Introduction to Computer Graphics
4. Shading

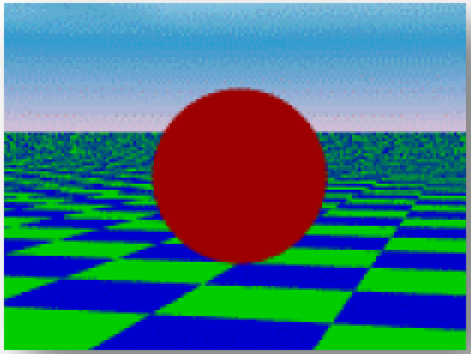
I-Chen Lin
National Chiao Tung University

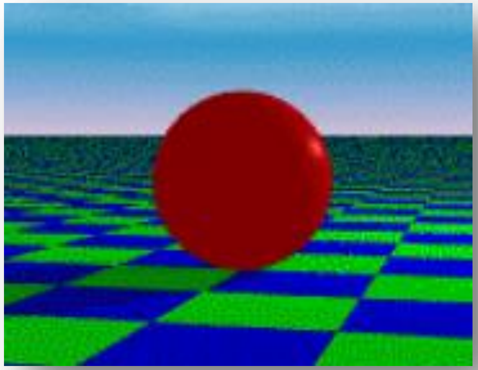
Textbook: E.Angel, D. Shreiner Interactive Computer Graphics, 6th Ed., Pearson
Ref: D.D. Hearn, M. P. Baker, W. Carithers, Computer Graphics with OpenGL, 4th Ed., Pearson
J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, R. L. Phillips. Introduction to Computer Graphics, Addison-Wesley

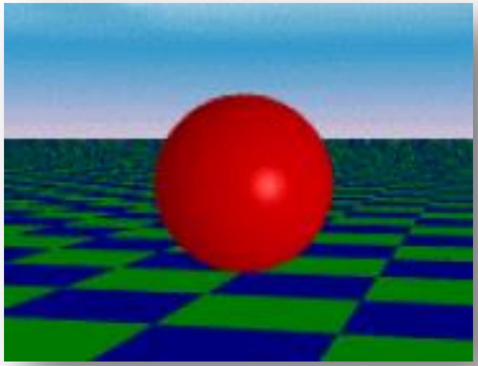
Illumination and Shading

- 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 ?

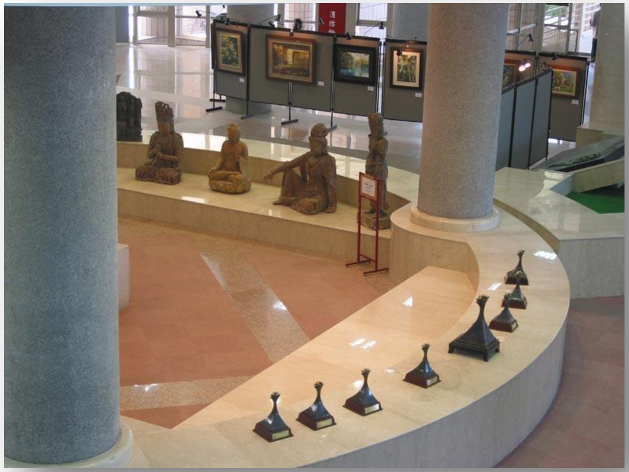
- Light-material interactions cause each point to have a different color or shade

- Need to consider
  - Light sources
  - Material properties
  - Location of the viewer
  - Surface orientation

Illumination and Shading

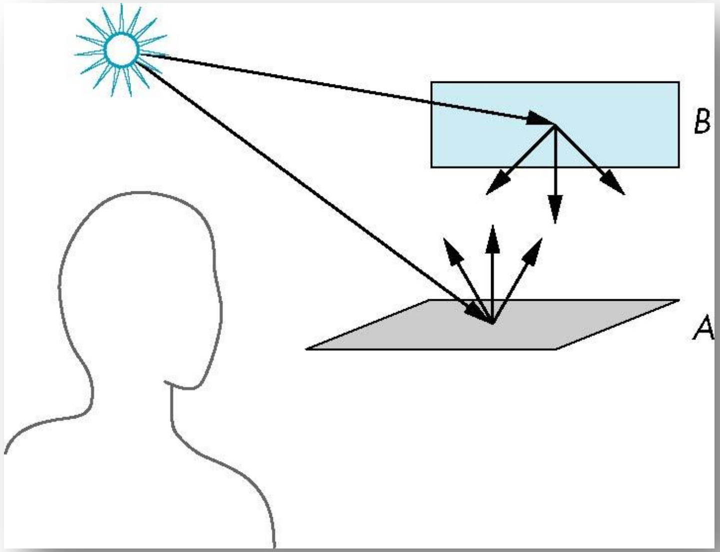
- Factors that affect the “color” of a pixel.
  - Light sources
    - Emittance spectrum (color)
    - Geometry (position and direction)
    - Directional attenuation

  - Objects’ surface properties
    - Reflectance spectrum (color)
    - Geometry (position, orientation, and micro-structure)
    - Absorption

