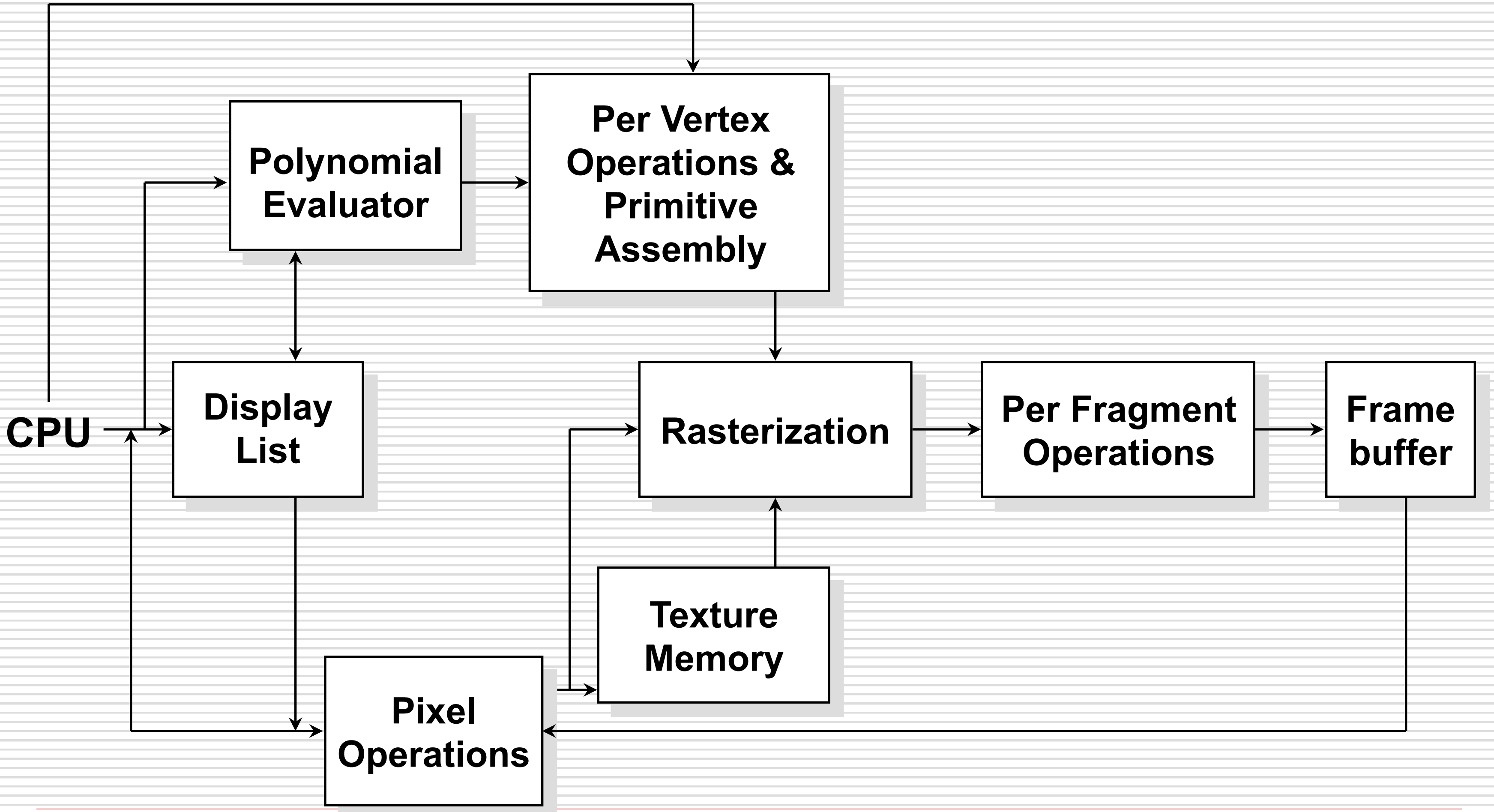
Game Programming

Robin Bing-Yu Chen
National Taiwan University

Game Lighting

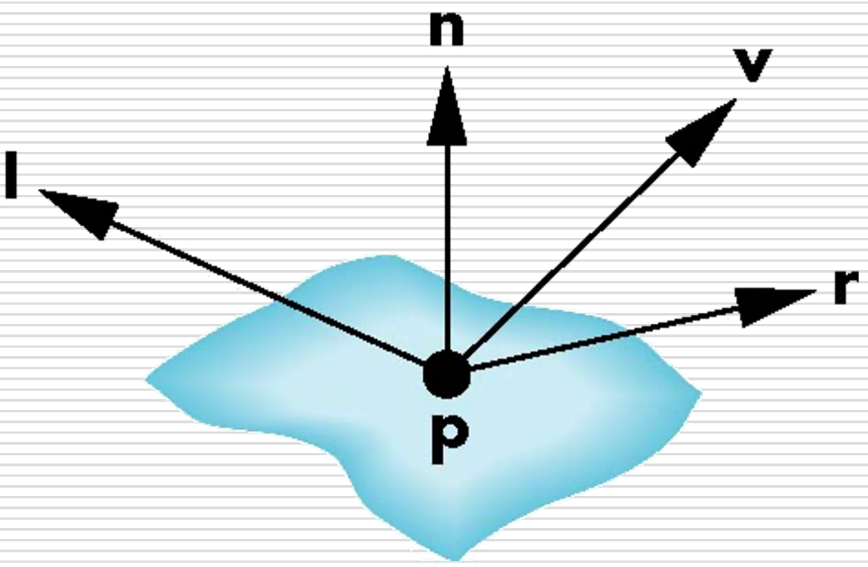
- Rendering Pipeline
- Illumination Model
- Shading Models

Rendering Pipeline



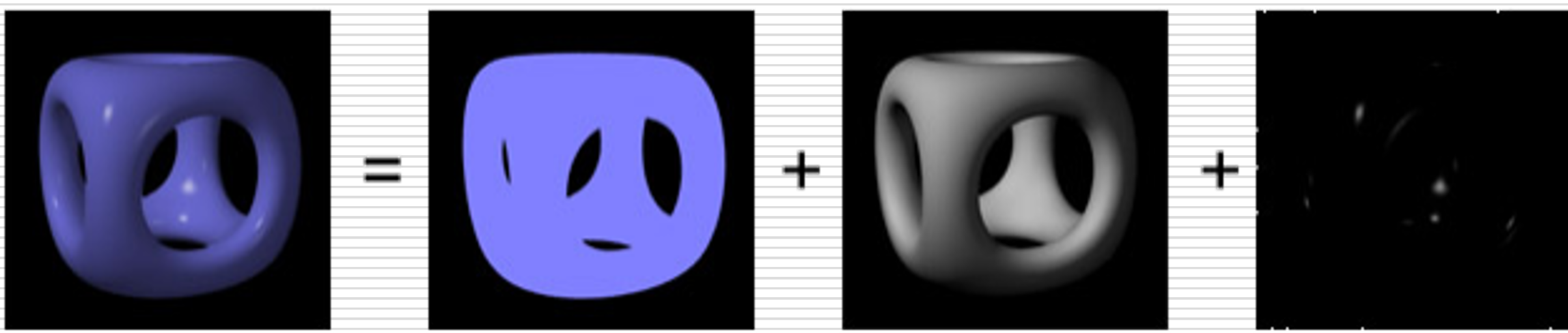
The Phong Illumination Model

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



Basics of Local Shading

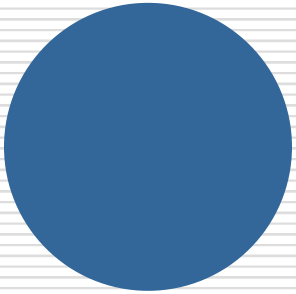
- Diffuse reflection
  - light goes everywhere; colored by object color
- Specular reflection
  - happens only near mirror configuration; usually white
- Ambient reflection
  - constant accounted for other source of illumination

