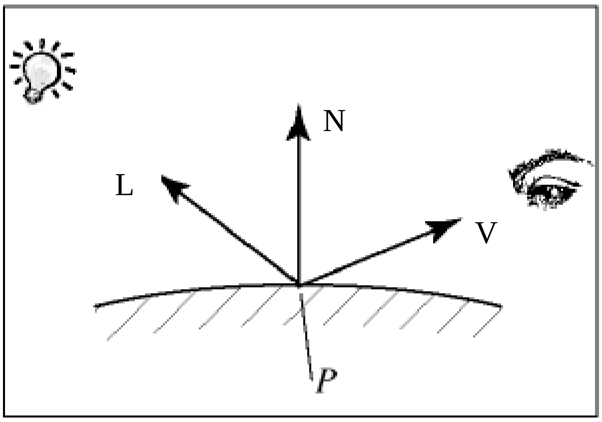
Illumination in Computer Graphics

Ann McNamara

Illumination in Computer Graphics

- Definition of light sources.
- Analysis of interaction between light and objects in a scene.
- Rendering images that are faithful to the physics of light.

Directions used in computing reflected light



Emissive Illumination Model

- Objects are emitting light.
- Each point on an object may emit a different color and/or amount of light.
- We define a vector of intensity levels for each point on the surface of an object:
  $I = (k_{er}, k_{eg}, k_{eb})$.
- Typically I is the same for all points on the surface of a single object.

Ambient Illumination Model

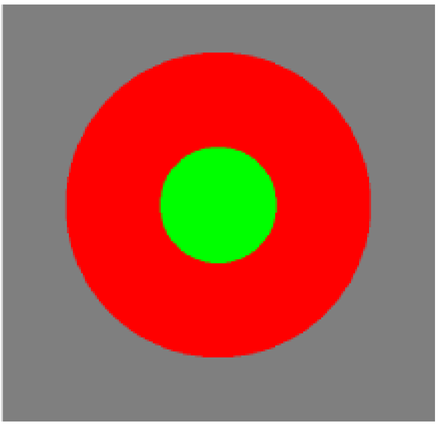
- Imagine a scene with many light sources and many reflecting surfaces.
- Equal amounts of light travel in all directions.
- The illumination of an object is independent of the position and orientation of the object.

Ambient Illumination Model

- Define a vector (I_{ar}, I_{ag}, I_{ab}) representing the ambient amounts of red, green and blue light.
- Determine the coefficients of ambient reflection (k_{ar}, k_{ag}, k_{ab}) for each point on each object.
- The color of a point is given by:

$$I = (I_{ar} k_{ar}, I_{ag} k_{ag}, I_{ab} k_{ab})$$

Green Sphere in front of Red Sphere – Ambient Light Only



Ambient Light

- Objects lit by ambient light are lit evenly on all surfaces in all directions
- Certain lights e.g. tube lights in class-rooms or kitchens try to achieve this by using large diffusers

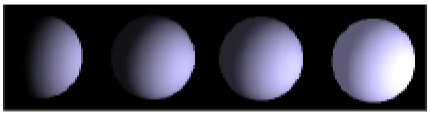
ambient illumination is characterised by an intensity L_a identical at every point in the scene

Different levels of ambient light:

$$I_d = L_a k_a$$

where k_a is proportion of ambient light reflected

How ambient light affects overall lighting:



$$I = L_a k_a + I_{rest}$$

Ambient Illumination in Phong

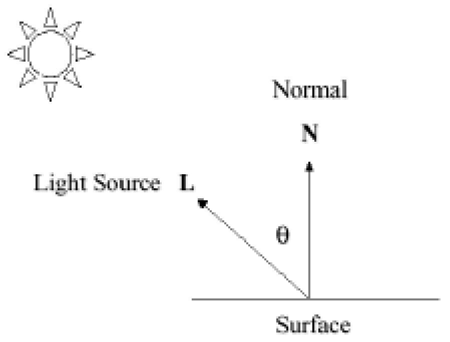
- Local illumination models account for light scattered from the light sources only.
- Light may be scattered from all surfaces in the scene
  - we are missing some light; in fact we are missing a lot of light, typically over 50%.
- *Ambient term* = a coarse approximation to this missing flux
- The ambient term is a constant everywhere in the scene but is sometimes estimated from the total powers and geometries of the light sources.

