


Objectives

- Learn to shade objects so their images appear three-dimensional
- Introduce the types of light-material interactions
- Build a simple reflection model---the Phong model--- that can be used with real time graphics hardware

Angel: Interactive Computer Graphics 5E © Addison-Wesley 2009

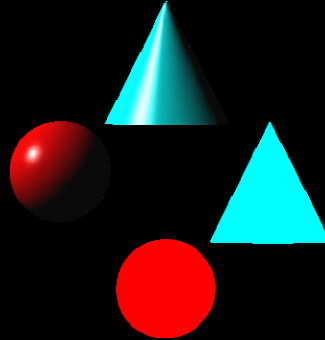


A presentation slide with a black background. In the top right corner, there is a green circular logo with a white ring inside, and the text "SIGGRAPH2006" below it. A horizontal dotted line is positioned below the logo. The word "Objectives" is written in a large, white, sans-serif font in the center of the slide. Below it, there are three bullet points in white text. At the bottom left, there is a small white text line: "Angel: Interactive Computer Graphics 5E © Addison-Wesley 2009". In the bottom right corner, there is a white rectangular logo with the text "OpenGL" in blue.

Lighting Principles



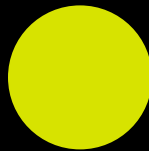
- Lighting simulates how objects reflect light
 - material composition of object
 - light's color and position
 - global lighting parameters
 - ambient light
 - two sided lighting
 - available in both color index and RGBA mode



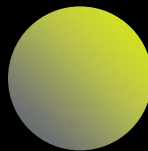
Why we need shading



- Suppose we build a model of a sphere using many polygons and color it with `glColor`. We get something like



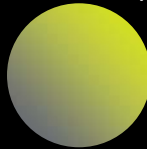
- But we want



Shading



- Why does the image of a real sphere look like



- Light-material interactions cause each point to have a different color or shade
- Need to consider
 - Light sources
 - Material properties
 - Location of viewer
 - Surface orientation

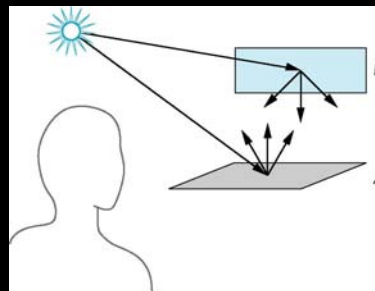
Angel: Interactive Computer Graphics 5E © Addison-Wesley 2009



Scattering



- Light strikes A
 - Some scattered
 - Some absorbed
- Some of scattered light strikes B
 - Some scattered
 - Some absorbed
- Some of this scattered light strikes A and so on



Angel: Interactive Computer Graphics 5E © Addison-Wesley 2009



Rendering Equation

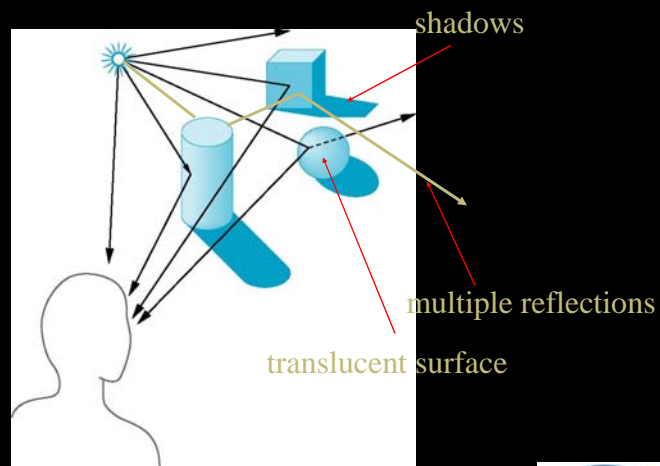


- The infinite scattering and absorption of light can be described by the *rendering equation*
 - Cannot be solved in general *analytically*
 - Ray tracing is a special case for perfectly reflecting surfaces ✓
- Rendering equation is global and includes
 - Shadows
 - Multiple scattering from object to object

Angel: Interactive Computer Graphics 5E © Addison-Wesley 2009



Global Effects



Angel: Interactive Computer Graphics 5E © Addison-Wesley 2009



Local vs Global Rendering



- Correct shading requires a global calculation involving all objects and light sources
 - Incompatible with pipeline model which shades each polygon independently (local rendering)
- However, in computer graphics, especially real time graphics, we are happy if things “look right”
 - Exist many techniques for approximating global effects

Angel: Interactive Computer Graphics 5E © Addison-Wesley 2009



Light-Material Interaction



- Light that strikes an object is partially absorbed and partially scattered (reflected)
- The amount reflected determines the color and brightness of the object
 - A surface appears red under white light because the red component of the light is reflected and the rest is absorbed
- The reflected light is scattered in a manner that depends on the smoothness and orientation of the surface

