angle is large

The Blinn Lighting Model

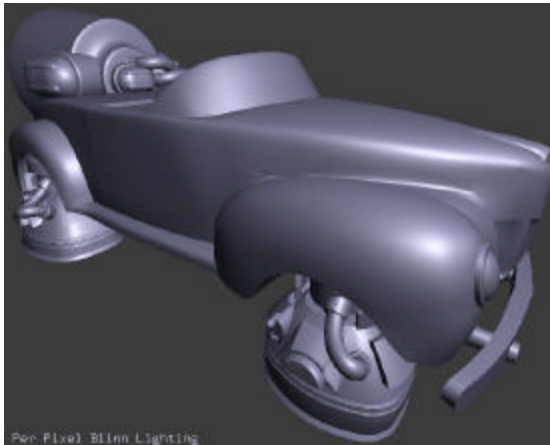
The Jim Blinn model for specular reflection is built on top of the work done in 1967 by Torrance and Sparrow who worked on a model to explain the fact that the specular intensity varies with both the direction of the light source and the direction of the viewer, whereas previous models had ignored the direction of the viewer. The Blinn Model is a modification of the Torrance-Sparrow model that yields similar results, but is significantly cheaper to compute.

The Blinn Lighting Equation:

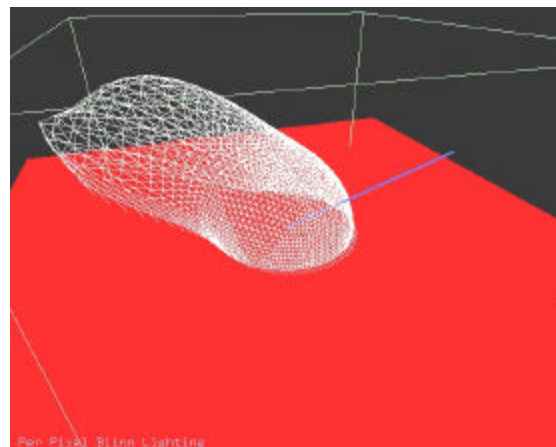
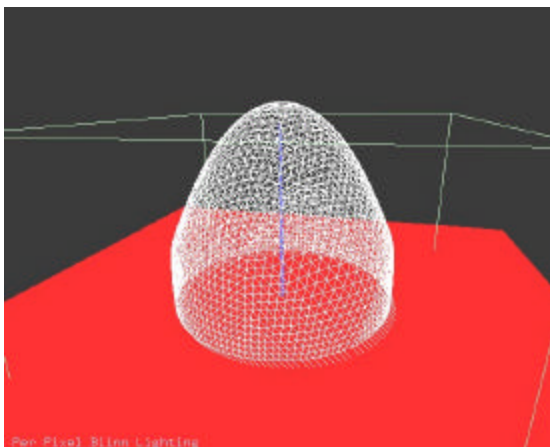
$$I = k_s f_{att} \frac{DGF_1(\mathbf{q}_i)}{(E \cdot N)}$$

$$D = \left(\frac{c^2}{(N \cdot H)^2 (c^2 - 1) + 1} \right)^2$$

The lighting equation has the same form as the Cook-Torrance lighting equation above, with a geometric attenuation factor and a Fresnel conductance term. The distribution function (D) is now significantly easier to compute.



Screenshot of Blinn lighting



Diagrams depicting the Blinn BRDF clearly show that Blinn lighting is similar to Cook-Torrance Lighting

It is evident from the screenshots that it is now possible to make surfaces that look more like metal than plastic using this equation, but without the huge cost of the Cook-Torrance or the Torrance-Sparrow models. Just like all the other per pixel based methods, the per pixel version of this lighting model is also amenable to add-ons such as bump mapping.

The Oren-Nayar Lighting Model

Oren and Nayar created a new BRDF in the hope to ‘generalize’ the Lambertian diffuse lighting model. This BRDF can reproduce several rough surfaces very well, including wall plaster, sand, sand paper, clay, and others. It is however very computationally expensive, and it requires the calculation of azimuth and zenith angles.

