Topics

Lighting and shading

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

Work on fragment shaders for different types of lighting

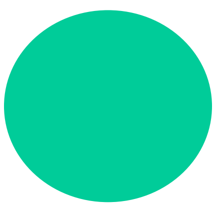
Generate shadows

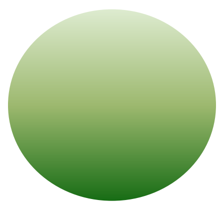
Modeling

How do we represent objects/environments?

- shape — the geometry of the object
- appearance — proper shading

Which one looks like a ball?

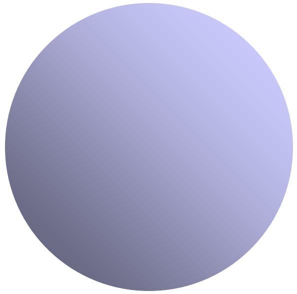




http://www.sketchwiki.com/shading/shading-sphere.php

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

Scattering

Light strikes A

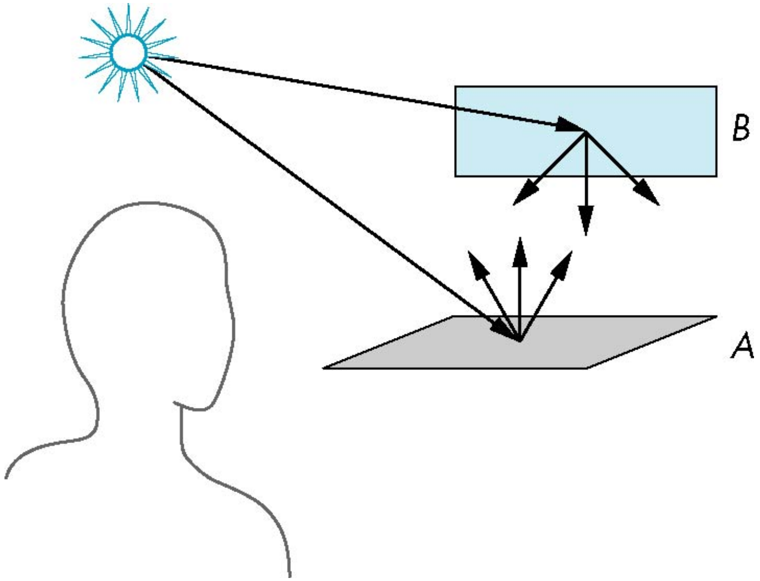
- Some scattered
- Some absorbed

Some of scattered light strikes B

- Some scattered
- Some absorbed

Some of this scattered light strikes A

and so on



E. Angel and D. Shreiner

Rendering Equation

Assign a color for every point

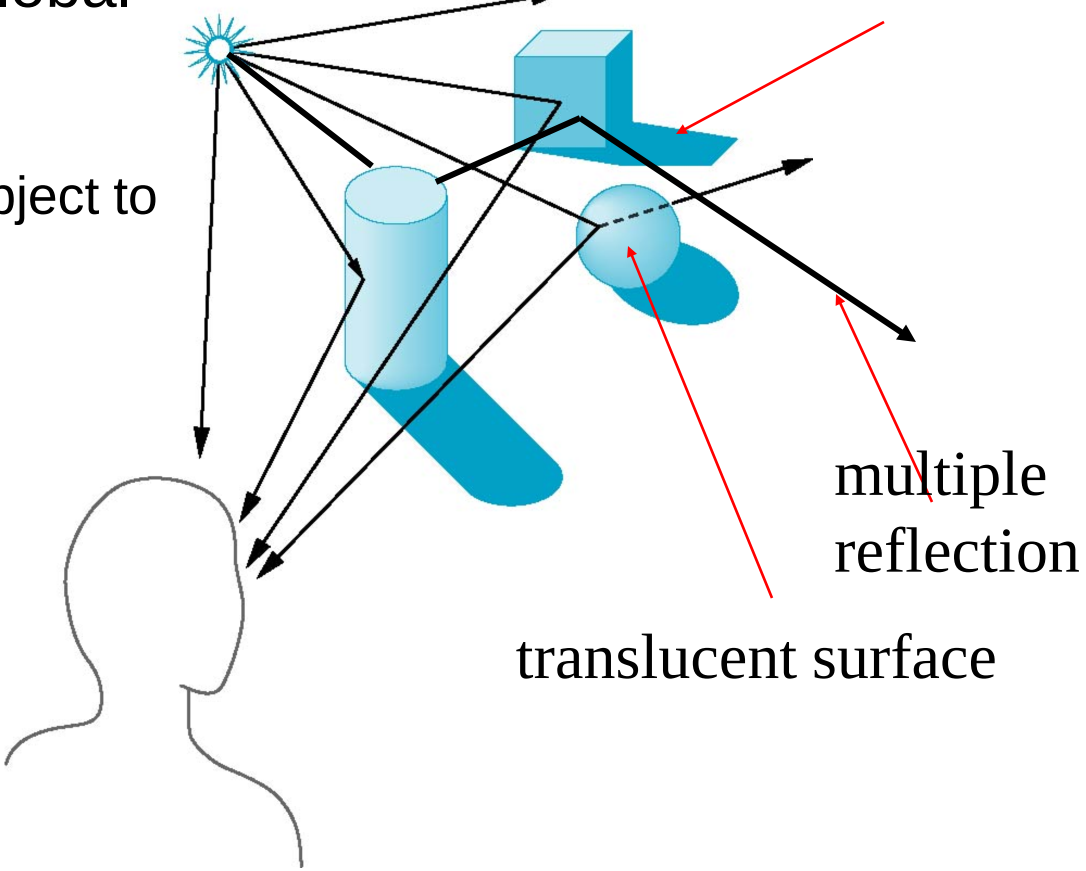
The infinite scattering and absorption of light can be described by a *rendering equation*