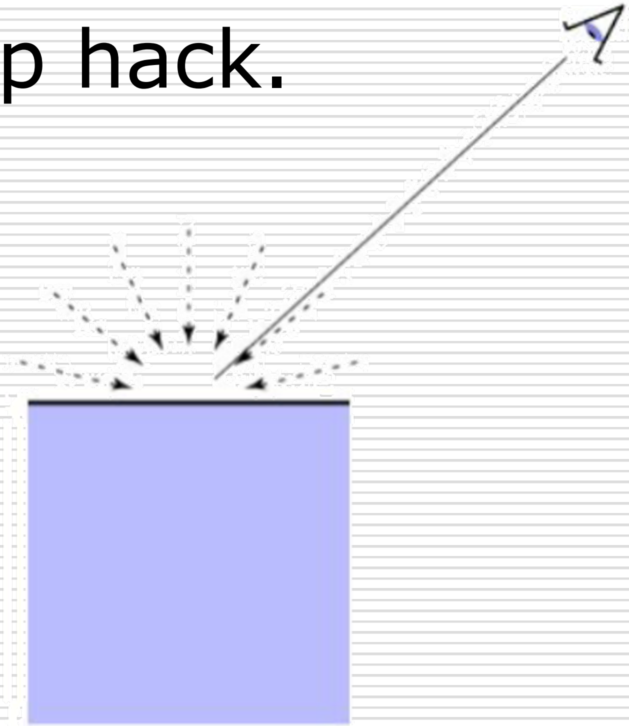


ambient diffuse specular

Ambient Shading

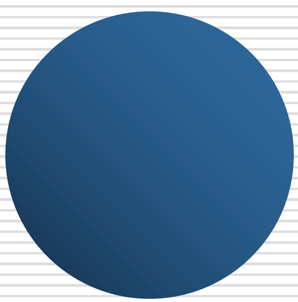
- add constant color to account for disregarded illumination and fill in black shadows; a cheap hack.

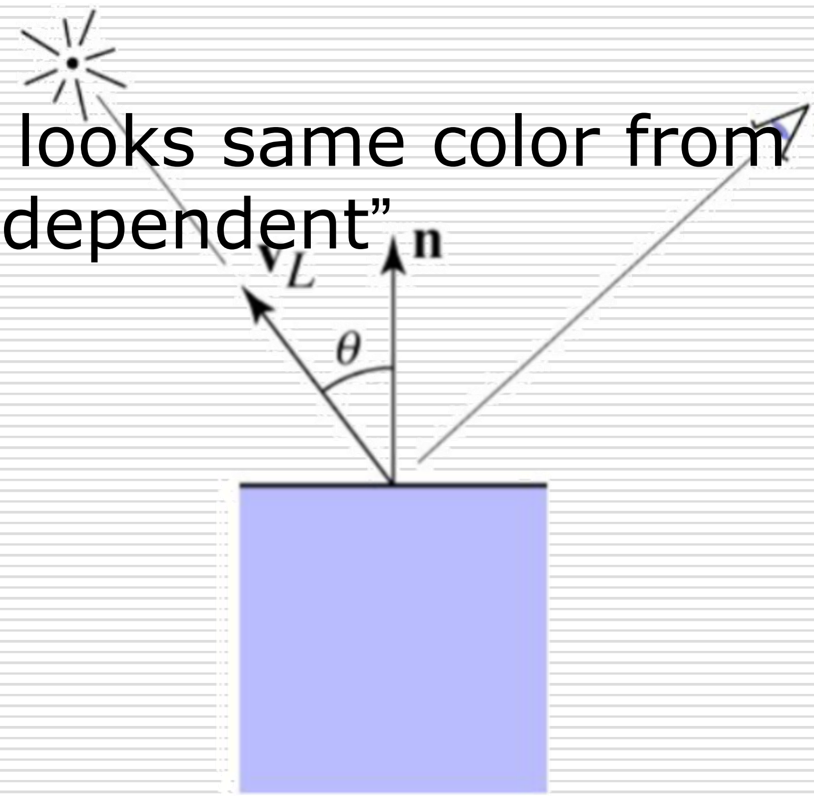




Diffuse Shading

- Assume light reflects equally in all directions
  - Therefore surface looks same color from all views; “view independent”