$$I_a = k_a L_a$$

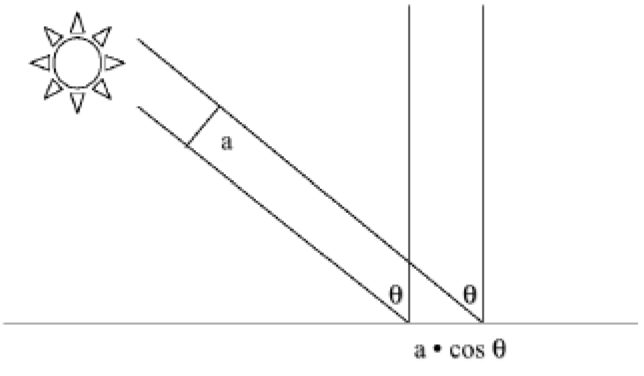
Diffuse Reflection Model

- Interaction of light with surfaces that scatter light in all directions.
- Color and intensity of light reflected from a point on an object depend on the following:
  – Color and intensity of the light source.
  – Distance and direction to the light source.
  – Orientation of the surface on which the point lies.
  – Reflecting properties of the object's material.

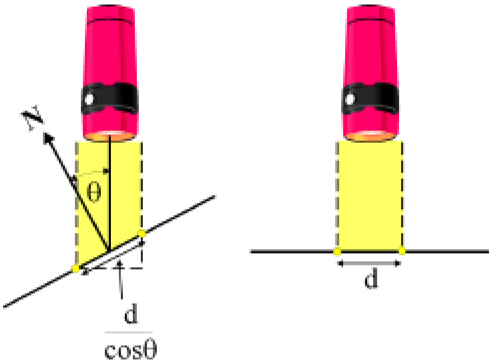
Diffuse Reflection Model



Diffuse Reflection Model

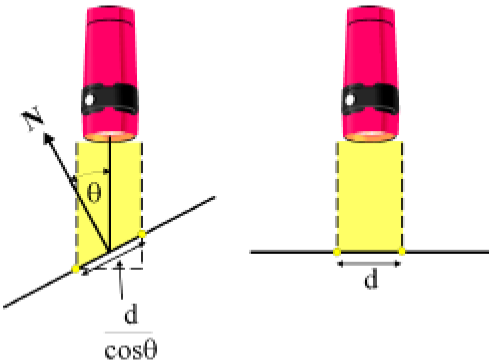


Diffuse Reflection Model



A surface which is oriented perpendicular to a light source will receive more energy (and thus appear brighter) than a surface oriented at an angle to the light source

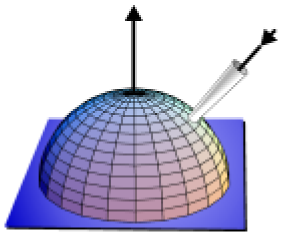
Diffuse Reflection Model



As θ increases, the brightness of a surface decreases by cos θ

Diffuse Reflection Model

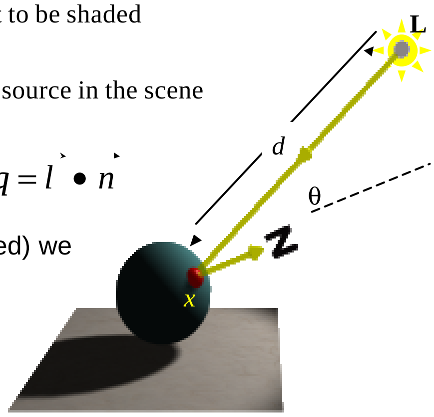
- We can use the cosine rule to implement shading of *Lambertian* or *diffuse* surfaces.



