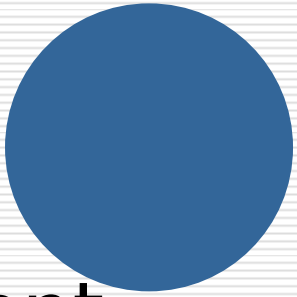
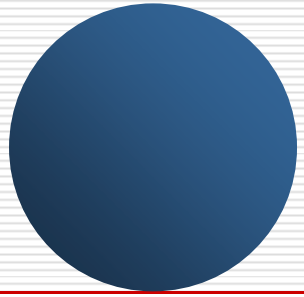
Why We Need Shading ?

- Suppose we build a model of a sphere using many polygons and color it with only one color. We get something like

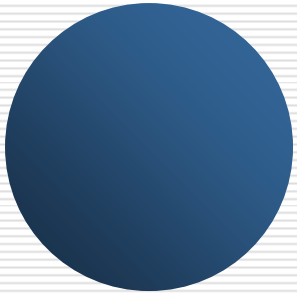


- But we want



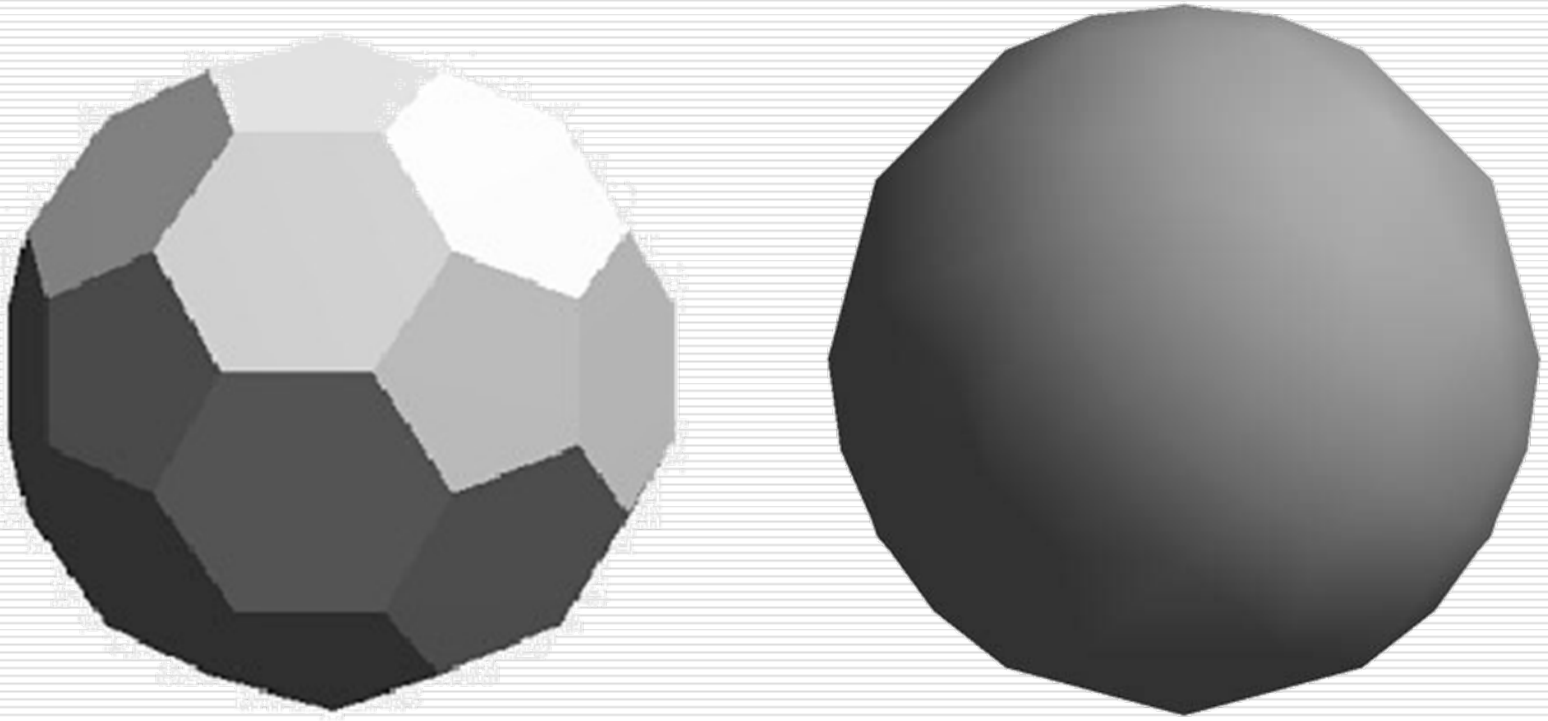
Shading

- 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 viewer
  - Surface orientation

What is Normal?