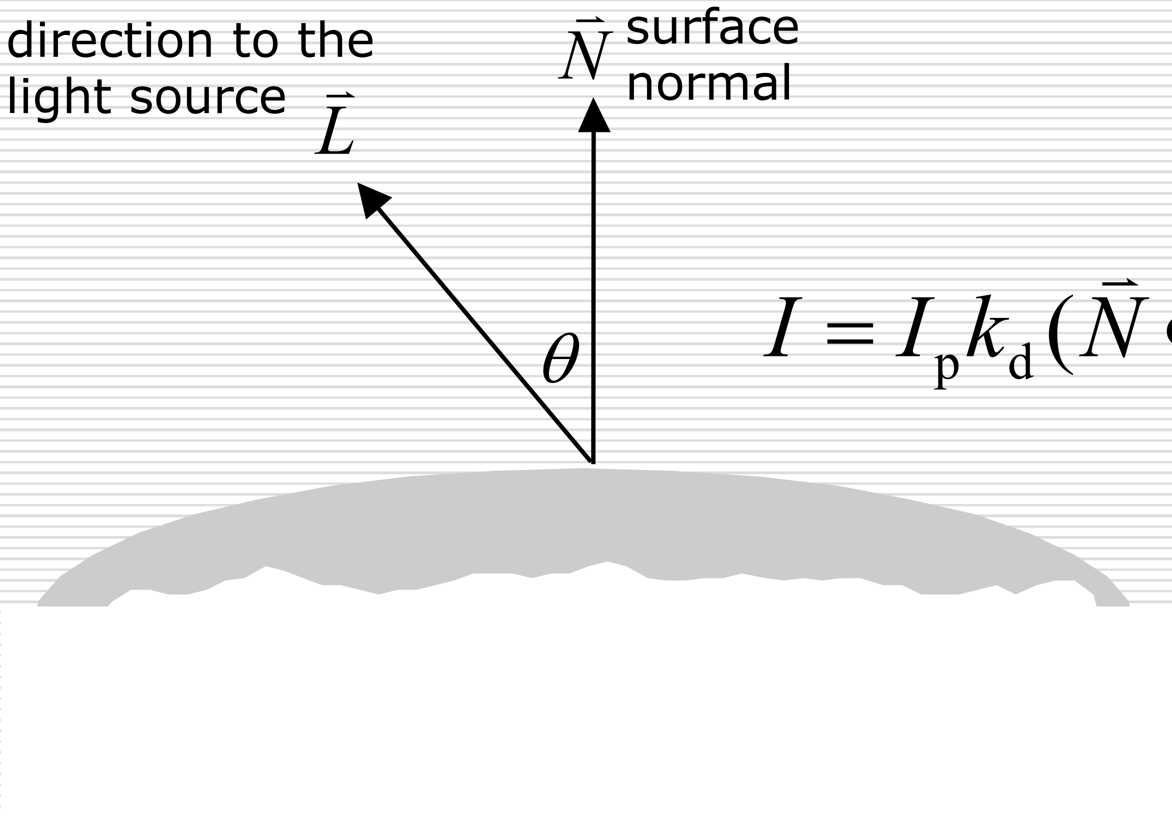


Illumination Models

- Ambient Light: $I = I_a k_a$
  - I_a : intensity of the ambient light
  - k_a : ambient-reflection coefficient: 0 ~ 1

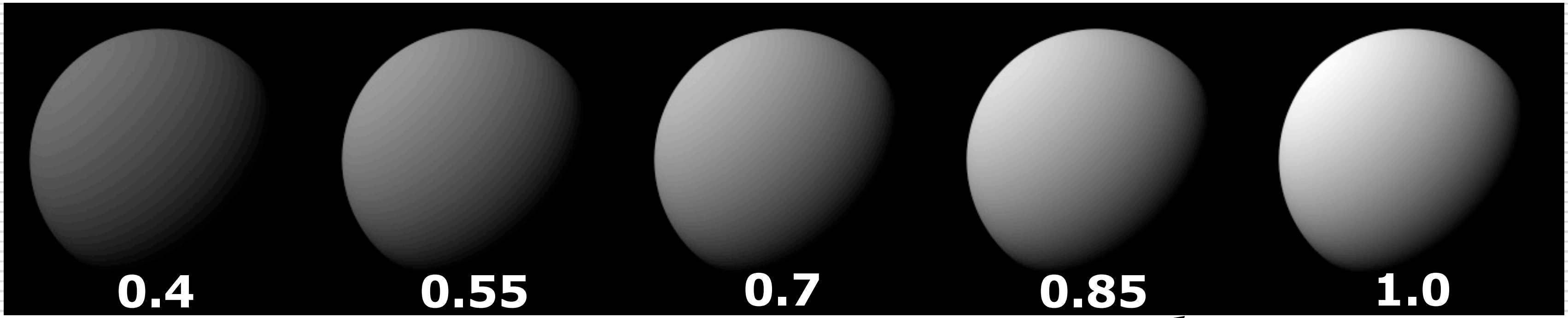
- Diffuse Reflection: $I = I_p k_d \cos\theta$
  - I_p : point light source's intensity
  - k_d : diffuse-reflection coefficient: 0 ~ 1
  - θ : angle: 0° ~ 90°

Diffuse Reflection

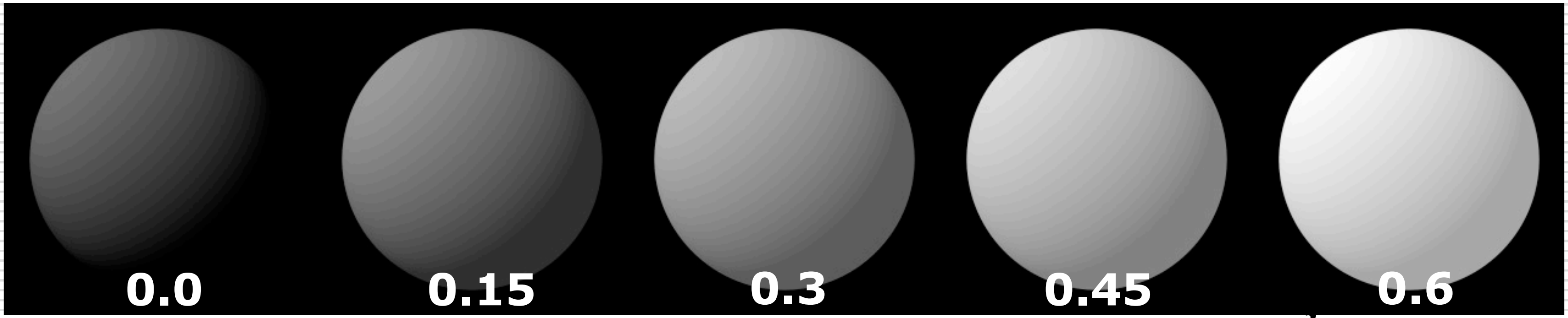


$$I = I_{\mathrm{p}} k_{\mathrm{d}} (\vec{N} \bullet \vec{L})$$

Examples



diffuse-reflection model with different k_{d}



ambient and diffuse-reflection model with different k_{a}
and $I_{\mathrm{a}} = I_{\mathrm{p}} = 1.0, k_{\mathrm{d}} = 0.4$

