



Introduction to Computer Graphics

3. Shading

National Chiao Tung Univ, Taiwan

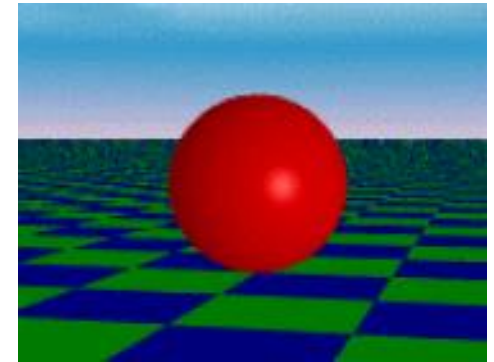
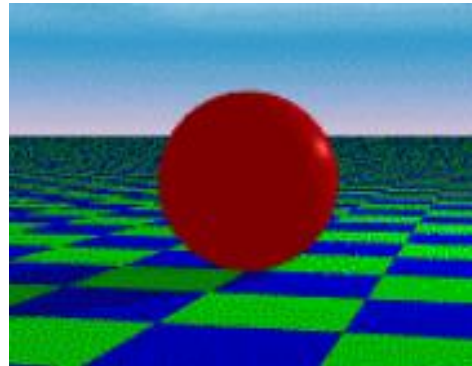
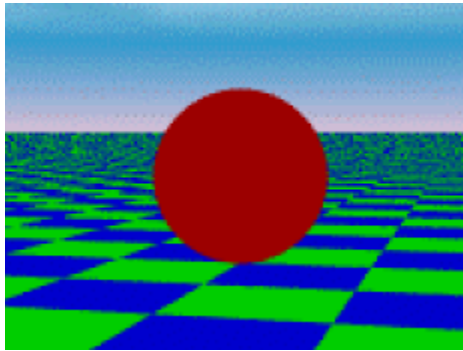
By: I-Chen Lin, Assistant Professor

Textbook: E. Angel, Interactive Computer Graphics, 5th Ed., Addison Wesley

Ref: Hearn and Baker, Computer Graphics, 3rd Ed., Prentice Hall

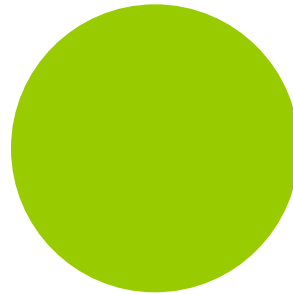
Illumination and Shading

- Is it a ball or a plate?
- What color should I set for each pixel?

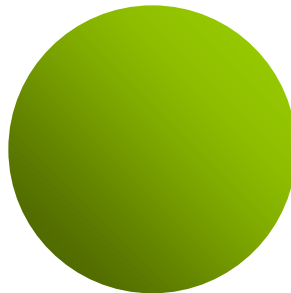


Why Do We Need Shading?

- Suppose we color a sphere model. We get something like

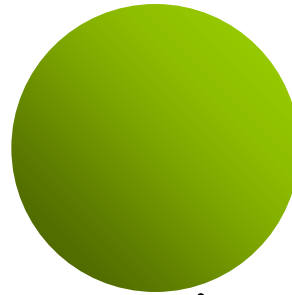


- But we want



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 the viewer
 - Surface orientation

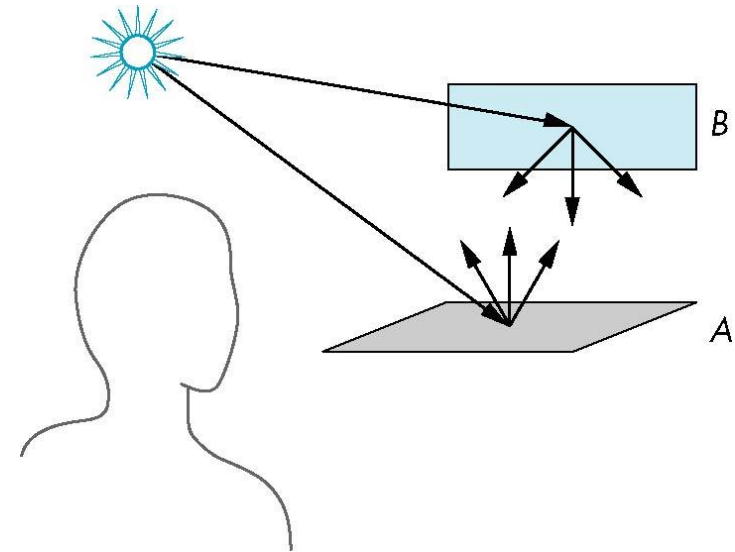
Illumination and Shading

- Factors that affect the “color” of a pixel.
 - Light sources
 - Emittance spectrum (color)
 - Geometry (position and direction)
 - Directional attenuation
 - Objects’ surface properties
 - Reflectance spectrum (color)
 - Geometry (position, orientation, and micro-structure)
 - Absorption