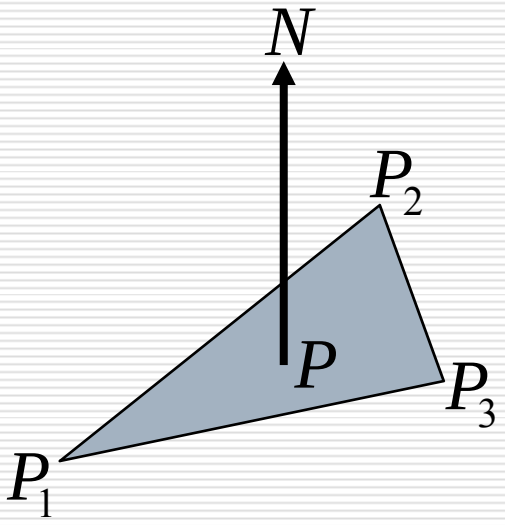


Recall: Normal for Triangle

- Plane $N \bullet (P - P_1) = 0$

$$N = P_1P_2 \times P_1P_3$$
$$= (P_3 - P_1) \times (P_2 - P_1)$$

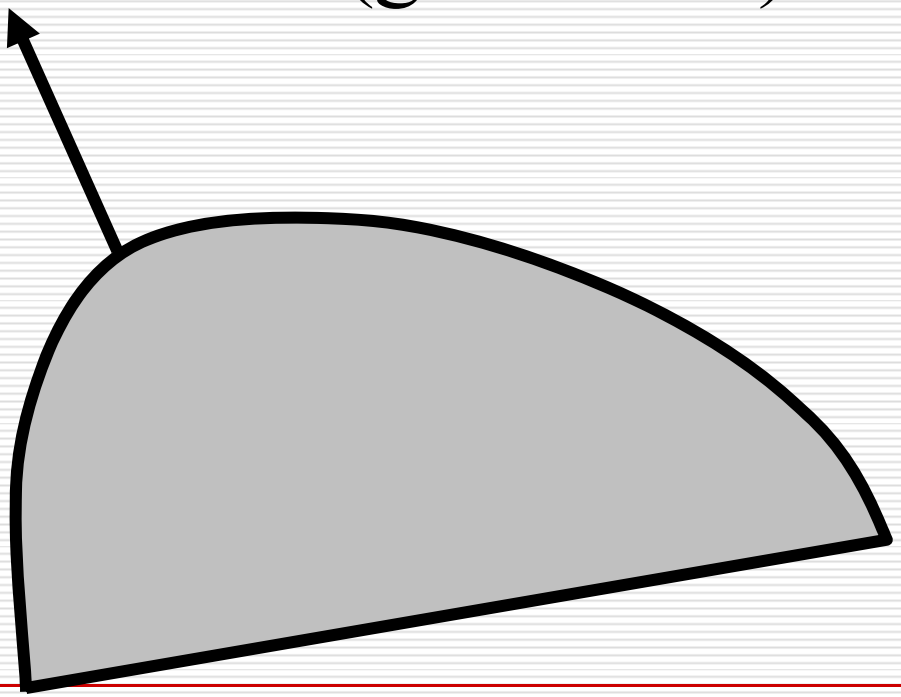
- Normalize $N \leftarrow N / |N|$



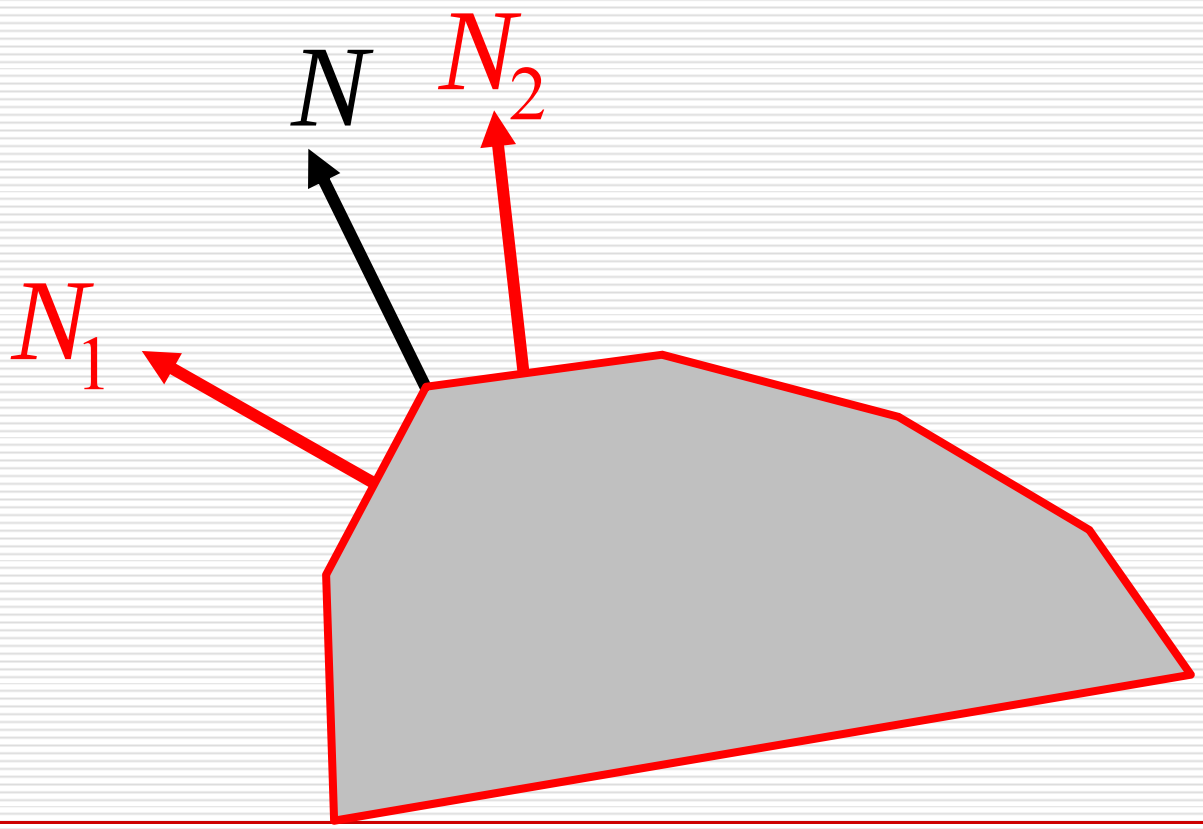
- Note that
  - right-hand rule determines outward face

