Introduction to Computer Graphics

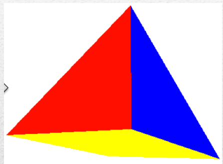
Lecture 10
Lighting and Shading

Outline

- Global and Local Illumination
  - Light Sources
  - Phong Illumination Model
  - Normal Vectors

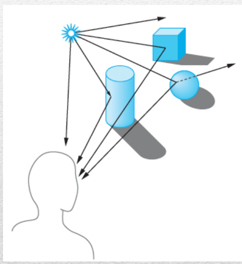
Introduction

- What gives 2D images the appearance of being 3D?
  - the gradations of color
  - shades of color
  - etc…



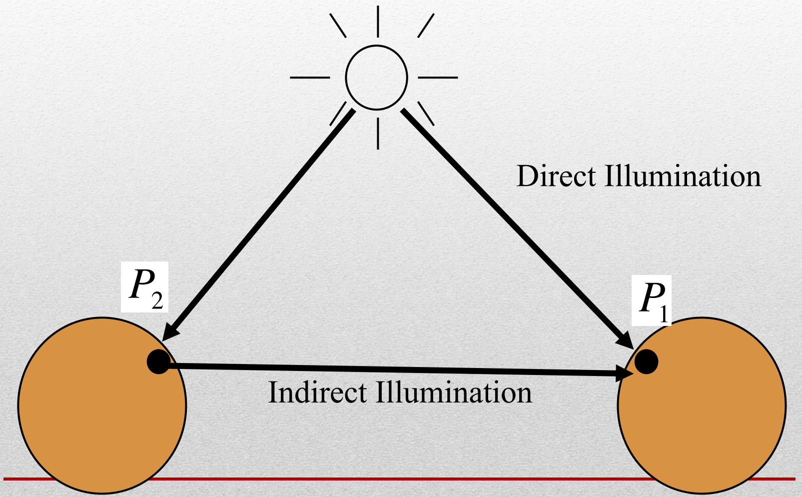
- Without lighting/shading….
  - a sphere is a uniformly colored circle, and a cube appears as a flat hexagon
- in 3D computer graphics, a group of algorithms are used to add more realistic lighting to 3D scenes

Global illumination



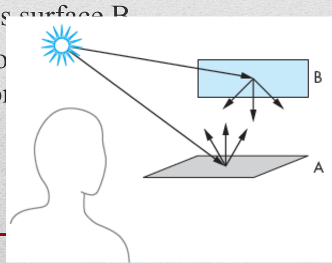
- a group of algorithms are used to add more realistic lighting to 3D scenes
  - Consider both
    - **direct illumination**
      - the light comes directly from a light source
    - **indirect illumination**
      - subsequent cases in which light rays from the same source are reflected by other surfaces in the scene, whether reflective or not

Global versus Local Illumination



Global illumination

- Recursive process:
  - Some light from the source that reaches surface A is scattered
  - Some of this reflected light reaches surface B
  - Some of it is then scattered back to again reflected back to B, and so on



Global Illumination

- Follow light rays through a scene
- Accurate, but expensive (off-line)
  - Numerical methods for computing a solution for the recursive process are not fast enough for real-time rendering.
- E.g.,
  - Ray tracing
  - Radiosity
  - Photon Mapping