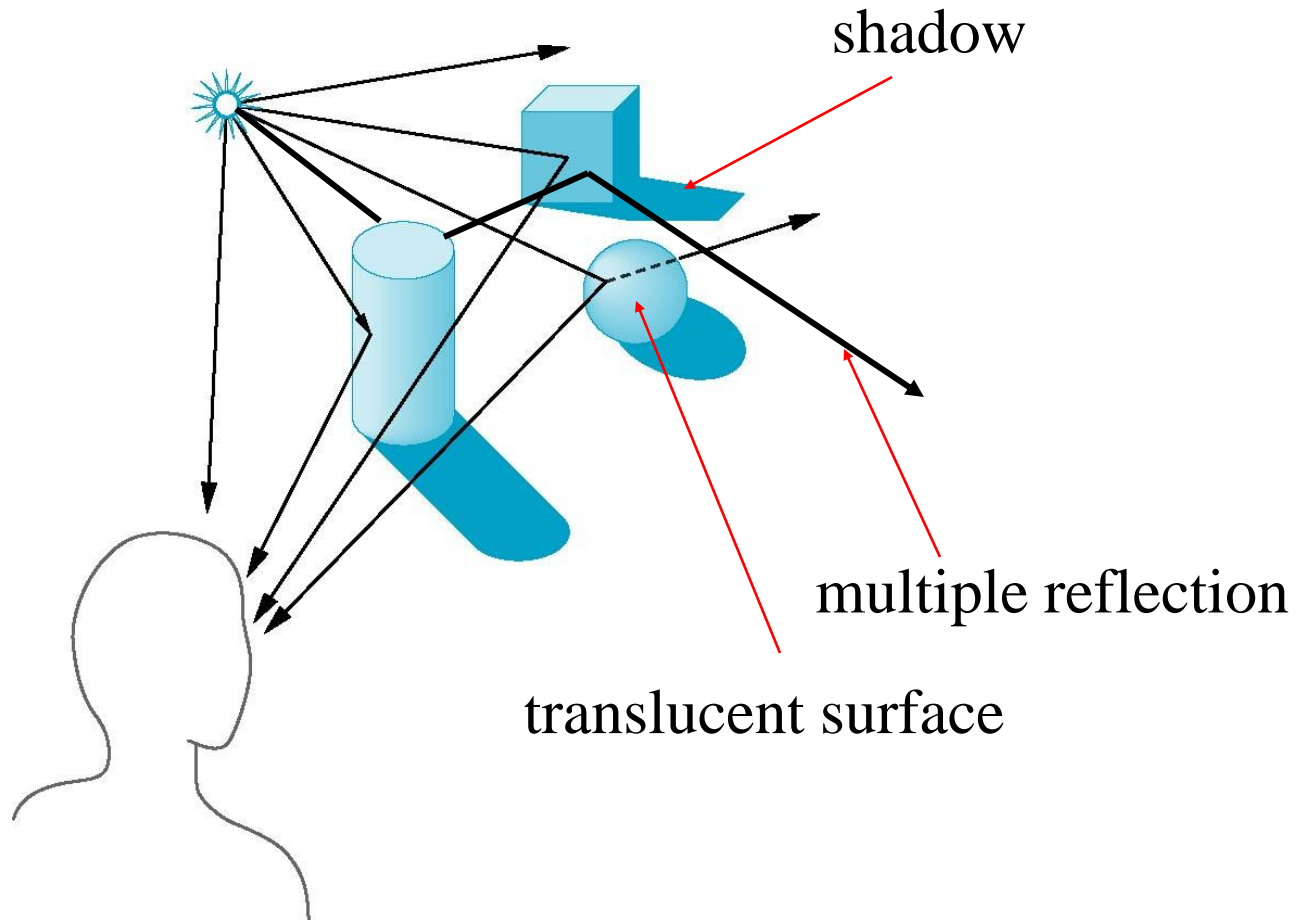


Scattering

- 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



Global Effects





Rendering Equation

- The infinite scattering and absorption of light can be described by the *rendering equation*
 - $[outgoing] - [incoming] = [emitted] - [absorbed]$
 - $[outgoing] = [emitted] + [reflected] (+ [transmitted])$
 - Cannot be solved in general
 - Ray tracing is a special case for perfectly reflecting surfaces
- Rendering equation is global and includes
 - Shadows
 - Multiple scattering from object to object

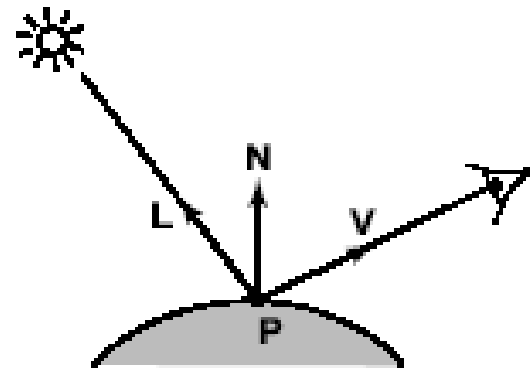
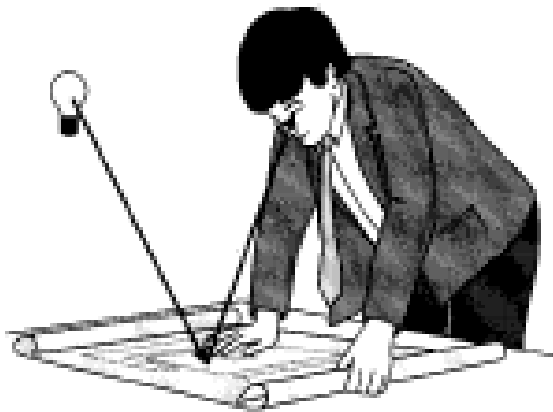


Local vs Global Illumination

- Correct shading requires a global calculation
 - Incompatible with a 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