The need for a better diffuse lighting model seems very real, and in their original paper, Oren and Nayar presented a clay vase that was rendered using the Lambertian and their proposed models compared to a real image. It is glaringly obvious from that demonstration that the Lambertian model was not suitable for representing certain materials, whereas in this case their model was much better.

This lighting equation can be calculated either per-pixel or per-vertex, and a per-pixel implementation is amenable to bump mapping, although the pixel program may get prohibitively expensive.

The Oren-Nayar Lighting Equation (the simplified qualitative model):

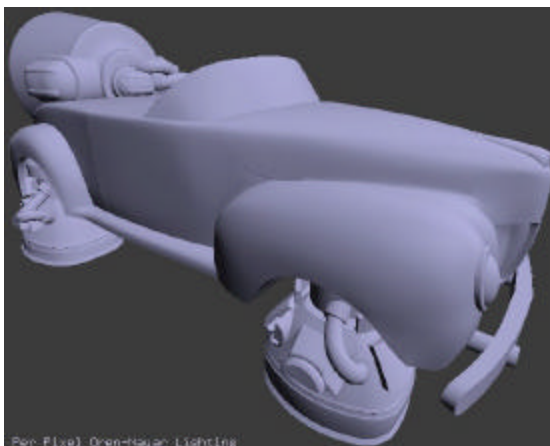
$$I = k_d f_{att} \frac{\mathbf{r}}{\mathbf{p}} E_0 \cos(\mathbf{q}_i) (A + B(\max[0, \cos(\mathbf{f}_r - \mathbf{f}_i)] \sin(\mathbf{a}) \tan(\mathbf{b})))$$

$$A = 1.0 - 0.5 \frac{\mathbf{s}^2}{\mathbf{s}^2 + 0.33}$$

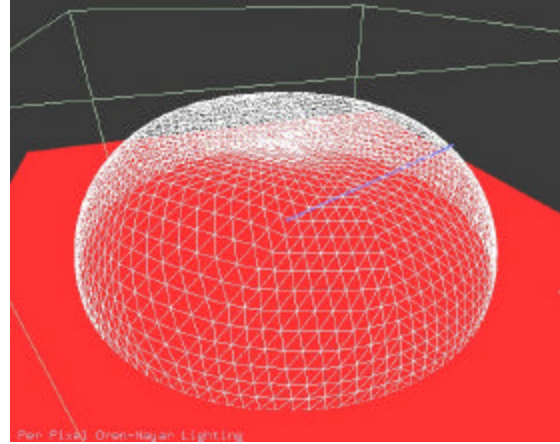
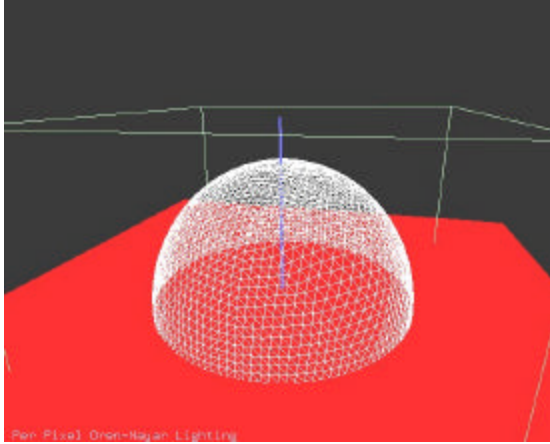
$$B = 0.45 \frac{\mathbf{s}^2}{\mathbf{s}^2 + 0.09}$$

The lighting equation above is simplified model that Oren and Nayar presented in their paper. It ignores terms like inter-reflections, in an effort to make the model cheaper to calculate.

Since hardware like nv30 can calculate the above equation in hardware, at the fragment shader level, no preprocessing is necessary, which is helpful if the light configuration or the geometry is modified.