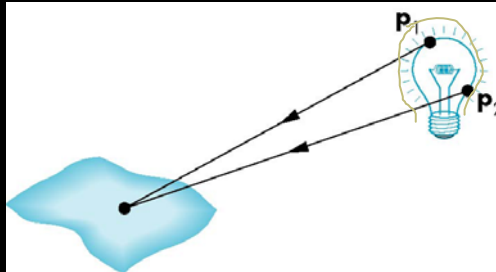
Angel: Interactive Computer Graphics 5E © Addison-Wesley 2009



Light Sources



General light sources are difficult to work with because we must integrate light coming from all points on the source



Angel: Interactive Computer Graphics 5E © Addison-Wesley 2009



Simple Light Sources



* • Point source $(x, y, z, w)^T$

$w \neq 0$ * – Model with position and color

$w = 0$ – Distant source = infinite distance away (parallel)

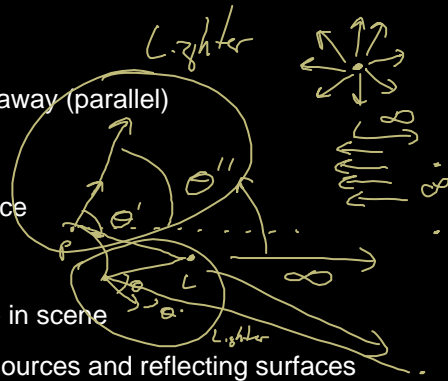
• Spotlight

– Restrict light from ideal point source

• Ambient light

– Same amount of light everywhere in scene

– Can model contribution of many sources and reflecting surfaces



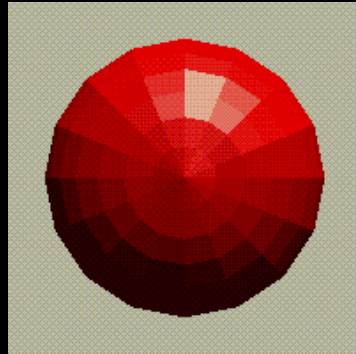
Angel: Interactive Computer Graphics 5E © Addison-Wesley 2009



Shading Schemes



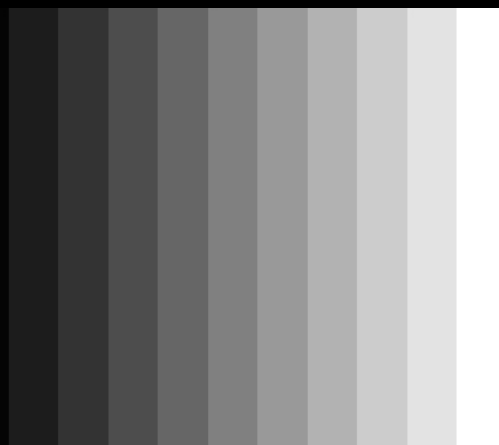
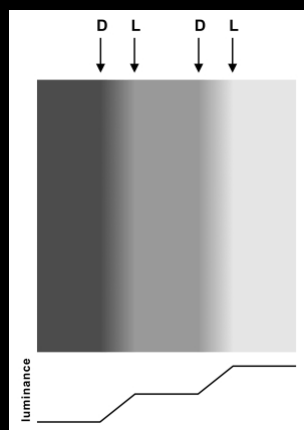
Flat Shading: same shade to entire polygon



CS5600



Mach Band Illusion



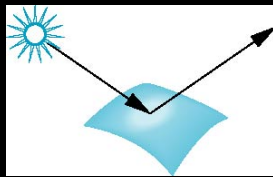
CS5600



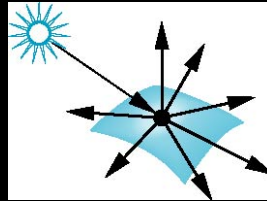
Surface Types



- The smoother a surface, the more reflected light is concentrated in the direction a perfect mirror would reflected the light
- A very rough surface scatters light in all directions



smooth surface



rough surface

Angel: Interactive Computer Graphics 5E © Addison-Wesley 2009

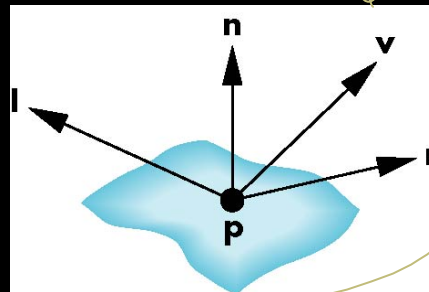


Phong Model

Local



- A simple model that can be computed rapidly
- Has three components
 - Diffuse
 - Specular
 - Ambient
- Uses four vectors
 - To source
 - To viewer
 - Normal
 - Perfect reflector