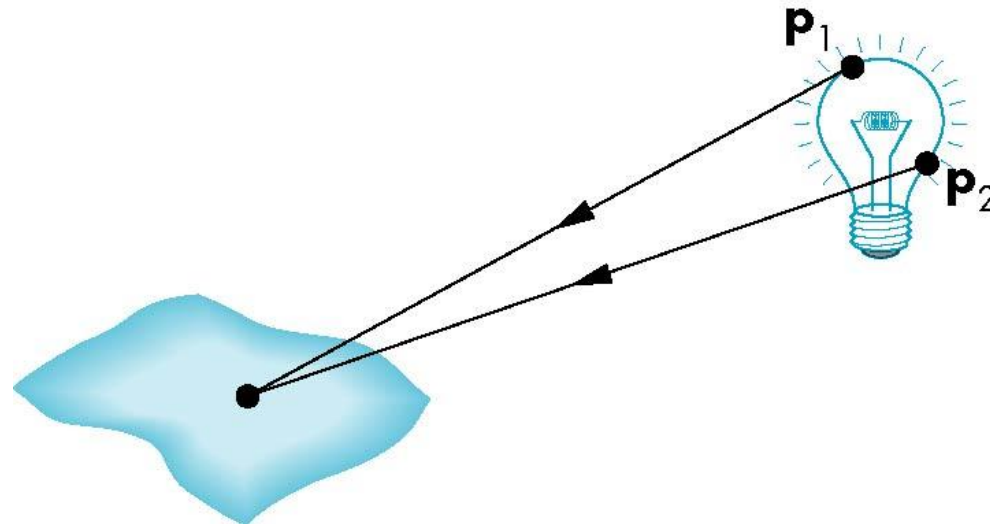
Local Illumination

- Adequate for real-time graphics.
- No inter-reflection, no refraction, no realistic shadow



Light Sources

- General light sources are difficult to simulate
 - because we must integrate light coming from all points on the source.



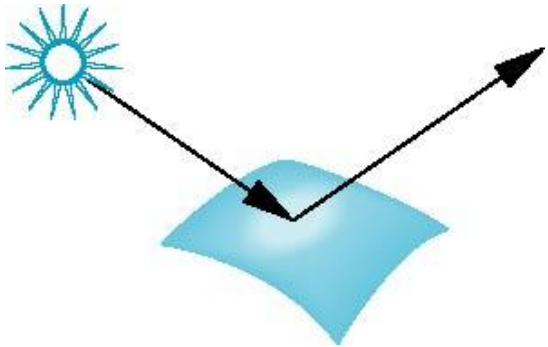


Simple Light Sources

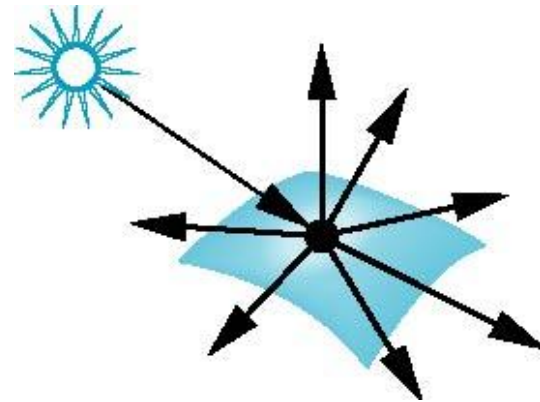
- Point source
 - Model with position and color
 - Distant source = infinite distance away (parallel)
- Spotlight
 - Restrict light from ideal point source
- Ambient light
 - Same amount of light everywhere in scene
 - Can model contribution of many sources and reflecting surfaces