Screenshot of Oren-Nayar lighting



At low angles of light incidence, the Oren-Nayar approximation is similar to the Lambertian model, but at high angles of incidence, the result is flatter, making the edges brighter.

As the screenshots above show, the benefit to using the Oren-Nayar lighting model over the Lambertian model is probably not worth the increase computational requirements.

The Minnaert Lighting Model

Minnaert added darkening limbs to the lighting equations to make the surface seem darker from certain viewing/lighting directions. This effect is seen in some types of clothing (such as velvet).

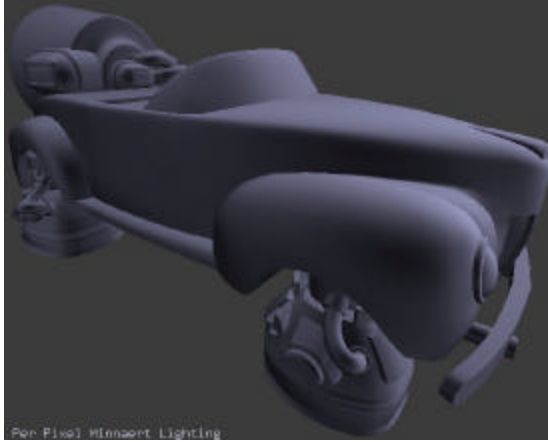
The computational power required to implement Minnaert style lighting is not very high, and can easily be optimized through the use of a texture lookup.

The Minnaert Lighting Equation:

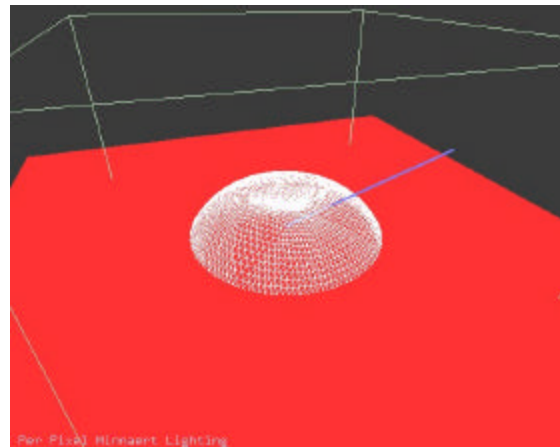
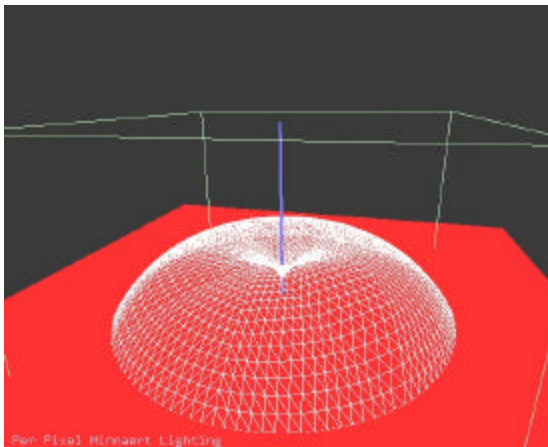
$I = k_d f_{att} (N \cdot L) * (N \cdot L)^k (1 - N \cdot E)^{1-k}$

Note that the first dot product can be combined with the second making the exponent $k+1$. The first part of the equation is simply Lambertian lighting, which is then modulated by the darkening factor.

This lighting model is well suited for bump mapping in it's per-pixel variant.