- Cannot be solved analytically in general
- Numerical methods are not fast enough
- Approximate solutions for special surfaces
  - E.g., ray tracing is a special case for perfectly reflecting surfaces
  - Not fast enough
- Phong reflection model
  - Simple
  - A compromise between correctness and efficiency

Global Effects

Rendering equation is global including

- Shadows
- Multiple scattering from object to object



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
- Only consider rays leaving the illumination source and reach the viewer’s eye or passing through the COP

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

- Opaque surface: reflection and absorption
- Translucent surface: reflection, absorption, and transmission
- E.g., a surface appears red under white light because the red component of the light is reflected and the rest is absorbed

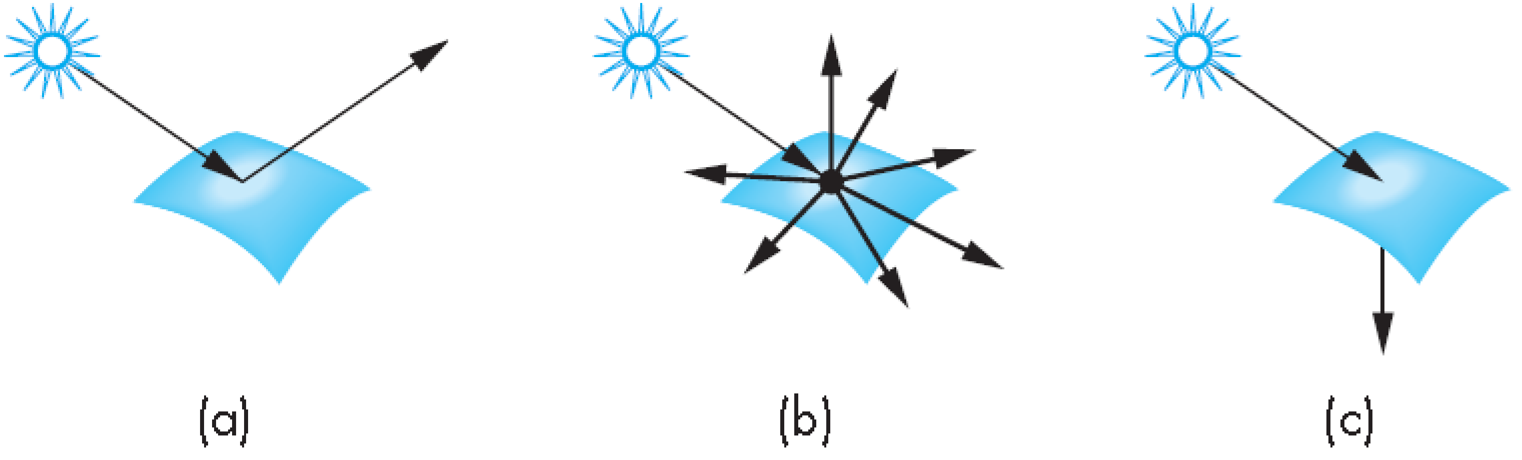
The reflected light is scattered in a manner that depends on the smoothness and orientation of the surface