Surface Types

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



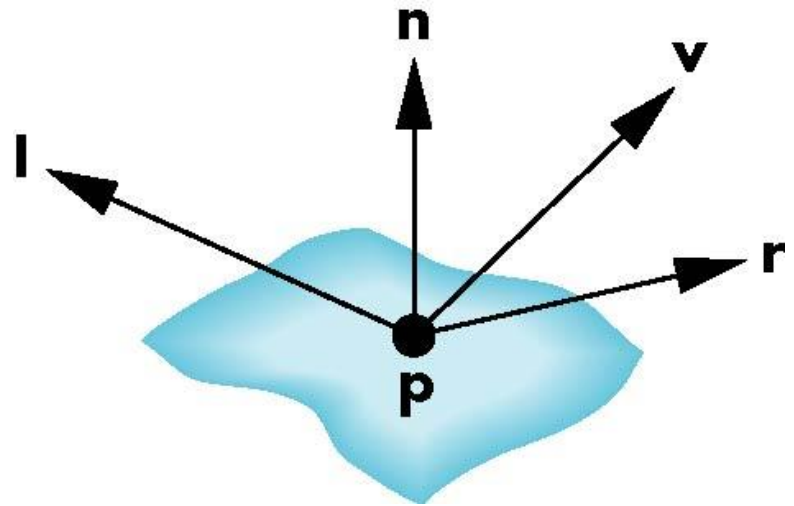
smooth surface



rough surface

Phong Reflection Model

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

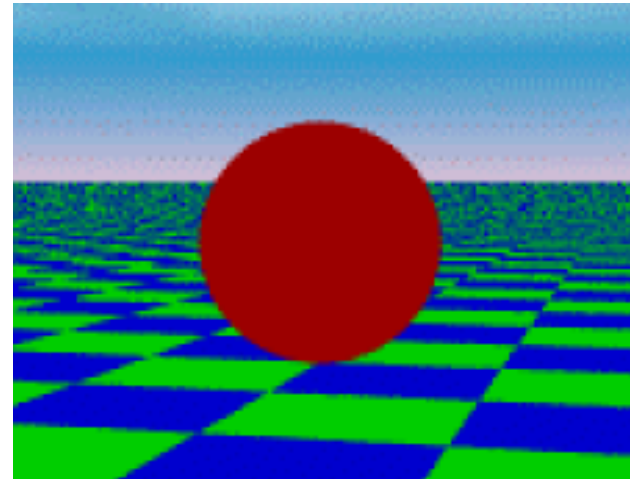


$$\mathbf{r} = 2 (\mathbf{l} \cdot \mathbf{n}) \mathbf{n} - \mathbf{l}$$

Ambient Light

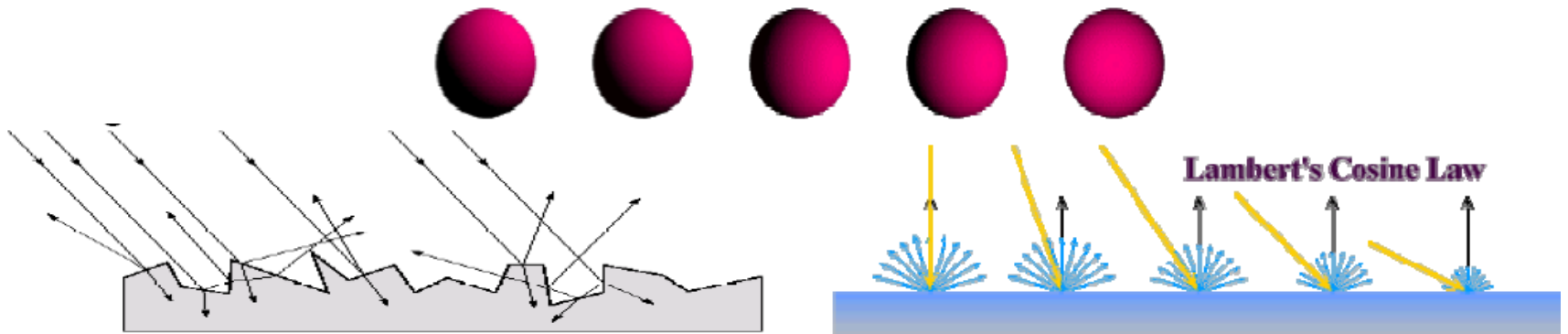
- The result of multiple interactions between (large) light sources and the objects in the environment.

- $I_{ambient} = K_a \cdot I_a$



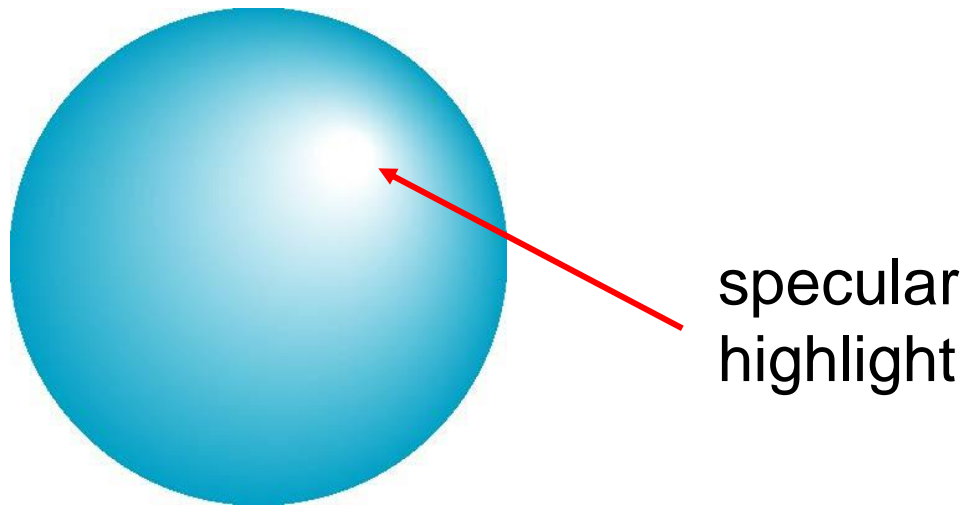
Diffuse Reflection

- Light scattered equally in all directions
- Reflected intensities vary with the direction of the light.
- Lambertian Surface
 - Perfect diffuse reflector
- $I_{diffuse} = K_d \cdot I_d (n \cdot v)$



Specular Surfaces

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



Modeling Specular Reflections

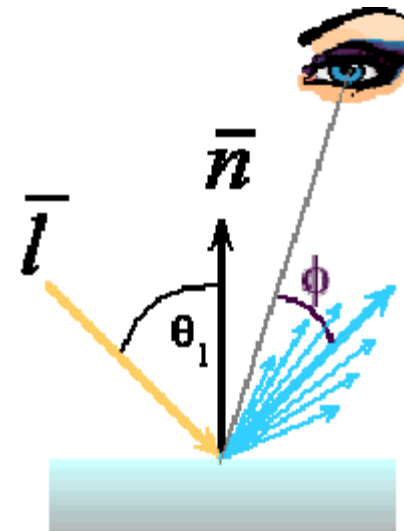
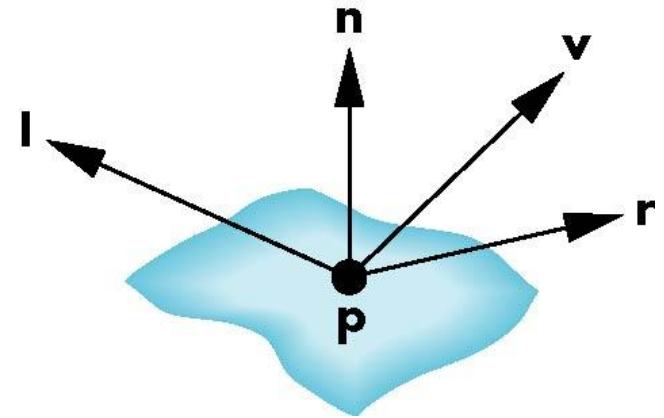
- Phong proposed