Lambertian Illumination Model

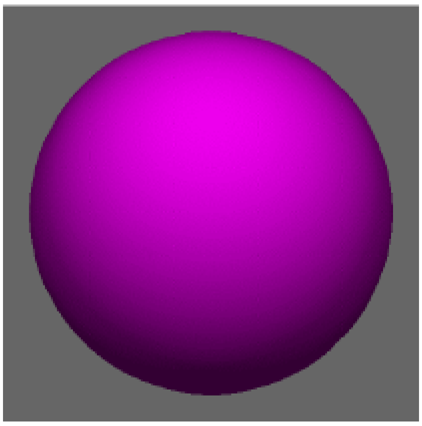
- To shade a diffuse surface we need to know:
  - *normal* to the surface at the point to be shaded
  - diffuse *reflectance* of the surface
  - *positions* and *powers* of the light source in the scene
- Lambert's Law: $R_d \propto \cos q$
- If ***L*** and ***N*** are unit vectors… $\cos q = \vec{l} \bullet \vec{n}$
- adding the reflection coefficient (proportion of incoming light reflected) we have the **diffuse reflection term**

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



Diffuse Reflection Model

- Define a vector (I_{dr} ,I_{dg} ,I_{db}) representing the amounts of red, green and blue light in the light source.
- Determine the coefficients of diffuse reflection (k_{dr} ,k_{dg} ,k_{db}) for each point on each object.
- Determine the unit vectors **L** and **N**.
- The color of a point is given by:
  - I= (I_{dr} k_{dr} , I_{dg} k_{dg} , $I_{db}k_{db}$) Max(L•N,0)



Magenta Sphere

Diffuse and Ambient Lighting

Light Source Attenuation

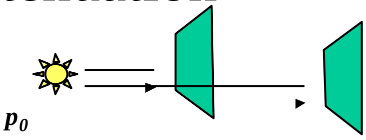
- Our model ignores the distance from the object to the light source.
- We can account for distance by including an additional factor f att in the formula:

$$I= f_{att} (I_{dr} k_{dr} , I_{dg} k_{dg} , I_{db} k_{db}) \text{Max}(L \bullet N,0)$$

  - $f_{att} = 1/(a + bd + cd^2)$

- Where d is the distance from the object point to the light source.

Light Source Attenuation



- Using the model so far, two parallel planes at different distances from the light source would be rendered exactly the same: distance from source seems to have no effect
- We need to account for energy transport falling off with distance from source:

$$I = L_a k_a + f_{att} I_d k_d (\overline{N} \bullet \overline{L})$$

- For a point light source the inverse square law (intensity falls off in proportion to the square of distance from source) is a correct model:

$$f_{att} = \frac{1}{d_L^{\ 2}} = \frac{1}{|p - p_0|^2}$$

Light Source Attenuation (2)

- However this is not a good model in practice (largely because most objects in the real world are not lit by point sources).
- A better approximation which allows for a richer range of effects is:

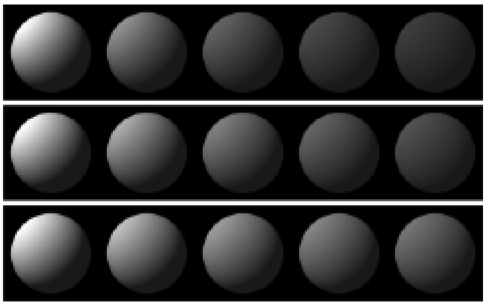
$$f_{att} = \frac{1}{a + bd_L + cd_L^2}$$

Spheres at increasing distances from light source

a=b=0; c=1

a=b=0.25; c=0.5

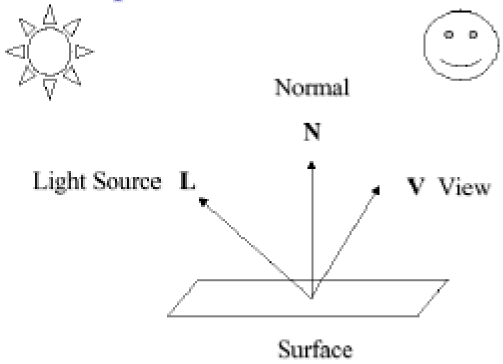
a=0; b=1; c=0



Specular Reflection Model

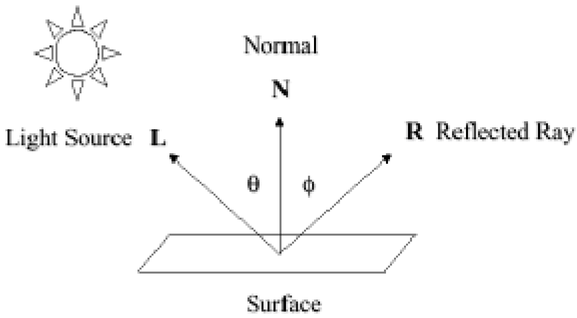
- Mirrors and shiny surfaces are not properly modeled by the diffuse reflection model.
- They reflect light more strongly in some directions than in others.
- To model these types of objects, we must consider the direction from which they are viewed.