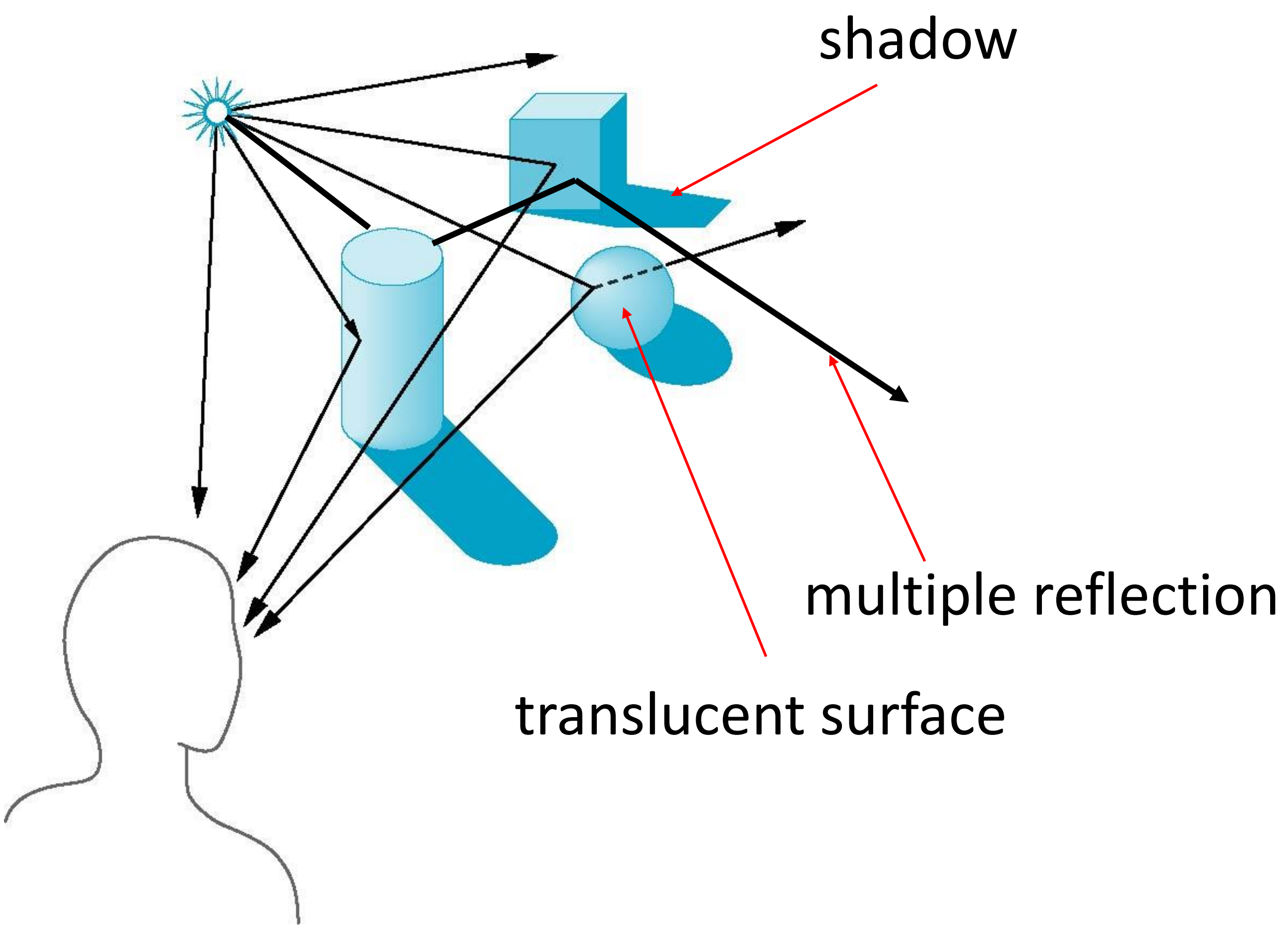


Scattering

- 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



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

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

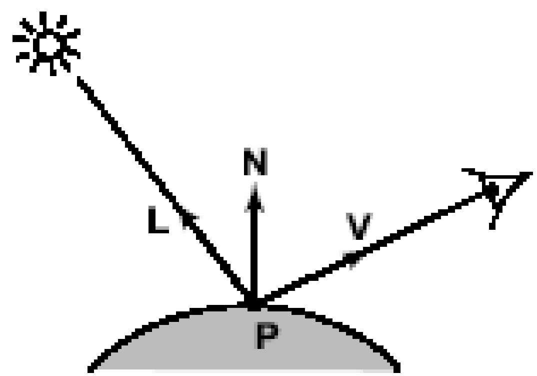
Local vs Global Rendering

- 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

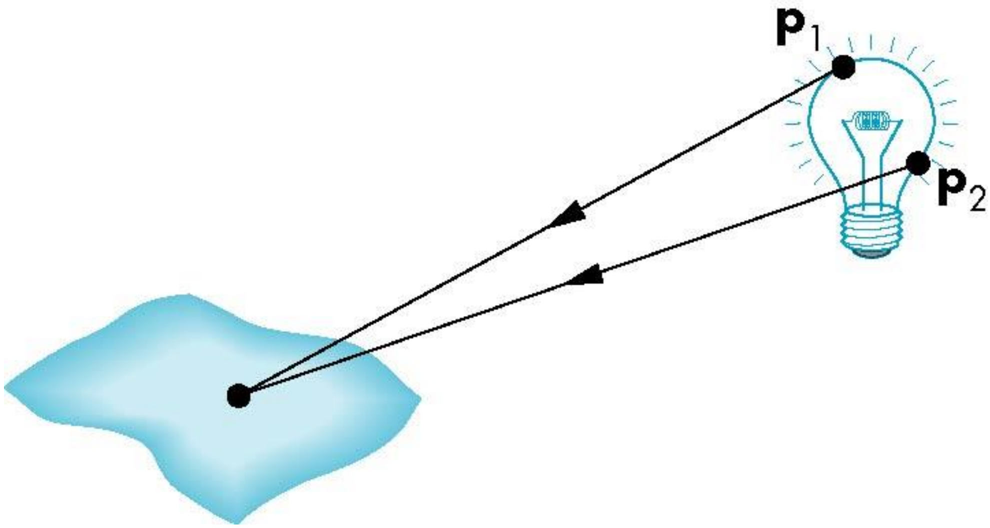
Local Illumination

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



Light Sources

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

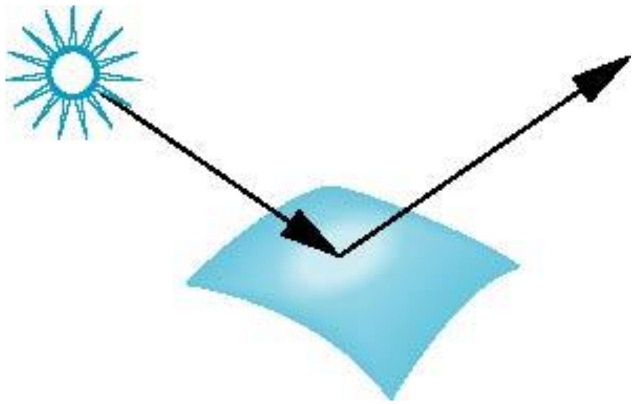


Simple Light Sources

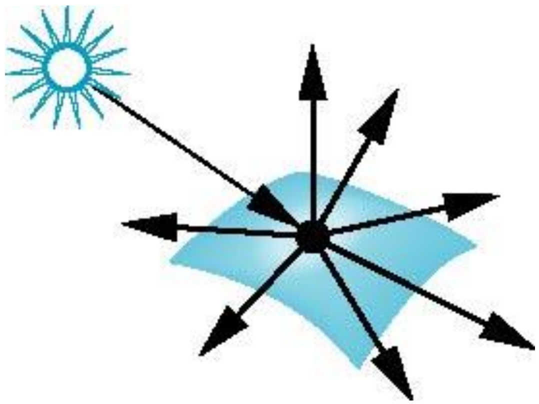
- Point source
  - Model with position and color