Angel: Interactive Computer Graphics 5E © Addison-Wesley 2009



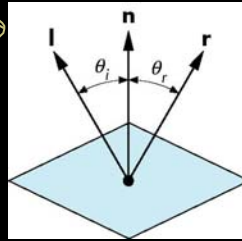
Ideal Reflector



- Normal is determined by local orientation
- Angle of incidence = angle of reflection
- The three vectors must be coplanar ↙

$$\mathbf{r} = 2(\mathbf{l} \cdot \mathbf{n})\mathbf{n} - \mathbf{l}$$

Computed



Angel: Interactive Computer Graphics 5E © Addison-Wesley 2009



Lambertian Surface



- Perfectly diffuse reflector
- Light scattered equally in all directions
- Amount of light reflected is proportional to the vertical component of incoming light

– reflected light $\sim \cos \theta_i$

→ $\cos \theta_i = \mathbf{l} \cdot \mathbf{n}$ if vectors normalized

– There are also three coefficients, k_r, k_b, k_g that show how much of each color component is reflected

Material



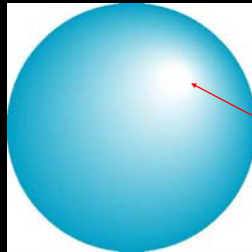
Angel: Interactive Computer Graphics 5E © Addison-Wesley 2009



Specular Surfaces



- Most surfaces are neither ideal diffusers nor perfectly specular (ideal reflectors)
- Smooth surfaces show specular highlights due to incoming light being reflected in directions concentrated close to the direction of a perfect reflection



specular highlight

Angel: Interactive Computer Graphics 5E © Addison-Wesley 2009



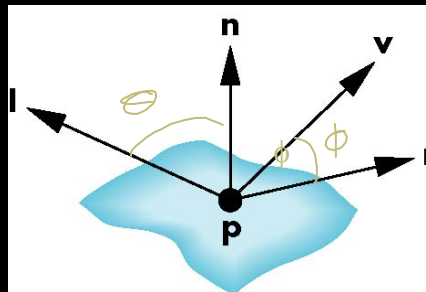
Modeling Specular Reflections



- Phong proposed using a term that dropped off as the angle between the viewer and the ideal reflection increased

$$I_r \sim k_s I \cos^\alpha \phi$$

reflected intensity absorption coef shininess coef
incoming intensity



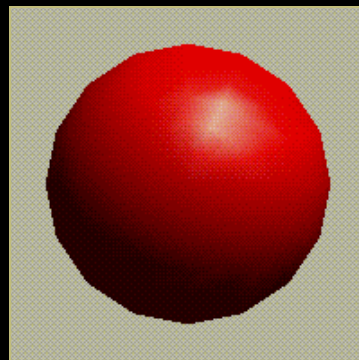
Angel: Interactive Computer Graphics 5E © Addison-Wesley 2009



Shading Schemes



Gouraud Shading:
smoothly blended
intensity across
each polygon



CS5600



Shading Schemes



Phong Shading:
interpolated
normals to
compute intensity
at each point



Bui Toung Phong Thesis

CS5600



Scan Convert Polygon P

The diagram shows a triangle P on a grid. The vertices are labeled I_1 , I_2 , and I_3 . A horizontal scan line y_s intersects the triangle. The intersection points are labeled I_a and I_b . A pixel I_p is shown at the intersection of the scan line and the triangle. The diagram is labeled CS5600 and includes the SIGGRAPH2003 and OpenGL logos.

Intensity Interpolation

I_1, I_2, I_3 : Compute by direction evaluation of *illumination expression*, whichever formula is being used

The diagram shows a triangle with vertices I_1 , I_2 , and I_3 . Green arrows point from each vertex towards the center of the triangle, indicating the direction of intensity interpolation. The diagram is labeled CS5600 and includes the SIGGRAPH2003 and OpenGL logos.

Surface Normals



- Normals define how a surface reflects light

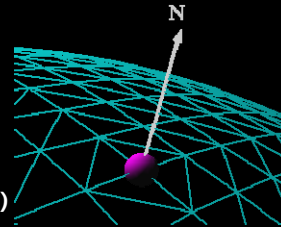
`glNormal3f(x, y, z)`

- Current normal is used to compute vertex's color
- Use *unit* normals for proper lighting
 - scaling affects a normal's length