Specular Reflection



Appearance of a surface depends on the direction L of the light source, direction of the surface normal N, and direction V of viewing

Reflection by a perfect mirror

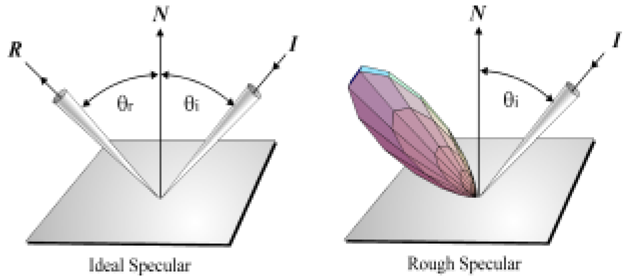


The angle ? of incidence equals the angle ? of reflection.

The vectors L, N and R all lie in one plane

Phong Illumination Model

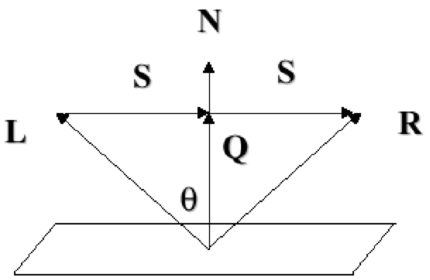
- *Specular* surfaces exhibit a high degree of *coherence* in their reflectance, i.e. the reflected radiance depends very heavily on the outgoing direction.
  - An *ideal specular* surface is *optically smooth* (smooth even at resolutions comparable to the wavelength of light).
  - Most specular surfaces (rough specular) reflect energy in a tight distribution (or *lobe*) centered on the *optical reflection direction*:



Phong's Model of Specular Reflection

- Determine the angle α between the direction **V** of viewing and the direction **R** of reflection by an ideal mirror.
- Assume the intensity of reflected light is proportional to (cos(α))s .
- The exponent s ("shine") is determined empirically.
- Large values of s make the surface behave more like an ideal mirror

Calculating the Reflection Vector



$$Q = N\,(N \cdot L)$$

$$S = Q - L$$

$$R = L + 2S$$

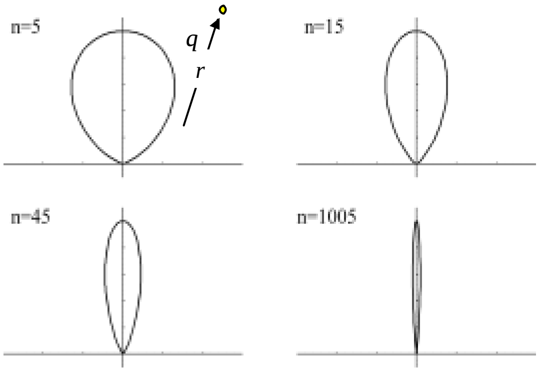
Specular Reflection Model

- Define a vector (I_{sr}, I_{sg}, I_{sb}) representing the amounts of red, green and blue light in the light source.
- Determine the coefficients of specular reflection (k_{sr}, k_{sg}, k_{sb}) for each point on each object.
- Determine the unit vectors **L, N** and **V**.
- Compute **R** from **L** and **N.**
- The color of a point is given by:
- $I = f_{att}\,(I_{sr}\, k_{sr},\ I_{sg}\, k_{sg},\ I_{sb}\, k_{sb})\ [\mathrm{Max}(\mathbf{R}\cdot\mathbf{V},0)]^s$

The Cos^s Function

The cosine function (defined on the sphere) gives us a lobe shape which approximates the distribution of energy about a reflected direction controlled by the shinyness parameter *a* known as the *Phong exponent*.