  - Distant source = infinite distance away (parallel)
- Spotlight
  - Restrict light from ideal point source
- Ambient light
  - Same amount of light everywhere in scene
  - Can model contribution of many sources and reflecting surfaces

Surface Types

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



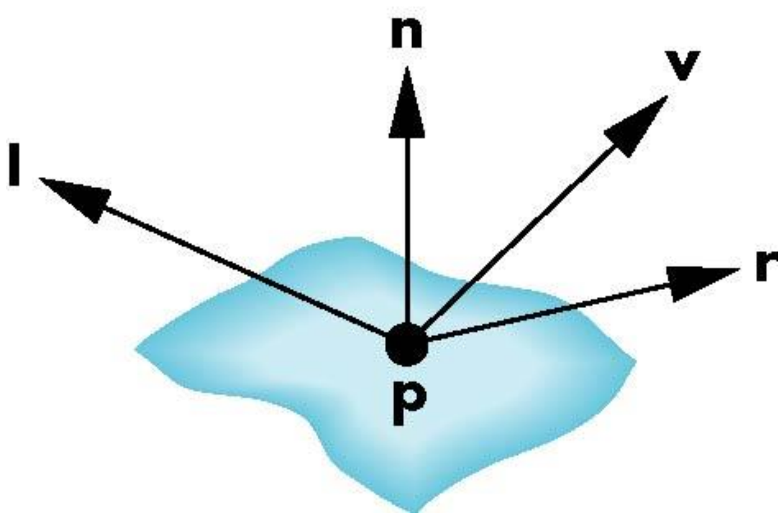
smooth surface



rough surface

Phong Reflection Model

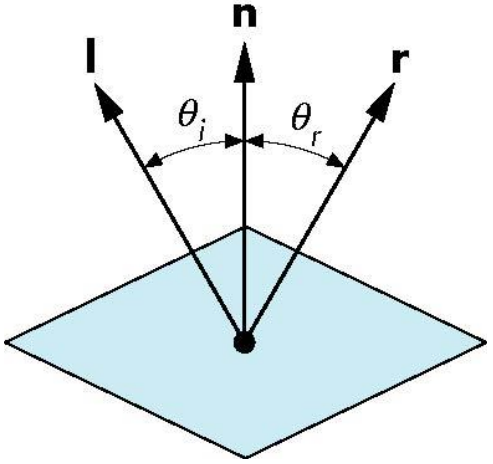
- A simple model that can be computed rapidly
- Has three components
  - Ambient
  - Diffuse
  - Specular
- Uses four vectors
  - To source ***l***
  - To viewer ***v***
  - Normal ***n***
  - Perfect reflector ***r***