Light-Material Interaction

Specular surface

Diffuse surface

Translucent surface

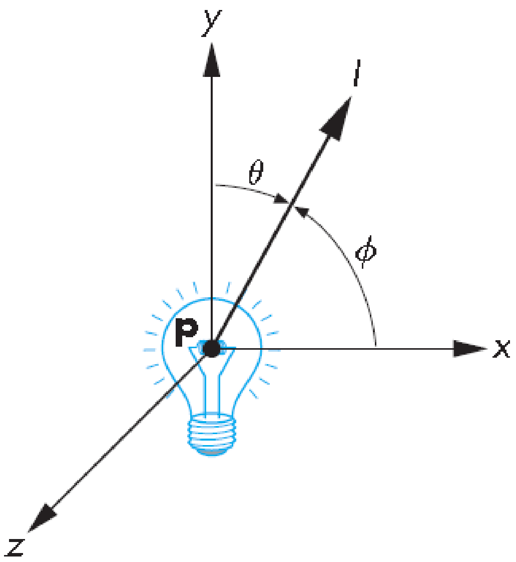


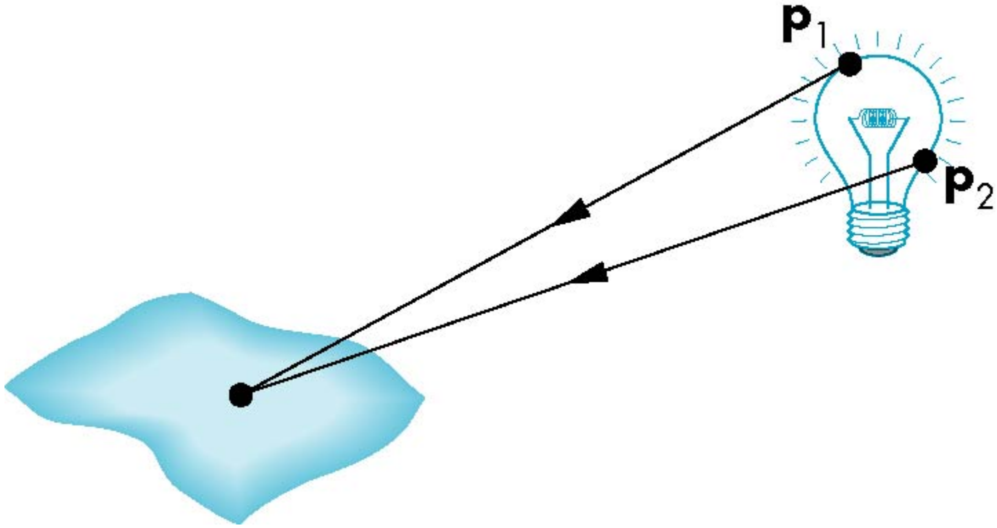
Light Sources

Light

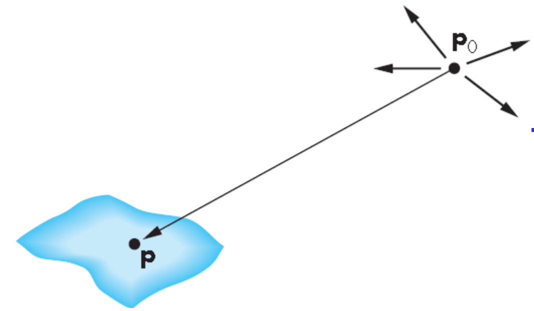
- Self-emission → Light sources
- reflection

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



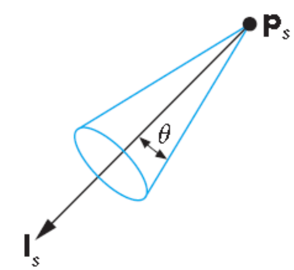


Simple Light Sources



Point source

- Emits light equally in all direction
- Model with position and color – proportional to the inverse square of the distance
- Distant source = infinite distance away (parallel)



Spotlight

- Restrict light from ideal point source

Ambient light

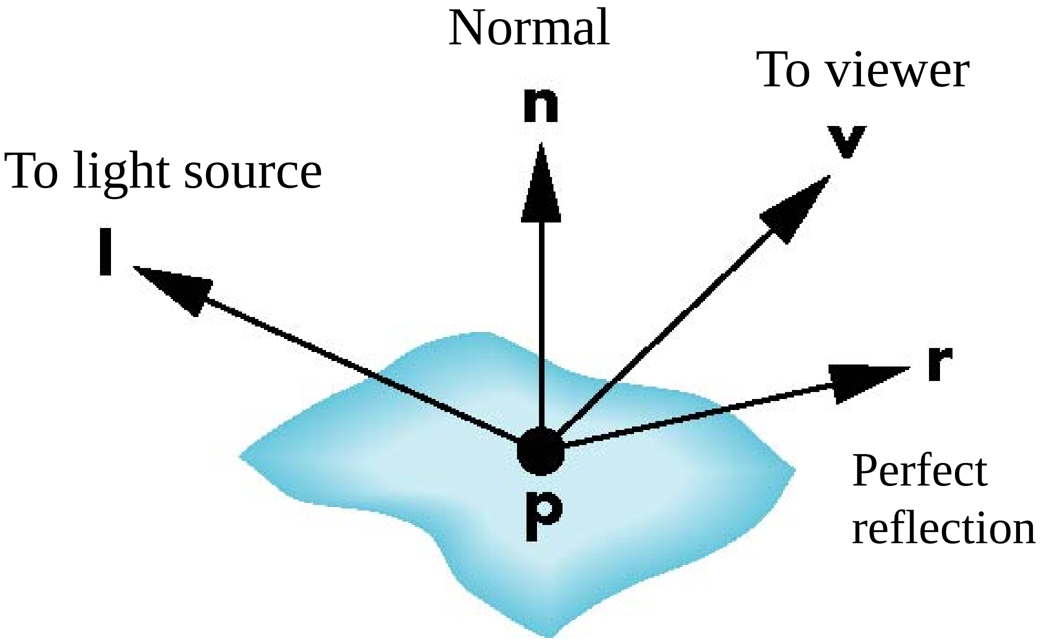
- Can model contribution of many sources and reflecting surfaces
- Uniform illumination everywhere in scene -- An intensity identical at every point

Phong Model

Uses four unit vectors to calculate a color on a surface

- Surface normal **n**
- To viewer **v**
- To light source **l**
- Perfect reflector **r**