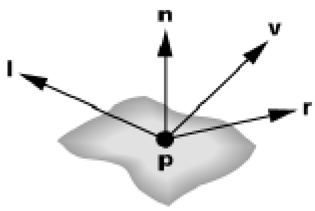
$r = \cos^s q$



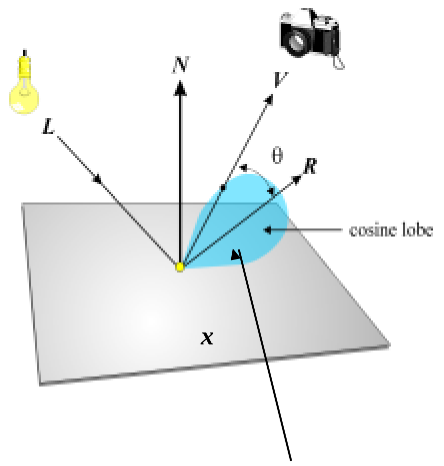
In the limit (n $\rightarrow \infty$) the function becomes a single spike (i.e. ideal specular).

Phong Illumination

- Supports:
  - Lambertian model for diffuse reflection
  - Cosine lobe for specular reflection
  - Ambient term to approximate all other light
- Based on 4 important Vectors:
  - ***p*** *is a surface point*
  - ***l*** *is direction to light source*
  - ***n*** *is surface normal*
  - ***v*** *is direction to COP*
  - ***r*** *(depends on* ***l*** *and* ***n****) is direction of perfectly reflected ray*



The Phong Model



$$I_s = k_s L_s \cos^s f$$

a : shininess (phong exponent)