$$\mathbf{I}_r \sim k_s \mathbf{I} \cos^a \phi$$

reflected
intensity

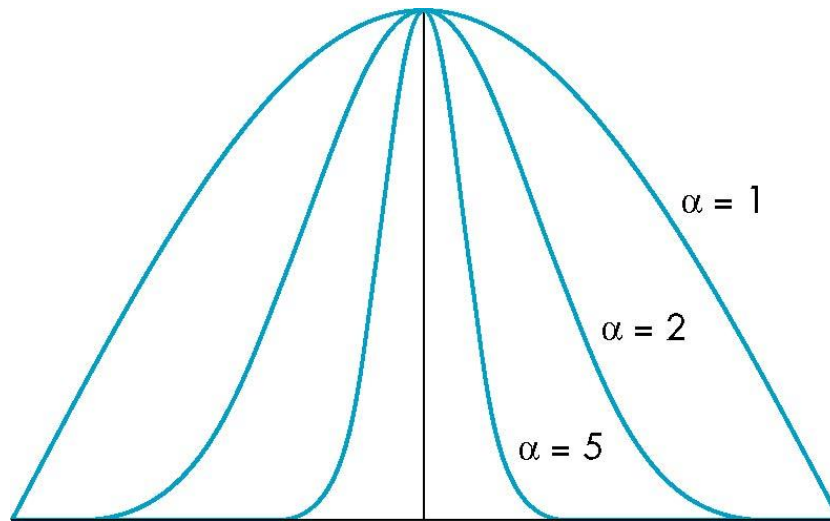
absorption coef

shininess coef
incoming intensity



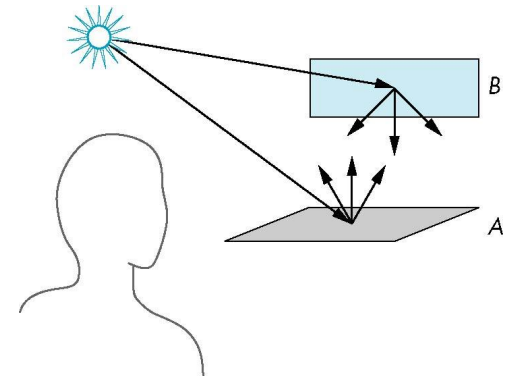
The Shininess Coefficient

- Values of α between 100 and 200 correspond to metals
- Values between 5 and 10 give surface that look like plastic



Distance Terms

- Inversely proportional to the square of the distance between them
- Add a factor of the form $1 / (ad + bd + cd^2)$ to the diffuse and specular terms
- The constant and linear terms soften the effect of the point source

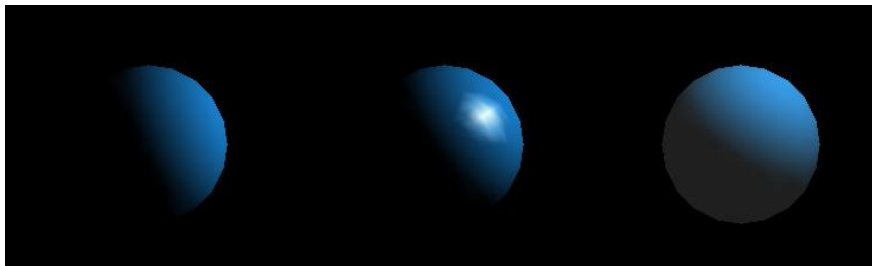


Coefficients

- 9 coefficients for each point light source
 - $I_{dr}, I_{dg}, I_{db}, I_{sr}, I_{sg}, I_{sb}, I_{ar}, I_{ag}, I_{ab}$
- Material properties
 - Nine absorption coefficients
 - $k_{dr}, k_{dg}, k_{db}, k_{sr}, k_{sg}, k_{sb}, k_{ar}, k_{ag}, k_{ab}$
 - Shininess coefficient a

Adding up the Components

- A primitive virtual world with lighting can be shaded by combining the three light components .
- $I = I_{ambient} + I_{diffuse} + I_{specular}$
 $= k_a I_a + k_d I_d (l \cdot n) + k_s I_s (v \cdot r)^a$



diffuse

diffuse

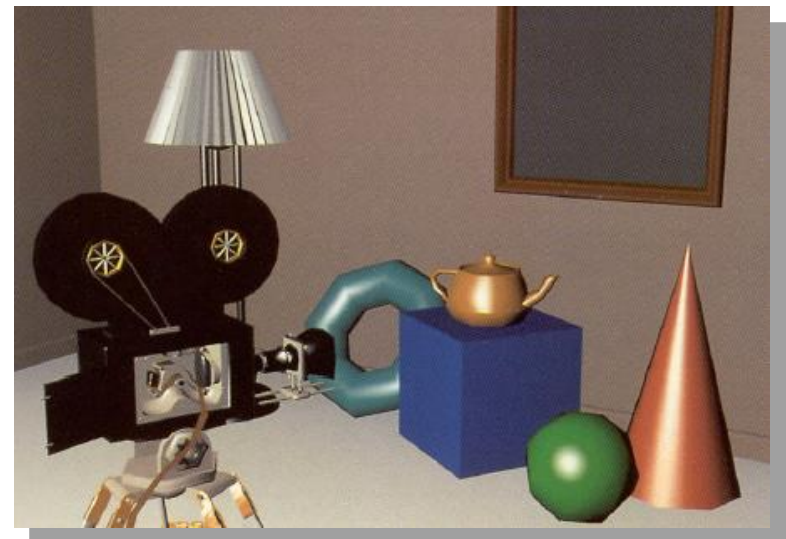
diffuse

+

specular

+

ambient





Modified Phong Model

- Problem: In the specular component of Phong model, it requires the calculation of a new reflection vector and view vector for each vertex