Screenshot of Minnaert lighting



It is clear from the diagrams that the darkening limbs bring the amount of light reflected towards zero when the viewer looks onto the surface at perpendicular angles

Ward's Anisotropic Lighting Model

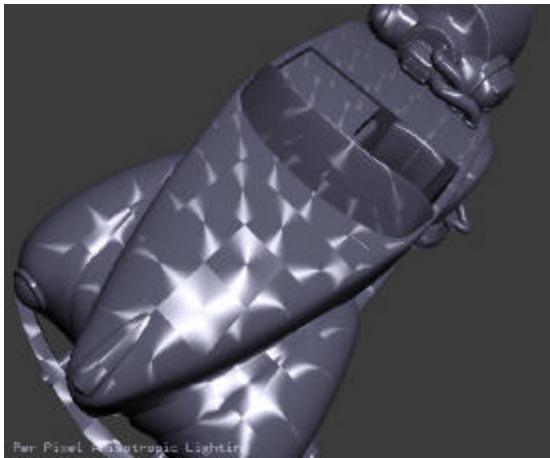
An isotropic surface has the property that for any given point on the surface, the light reflected does not change when the surface is rotated about the normal. This is the case for many materials, but some materials such as brushed metal or hair this is not the case. The reason for these *anisotropic* surfaces is that the micro facets that make up the surface have a preferred direction in the form of parallel grooves or scratches. There are several ad-hoc models for lighting anisotropic surfaces that have been developed for use in real time graphics. Other nVidia demos for anisotropic lighting use a texture lookup based on the cosine of the angle between the surface normal and the light vector one axis, and the cosine of the angle between the surface normal and the half angle vector on the other axis. If the texture map has a bright line down the diagonal, then the surface will be bright when those two values are approximately the same. The approach presented here does not use any textures, but is instead based on the BRDF introduced by Greg Ward Larson in 1992.

The Ward Anisotropic Reflection Model:

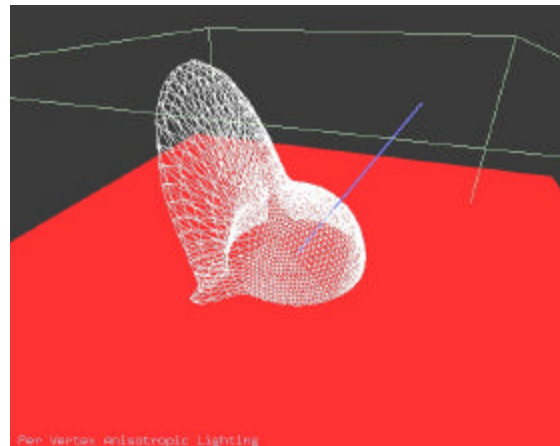
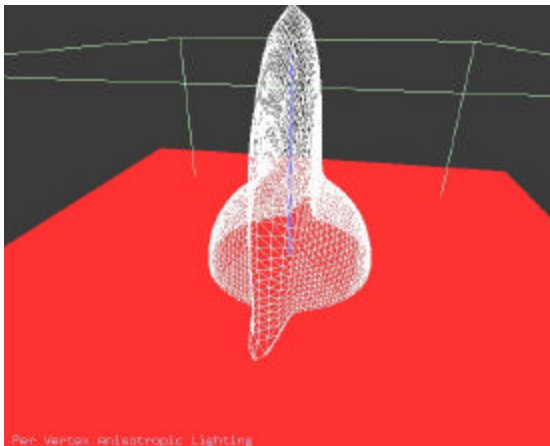
$$I = k_d f_{att}(N \cdot L) + k_s f_{att} \frac{1}{\sqrt{(N \cdot L)(N \cdot E)}} \frac{(N \cdot L)}{4a_x a_y} e^{-2 \frac{\left(\frac{H \cdot X}{a_x}\right)^2 + \left(\frac{H \cdot Y}{a_y}\right)^2}{1 + (H \cdot N)}}$$

The X and Y terms are two perpendicular tangent directions on the surface. They give represent the direction of the grooves in the surface. The α terms are the standard deviations of the slope in the X and Y direction (given by their respective subscripts).

Proper tessellation is essential when per vertex calculations are used; otherwise per pixel calculations should be used.