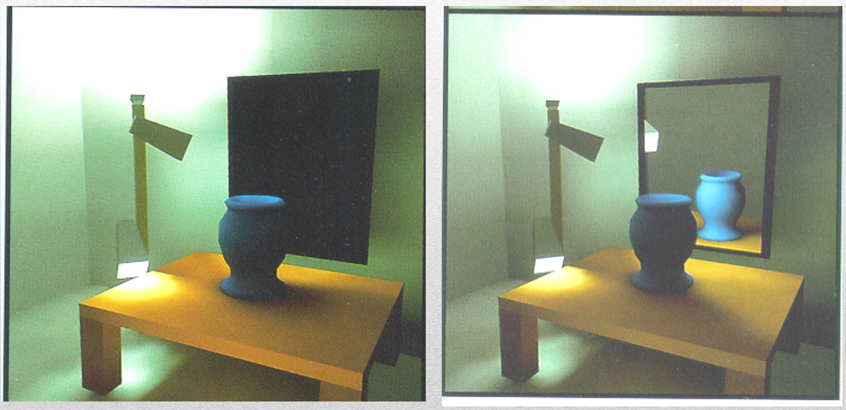
Ray tracing Example

Martin Moeck, Siemens Lighting

Global versus Local Illumination

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

Global versus Local Illumination

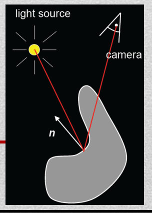


Images courtesy of Foley & van Dam, Computer Graphics

We start with simple local illumination and then extend those concepts/models to global illumination

Global versus Local Illumination

- **Local lighting** models
  - Approximate model
    - To add shading to a **fast** pipeline graphics architecture
      - E.g., OpenGL applications
  - Local interaction between light, surface, viewer
    - depend only on
      - the locations and properties of the light sources.
      - the material properties & the local geometry of the surface
      - Viewer positions
    - independent of any other surfaces in the scene
      - as opposed to global lighting models



Light Sources and Material Properties

- Appearance depends on
  - Light sources, their locations and properties
  - Material (surface) properties:
  - Viewer positions

Local Illumination

we consider only single interactions between light sources and surfaces

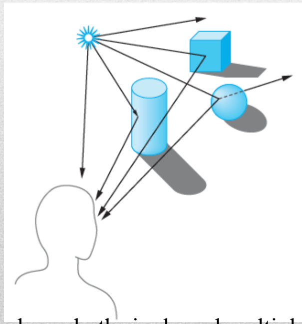
- Two independent parts of the problem：
  - model the **light sources** in the scene
    - follow rays of light from light-emitting surfaces
  - build a **reflection model** that deals with the interactions between materials and light.
    - model what happens to these rays as they interact with reflecting surfaces in the scene

Light Sources & Reflection Model

- What color we see?

If a ray of light enters her eye directly from the source:
→ she sees the color of the source

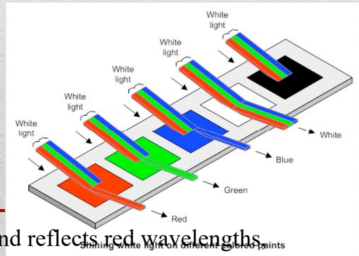
If the ray of light hits a surface visible to our viewer:
→ she sees the color that based on the light reflected from the surface toward her eyes



shows both single and multiple interactions between rays and objec

Light Sources & Reflection Model

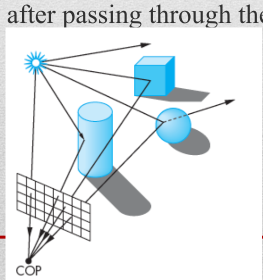
- What color?
  - the color that we see is determined by multiple interactions among light sources and reflective surfaces
  - Emission is what light sources do
    - Emission produces light
  - Adsorption is what paints, inks, dyes etc. do
    - adsorption removes light



E.g., Red paint absorbs green and blue wavelengths, and reflects red wavelengths, resulting in you seeing a red appearance

Light Sources & Reflection Model

- Which rays we should trace?
  - Only need to consider those rays that leave the source and reach **the viewer's eye**
  - replace the viewer by the projection plane
    - → Only consider the rays that reach **the center of projection** (COP) after passing through the clipping rectangle



most rays leaving a source do not contribute to the image and are thus of no interest to us

Local Illumination

- In OpenGL
  - Phong model
    - fast
    - provides a compromise between physical correctness and efficient calculation
    - consider only single interactions between light sources and surfaces
  - Two independent parts of the problem
    - model the **light sources** in the scene
    - build a **reflection model**
      - that deals with the interactions between materials and light.
      - model what happens to these rays as they interact with reflecting surfaces in the scene