Ideal Reflector

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

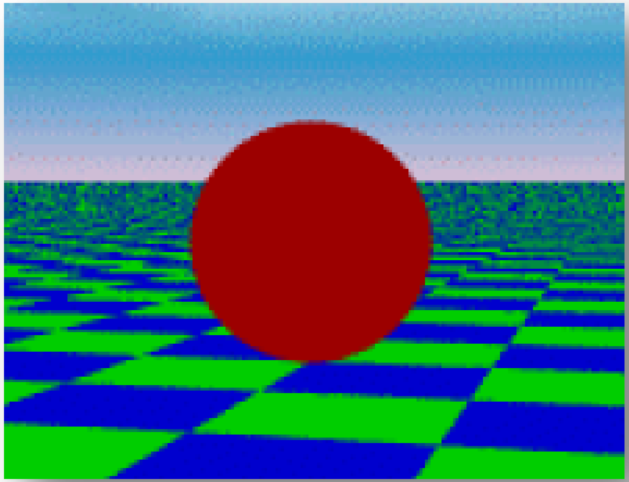
$$\mathbf{r} = 2\,(\mathbf{l} \cdot \mathbf{n})\,\mathbf{n} - \mathbf{l}$$



Ambient Light

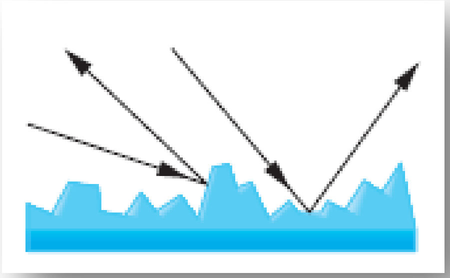
- The result of multiple interactions between (large) light sources and the objects in the environment.

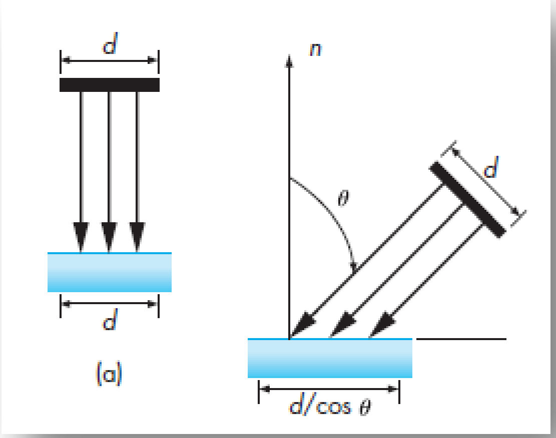
- $I_{ambient} = K_a \cdot I_a$

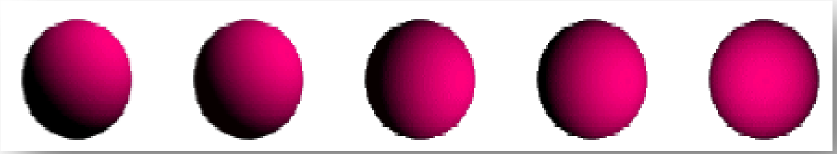


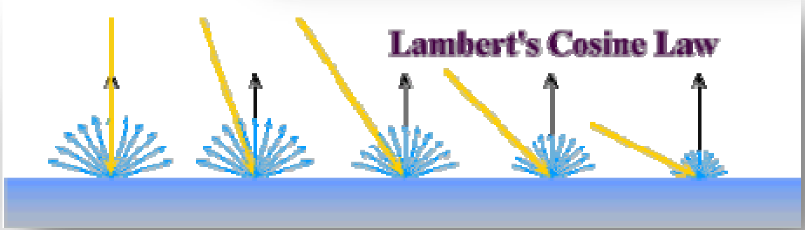
Diffuse Reflection

- Light scattered equally in all directions
- Reflected intensities vary with the direction of the light.
- Lambertian Surface
  - Perfect diffuse reflector
  - reflected light ~$\cos\theta_i$
  - $\cos\theta_i = \mathbf{l} \cdot \mathbf{n}$ if vectors normalized
- $I_{diffuse} = K_d \cdot I_d\,(n \cdot l)$