Known given the information of surface, viewer, and light source



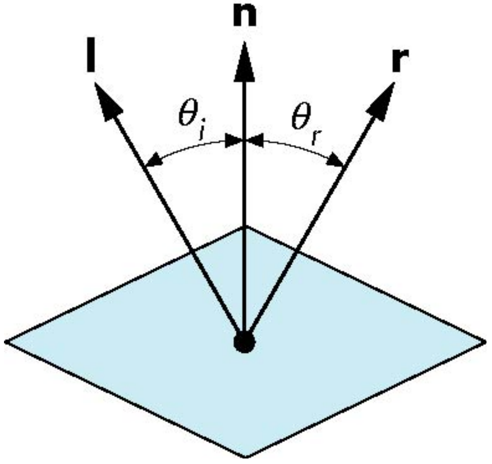
Ideal Reflector

Normal is determined by local orientation

Angle of incidence = angle of relection

The three vectors must be coplanar

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



See a proof in page 275

Phong Model

A

A simple model that can be computed rapidly

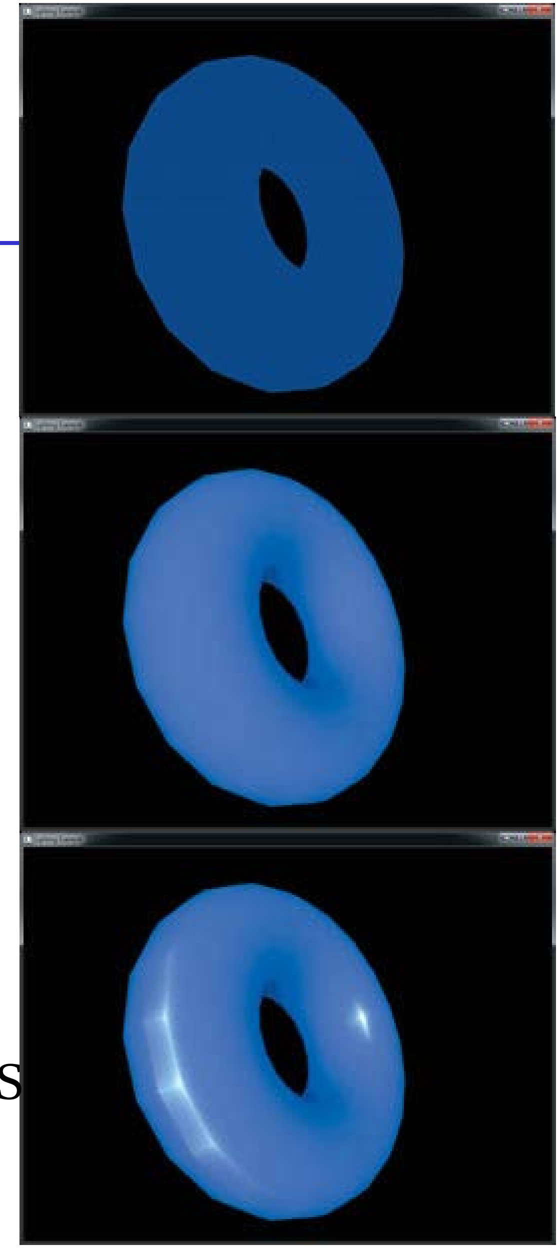
Each light source has three components

- Ambient
- Diffuse
- Specular

Point light source (applies to Diffuse and Specular)

A+D

A+D+S



Shreiner et al

Phong Model

Since the amount reflected determines the color and brightness of the object, a color intensity (r, g, and b) at a surface point P from each source can be computed as

$$\mathbf{I}_i = \mathbf{L}_i{}^T \mathbf{R}_i$$

Consider all sources, the intensity at a surface point P is

$$\mathbf{I} = \sum_{i} \mathbf{I}_i + \mathbf{I}_a$$

↓ Global ambient

Light Sources

Suppose we have a total of M light sources. In the Phong Model, we add the results from each light source

- Each light source has separate diffuse, specular, and ambient terms to allow for maximum flexibility even though this form does not have a physical justification

- Separate red, green and blue components

Hence, 9 coefficients for each point source

$$\mathbf{L}_i = \begin{bmatrix} L_{ira} & L_{iga} & L_{iba} \\ L_{ird} & L_{igd} & L_{ibd} \\ L_{irs} & L_{igs} & L_{ibs} \end{bmatrix}$$

Material Properties

Reflection term for a point P on the surface

$$\mathbf{R}_i = \begin{bmatrix} R_{ira} & R_{iga} & R_{iba} \\ R_{ird} & R_{igd} & R_{ibd} \\ R_{irs} & R_{igs} & R_{ibs} \end{bmatrix}$$

- **Material properties match light source properties → $\mathbf{R}_i$ is proportional to**
  - Nine reflection coefficients related to materials
    - k_{dr}, k_{dg}, k_{db}, k_{sr}, k_{sg}, k_{sb}, k_{ar}, k_{ag}, k_{ab}
  - Shininess coefficient α

Example of Phong Model

For example, for each source, the red intensity is

$$I_{ir} = I_{ira} + I_{ird} + I_{irs} = R_{ira}L_{ira} + R_{ird}L_{ird} + R_{irs}L_{irs}$$

ambient diffuse specular

For all sources, the red intensity is

$$I_r = \sum_i (I_{ira} + I_{ird} + I_{irs}) + I_{ar}$$

We can consider each component individually

In the later slides, we will omit the subscript for the light source

Ambient Reflection

Ambient light is the result of multiple interactions between (large) light sources and the objects in the environment

$\mathbf{I}_a$ is the same for every point on the surface

Amount and color depend on both the color of the light(s) and the material properties of the object ***regardless of the position of the reflection***

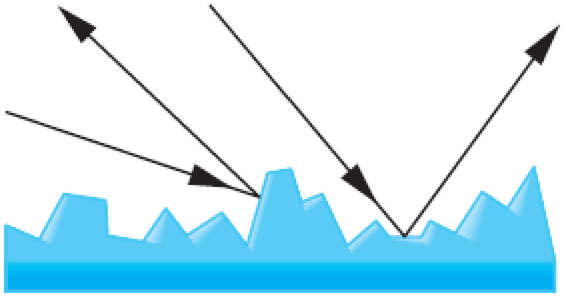
$$\mathbf{I}_a = \mathbf{k}_a \mathbf{L}_a$$

reflection coef intensity of ambient light

$0 \leq k_{ar}, k_{ag}, k_{ab} \leq 1$ are ambient reflection coefficient, constants to the material for r, g, b

Diffuse Reflection

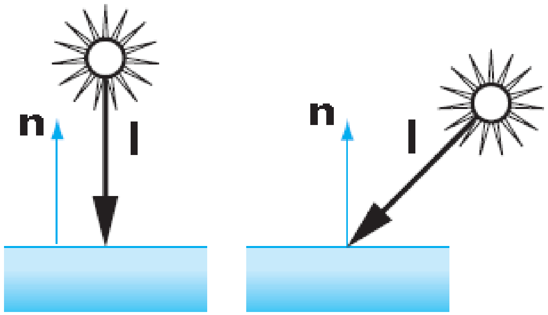
A perfect diffuse reflector reflects light in all directions