`glEnable(GL_NORMALIZE)`

or

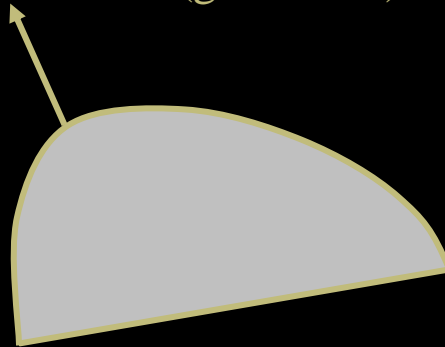
`glEnable(GL_RESCALE_NORMAL)`



Using Average Normals

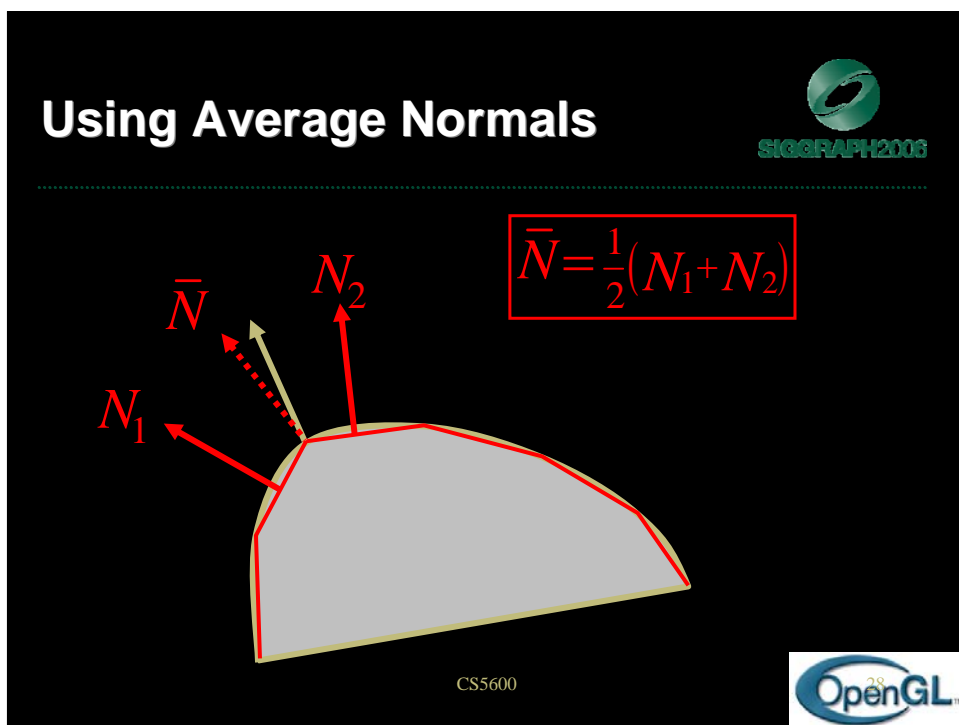
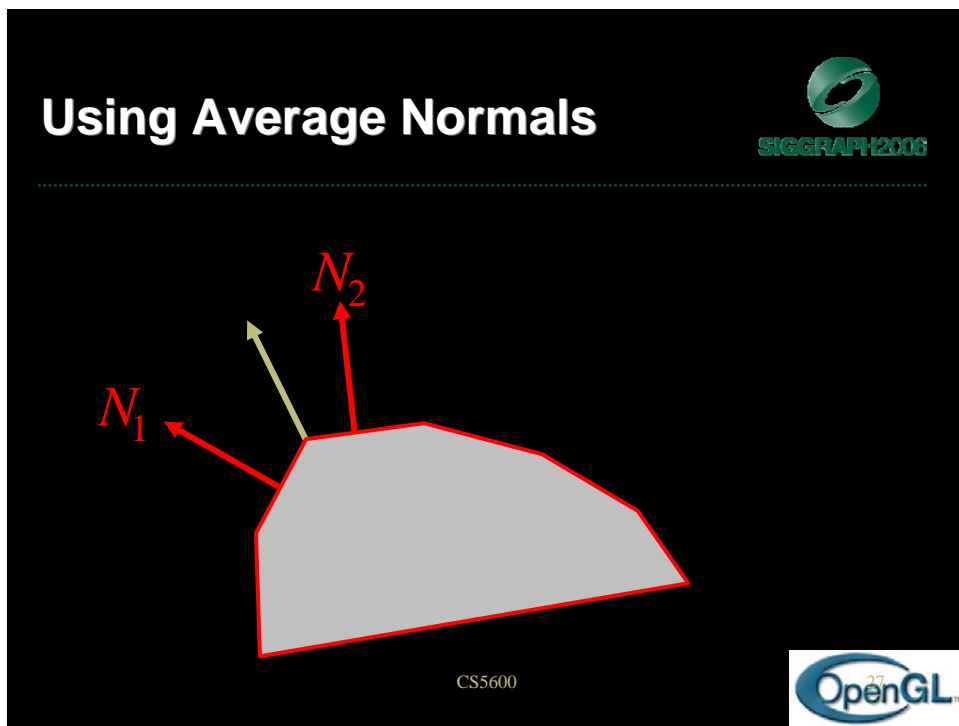