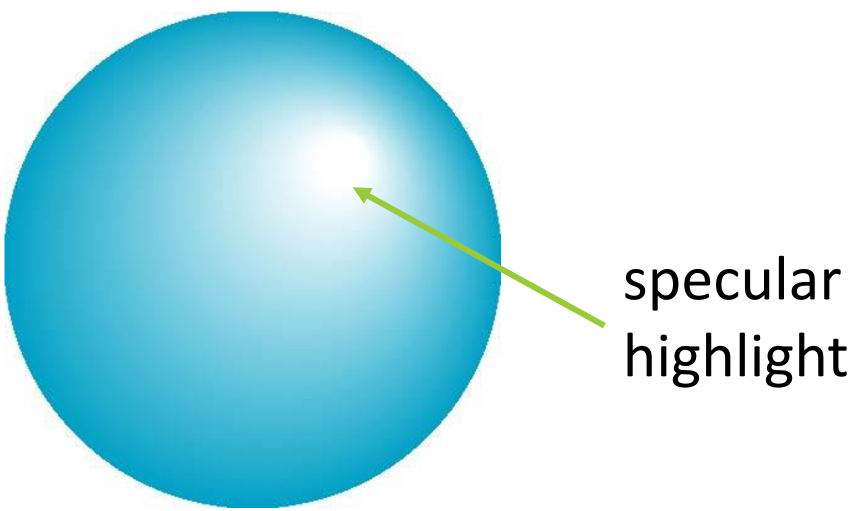






Specular Surfaces

- 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

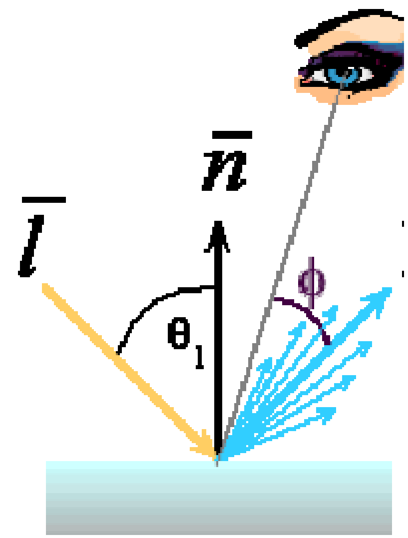
$$\boldsymbol{I}_r \sim k_s\, \boldsymbol{I} \cos^a \boldsymbol{\phi}$$

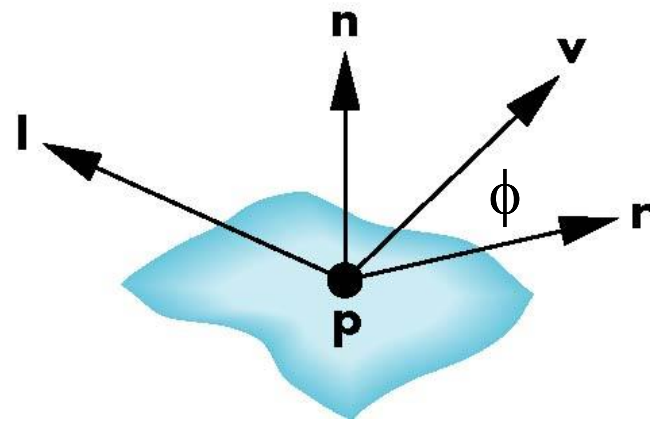
reflected intensity

absorption coef

incoming intensity

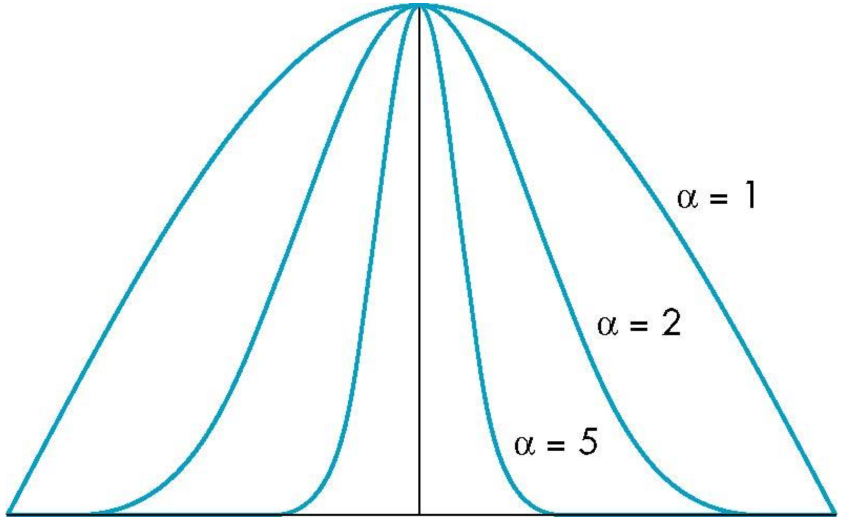
shininess coef





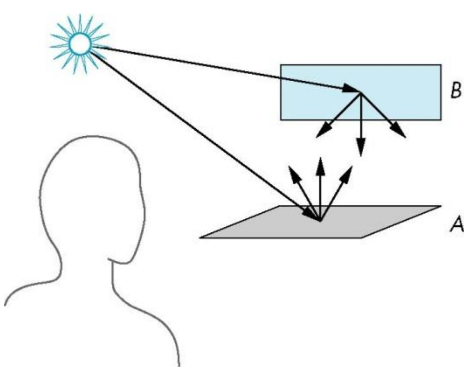
The Shininess Coefficient

- Values of a 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 $1/(a + bd + cd^2)$ to the diffuse and specular terms
- The constant and linear terms soften the effect of the point source



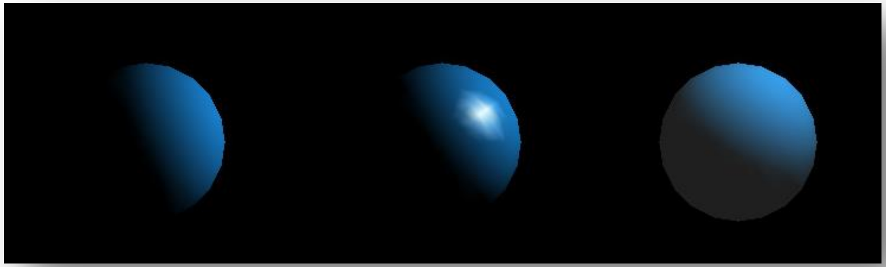
Coefficients

- 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/reflection coefficients
    - $k_{dr}, k_{dg}, k_{db}, k_{sr}, k_{sg}, k_{sb}, k_{ar}, k_{ag}, k_{ab}$
  - Shininess coefficient α