Light-Source Attenuation

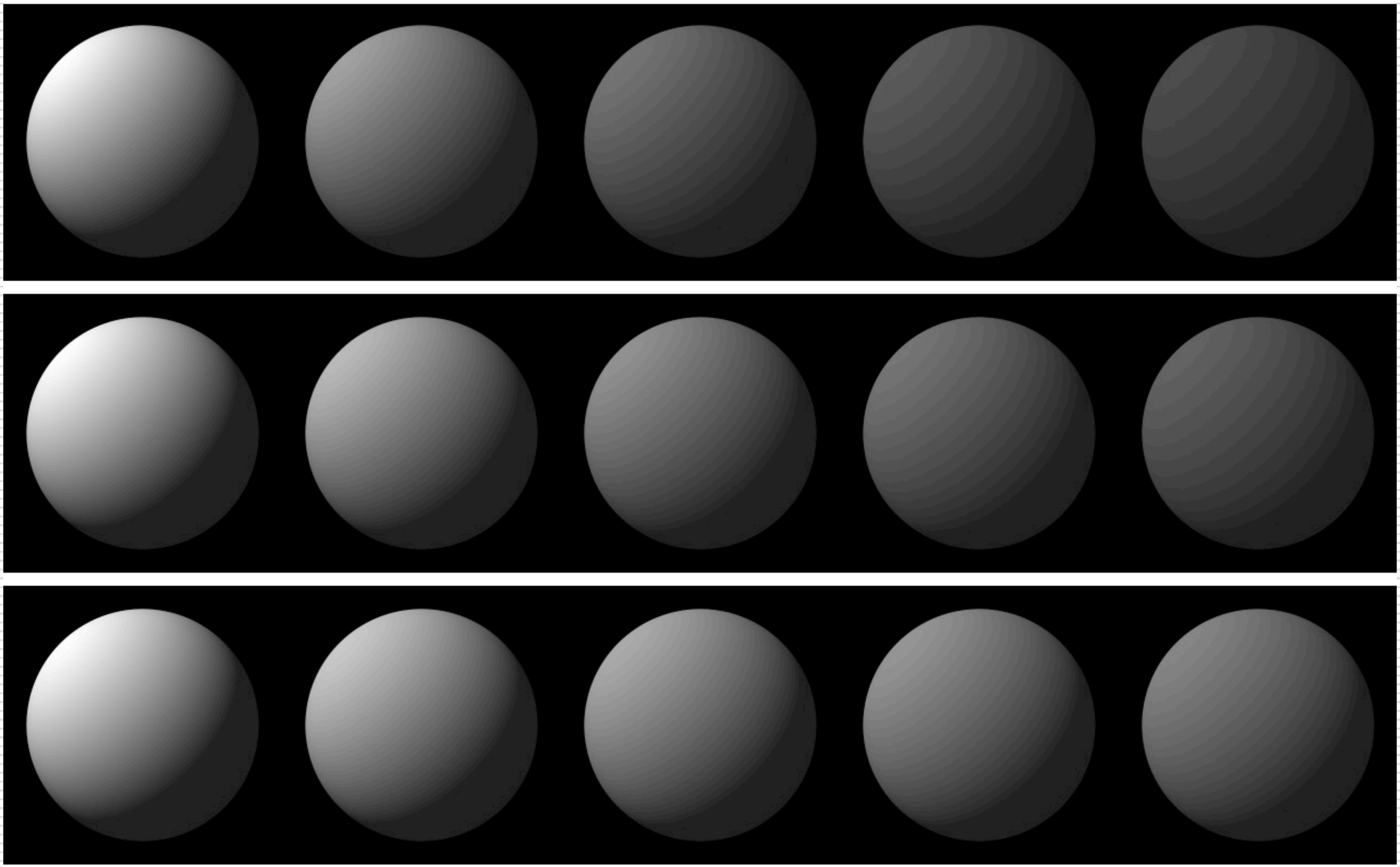
- $I = I_a k_a + f_{att} I_p k_d (\vec{N} \bullet \vec{L})$
  - f_{att} : light-source attenuation factor
  - if the light is a point source

$$f_{att} = \frac{1}{d_L^2}$$

  - where d_L is the distance the light travels from the point source to the surface

$$f_{att} = \min\left(\frac{1}{c_1 + c_2 d_L + c_3 d_L^2}, 1\right)$$

Examples



$$\frac{1}{d_L^2}$$

$$\frac{1}{d_L}$$

Colored Lights and Surfaces

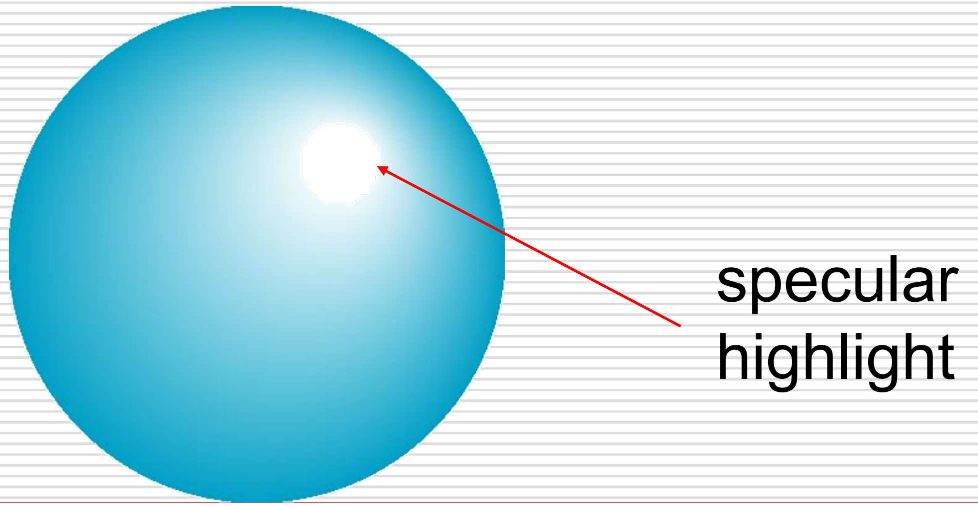
- If an object's **diffuse color** is $O_{\mathrm{d}} = (O_{\mathrm{dR}}, O_{\mathrm{dG}}, O_{\mathrm{dB}})$ then $I = (I_{\mathrm{R}}, I_{\mathrm{G}}, I_{\mathrm{B}})$ where for the red component $I_{\mathrm{R}} = I_{\mathrm{aR}} k_{\mathrm{a}} O_{\mathrm{dR}} + f_{\mathrm{att}} I_{\mathrm{pR}} k_{\mathrm{d}} O_{\mathrm{dR}} (\vec{N} \bullet \vec{L})$ however, it should be $I_{\lambda} = I_{\mathrm{a}\lambda} k_{\mathrm{a}} O_{\mathrm{d}\lambda} + f_{\mathrm{att}} I_{\mathrm{p}\lambda} k_{\mathrm{d}} O_{\mathrm{d}\lambda} (\vec{N} \bullet \vec{L})$ where λ is the **wavelength**