Screenshot of Ward's Anisotropic lighting (The grooves in the surface were made to look like circular brushed metal patterns)



The diagrams of the function clearly show that the lobe of reflected light is oriented perpendicularly to the groove in the surface (the groove is oriented with the wireframe box in the left-right direction)

Environment Mapping Based Lighting Models

A lot of research has been done recently on environment map based lighting. Most techniques are concerned with different ways of creating the environment map, to create a realistic looking lighting model. The nice things about environment maps is that it's all precomputed and as long as the lights are static, and the surface being lit doesn't change position too much, environment mapping is a very good looking, and very cheap way of lighting objects.

The general idea of environment map based lighting is that all of the lighting computations are done off-line for all directions at a given point, P. The data is then stored in a cube map, which is used to fetch the light value for any point on a surface that is approximately located where the original point P was. This means that incredibly complex and expensive lighting solutions can be computed, and then cheaply applied to an object in real-time when required.

There are several drawbacks to the method as well, many of which should be apparent from the paragraph above. If the surface being lit is not at the location where the point P was, then the cube map may be useless, and creating a new one may be prohibitively expensive. If any of the lights lighting the scene are moved, then the cube map will also have to be computed, so in general, the technique is only great for a complex object that is relatively far away from static lights in comparison to it's size and the movements it will make.

Appendix A

The following table gives each symbol used in this paper and an explanation.

Symbol	Name	Explanation
I	Intensity	The output intensity from the lighting equation. This is the final intensity of the fragment.
k_d	Diffuse Reflection Coefficient	The fraction of light that is reflected through diffuse reflection
k_s	Specular Reflection Coefficient	The fraction of light that is reflected through specular reflection (note that the diffuse and specular coefficients should add up to one)
N	Surface Normal	Normalized direction vector that is perpendicular to the surface
L	Light Vector	Normalized direction vector that points from the surface point to the light location
E	Eye Vector	Normalized direction vector that points from the surface to the viewer (camera location)
H	Half Vector	Normalized direction vector that is the average of the light and viewing vector (normalize (L+V))

R	Reflection Vector	Normalized vector that is in the direction of the Light Vector, rotated around the Normal Vector 180 degrees. Calculated as follows: $R = 2(L \cdot N)N - L$
n	Specular Exponent	The specular exponent determines how 'sharp' the specular highlight is (higher is 'sharper')
c	Ellipsoid Eccentricity	The eccentricity of the micro facets (0 for very shiny, 1 for very dull)
c_1, c_2, c_3	Attenuation Coefficients	The coefficients for constant, linear and quadratic attenuation of the light source, respectively.
d	Light Distance	The distance to the light
G	Geometric Attenuation	The attenuation factor due to self-shadowing and masking of micro facets on the surface. Calculated as follows: $G = \min \left\{ 1, \frac{2(N \cdot H)(N \cdot E)}{E \cdot H}, \frac{2(N \cdot H)(N \cdot L)}{V \cdot H} \right\}$
F	Fresnel Conductance Term	Fresnel determines the amount of reflection of the surface (increases as the zenith angle becomes larger). Calculated as follows: $F = \frac{1}{2} \frac{(g - c)^2}{(g + c)^2} \left[1 + \frac{(c(g + c) - 1)^2}{(c(g - c) + 1)^2} \right]$ $c = \cos(\mathbf{q}_i) = L \cdot H$ $g = \sqrt{n^2 + c^2} - 1$ For simplification, the Fresnel conductance term can also be approximated by the following (much cheaper) equation: $F = (1 - (1 - (N \cdot E))^5) + n(1 - (N \cdot E))^5$
f_{att}	Attenuation Term	The fraction of light that reaches the surface as an effect of light attenuation. Calculated as follows: $f_{att} = \min\left(\frac{1}{c_1 + c_2 d + c_3 d^2}, 1\right)$ Note that in real life, light is attenuated with the square of the distance, but often, the heuristic (hack) above works better.

Appendix B

The following table shows the relative costs of the different lighting models in terms of vertex shaders, pixel shaders, and texture lookups.