Defining a Light Source

- Use vectors {r, g, b, a} for light properties
- Beware: light positions will be transformed by the modelview matrix

```
GLfloat light_ambient[] = {0.2, 0.2, 0.2, 1.0};
GLfloat light_diffuse[] = {1.0, 1.0, 1.0, 1.0};
GLfloat light_specular[] = {1.0, 1.0, 1.0, 1.0};
GLfloat light_position[] = {-1.0, 1.0, -1.0, 0.0};

glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

Defining Material Properties

- OpenGL is a state machine:
- material properties stay in effect until changed

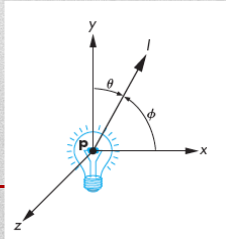
```
GLfloat mat_a[] = {0.1, 0.5, 0.8, 1.0};
GLfloat mat_d[] = {0.1, 0.5, 0.8, 1.0};
GLfloat mat_s[] = {1.0, 1.0, 1.0, 1.0};
GLfloat low_sh[] = {5.0};

glMaterialfv(GL_FRONT, GL_AMBIENT, mat_a);
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_d);
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_s);
glMaterialfv(GL_FRONT, GL_SHININESS, low_sh);
```

Outline

- Global and Local Illumination
- **Light Sources**
- Phong Illumination Model
- Normal Vectors

Light Sources

- Light can leave a surface through two fundamental processes:
  - **self-emission** and **reflection**
  - e.g., a light bulb can also reflect light that is incident on it from the surrounding environment.
  - the emissive term in our simple models is usually omitted

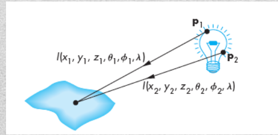


Light Sources

- Problem: How much is a surface illuminated by this source?
  - Obtain the total contribution of the source by integrating over its surface
    - accounts for the emission angles that reach this surface and the distance between the source and the surface

→Too difficult!!

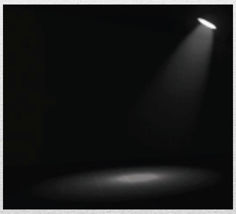
→it is easier to model the distributed source with
- Polygons (each of which is a simple source)
- Or an approximating set of point sources



Types of Light Sources

These lighting types are sufficient for rendering most simple scenes:

- Ambient light
  - no identifiable source or direction
- Point source
  - given only by point
- Distant light
  - given only by direction
- Spotlight
  - from source in direction
  - Cut-off angle defines a cone of light
  - Attenuation function (brighter in center) (衰減)



Ambient Light

- Uniform lighting
- Lights entire scene
- Computationally inexpensive
- Simply add [I_{ar} I_{ag} I_{ab}] to every pixel on every object
- A cheap hack to make the scene brighter.

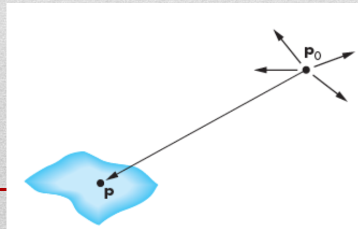
$$\mathbf{I}_a = \begin{bmatrix} I_{ar} \\ I_{ag} \\ I_{ab} \end{bmatrix}$$

Point Source

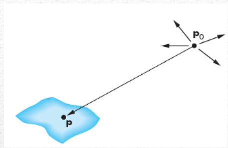
$$\mathbf{I}(\mathbf{p}_0) = \begin{bmatrix} I_r(\mathbf{p}_0) \\ I_g(\mathbf{p}_0) \\ I_b(\mathbf{p}_0) \end{bmatrix}$$

- Given by a point p_0
- Light emitted equally in all directions
- Intensity decreases with square of distance
  - At a point p, the intensity of light received from the point source p_0:

$$\mathbf{i}(\mathbf{p}, \mathbf{p}_0) = \frac{1}{|\mathbf{p} - \mathbf{p}_0|^2}\mathbf{I}(\mathbf{p}_0)$$



Point Sources



Limitations

- Shading and shadows inaccurate
  - objects appear either bright or dark

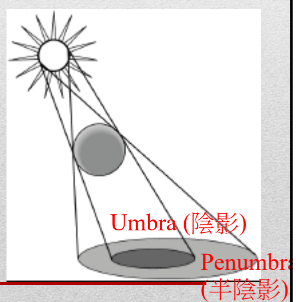
Example: penumbra (partial "soft" shadow)

- Compensate with attenuation

$$\frac{1}{a + bq + cq^2}$$

q: distance $|p - p_0|$
a, b, c: constants

- Softens lighting

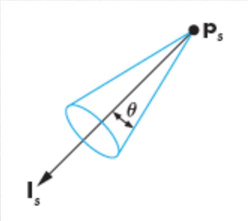


Shadows created by finite-size light source

Spotlight

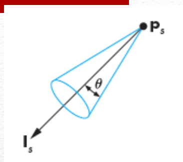
- Light still emits from point
  - Spotlight is constructed from a point source by limiting the angles at which light can be seen
- Cut-off by a cone
  - apex is at P_s
  - width is determined by an angle θ
  - points in the direction l_s

a narrow range of angles through which light is emitted



If θ = 180, the spotlight becomes a point source

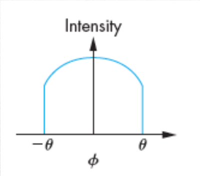
Spotlight



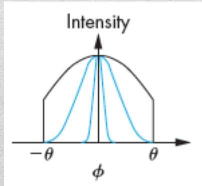