Specular Shading

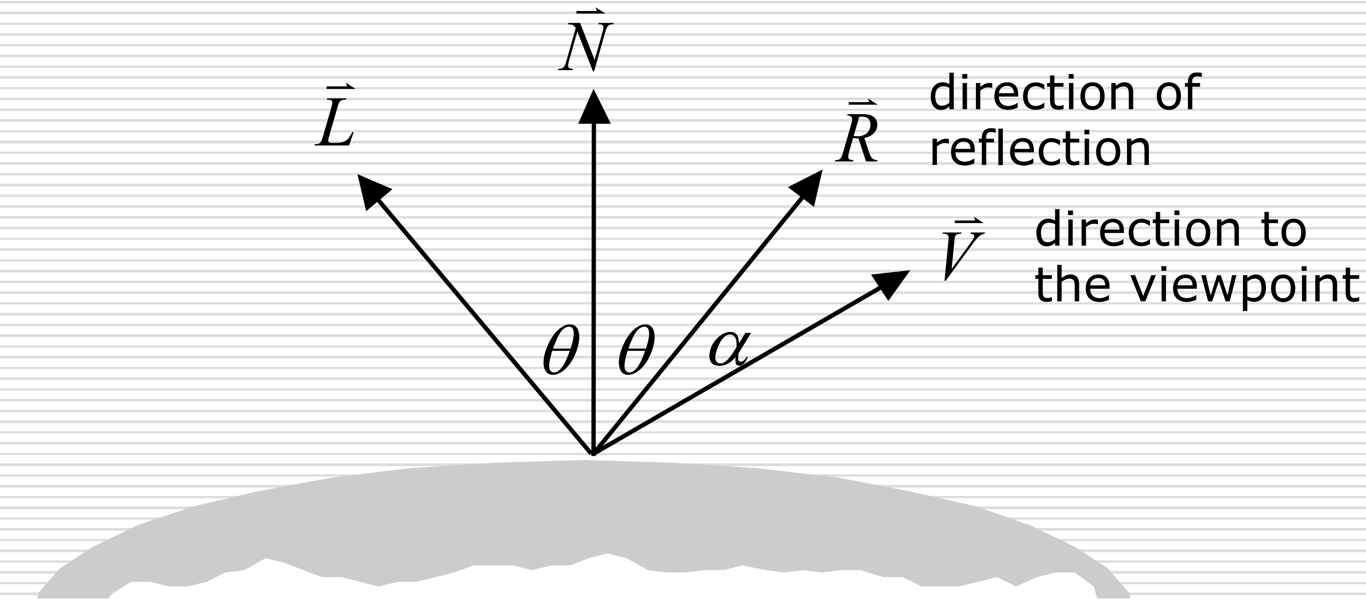
- Some surfaces have highlights, mirror like reflection; view direction dependent; especially for smooth shinny surfaces

Specular Surfaces

- Most surfaces are neither ideal diffusers nor perfectly specular (ideal refectors)
- Smooth surfaces show specular highlights due to incoming light being reflected in directions concentrated close to the direction of a perfect reflection



Specular Reflection



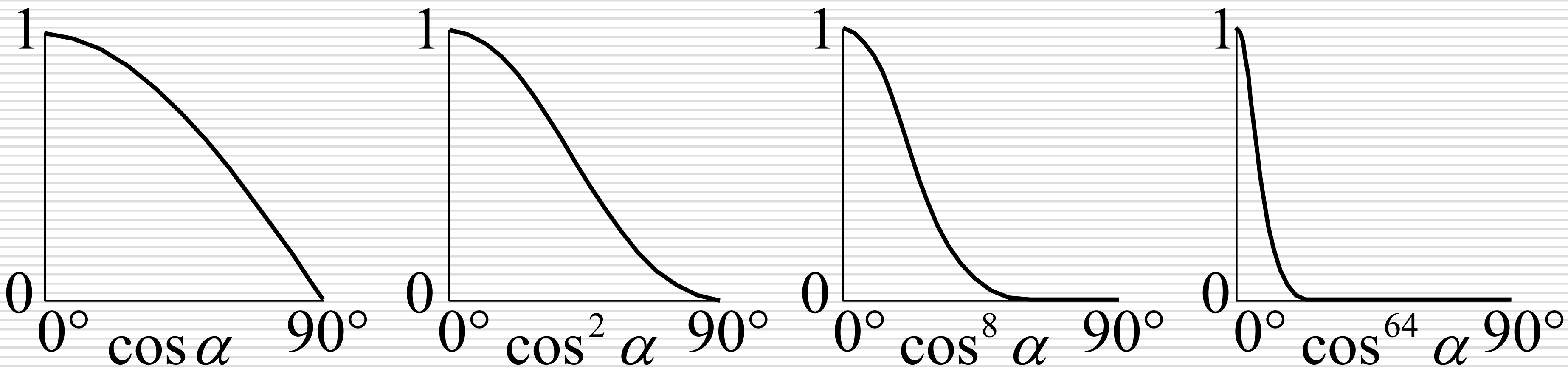
The Phong Illumination Model

- $I_\lambda = I_{a\lambda} k_a O_{d\lambda} + f_{att} I_{p\lambda} [k_d O_{d\lambda} \cos\theta + W(\theta) \cos^n \alpha]$
  - $W(\theta) = k_s$: specular-reflection coefficient:0~1
- so, the Eq. can be rewritten as
  $I_\lambda = I_{a\lambda} k_a O_{d\lambda} + f_{att} I_{p\lambda} [k_d O_{d\lambda} (\vec{N} \bullet \vec{L}) + k_s (\vec{R} \bullet \vec{V})^n]$
- consider the object's **specular color**

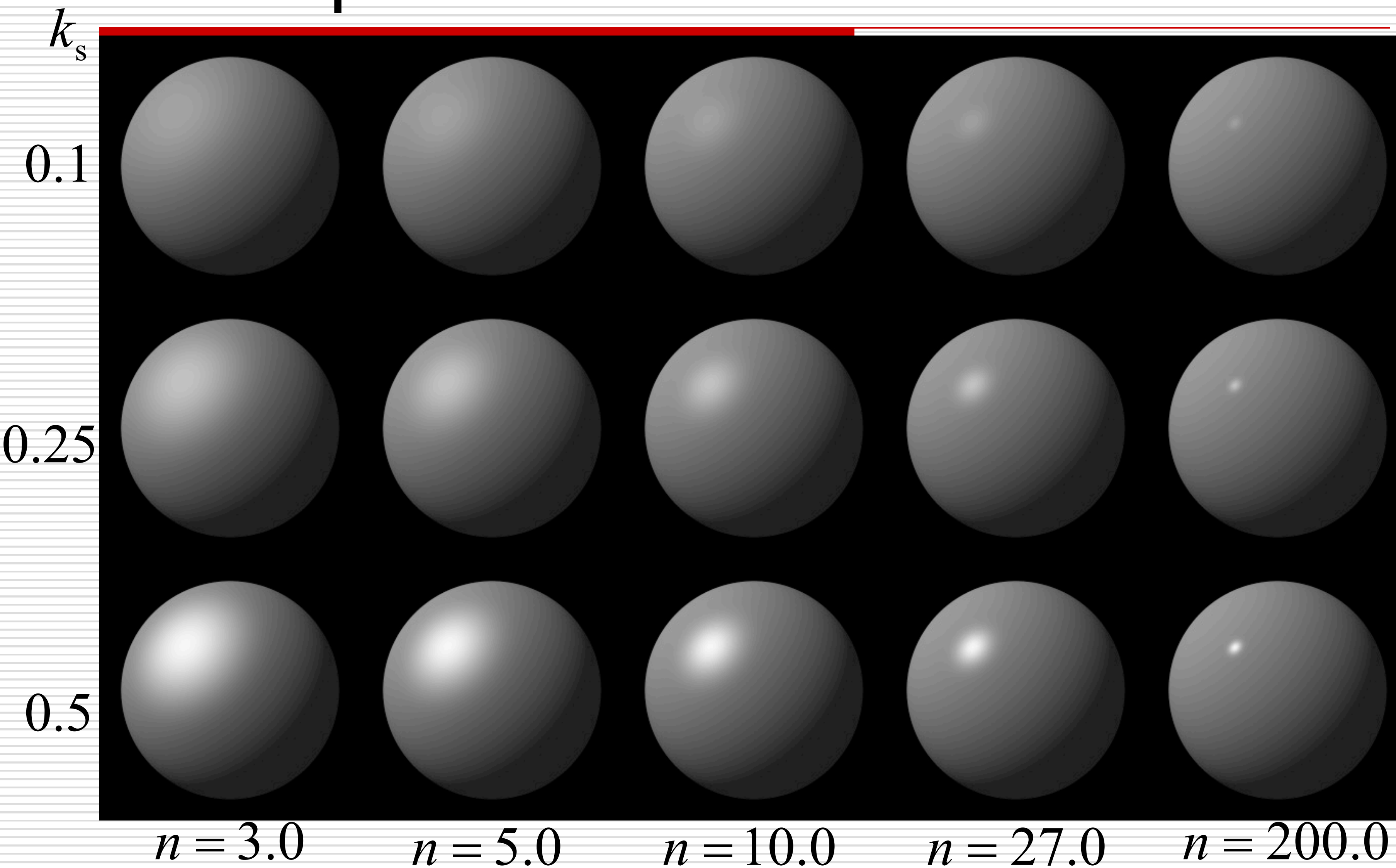
$$I_\lambda = I_{a\lambda} k_a O_{d\lambda} + f_{att} I_{p\lambda} [k_d O_{d\lambda} (\vec{N} \bullet \vec{L}) + k_s O_{s\lambda} (\vec{R} \bullet \vec{V})^n]$$