Adding up the Components

- A primitive virtual world with lighting can be shaded by combining the three light components .
- $I = I_{ambient} + I_{diffuse} + I_{specular}$

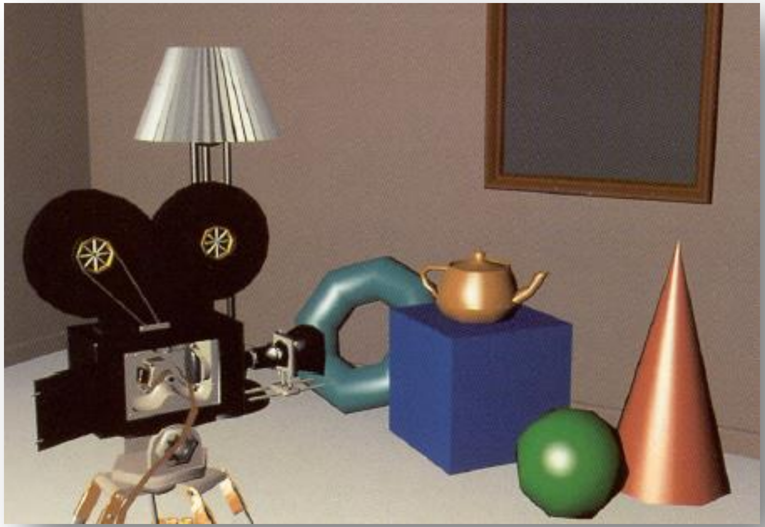
  $= k_a\, I_a + k_d\, I_d(\mathbf{l} \cdot \mathbf{n}) + k_s\, I_s\, (\mathbf{v} \cdot \mathbf{r})^{\alpha}$



diffuse

diffuse + specular

diffuse + 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

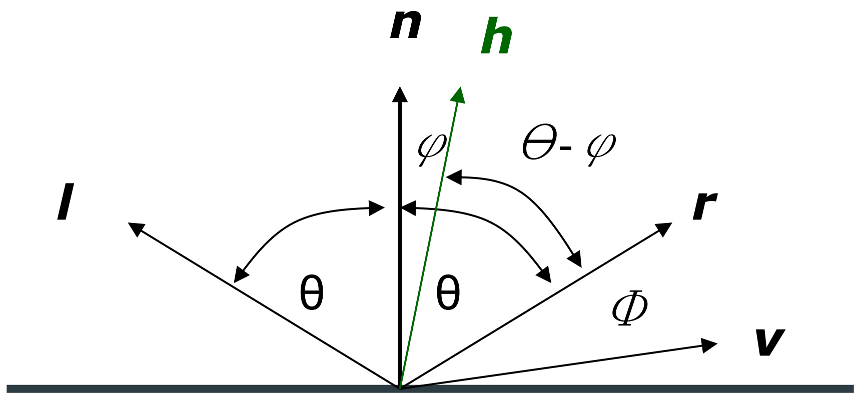
$$\boldsymbol{r} = 2\,(\boldsymbol{l} \cdot \boldsymbol{n})\,\boldsymbol{n} - \boldsymbol{l}$$