- Attenuation of a spotlight (More realistic)
  - most of the light concentrated in the center of the cone
    - Use the intensity function

$$\cos^e \varphi$$

    - Φ: the angle between the direction of the source l_s and a vector s to a point on the surface
    - $\Phi < \theta$
    - exponent e: determines how rapidly the light intensity drops off
    - Easy to compute: $\cos \Phi = u \bullet v$, if u, v are unit vector



$\cos \varphi$



$\cos^e \varphi$

Distant Light Source

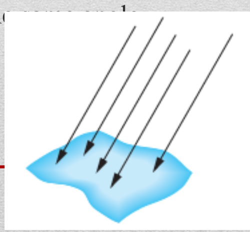
- Given by a direction vector
- Simplifies some calculations
  - most shading calculations require the direction
    - from the point on the surface to the light source position
  - if the light source is far from the surface, the vector does not change much as we move from point to point
    - E.g., the sun strikes all objects at proximity th
- In OpenGL:

$$\mathbf{p}_0 = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Point source

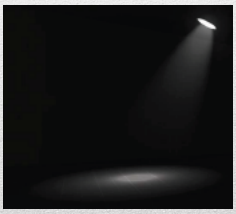
$$\mathbf{p}_0 = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$$

Distant source



Light Sources Summary

- Ambient light
  - no identifiable source or direction
- Point source
  - given only by point
- Distant light
  - given only by direction
- Spotlight
  - from source in direction
  - Cut-off angle defines a cone of light
  - Attenuation function (brighter in center) (衰減)



Outline

- Global and Local Illumination
- Light Sources
- **Phong Reflection Model**
- Normal Vectors

Phong Reflection Model

- Introduced by Phong
  - later modified by Blinn.
- An efficient approach compromises between realism and efficiency
  - can be a close-enough approximation to physical reality to produce good renderings

Phong Reflection Model

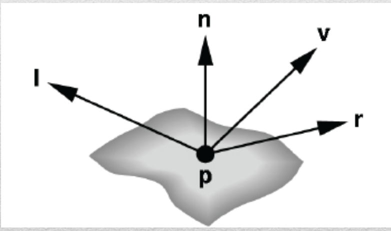
$$I = \frac{1}{a+bq+cq^2}(k_d L_d (l \cdot n) + k_s L_s (r \cdot v)^{\alpha}) + k_a L_a$$

- Light components for each color:
  - Ambient (L_a), diffuse (L_d), specular (L_s)
- Material coefficients for each color:
  - Ambient (k_a), diffuse (k_d), specular (k_s)
- Distance q for surface point from light source

l: unit vector to light
r: l reflected about n
n: surface normal
v: vector to viewer

Phong Reflection Model

- The Phong model uses the four vectors, (l,v,n,r)
  - to calculate a color for an arbitrary point p on a surface
- Basic inputs are material properties and l, n, v



l = unit vector to light source
v = unit vector to viewer
n = surface normal
r = reflection of l at p
(determined by l and n)

Phong Reflection Model Overview

The Phong model supports the three types of material–light interactions:

- Light source contributions decomposed into
  - Ambient reflection
  - Diffuse reflection
  - Specular reflection
- Calculate each color channel (R,G,B) separately

Light – Material Interactions

Diffuse surfaces:

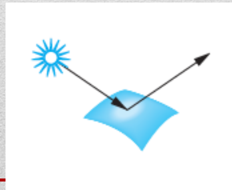
- reflected light being scattered in all directions.
- Perfectly diffuse surfaces scatter light equally in all directions
  - appears the same to all viewers.
- E.g., many natural materials
  - such as terrain viewed from an airplane
  - Walls painted with matte/flat paint

Diffuse surface

Light – Material Interactions

Specular surfaces (appear shiny)

- appear shiny
- Light is reflected in a narrow range of angles
  - close to the angle of reflection
  - the angle of incidence = the angle of reflection
  - may be partially absorbed, but all reflected light emerges at a single angle
- E.g., Mirrors



Specular surface

Phong Reflection Model Overview

1. Start with global ambient light [I_{ar} I_{ag} I_{ab}]
2. Add contributions from each light source
   - Light source contributions decomposed into
     - Ambient reflection
     - Diffuse reflection
     - Specular reflection
   - Calculate each color channel (R,G,B) separately
3. Clamp the final result to [0, 1]