  - $O_{s\lambda}$: specular color

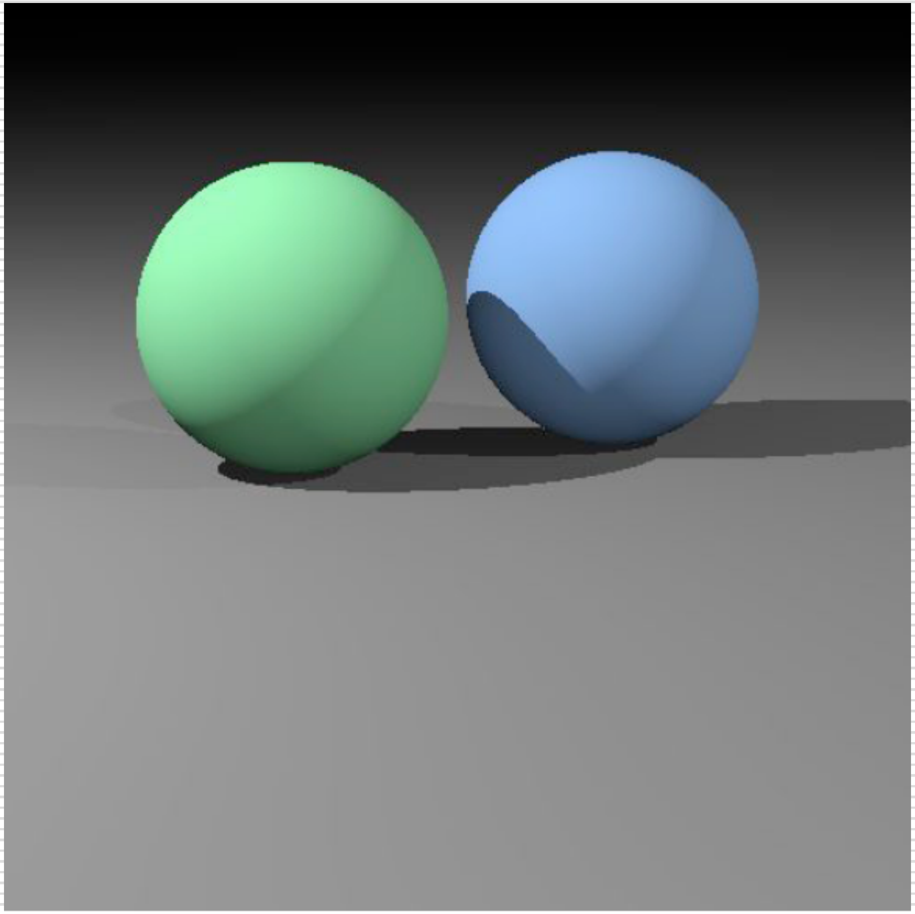
The Phong Illumination Model



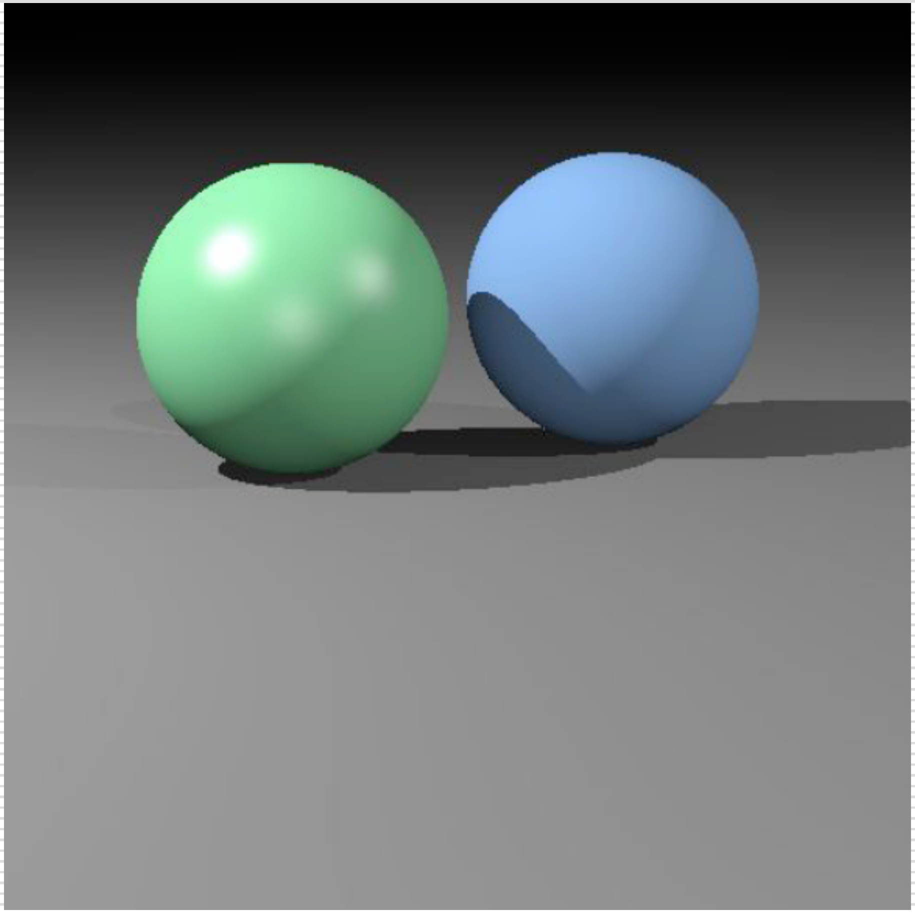
Examples



Specular Shading



diffuse



diffuse + specular

Multiple Light Sources

- If there are m light sources, then

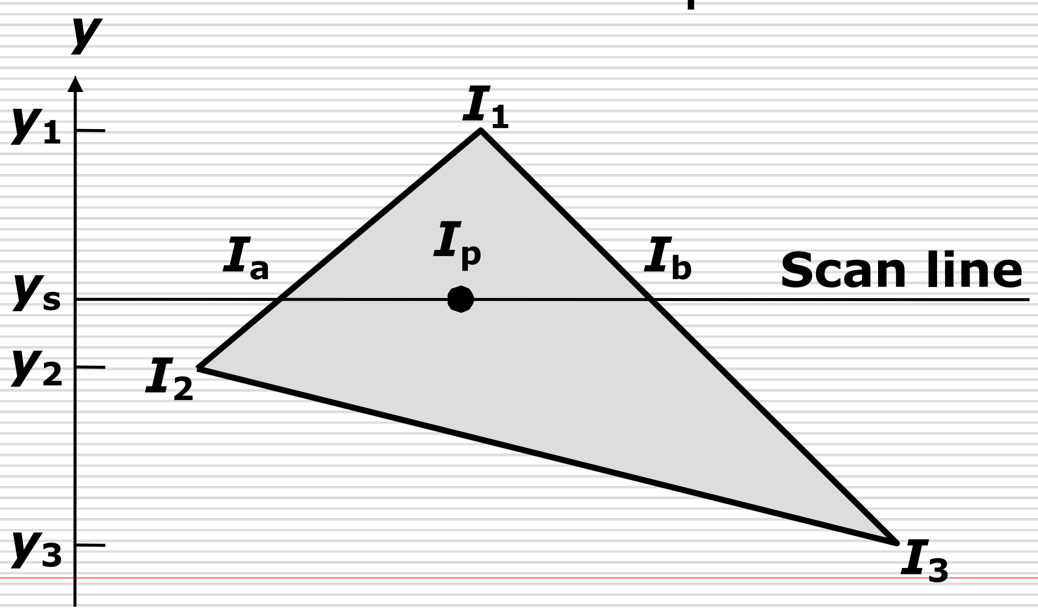
$$I_\lambda = I_{a\lambda} k_a O_{d\lambda} + \sum_{1 \le i \le m} f_{att_i} I_{p\lambda_i} [k_d O_{d\lambda} (\vec{N} \bullet \vec{L}_i) + k_s O_{s\lambda} \cos^n \alpha_i]$$

$$\approx I_{a\lambda} k_a O_{d\lambda} + \sum_{1 \le i \le m} f_{att_i} I_{p\lambda_i} [k_d O_{d\lambda} (\vec{N} \bullet \vec{L}_i) + k_s O_{s\lambda} (\vec{R}_i \bullet \vec{V})^n]$$

$$\approx I_{a\lambda} k_a O_{d\lambda} + \sum_{1 \le i \le m} f_{att_i} I_{p\lambda_i} [k_d O_{d\lambda} (\vec{N} \bullet \vec{L}_i) + k_s O_{s\lambda} (\vec{N} \bullet \vec{H}_i)^n]$$

