Game Architecture

3/1/16: Lighting

Old School Lighting

Ambient

Ambient + Diffuse

Ambient + Diffuse + Specular

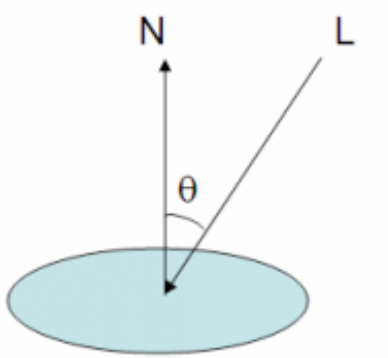
Diffuse

- The radiant intensity observed from a surface or a radiator is directly proportional to the cosine of the angle θ between the direction of the incident light and the surface normal

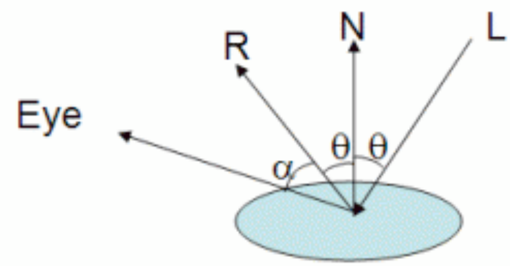
$$I_o = L_d M_d cos\theta$$

reflected intensity   light diffuse color   material diffuse coefficient

Specular



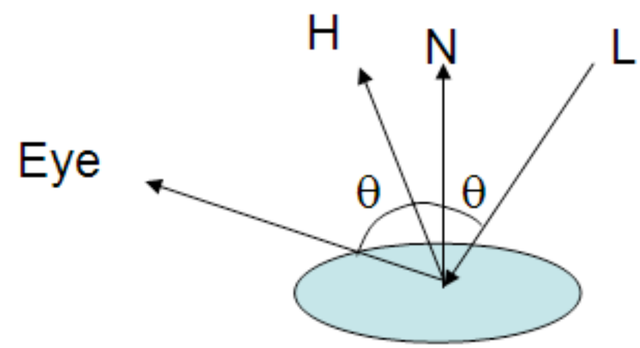
Lambert



Phong

$$R = -2N(L \cdot N) + L$$

$$\text{Spec} = L_S M_S (R \cdot \text{eye})^S$$



Blinn-Phong

$$H = \text{eye} - L$$

$$\text{Spec} = L_S M_S (N \cdot H)^S$$

Eye Space

- Most lighting calculation involves the relationship between the location of the light, the location of the model, and the location of the camera (eye)
- Often necessary to transform the incoming vertex into eye coordinates (ec)

```
ecPosition = ViewMatrix * ModelMatrix * in_Position;
```

Eye Space

- Incoming normals must also be transformed, so we store the normal matrix transformation as a uniform variable

```
tnorm = NormalMatrix * in_Normal;
```

Normal Matrix Transformation?

- Why don't we just do what we did with the in_Position? In other words, why not:

```
tnorm = ViewMatrix * ModelMatrix * in_Normal;
```

Two Things:

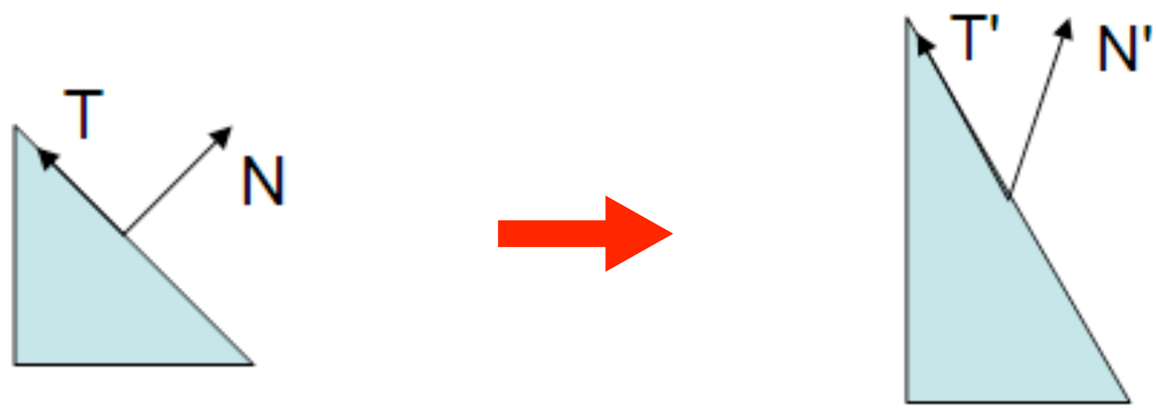
- ModelMatrix is a 4x4 and in_Normal is a vec3
- We want a vec3 output, since a normal is just a direction (or a vec4 with a w of 0)
- Okay, so how about:

```
tnorm = vec3(ViewMatrix * ModelMatrix * vec4(in_Normal, 0.0));
```

- This works *most* of the time

Except...