```
GLfloat  ambientLight[] = { 0.5f, 0.5f, 0.5f, 1.0f};
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLight);
glLightfv(GL_LIGHT0,GL_DIFFUSE,ambientLight);
glLightfv(GL_LIGHT0,GL_SPECULAR,specular);
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
```

Phong Reflection Model Overview

1. Start with global ambient light [I_{ar} I_{ag} I_{ab}]
2. Add contributions from each light source
3. Clamp the final result to [0, 1]

$$I = \sum_i (I_{ia} + I_{id} + I_{is}) + I_a$$

Example

```
//Example Lighting Properties
 GLfloat lmodel_ambient[] = { 0.2, 0.2, 0.2, 1.0 };
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient);

GLfloat light_ambient[]={0.2, 0.2, 0.2, 1.0};
GLfloat light_diffuse[]={1.0, 1.0, 1.0, 1.0};
GLfloat light_specular[]={1.0, 1.0, 0.0, 1.0};
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);

//Example Material Properties
GLfloat mat_ambient[]={0.8, 0.6, 0.4, 1.0};
GLfloat mat_specular[]={0.0, 0.0, 0.0, 1.0};
GLfloat mat_diffuse[]={0.8, 0.6, 0.4, 1.0};
GLfloat mat_shininess={20.0};

glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);
```

Phong Reflection Model

$$I = \frac{1}{a + bq + cq^2}\left(k_d L_d (l \cdot n) + k_s L_s (r \cdot v)^{\alpha}\right) + k_a L_a$$

- Light components for each color:
  - Ambient (L_a), diffuse (L_d), specular (L_s)
- Material coefficients for each color:
  - Ambient (k_a), diffuse (k_d), specular (k_s)
- Distance q for surface point from light source

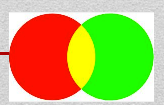
l: unit vector to light
n: surface normal
r: l reflected about n
v: vector to viewer

Ambient Reflection

$$I_a = k_a L_a$$

- L_a : ambient component of light source
  - L_a can be any of the individual light sources, or a global ambient term
- K_a :Ambient reflection coefficient $k_a = R_a$, $0 \le k_a \le 1$
  - A surface has three k_a components —k_{ar}, k_{ag}, k_{ab}

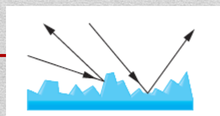
    E.g. a sphere appears yellow under white ambient light if its k_{ab} is small and its k_{ar}, k_{ag} are large
  - k_a may be different for every surface
- Intensity of ambient light is uniform at every point



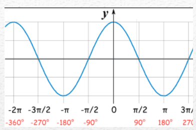
Diffuse Reflection

$$I_d = k_d L_d (l \bullet n)$$

- K_d: Diffuse reflection coefficient , $0 \leq k_d \leq 1$
- the amount of light reflected depends both on
  - the material
  - the position of the light source relative to the surface
    - Angle of incoming light is important
    - can be modeled mathematically with Lambert's law

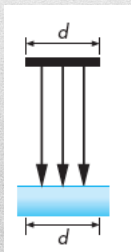


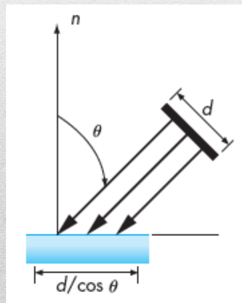
Diffuse Reflection



$R_d \propto \cos \theta$

- Lambert's Law:
  - Intensity depends on angle of incoming light.





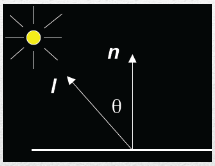
the same amount of light is spread over a larger area, and the surface appears dimmer.

Diffuse Reflection

Diffuse Light Intensity Depends On Angle Of Incoming Light

$\mathbf{I_d = k_d\ L_d(l \bullet n)}$

- $\mathbf{I_d = k_d\ L_d}\cos\theta$
  - $\cos\theta = l \bullet n$



l : unit vector to light
n : unit surface normal
Θ : angle to normal

k_d: the fraction of incoming diffuse light that is reflected
L_d: diffuse component of light

$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\|\,\|\mathbf{B}\| \cos\theta$

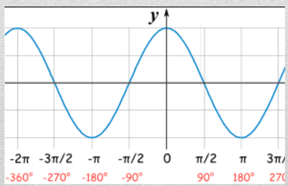
- Attenuation with distance:

$$I_d = \frac{k_d L_d}{a + bq + cq^2}(l \cdot n)$$

q: distance to light source,

Diffuse Reflection

- Potential Problem: $\mathbf{I_d = k_d\ L_d(l \bullet n)}$
  - $\mathbf{(l \bullet n)}$ will be negative if the light source is below the horizon
  - use zero rather than a negative value
  - use max($\mathbf{l \bullet n}$, 0)
  - $\mathbf{I_d = k_d\ L_d\ \ max((l \bullet n), 0)}$

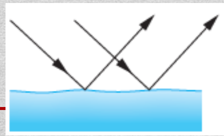


Specular Reflection

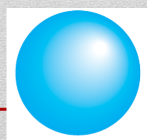
$$I_s = k_s \, L_s \, (\cos \varphi)^{\alpha}$$

K_s: Specular reflection coefficient , $0 \le k_s \le 1$

- Used to model highlights
  - a diffuse surface is rough, a specular surface is smooth
  - Shiny surfaces have high specular coefficient
  - Does not give mirror effect (need other techniques)

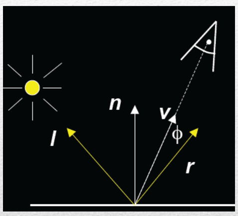


specular reflection

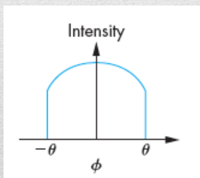


specular highlights

Specular Reflection



v = unit vector to camera
r = unit reflected vector
φ = angle between v and r



$$\boxed{I_s = k_s \, L_s \, (\cos \varphi)^{\alpha}}$$

- L_s is specular component of light
- α is shininess coefficient
- Can add distance term as well

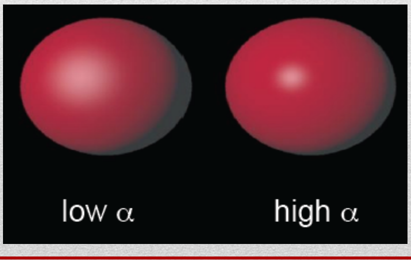
$$I_s = \frac{k_s \, L_s \, (\cos \varphi)^{\alpha}}{a + bq + cq^2}$$

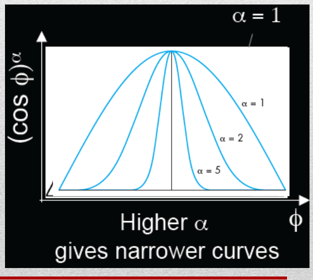
```
//Example Material Properties
    GLfloat mat_ambient[]={0.8, 0.6, 0.4, 1.0};
    GLfloat mat_diffuse[]={0.8, 0.6, 0.4, 1.0};
    GLfloat mat_specular[]={0.0, 0.0, 0.0, 1.0};
    GLfloat mat_shininess={50.0};