$N = \text{true (geometric) normal}$

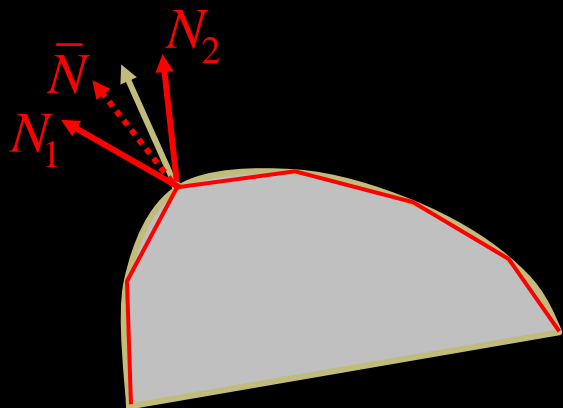


CS5600





Using Average Normals



CS5600



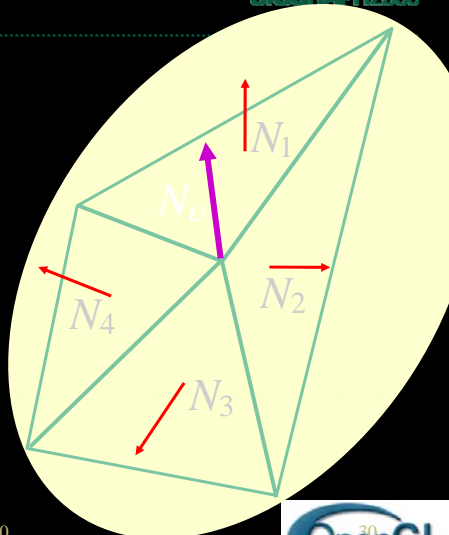
What should corner normals be?



$$N_v = \frac{(N_1 + N_2 + N_3 + N_4)}{\|N_1 + N_2 + N_3 + N_4\|}$$

More generally,

$$N_v = \frac{\sum_{i=1}^n N_i}{\left\| \sum_{i=1}^n N_i \right\|}$$



CS5600



Relevant Light (unit) Vectors

Surface Normal

Point light source direction

Reflection direction

Viewpoint direction

The diagram illustrates the geometry of light reflection on a surface. A green oval represents the surface. A red vector N points vertically upwards from the center of the surface, representing the surface normal. Three other red vectors originate from the same point: L (point light source direction) pointing to the left, R (reflection direction) pointing to the right, and V (viewpoint direction) pointing downwards and to the right. The angle between N and L is labeled θ . The angle between N and R is also labeled θ . The angle between N and V is labeled α . The angle between L and V is labeled ϕ .