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

Using the Halfway Angle

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

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



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

$$\boxed{2\varphi = \phi}$$

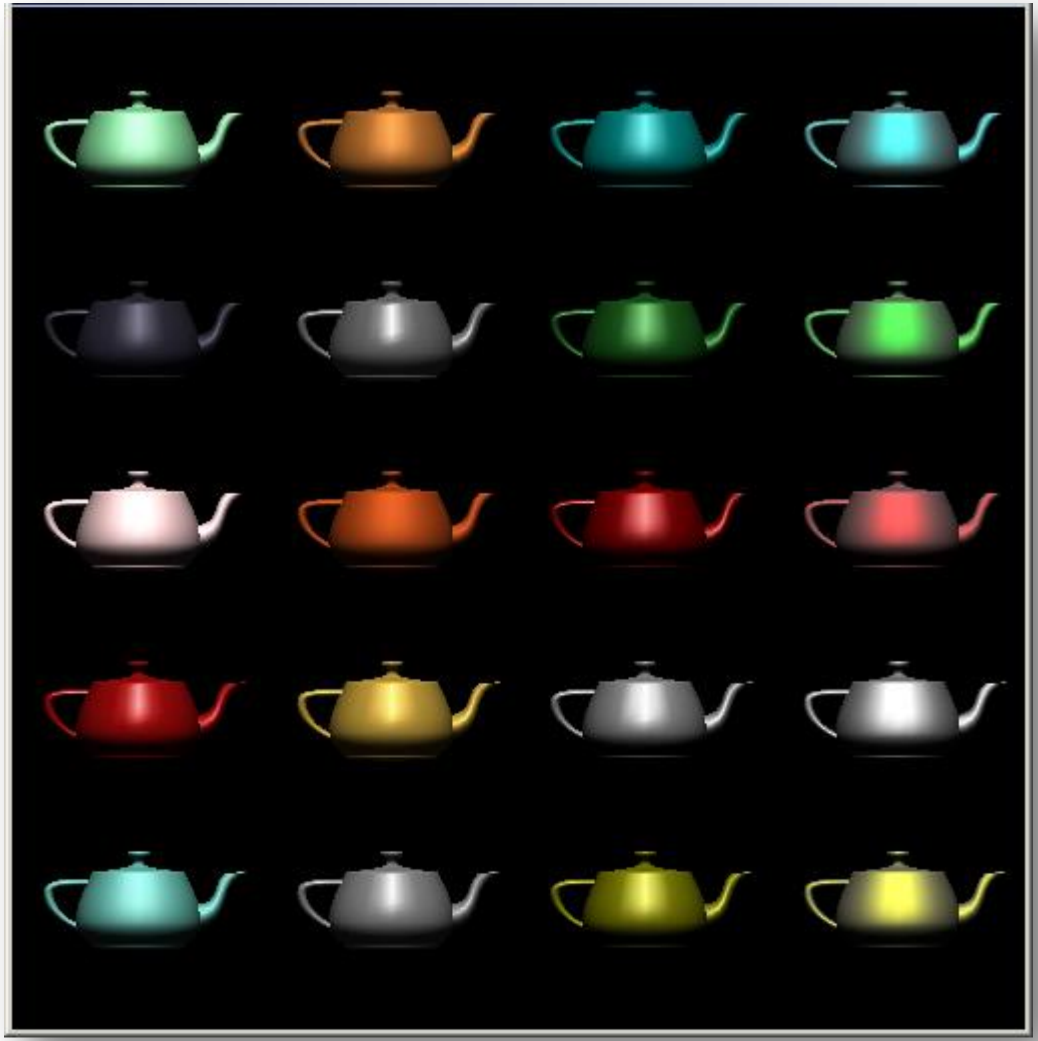
Using the Halfway Angle

- 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 in the modified Phong model.



Computation of Vectors

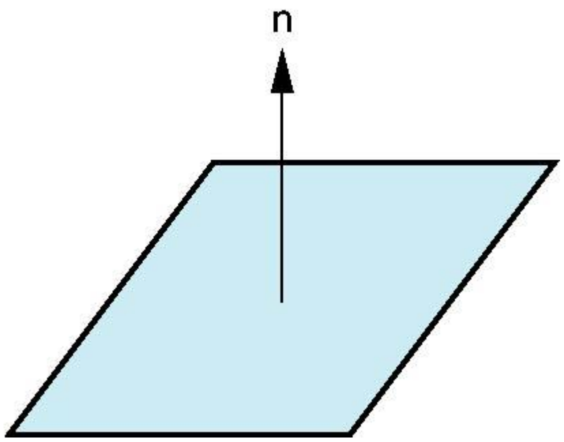
- **l** and **v** : specified by the application
- **r:** computed from **l** and **n**
- Determine **n**
  - Depending on underlying representation of surface
  - OpenGL leaves determination of normal to application
    - Exception for GLU quadrics and Bezier surfaces

Plane Normals

- Equation of plane: $ax+by+cz+d = 0$

- Normal can be obtained by

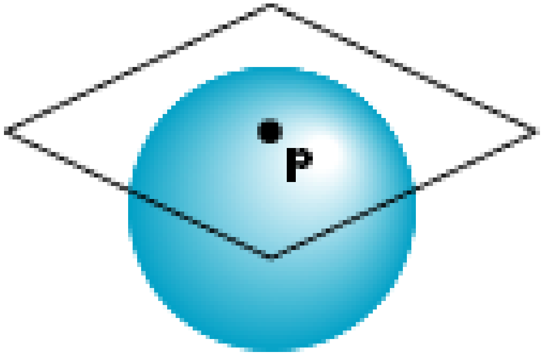
$$\mathbf{n} = (p_2 - p_0) \times (p_1 - p_0)$$



Normal to Sphere

- Implicit function $f(x,y.z)=0$

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



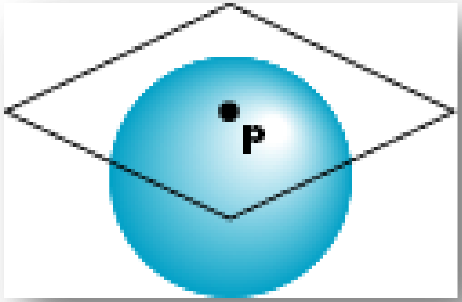
Parametric Form

- 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

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

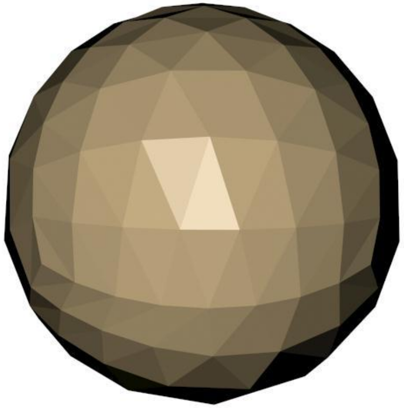
Polygonal Shading

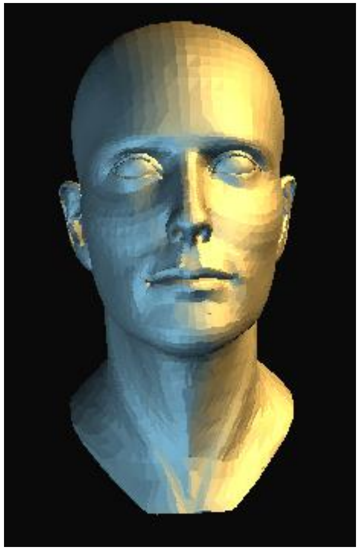
- Practical implementation to fill color within a polygon.
  - Flat shading
  - Gouraud shading (smooth shading)
  - Phong shading

Flat Shading

- Flat or constant shading.

