Lecture 9

Shading (coloring, lighting)

Shading

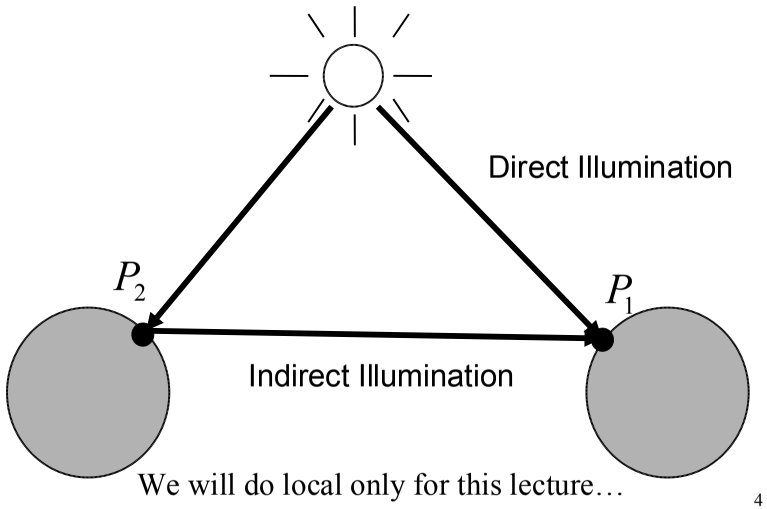
Determining the light traveling from a point in the scene to the viewer's eye

Images courtesy of Watt, Watt & Watt, and Foley & van Dam

Local versus Global Illumination

- Local Illumination
  - Only considers direct illumination
  - No reflection
  - No refraction
  - Shadows possible
- Global Illumination
  - Considers indirect illumination
  - Reflection
  - Refraction
  - Shadows

Local versus Global Illumination



We will do local only for this lecture…

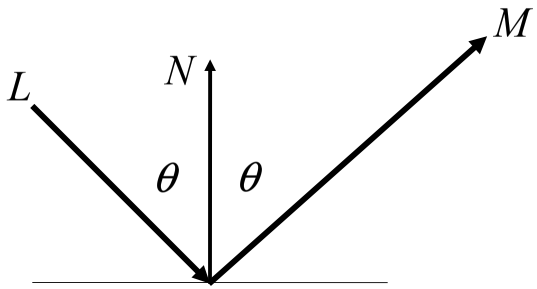
Local versus Global Illumination


Images courtesy of Francois Sillion

To understand shading properly, we need to review some basic notions of physics…

Ideal Specular Reflection

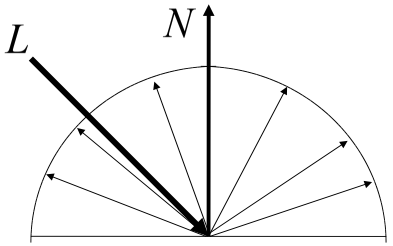
Reflection is only in the mirror direction



Ideal Diffuse Reflection

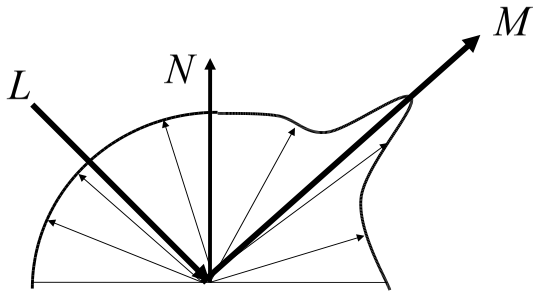
Reflection is equal in all directions

$$f(x, \omega_i, \omega_r) = \rho_{diffuse}$$



Directional Diffuse Reflection

Reflection is concentrated around the mirror direction



Ambient Light

If a surface is visible from the eye, but not a light, it will be rendered black (if indirect light is not considered).

Ambient light is an approximation to indirect light

- Difficult to compute
- Approximate this as a constant
- Or a light source at the eye

Phong Reflection

Assume point lights and direct illumination only

$$I = I_{ambient} + I_{diffuse} + I_{specular}$$

Phong Diffuse Component

Diffuse component depends only on incident angle.

$$I_{diffuse} = I_l \; k_d \cos\theta$$

$$= I_l \; k_d (N \cdot L)$$

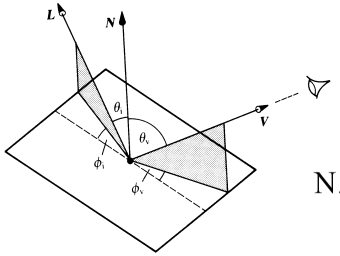


Image courtesy of Watt, 3D Computer Graphics

N.B: L and N are unit…

Phong Specular Component

Phong combines directional diffuse & ideal specular

$$I_{specular} = I_l \; k_s \cos^n \Phi$$

$$= I_l \; k_s (R \cdot V)^n$$

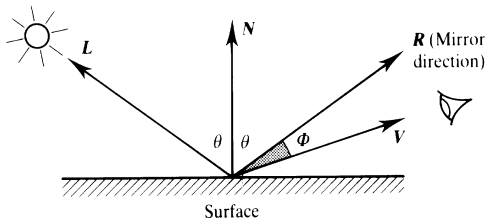


Image courtesy of Watt, 3D Computer Graphics

Phong Specular Component

We reparameterize this as:

$$I_{specular} = I_l \, k_s (N \cdot H)^n$$

$$H = |L + V| \quad \text{(renormalized)}$$

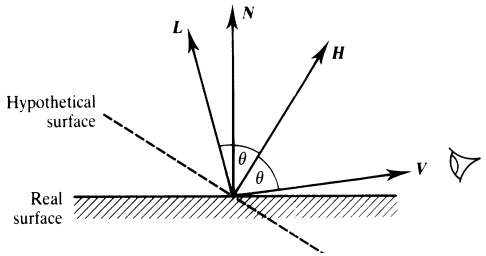


Image courtesy of Watt, 3D Computer Graphics

Phong Ambient Component

Treat it as a constant:

$$I_{ambient} = I_a \, k_a$$

Where I_a is the ambient light in the scene.

Adding Color

$$\begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix} = \begin{bmatrix} I_{a,r} k_{a,r} \\ I_{a,g} k_{a,g} \\ I_{a,b} k_{a,b} \end{bmatrix} + (N \cdot L) \begin{bmatrix} I_{l,r} k_{d,r} \\ I_{l,g} k_{d,g} \\ I_{l,b} k_{d,b} \end{bmatrix} + (N \cdot H)^n \begin{bmatrix} I_{l,r} k_{s,r} \\ I_{l,g} k_{s,g} \\ I_{l,b} k_{s,b} \end{bmatrix}$$

Adding Lights

$$\begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix} = \begin{bmatrix} I_{a,r} k_{a,r} \\ I_{a,g} k_{a,g} \\ I_{al,b} k_{a,b} \end{bmatrix} + \sum_{l=lights} \left((N \cdot L) \begin{bmatrix} I_{l,r} k_{d,r} \\ I_{l,g} k_{d,g} \\ I_{l,b} k_{d,b} \end{bmatrix} + (N \cdot H)^n \begin{bmatrix} I_{l,r} k_{s,r} \\ I_{l,g} k_{s,g} \\ I_{l,b} k_{s,b} \end{bmatrix} \right)$$

$$I = k_a \, I_a + \sum_{l=lights} \left(k_d (N \cdot L) + k_s (N \cdot H)^n \right) I_l$$

Phong Reflection

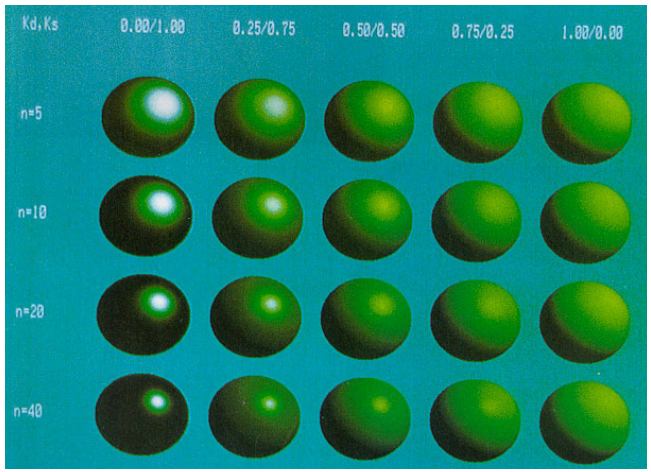
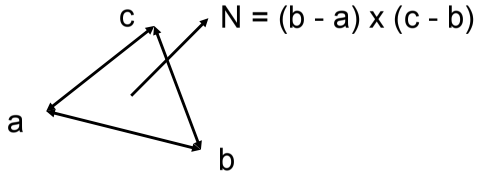


Image courtesy of Watt, 3D Computer Graphics

Vertex Normals vs. Face Normals

What are the normals to the surface?

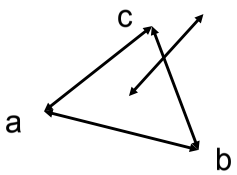
Each polygonal face has a normal.



We call these face normals.

Flat Shading

Assume a constant color across the polygon



Uses face normals

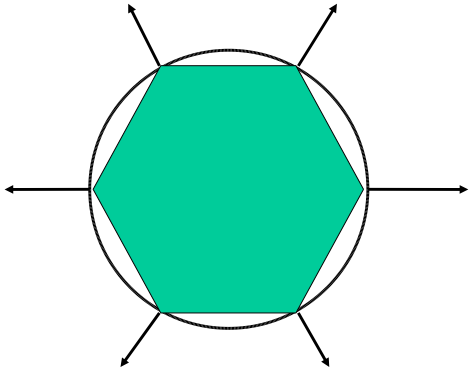
Equivalent to single point sampling…

Polygon mesh is only an approximation.

Can we do better?