Determines by

- Materials – how much reflected
  - There are also three coefficients, $0 \leq k_{dr}, k_{dg}, k_{db} \leq 1$ that show how much of each color component is reflected
- Relative position of the light source to the surface

Lambertian surfaces

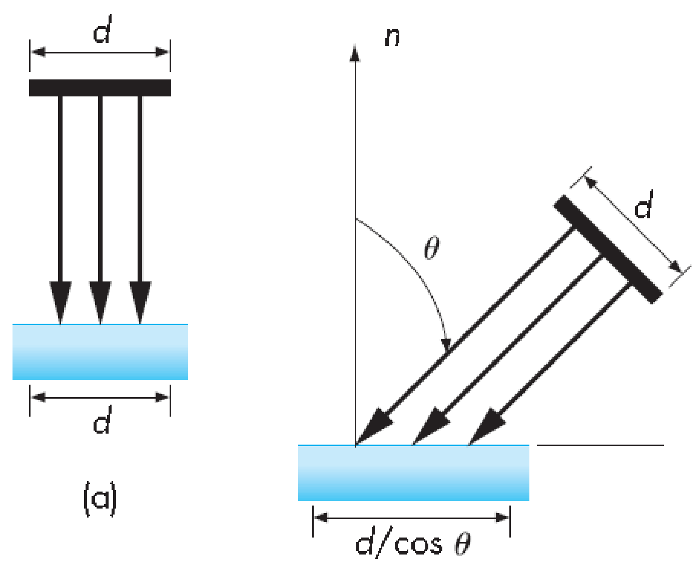


Lambertian Surface

Amount of light reflected is proportional to the vertical component of incoming light

- reflected light $R_d \propto \cos\theta$
- $\cos\theta = \mathbf{l} \cdot \mathbf{n}$ if vectors normalized
- $\mathbf{I}_d = \mathbf{k}_d(\mathbf{l} \cdot \mathbf{n})\mathbf{L}_d$

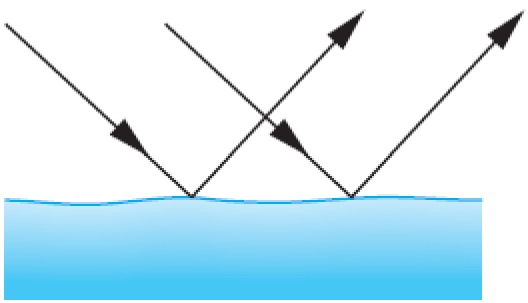
diffuse coef

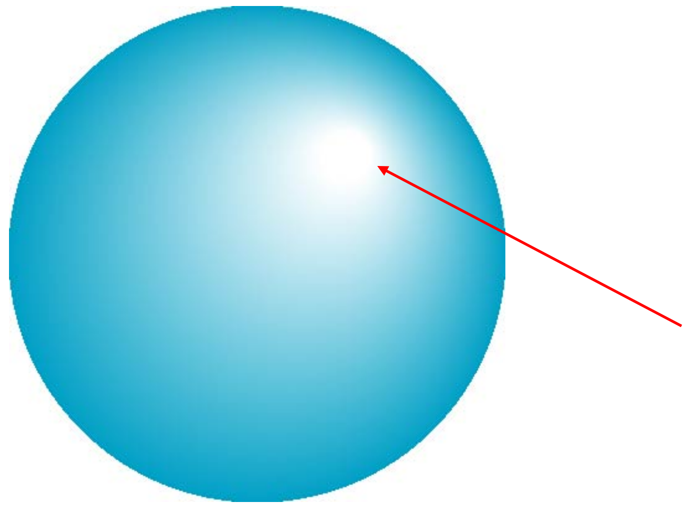


Specular Surfaces

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

Modeling Specular Reflections

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

shininess coef

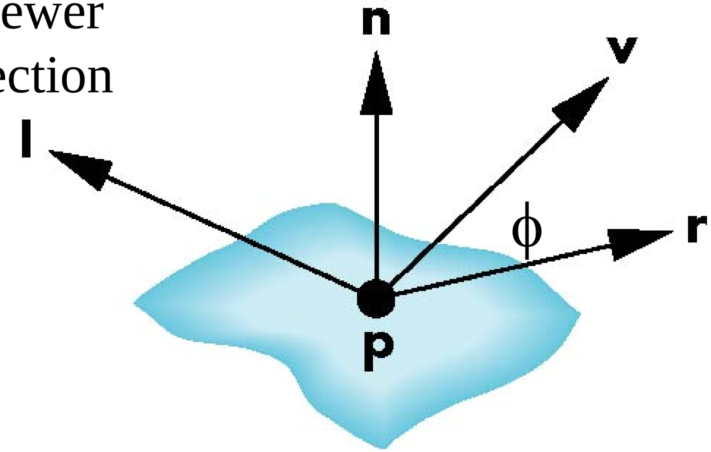
$$\mathbf{I}_s = \mathbf{k}_s\, \mathbf{L}_s \cos^{\alpha}\phi \;\Longrightarrow\; \boldsymbol{I}_s = \boldsymbol{k}_s\, \boldsymbol{L}_s \max((\mathbf{r}\cdot\mathbf{v})^{\alpha}, 0)$$