    glMaterialfv(GL_FRONT, GL_SPECULAR,
mat_specular);
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);
```

Shininess Coefficient

$$I_s = k_s L_s (\cos \varphi)^\alpha$$

- α is the shininess coefficient



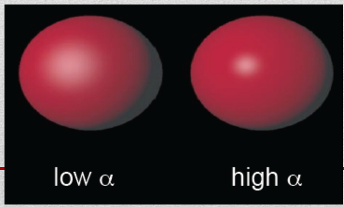


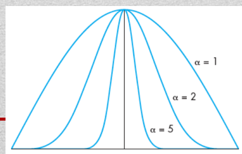
Specular Reflection

$$I_s \quad = \quad k_s\, L_s\, (\cos \varphi)^\alpha \quad = \quad k_s L_s \max((r \cdot v)^\alpha, 0)$$

$\cos \varphi = v \bullet r$

- α:
  - Larger value: most metallic surfaces
  - smaller values : materials that show broad highlights





Summary of Phong Model

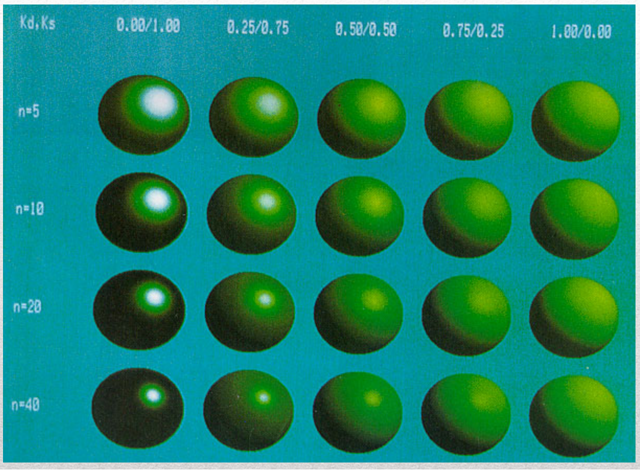
- Light components for each color:
  - Ambient (L_a), diffuse (L_d), specular (L_s)
- Material coefficients for each color:
  - Ambient (k_a), diffuse (k_d), specular (k_s)
- Distance q for surface point from light source

$$I = \frac{1}{a + bq + cq^2}(k_d L_d (l \cdot n) + k_s L_s (r \cdot v)^\alpha) + k_a L_a$$

l: unit vector to light n: surface normal
r: l reflected about n v: vector to viewer

Phong Example

$\alpha = n$



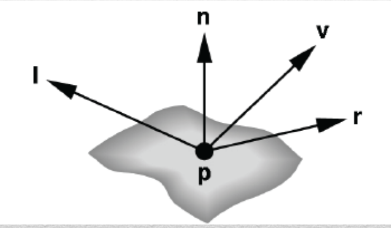
$$I = \frac{1}{a + bq + cq^2}(k_d L_d (l \cdot n) + k_s L_s (r \cdot v)^{\alpha}) + k_a L_a$$

Outline

- Global and Local Illumination
- Light Sources
- Phong Illumination Model
- Normal Vectors

Phong Reflection Model

- The Phong model uses the four vectors, (l,v,n,r)
  - to calculate a color for an arbitrary point p on a surface
- Basic inputs are material properties and l, n, v



l = unit vector to **l**ight source
v = unit vector to **v**iewer
n = surface normal
r = **r**eflection of l at p
(determined by l and n)

Normal Vectors

- The shading of objects also depends on the orientation of their surfaces
  - a factor that we shall see is characterized by the **normal vector** at each point
- Must calculate and specify the normal vector