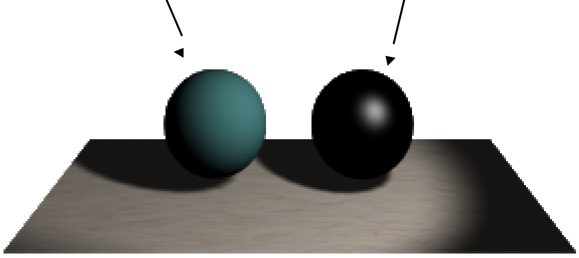
k_s: specular reflectivity coefficient

Radiance of reflected light given by cosine function

Pure Lambertian vs. Phong

Lambertian Surface

Phong Illuminated Specular Surface

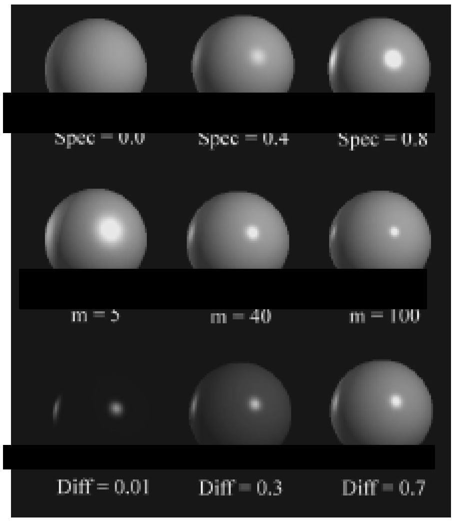


Phong Illumination Examples

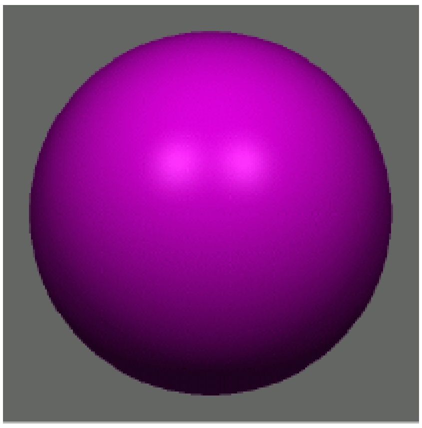
Increasing specular reflectance

Increasing shininess coefficient

Increasing diffuse reflectance



Diffuse, Ambient and Specular Lighting



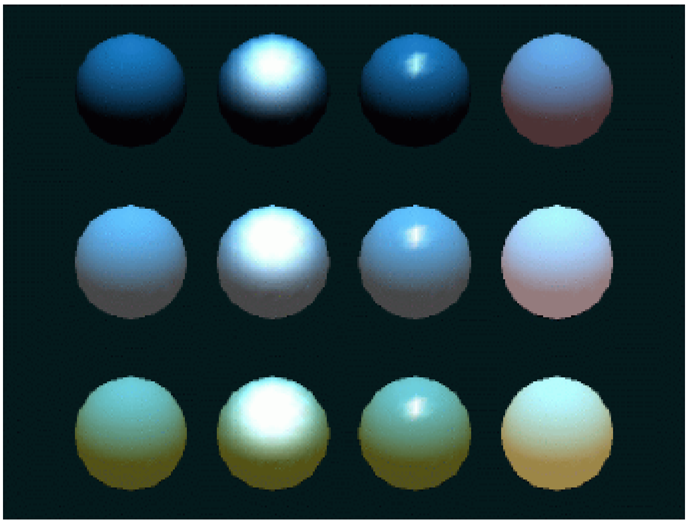
The Halfway Specular Reflection Model

- Let **H = (L+V)/(|L+V|)** be the unit vector that lies half way between the direction of the light source and the direction of viewing.
- The vector **H** is called the "direction of maximum highlights".
- The color of a point is given by:

$$I= f_{att} (I_{sr}\, k_{sr} ,\ I_{sg}\, k_{sg} ,\ I_{sb} k_{sb}) \, [Max(\mathbf{N} \cdot \mathbf{H},0)]^{s}$$

Advantage of the Halfway Model

- Suppose the light source and the viewer are taken to be at infinity.
- Then the vectors **L** and **V** are constant for all points in the scene.
- The direction **H** can be computed once for the entire scene.
- OpenGL uses this technique.



Lighting Examples

- First Column: Diffuse blue reflection only.
- Second Column: Same as first column, but with addition of specular reflection with low shininess.
- Third Column: Same as second column, but with high shininess in specular component.
- Fourth Column: Same as third column, but with addition of emissive light component.

Lighting Examples

- First row: No ambient reflection.
- Second row: Significant ambient reflection.
- Third Row: Coloured ambient reflection.

Phong Illumination Model

- To simulate reflection we should examine surfaces in the reflected direction to determine incoming light
  $\Rightarrow$ *global illumination*
- The Phong model is an *empirical* local model of shiny surfaces – A local model used to simulate effects which can be global in nature
- We only consider reflections of light sources. Assume that the BRDF of shiny surfaces may be approximated by a *spherical cosine function* raised to a power (known as the *Phong exponent*).
- A useful approximation for efficient computation of light-material interactions which produces good renderings under a variety of lighting conditions and material properties

Material Properties

- An object must have material data associated with it to define how diffuse, specular (and shiny) or ambient it is

$$\text{Surface Data} = \begin{cases} k_a : \text{ambient reflectance} \\ k_d : \text{diffuse reflectance} \\ k_s : \text{specular reflectance} \\ a : \text{phong exponent} \end{cases}$$

- Each reflectance factor (k_a, k_d, k_s, respectively for ambient, diffuse and specular reflectance) is the proportion of incoming light reflected due to each light-material interaction
- The phong exponent affects the shininess of the object

Putting it All Together

- Now we can sum the three light contributions – diffuse, specular and ambient to form the total amount of light I that reaches the eye from a point P
  - $I = I_a k_a + I_d k_d$ x Lambert + $I_{sp} k_s$ x phong^s
  - Lambert = max(0, L.N)
  - Phong = max (0, H.N)

Putting it All Together

- I_d and I_{sp} have been given different names because OpenGL allows you to set them separately, but usually they are set to same values.

Adding Colour

- Light of any colour can be constructed by adding certain amounts of red, green and blue.
- Calculate each colour component indiviually and simply add them to form the final colour of the reflected light

Adding Colour

- $I = I_{ar}k_{ar} + I_{dr}k_{dr}$ x Lambert + $I_{spr}k_{sr}$ x phong^s
- $I = I_{ag}k_{ag} + I_{dg}k_{dg}$ x Lambert +$I_{spg}k_{sg}$ x phong^s
- $I = I_{ab}k_{ab} + I_{db}k_{db}$ x Lambert + $I_{spb}k_{sb}$ x phong^s

- Note Lambert & Phong don't depend on colour and need to be computed only only once