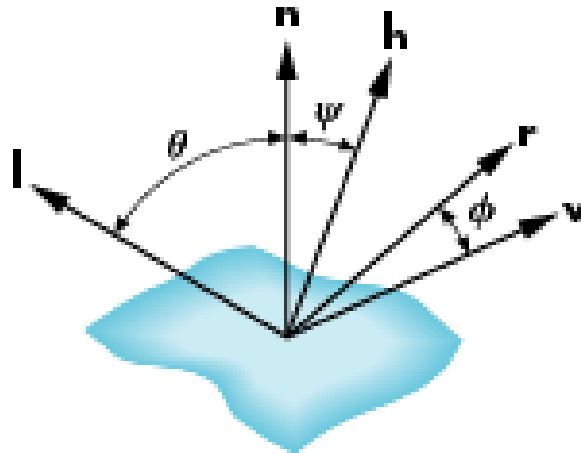
$$r = 2 (l \cdot n) n - l$$

- Blinn suggested an approximation using the halfway vector that is more efficient

The Halfway Vector

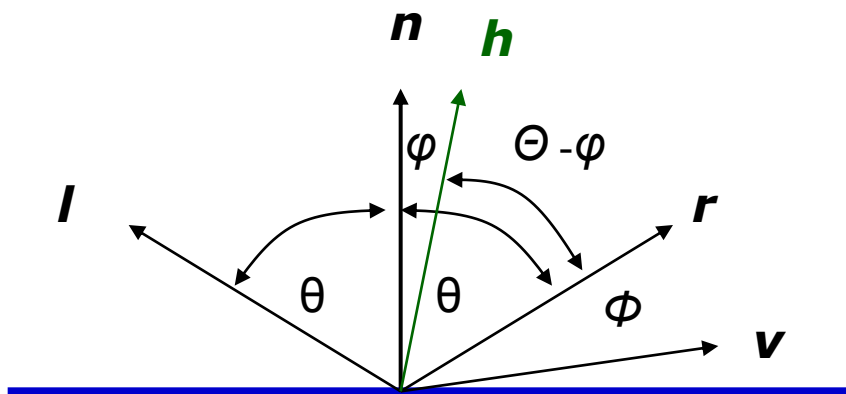
- \mathbf{h} is normalized vector halfway between \mathbf{l} and \mathbf{v}

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



Using the Halfway Angle

- Replace $(\mathbf{v} \cdot \mathbf{r})^a$ by $(\mathbf{n} \cdot \mathbf{h})^b$
- b is chosen to match shininess
- Note that halway angle is half of angle between \mathbf{r} and \mathbf{v} if vectors are coplanar