Vertex Normals vs. Face Normals

Should use the actual surface's normals



Usually stored at the vertices of the object

Can calculate as averages of face normals

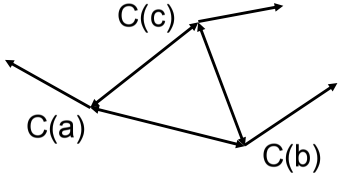
Interpolation

Given vertex normals, how to color the interior:

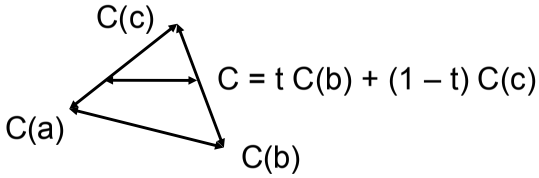
- Gouraud Interpolation
- Phong Interpolation

Gouraud Interpolation

Calculate the color at each vertex

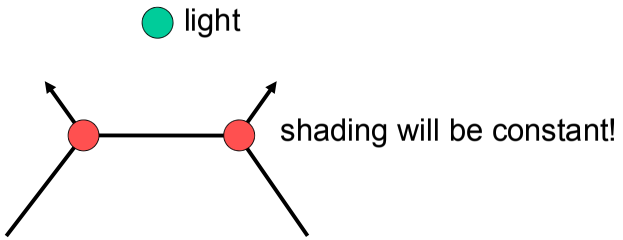


Interpolate the colors



Gouraud Interpolation Problems

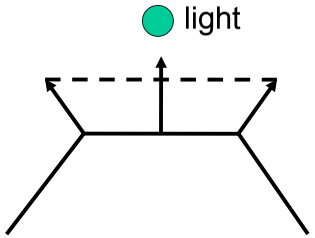
Misses some highlights



Shading is not linear

Phong Interpolation

Interpolate the normals, then compute the colors



Interpolation is usually done component-wise

Interpolation

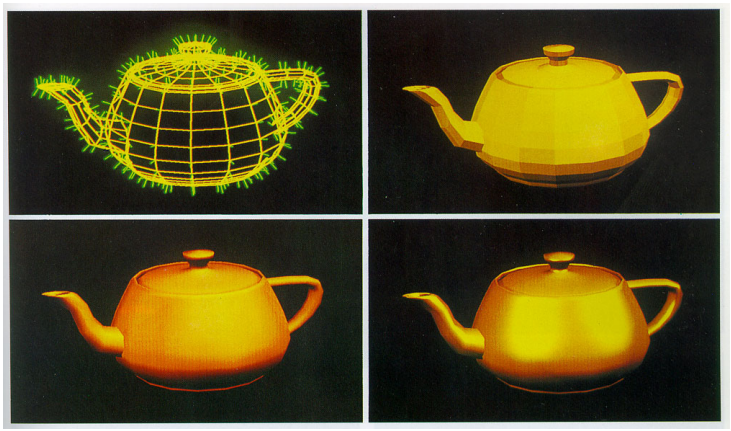
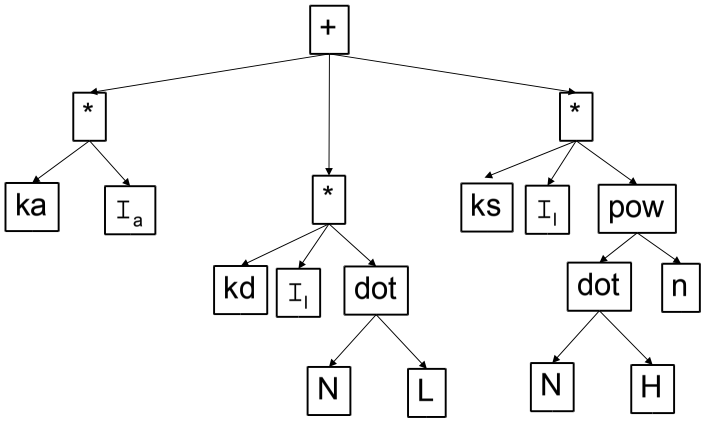


Image courtesy of Watt & Watt, Advanced Animation and Rendering Techniques

Shade Trees

Phong Shade Tree:

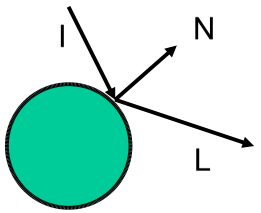


Shading Language

A language for implementing shading models

State is passed to/from the shader by global variables
Ci – Outgoing ray color
Oi – Outgoing ray opacity

Cs – Surface color
Os – Surface opacity
P – Surface point
N - Surface normal
I - Direction of viewing (eye ray)
L - Direction to the light source



Shading Language

Phong Surface Shader:

```
surface
plastic(float Ks = .5, float Kd = .5, float Ka = 1,
        float roughness = .1, color specularColor = 1)
{
  point Nf = faceforward(N,I);

  Oi = Os;
  Ci = Os*(Cs*(Ka*ambient()+Kd*diffuse(Nf)
       + specularColor*Ks*specular(Nf,-I,roughness);
}
```