- Non-uniform scaling can screw things up



- Bottom line, if you're going to allow non-uniform scaling, use:

$$\mathbf{N} = (\mathbf{M}_{3\times 3}^{-1})^T$$

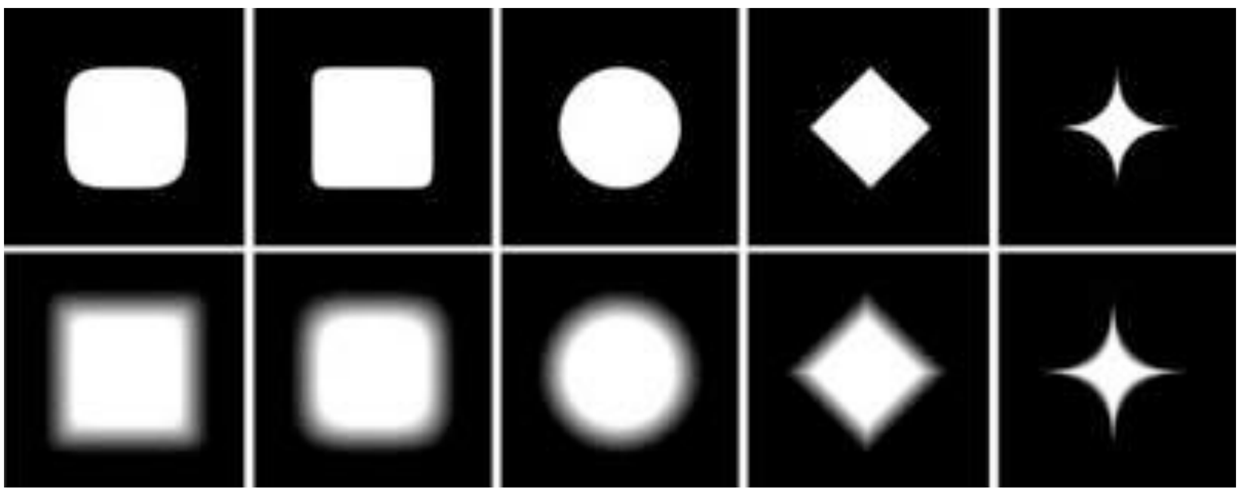
Der Überlichtshatten

$$\left(\frac{x}{a}\right)^{\frac{2}{d}} + \left(\frac{y}{b}\right)^{\frac{2}{d}} = 1$$

0

edge width
(smoothstepped
between ellipses)

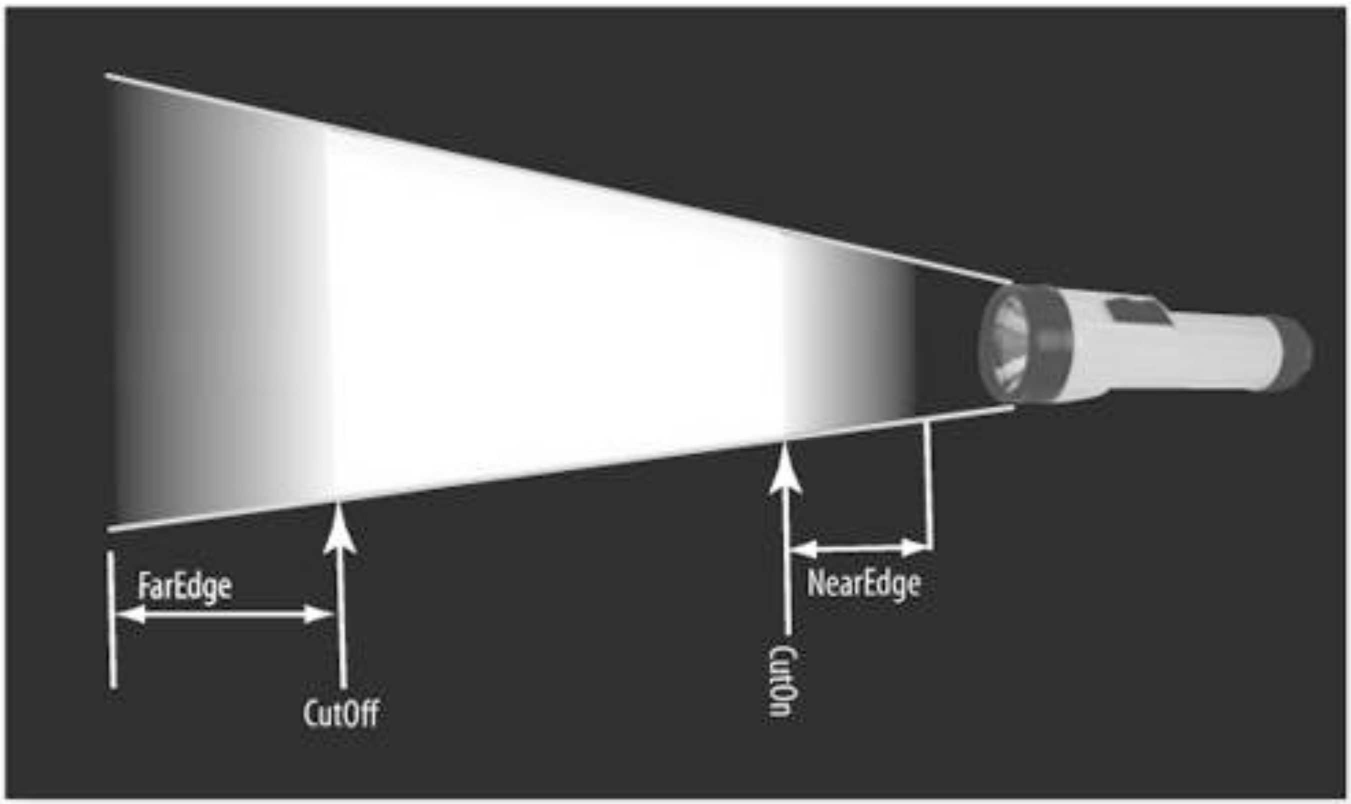
0.3



0.2 0.5 1.0 2.0 4.0

roundness (d)

The Überlight Shader



Homework 6

- Uniforms locations are for a program object, not an individual shader (i.e., you can share a uniform between vertex and fragment shaders)
- The world in light coordinates is the inverse of light in world coordinates
- Remember that ModelView is always changing!
- Due 3/11