Angle between viewer and the ideal reflection

reflected intensity

incoming intensity

absorption coef

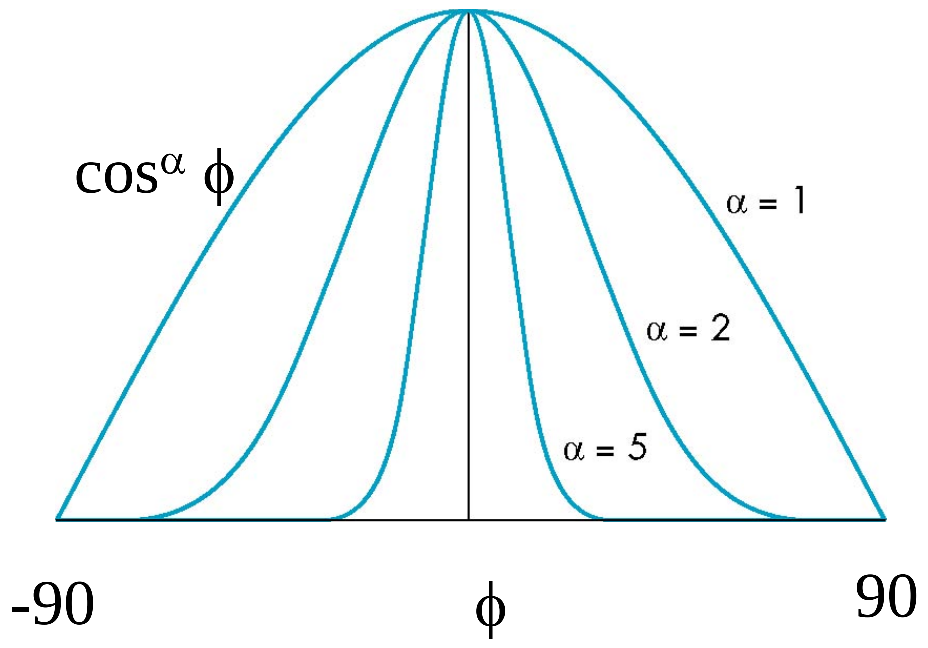


E. Angel and D. Shreiner

The Shininess Coefficient

Values of α between 100 and 200 correspond to metals

Values between 5 and 10 give surface that look like plastic

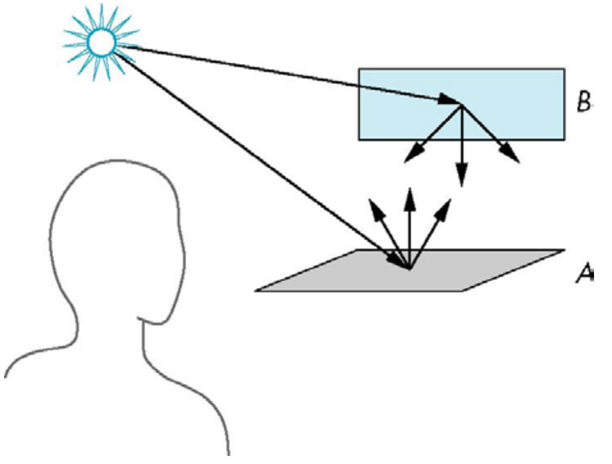


Distance Terms

The light from a point source that reaches a surface is inversely proportional to the square of the distance between them

Add a factor $\frac{1}{a + bd + cd^2}$ to the diffuse and specular terms

The constant and linear terms soften the effect of the point source



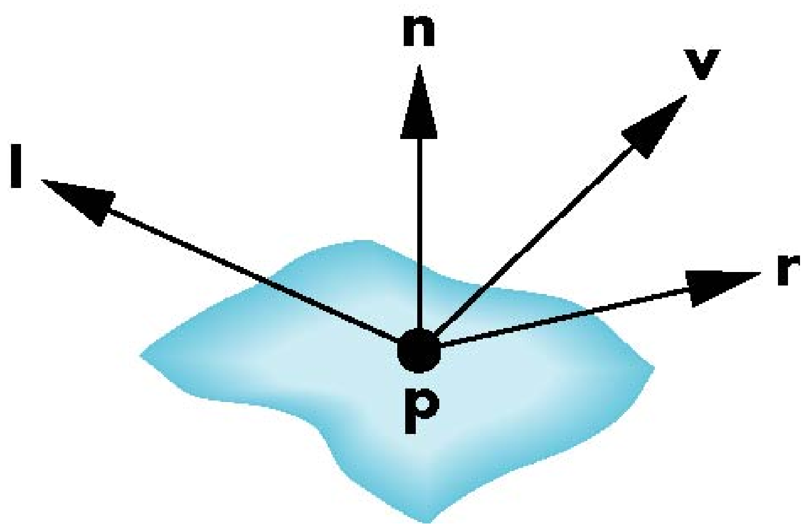
Overall Phong Model

For each light source and each color component, the Phong model can be written as

$$I = \frac{k_d L_d max(\mathbf{l} \cdot \mathbf{n}, 0) + k_s L_s \max\left((\mathbf{r} \cdot \mathbf{v})^{\alpha}, 0\right)}{a + bd + cd^2} + k_a I_a$$

For each color component we add

contributions from all sources



Modified Phong Model

The specular term in the Phong model is time consuming. For each vertex, it requires

- Calculation of the reflection vector
- Calculation of the dot product of the reflection vector and the view vector $\mathbf{r} \cdot \mathbf{v}$