$$\theta + \varphi = \theta - \varphi + \phi$$

$$2\varphi = \phi$$

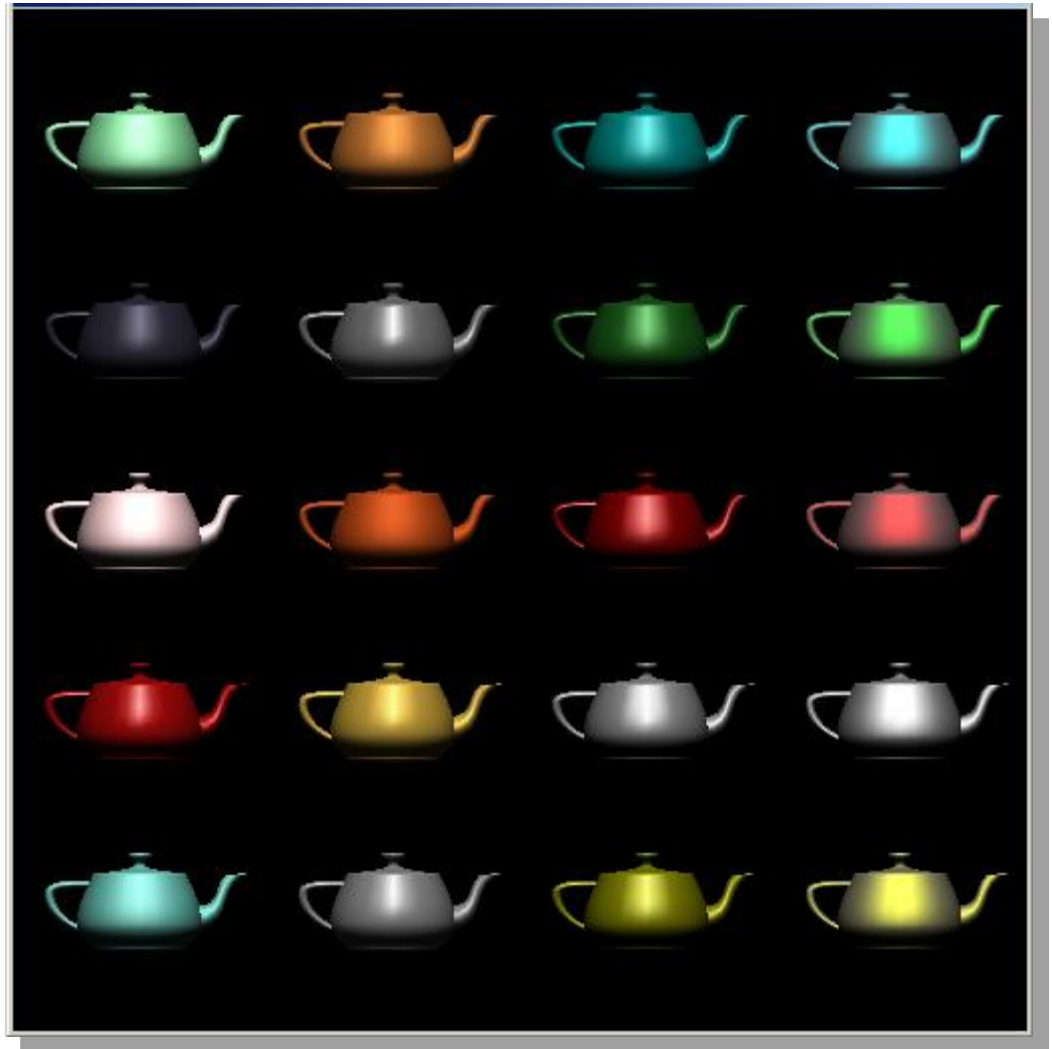


Using the Halfway Angle

- Resulting model is known as the modified Phong or Blinn lighting model
 - Specified in OpenGL standard and most real-time applications

Example

Teapots with
different
parameters.





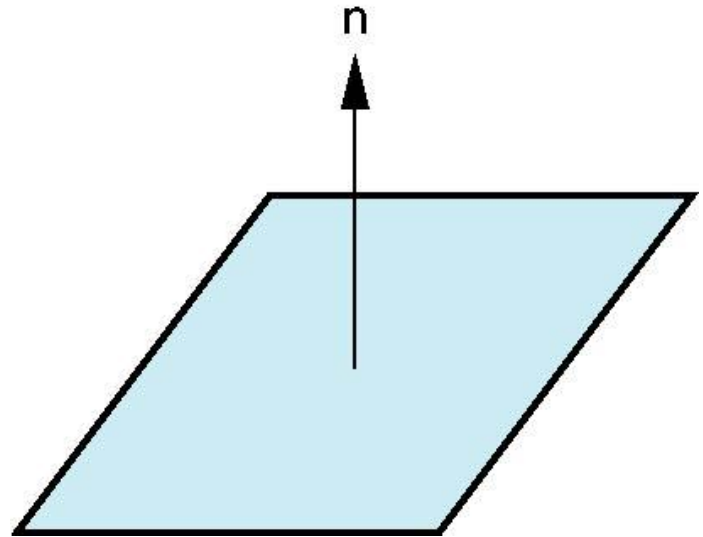
Computation of Vectors

- l and v : specified by the application
- r : computed from l and n
- Determine n
 - OpenGL leaves determination of normal to application
 - Exception for GLU quadrics and Bezier surfaces

Plane Normals

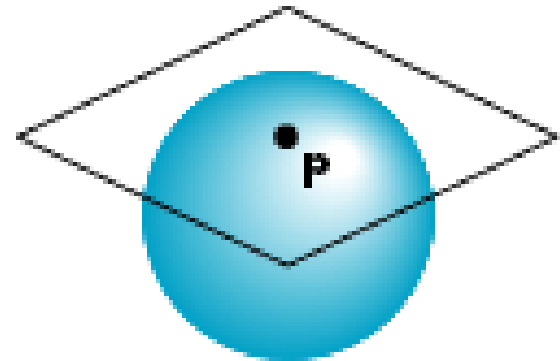
- Equation of plane: $ax+by+cz+d = 0$
- Normal can be obtained by

$$\mathbf{n} = (\mathbf{p}_2 - \mathbf{p}_0) \times (\mathbf{p}_1 - \mathbf{p}_0)$$



Normal to Sphere

- Implicit function $f(x,y,z)=0$
- Normal given by gradient
- Sphere
 - $n = [\partial f/\partial x, \partial f/\partial y, \partial f/\partial z]^T$



Parametric Form

- For sphere

$$x = x(u, v) = \cos u \sin v$$

$$y = y(u, v) = \cos u \cos v$$

$$z = z(u, v) = \sin u$$

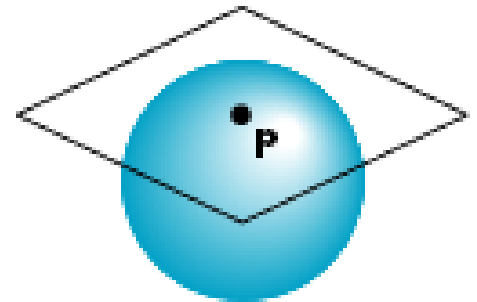
- Tangent plane determined by vectors

$$\partial \mathbf{p} / \partial u = [\partial x / \partial u, \partial y / \partial u, \partial z / \partial u]^T$$

$$\partial \mathbf{p} / \partial v = [\partial x / \partial v, \partial y / \partial v, \partial z / \partial v]^T$$

- Normal given by cross product

$$\mathbf{n} = \partial \mathbf{p} / \partial u \times \partial \mathbf{p} / \partial v$$