Computing Lighting at Each Pixel

- Most accurate approach: Compute component illumination at each pixel with individual positions, light directions, and viewing directions
- But this could be expensive...



Shading Models for Polygons

- Flat Shading
  - Faceted Shading
  - Constant Shading
- Gouraud Shading
  - Intensity Interpolation Shading
  - Color Interpolation Shading
- Phong Shading
  - Normal-Vector Interpolation Shading

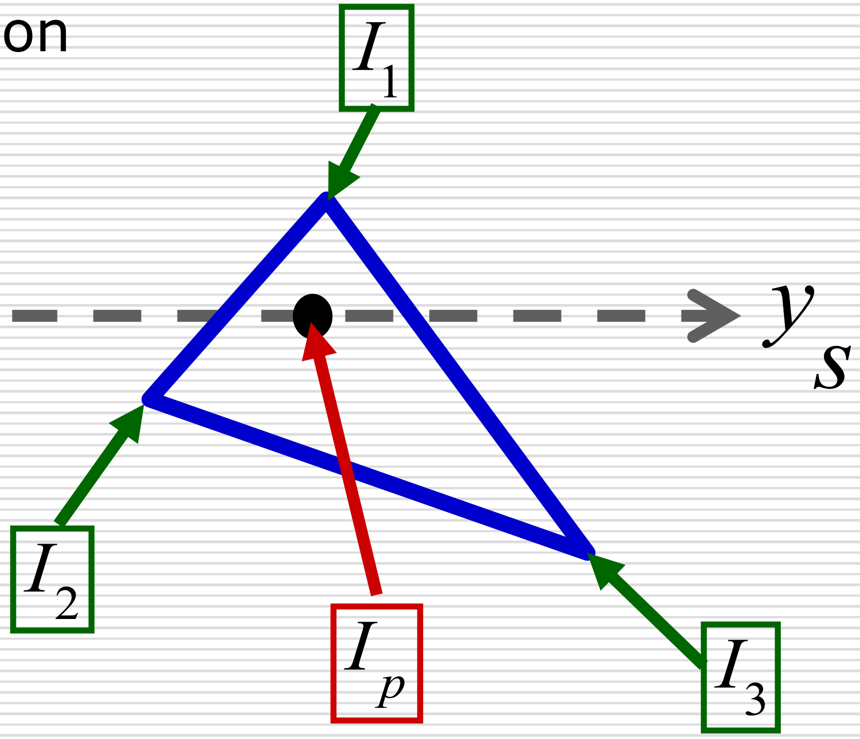
Flat Shading

- Assumptions
  - The light source is at infinity
  - The viewer is at infinity
  - The polygon represents the actual surface being modeled and is not an approximation to a curved surface

Flat Shading

- Compute constant shading function, over each polygon
- Same normal and light vector across whole polygon
- Constant shading for polygon