CS5600

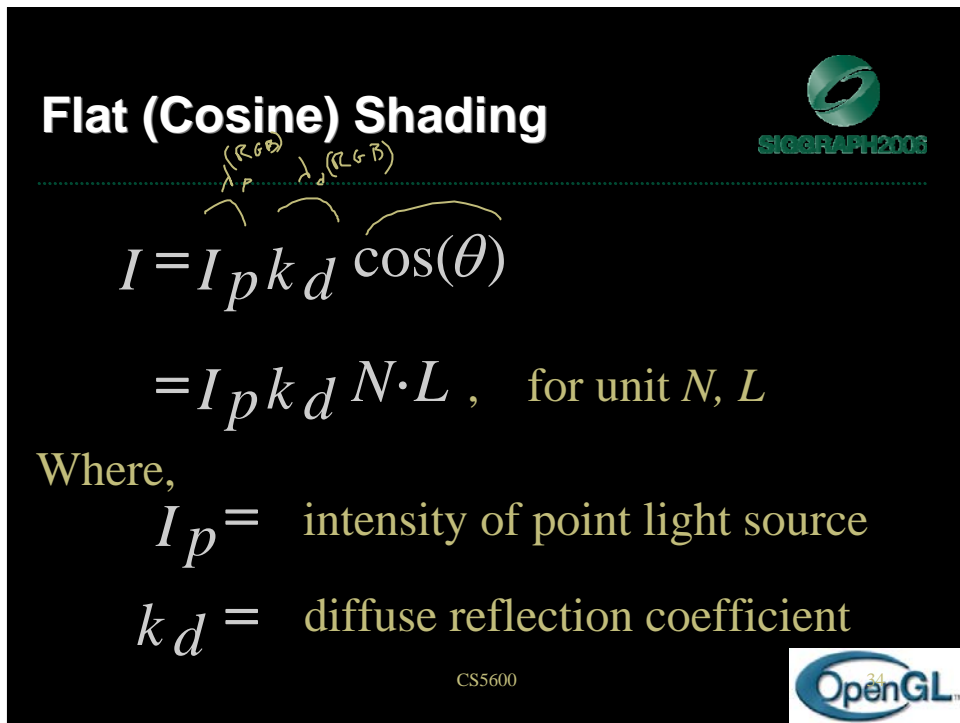
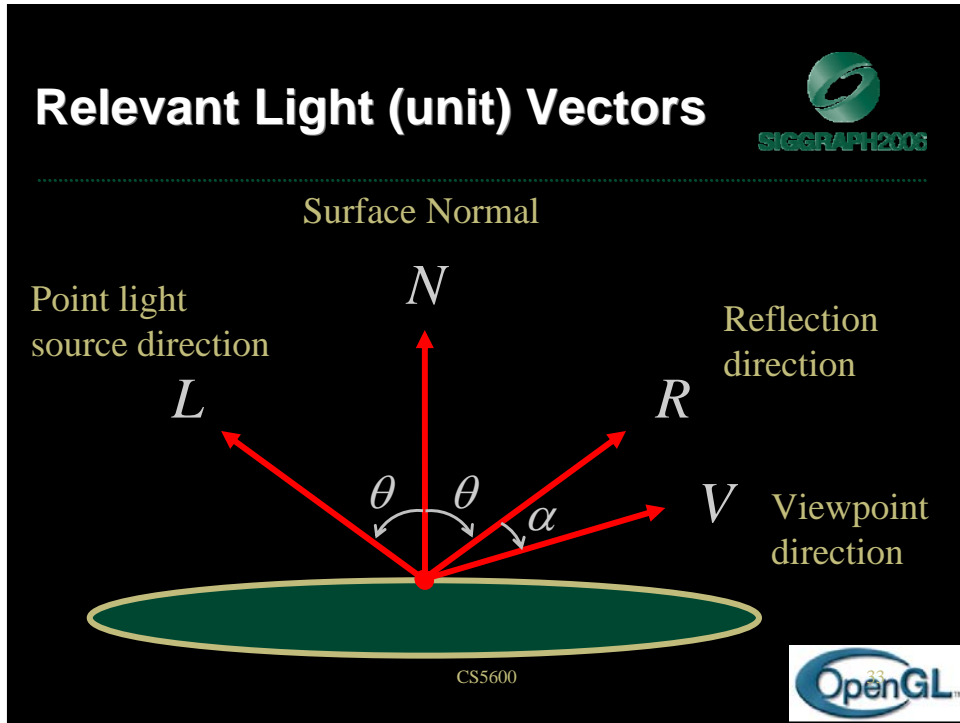
Flat (Cosine) Shading

- Compute constant shading function, over each polygon, based on simple cosine term
- Same normal and light vector across whole polygon
- Constant shading for polygon

$\sim N \cdot L$

The diagram shows a tetrahedron with four triangular faces. Each face has a red normal vector pointing outwards, labeled N_1 , N_2 , N_3 , and N_4 respectively.

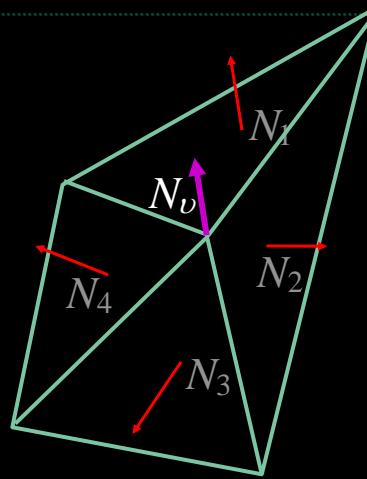
CS5600



Gouraud Shading



- Compute constant shading function, for each vertex, based on simple cosine term
- different normal and light vector for each vertex
- Interpolated shading for polygon



$$\sim N \cdot L$$

CS5600



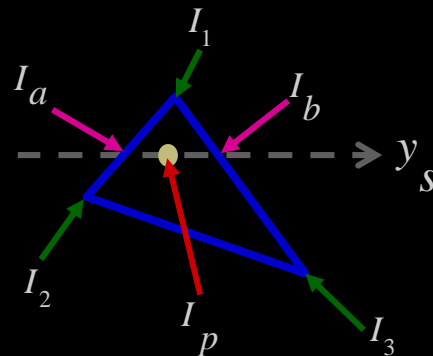
Intensity Interpolation (Gouraud)