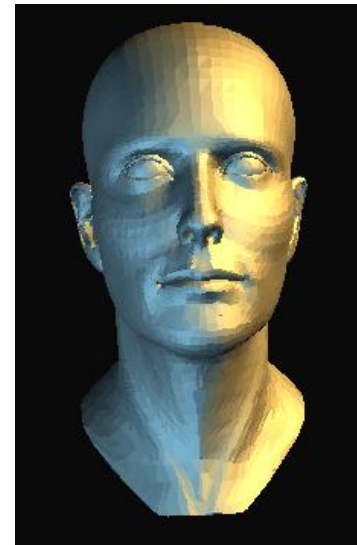
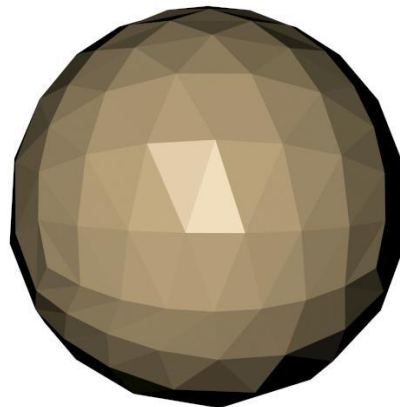


Polygonal Shading

- Practical implementation to fill color within a polygon.
 - Flat shading
 - Gouraud shading (smooth shading)
 - Phong shading

Flat Shading

- Flat or constant shading.
- Assume l , n , v are constant for a polygon.
 - Shading calculation: once for each polygon.



Gouraud Shading

- Find average normal at each vertex
- Apply Phong lighting model at each vertex
- Interpolate vertex shades across each polygon



Flat



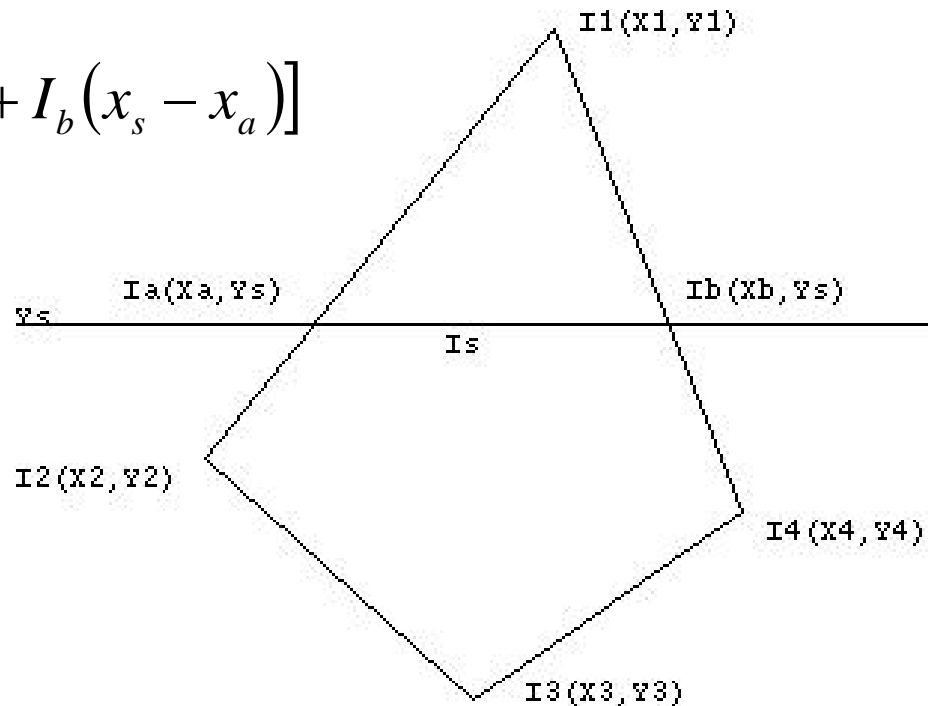
Gouraud

Gouraud Shading

$$I_a = \frac{1}{y_1 - y_2} [I_1(y_s - y_2) + I_2(y_1 - y_s)]$$

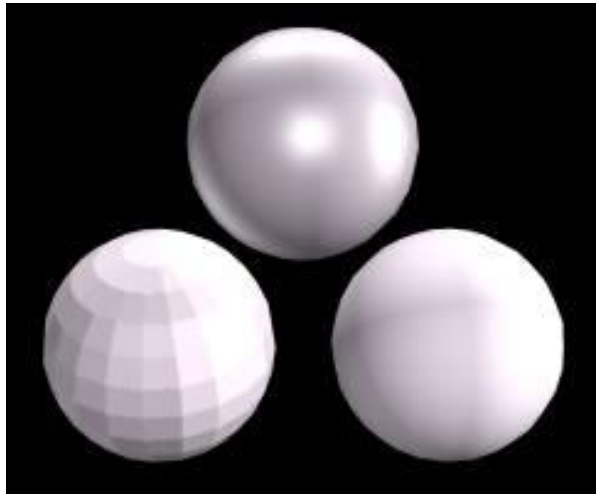
$$I_b = \frac{1}{y_1 - y_4} [I_1(y_s - y_4) + I_4(y_1 - y_s)]$$

$$I_s = \frac{1}{x_b - x_a} [I_a(x_b - x_s) + I_b(x_s - x_a)]$$



Phong Shading

- Find vertex normals
- Interpolate vertex normals across edges
- Find shades along edges
- Interpolate edge shades across polygons





Problems with Interpolated Shading

- Polygonal silhouette
- Perspective distortion
- Orientation dependence
- Problems at shared vertices
- Unrepresentative vertex normals