Blinn proposed an approximation using the halfway vector that is more efficient

The Halfway Vector

h is normalized vector halfway between **l** and **v**

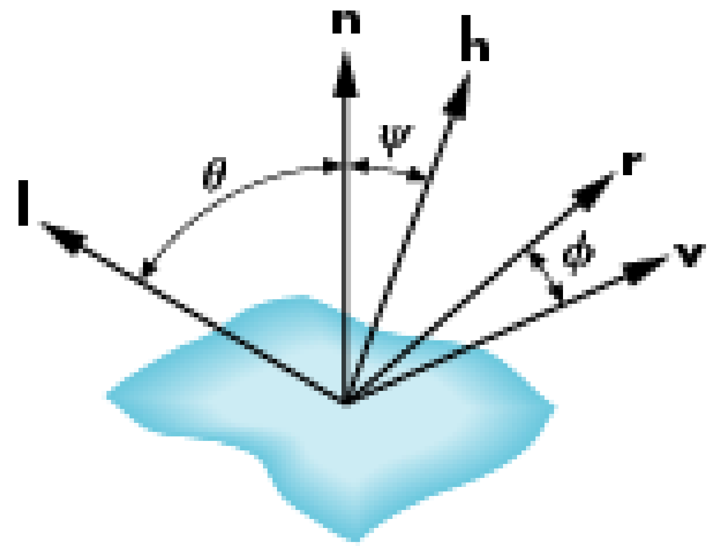
$$\mathbf{h} = (\mathbf{l} + \mathbf{v}) / |\mathbf{l} + \mathbf{v}|$$

If **v** is on the same plane of **l, n**, and **r**, the ***halfway angle*** ψ between **n** and **h** is

$$2\psi = \phi$$

Note that halfway angle is half of angle

between r and v if vectors are coplanar



Using the halfway vector

Replace $(\mathbf{v} \cdot \mathbf{r})^\alpha$ by $(\mathbf{n} \cdot \mathbf{h})^\beta$

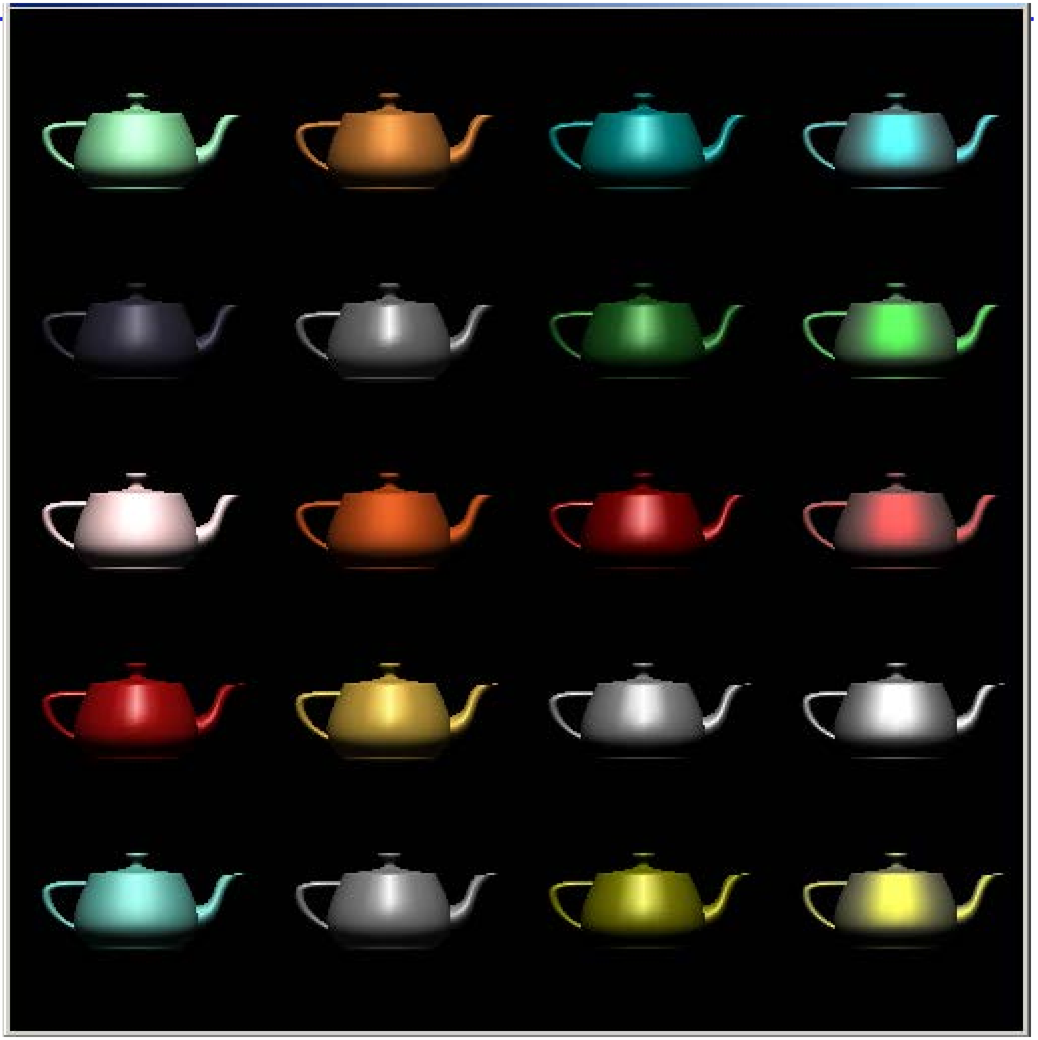
β is chosen to match shininess

Resulting model is known as the **modified Phong** or **Blinn-Phong lighting model**

It is the default lighting model in OpenGL pipeline

Example

Only differences in these teapots are the parameters in the modified Phong model



Computation of Vectors

Need to compute the four vectors

- Surface normal **n**
- To viewer **v**
- To light source **l**
- Perfect reflector **r**