$$I_a = I_1 \frac{y_s - y_2}{y_1 - y_2} + I_2 \frac{y_1 - y_s}{y_1 - y_2}$$

$$I_b = I_1 \frac{y_s - y_3}{y_1 - y_3} + I_3 \frac{y_1 - y_s}{y_1 - y_3}$$

$$I_p = I_a \frac{x_b - x_p}{x_b - x_a} + I_b \frac{x_p - x_a}{x_b - x_a}$$



CS5600

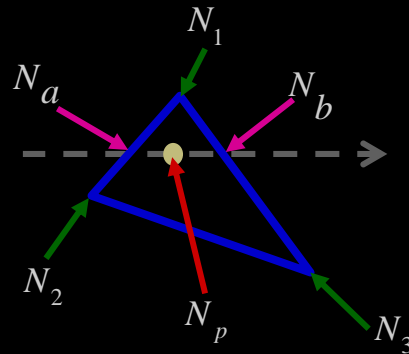


Normal Interpolation (Phong)



$$N_a = N_1 \frac{y_s - y_2}{y_1 - y_2} + N_2 \frac{y_1 - y_s}{y_1 - y_2}$$

$$N_b = N_1 \frac{y_s - y_3}{y_1 - y_3} + N_3 \frac{y_1 - y_s}{y_1 - y_3}$$



CS5600



Normal Interpolation (Phong)



$$\tilde{N}_p = \frac{N_a}{\|N_a\|} \begin{bmatrix} x_b - x_p \\ x_b - x_a \end{bmatrix} + \frac{N_b}{\|N_b\|} \begin{bmatrix} x_p - x_a \\ x_b - x_a \end{bmatrix}$$


$$N_p = \frac{\tilde{N}_p}{\|\tilde{N}_p\|}$$

Normalizing makes
this a unit vector

CS5600




Phong Illumination Formula (1/2)




$$I_{\lambda} = \underbrace{I_{a\lambda}}_{\text{ambient}} \underbrace{k_{a\lambda}}_{\text{coeff}} + f_{att} I_{l\lambda} \left[\underbrace{k_{d\lambda}}_{\text{diffuse}} (N \cdot L) + \underbrace{k_{s\lambda}}_{\text{specular}} (R \cdot V)^n \right]$$

OpenGL: $\cos^n(\alpha)$


CS5600 

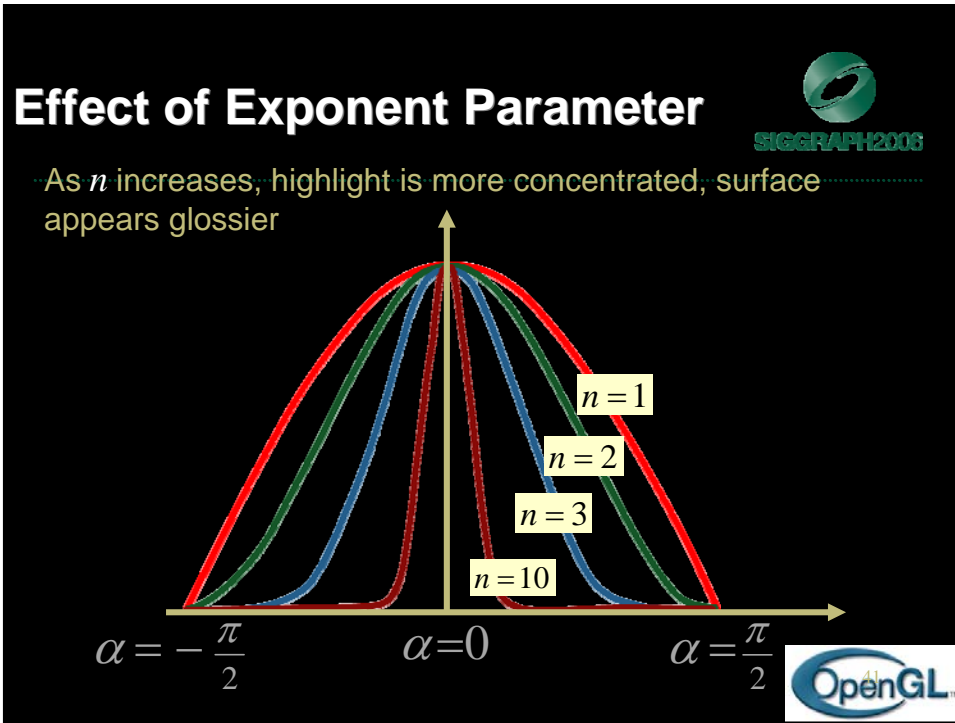
Illumination Formula (2/2)



Where,

- a denotes ambient term
- d denotes diffuse term
- s denotes specular term
- k denotes coefficient (RGB)
- I denotes intensity (light (RGB))

CS5600 



- ## How OpenGL Simulates Lights
- Phong lighting model
 - Computed at vertices
 - Lighting contributors
 - Surface material properties
 - Light properties
 - Lighting model properties
- OpenGL SIGGRAPH2006