$$\boxed{I_p = I}$$

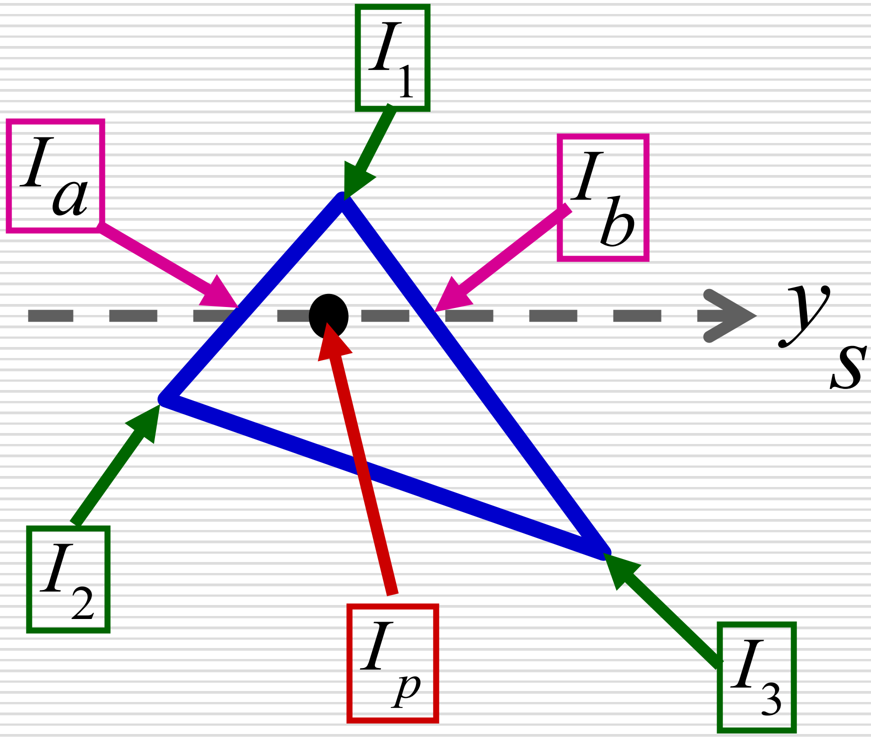


Intensity Interpolation (Gouraud)

$$I_a = I_1 \frac{y_s - y_2}{y_1 - y_2} + I_2 \frac{y_1 - y_s}{y_1 - y_2}$$

$$I_b = I_1 \frac{y_s - y_3}{y_1 - y_3} + I_3 \frac{y_1 - y_s}{y_1 - y_3}$$

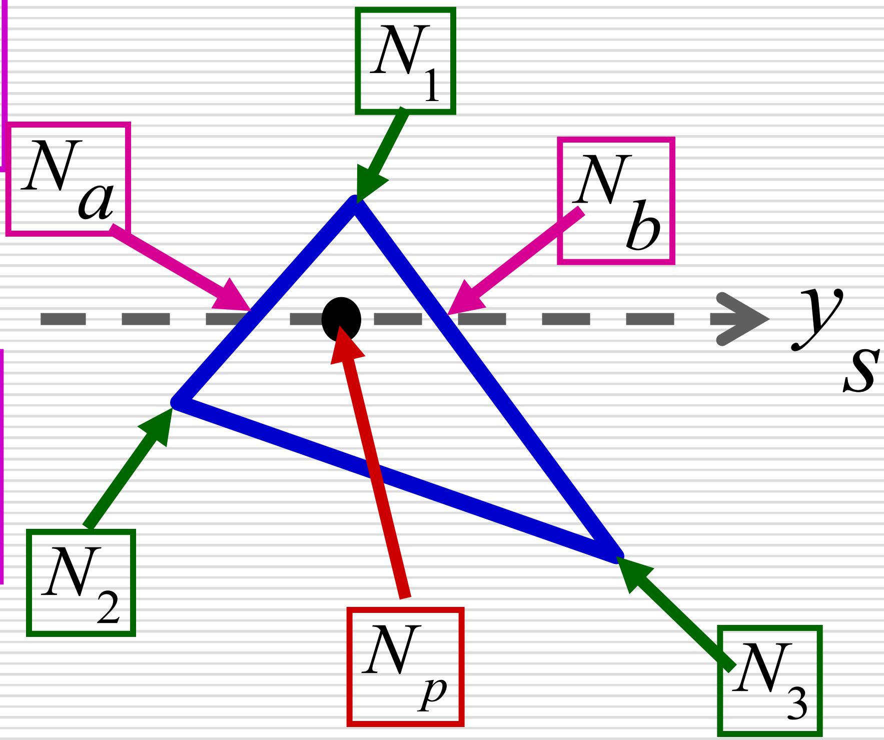
$$I_p = I_a \frac{x_b - x_p}{x_b - x_a} + I_b \frac{x_p - x_a}{x_b - x_a}$$



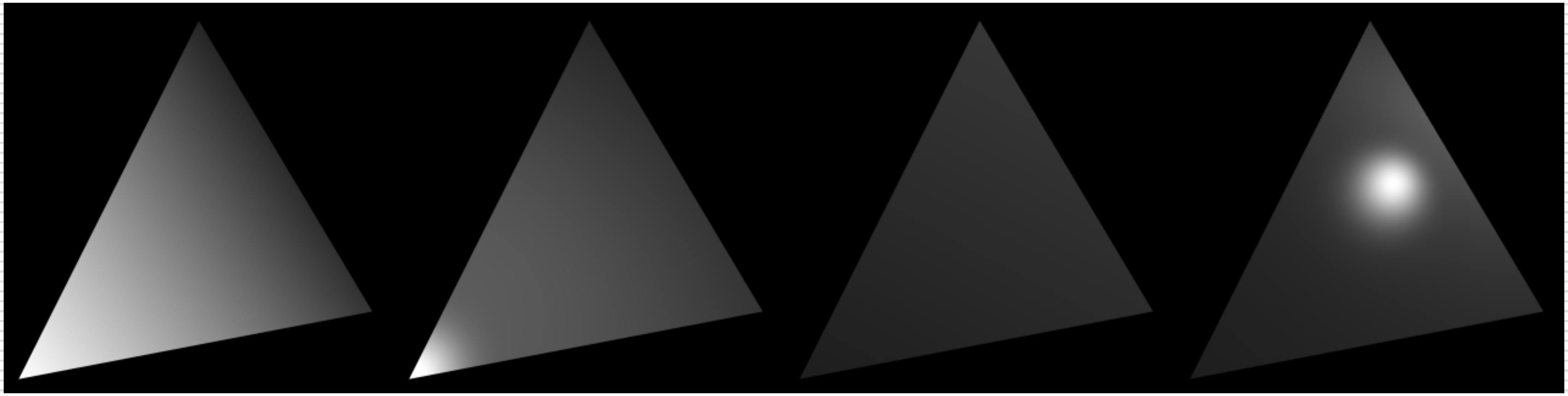
Normal Interpolation (Phong)

$$N_a = N_1 \frac{y_s - y_2}{y_1 - y_2} + N_2 \frac{y_1 - y_s}{y_1 - y_2}$$

$$N_b = N_1 \frac{y_s - y_3}{y_1 - y_3} + N_3 \frac{y_1 - y_s}{y_1 - y_3}$$



Gouraud v.s. Phong Shading



Gouraud **Phong** **Gouraud** **Phong**

Shadows

- $$I_\lambda = I_{a\lambda} k_a O_{d\lambda} + \sum_{1 \le i \le m} S_i f_{att_i} I_{p\lambda_i} [k_d O_{d\lambda} (\vec{N} \bullet \vec{L}_i) + k_s O_{s\lambda} (\vec{R}_i \bullet \vec{V})^n]$$

  - $$S_i = \begin{cases} 0, \text{ if light } i \text{ is blocked at this point} \\ 1, \text{ if light } i \text{ is not blocked at this point} \end{cases}$$