  - Even in OpenGL!

Normals of Plane

Method I:

a plane is given by the equation: **$ax + by + cz + d = 0$**

- Let p_0 be a known point on the plane
- Let p be an arbitrary point on the plane
- Recall: u • v = 0 if and only if u orthogonal to v
- $n \bullet (p - p_0) = n \bullet p - n \bullet p_0 = 0$
- Consequently $n_0 = [a\ b\ c\ ]^T$
- Normalize to $n = n_0/|n_0|$

Normals of Plane

Method II:

plane given by p_0, p_1, p_2

- Points must not be collinear
- Recall: u x v orthogonal to u and v
- $n_0 = (p_1 - p_0) \times (p_2 - p_0)$
- Order of cross product determines orientation
- Normalize to $n = n_0/|n_0|$

Normals of Sphere

- Implicit Equation： $f(x, y, z) = x^2 + y^2 + z^2 - 1 = 0$
- Vector form: $f(p) = p \bullet p - 1 = 0$
- Normal given by gradient vector

$$n_0 = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix} = \begin{bmatrix} 2x \\ 2y \\ 2z \end{bmatrix} = 2p$$

- Normalize $n_0/|n_0| = 2p/2 = p$

Angle of Reflection

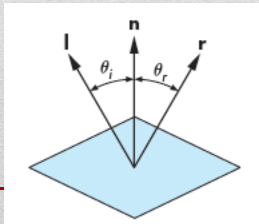
Perfect reflection (an ideal mirror):

angle of incident θi (入射角) ＝ angle of reflection: θr (反射角)

- Note: l, n, and r must lie in the same plane
  - In 3D, these two conditions are sufficient to determine r from n and l
- Normalize l, n
  - Such that $|l| = |n| = 1$, also we want $|r| = 1$

$l \bullet n = \cos(\theta i) = \cos(\theta r) = n \bullet r$
The coplanar condition implies that:
$r = \alpha\, l + \beta\, n$

→ $n \bullet r = \alpha(l \bullet n) + \beta = l \bullet n$ ---(1)
→ $r \bullet r = 1 = (\alpha\, l + \beta\, n)^2 = \alpha^2 + 2\alpha\beta(l \bullet n) + \beta^2$ ----(2)
Solution: $\alpha = -1$ and $\beta = 2\,(l \bullet n)$
→ **$r = 2\,(l \bullet n)\,n - l$**



Summary

- Global and Local Illumination
- Light Sources
- Phong Illumination Model
- Normal Vectors

Lighting Guide:
http://www.glprogramming.com/red/chapter05.html