Surface Normals



- Normals define how a surface reflects light

```
glNormal3f( x, y, z )
```

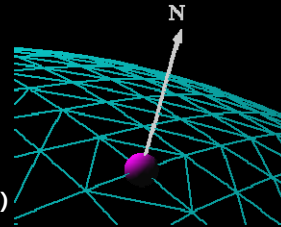
- Current normal is used to compute vertex's color
- Use *unit* normals for proper lighting

- scaling affects a normal's length

```
glEnable( GL_NORMALIZE )
```

or

```
glEnable( GL_RESCALE_NORMAL )
```



Material Properties



- Define the surface properties of a primitive

```
glMaterialfv( face, property, value );
```

GL_DIFFUSE	Base color
GL_SPECULAR	Highlight Color
GL_AMBIENT	Low-light Color
GL_EMISSION	Glow Color
GL_SHININESS	Surface Smoothness

- separate materials for front and back



Light Properties



```
glLightfv( light, property, value );
```

– *light* specifies which light

- multiple lights, starting with `GL_LIGHT0`

```
glGetIntegerv( GL_MAX_LIGHTS, &n );
```

– *properties*

- colors
- position and type
- attenuation



Light Sources (cont'd.)



- Light color properties

– `GL_AMBIENT`

– `GL_DIFFUSE`

– `GL_SPECULAR`



Types of Lights



- OpenGL supports two types of Lights
 - Local (Point) light sources
 - Infinite (Directional) light sources
- Type of light controlled by w coordinate

$w = 0$ *Infinite Light directed along* $(x \ y \ z)$

$w \neq 0$ *Local Light positioned at* $(x/w \ y/w \ z/w)$



Turning on the Lights



- Flip each light's switch
`glEnable(GL_LIGHTn);`
- Turn on the power
`glEnable(GL_LIGHTING);`

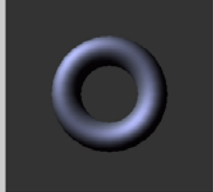


Light Material Tutorial

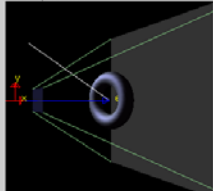


Light & Material

Screen-space view



World-space view



Command manipulation window

```
GLfloat light_pos[] = { -2.00, 2.00, 2.00, 1.00 };
GLfloat light_Ka[] = { 0.00, 0.00, 0.00, 1.00 };
GLfloat light_Kd[] = { 1.00, 1.00, 1.00, 1.00 };
GLfloat light_Ks[] = { 1.00, 1.00, 1.00, 1.00 };

glLightfv(GL_LIGHT0, GL_POSITION, light_pos);
glLightfv(GL_LIGHT0, GL_AMBIENT, light_Ka);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_Kd);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_Ks);

GLfloat material_Ka[] = { 0.11, 0.06, 0.11, 1.00 };
GLfloat material_Kd[] = { 0.43, 0.47, 0.54, 1.00 };
GLfloat material_Ks[] = { 0.33, 0.33, 0.52, 1.00 };
GLfloat material_Ke[] = { 0.00, 0.00, 0.00, 0.00 };
GLfloat material_Se = 10 ;

glMaterialfv(GL_FRONT, GL_AMBIENT, material_Ka);
glMaterialfv(GL_FRONT, GL_DIFFUSE, material_Kd);
glMaterialfv(GL_FRONT, GL_SPECULAR, material_Ks);
glMaterialfv(GL_FRONT, GL_EMISSION, material_Ke);
glMaterialfv(GL_FRONT, GL_SHININESS, material_Se);
```

Click on the arguments and move the mouse to modify values.



Controlling a Light's Position



- Modelview matrix affects a light's position
 - Different effects based on when position is specified
 - eye coordinates
 - world coordinates
 - model coordinates
 - Push and pop matrices to uniquely control a light's position



Light Position Tutorial



```
GLfloat pos[4] = { 1.50 , 1.00 , 1.00 , 0.00 };
gluLookAt( 0.00 , 0.00 , 2.00 , <- eye
          0.00 , 0.00 , 0.00 , <- center
          0.00 , 1.00 , 0.00 ); <- up
glLightfv(GL_LIGHT0, GL_POSITION, pos);
```

Click on the arguments and move the mouse to modify values.



Tips for Better Lighting



- Recall lighting computed only at vertices
 - model tessellation heavily affects lighting results
 - better results but more geometry to process
- Use a single infinite light for fastest lighting
 - minimal computation per vertex



Lighting Models



SIGGRAPH2006

	To	Somewhere	Everywhere
From			
Somewhere		<i>Specular</i>	<i>Lambertian</i>
Everywhere		<i>Caustics</i>	<i>Ambient</i>

CS5600