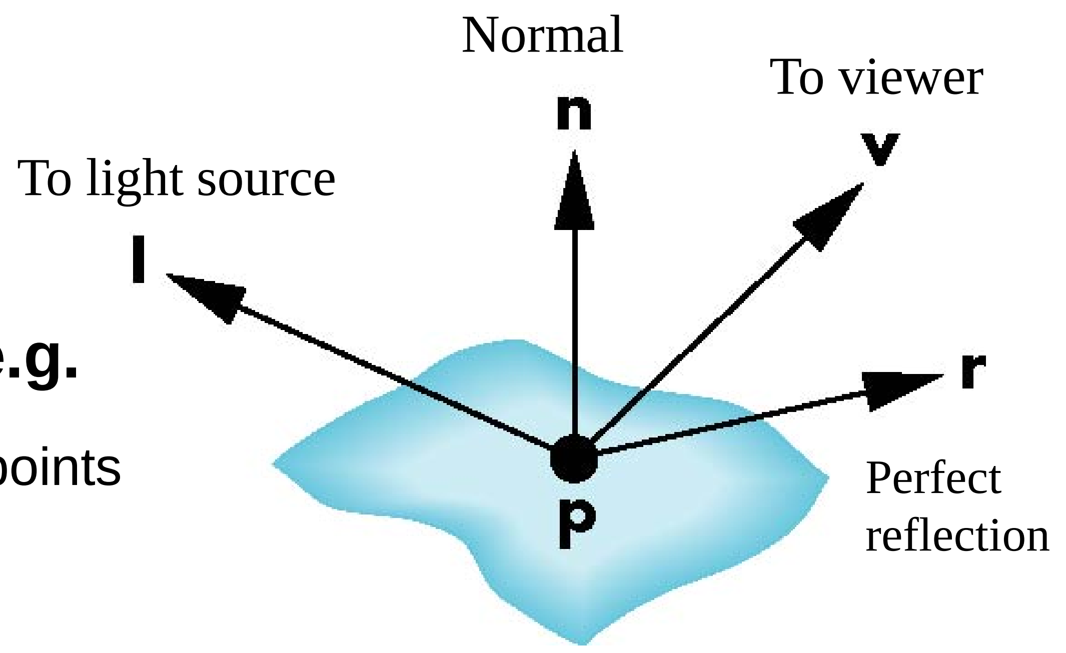
Simplifications can apply, e.g.

- Normal can be the same for all points

a flat polygon

- Light direction is the same for all points

if the light is far away from the surface



E. Angel and D. Shreiner: Interactive Computer Graphics 6E © Addison-Wesley 2012

Computation of Vectors

l and **v** are specified by the application

h can be computed from **l** and **v**

How to calculate **n?**

Depending on surface

OpenGL leaves determination of normal to application, e.g., the obj file contains the normals

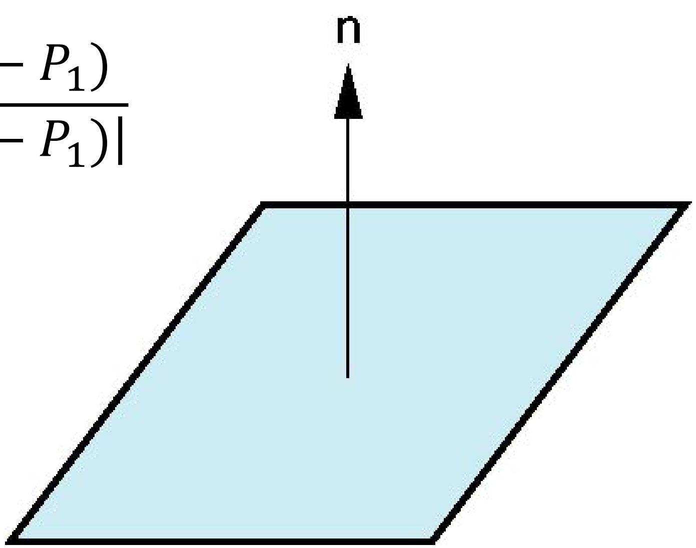
Plane Normals

Plane can be determined by three points P_1, P_2, P_3 or normal $\mathbf{n}$ and P_0

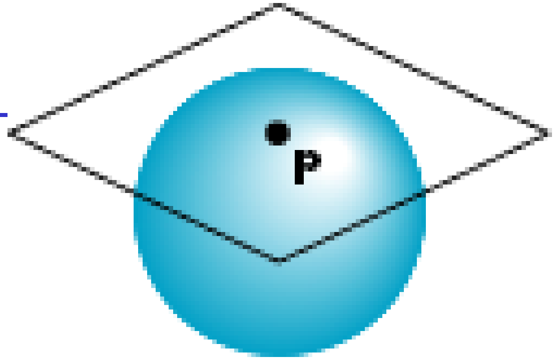
Given three noncolinear points, e.g., the three vertices of a triangle P_1, P_2, and P_3, normal can be obtained by

$$\mathbf{n} = \frac{(P_3 - P_1) \times (P_2 - P_1)}{|(P_3 - P_1) \times (P_2 - P_1)|}$$

Order of vectors is important!



Normal to Sphere



E. Angel and D. Shreiner: Interactive Computer Graphics 6E © Addison-Wesley 2012

How we compute normals for curved surfaces?

Depend on how we model the surface.

Implicit function *of a unit sphere centered at the origin*

$$f(x, y. z) = x^2 + y^2 + z^2 - 1 = 0$$

Or in vector form

$$f(\mathbf{p}) = \mathbf{p} \cdot \mathbf{p} - 1 = 0$$

Normal is given by gradient

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

$$\Rightarrow \quad \mathbf{n} = \frac{\mathbf{n}'}{|\mathbf{n}'|} = \mathbf{p}$$