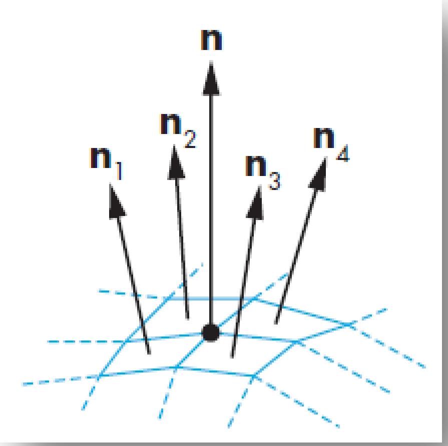
- Assume l, n, v are constant for a polygon.
  - Shading calculation: once for each polygon.



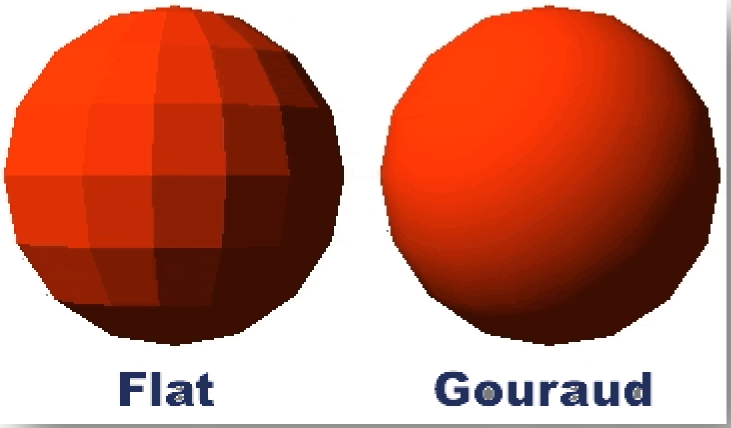


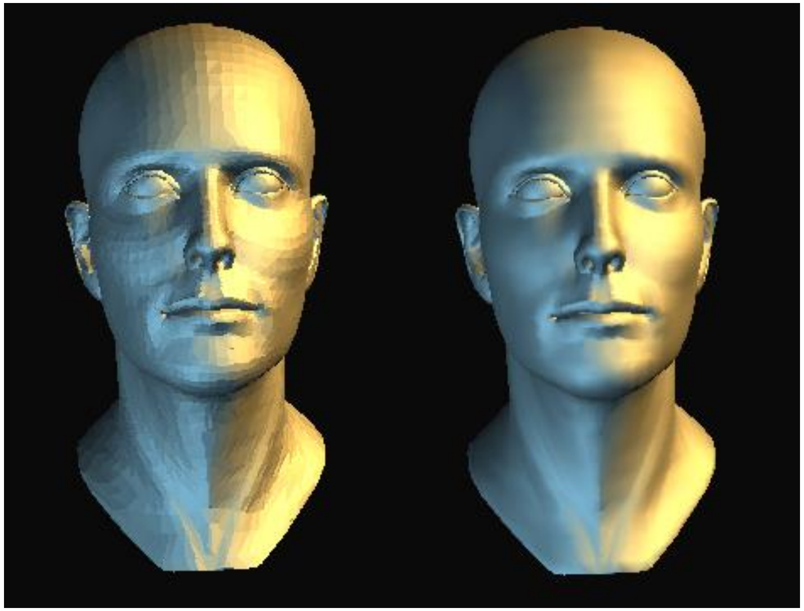
Gouraud Shading

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



$$\mathbf{n} = (\mathbf{n}_1+\mathbf{n}_2+\mathbf{n}_3+\mathbf{n}_4) / |\mathbf{n}_1+\mathbf{n}_2+\mathbf{n}_3+\mathbf{n}_4|$$



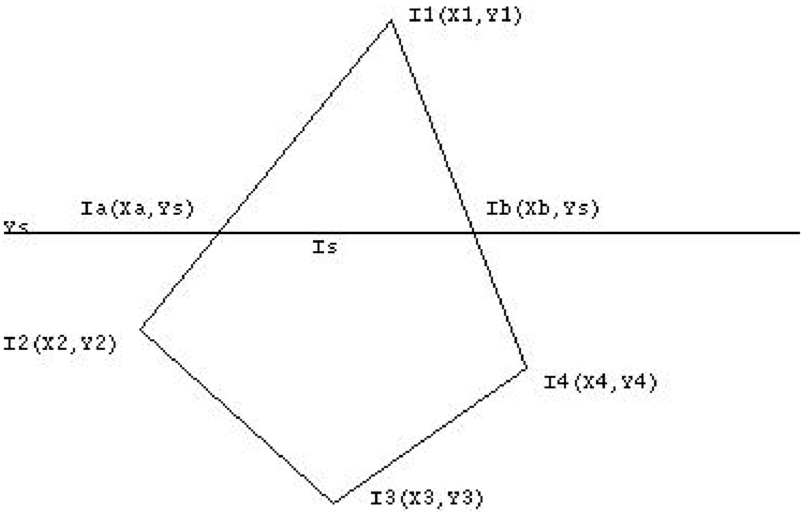


Gouraud Shading (cont.)

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

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

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



Phong Shading

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

Problems about Interpolated Shading on Polygonal Models

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