Using Average Normals

N = true (geometric) normal

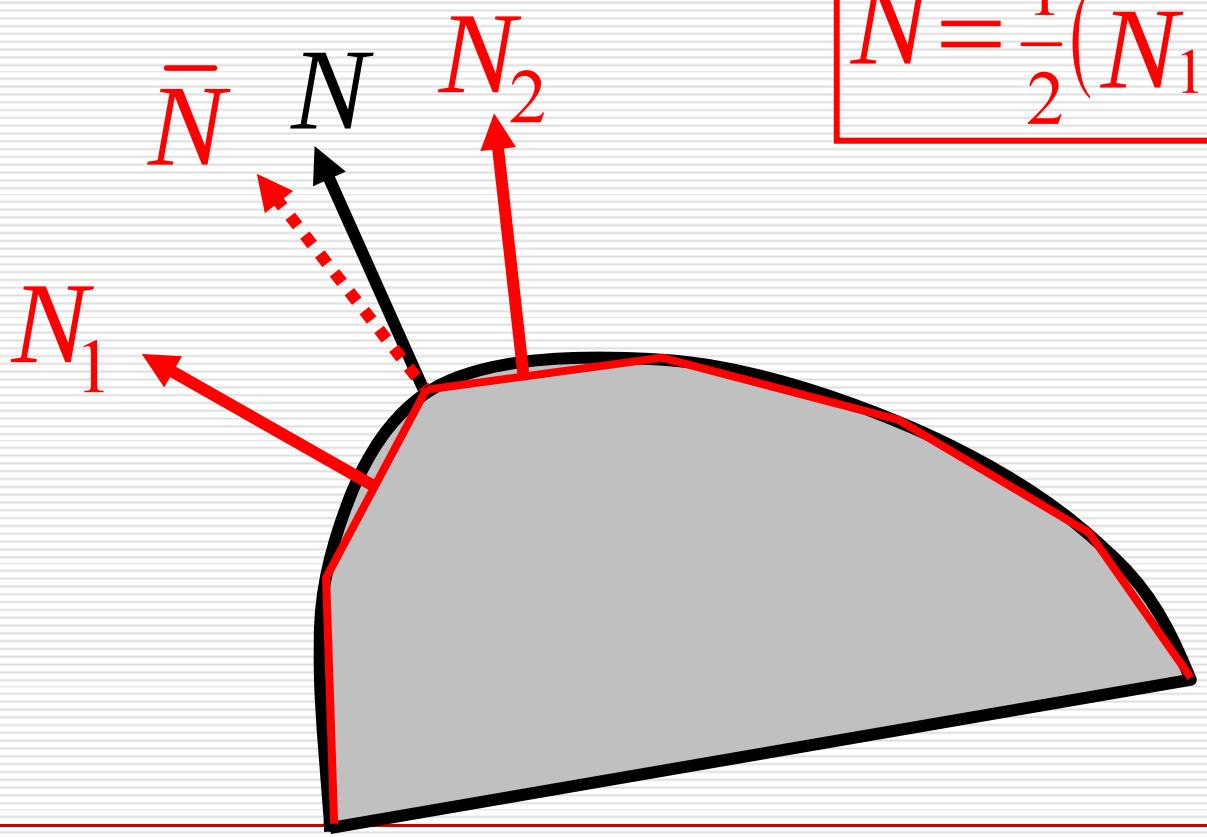


Using Average Normals



Using Average Normals

$$\bar{N} = \frac{1}{2}(N_1 + N_2)$$

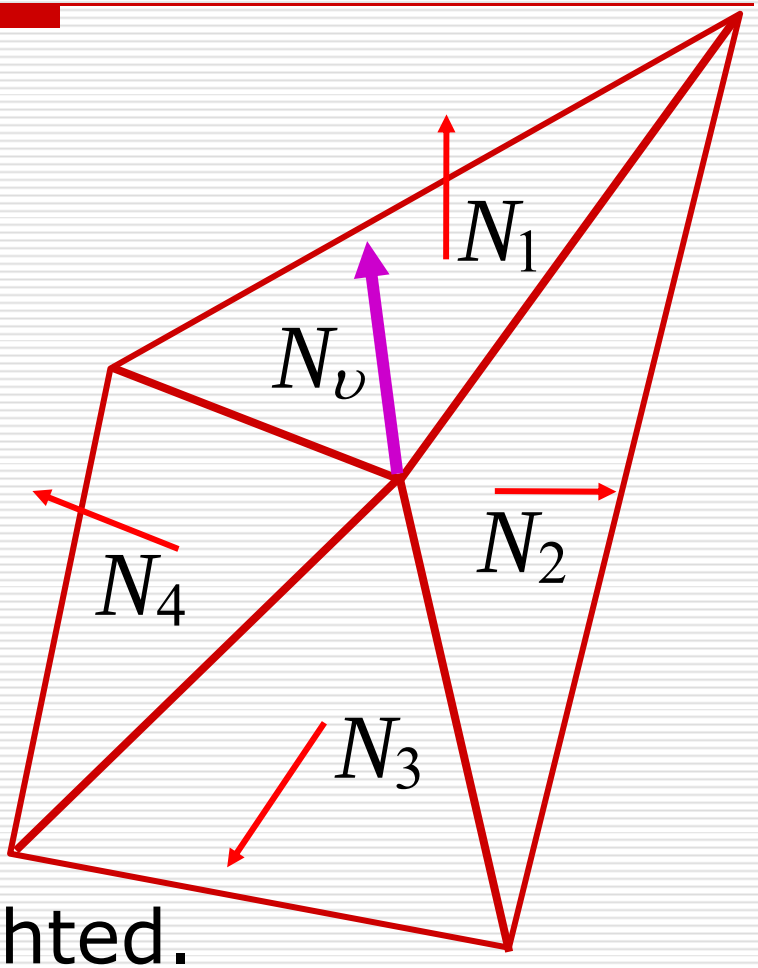


Using Average Normals

$$N_v = \frac{(N_1 + N_2 + N_3 + N_4)}{\|N_1 + N_2 + N_3 + N_4\|}$$

More generally,

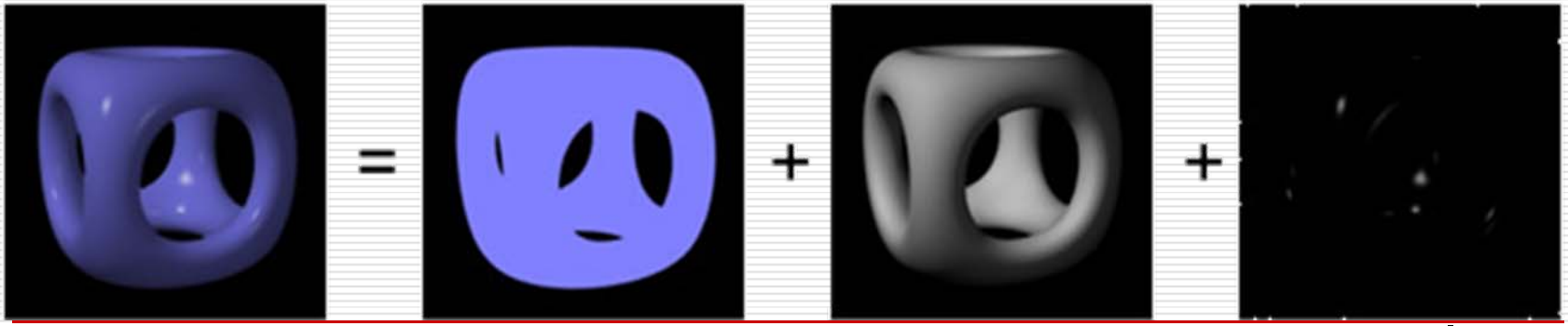
$$N_v = \frac{\sum_{i=1}^{n} N_i}{\left| \sum_{i=1}^{n} N_i \right|}$$



It can also be area-weighted.

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



Colored Lights and Surfaces

- If an object's **diffuse color** is $O_d = (O_{dR}, O_{dG}, O_{dB})$ then $I = (I_R, I_G, I_B)$ where for the red component

$$I_R = I_{aR} k_a O_{dR} + f_{att} I_{pR} k_d O_{dR} (\vec{N} \bullet \vec{L})$$

however, it should be

$$I_\lambda = I_{a\lambda} k_a O_{d\lambda} + f_{att} I_{p\lambda} k_d O_{d\lambda} (\vec{N} \bullet \vec{L})$$

where λ is the **wavelength**

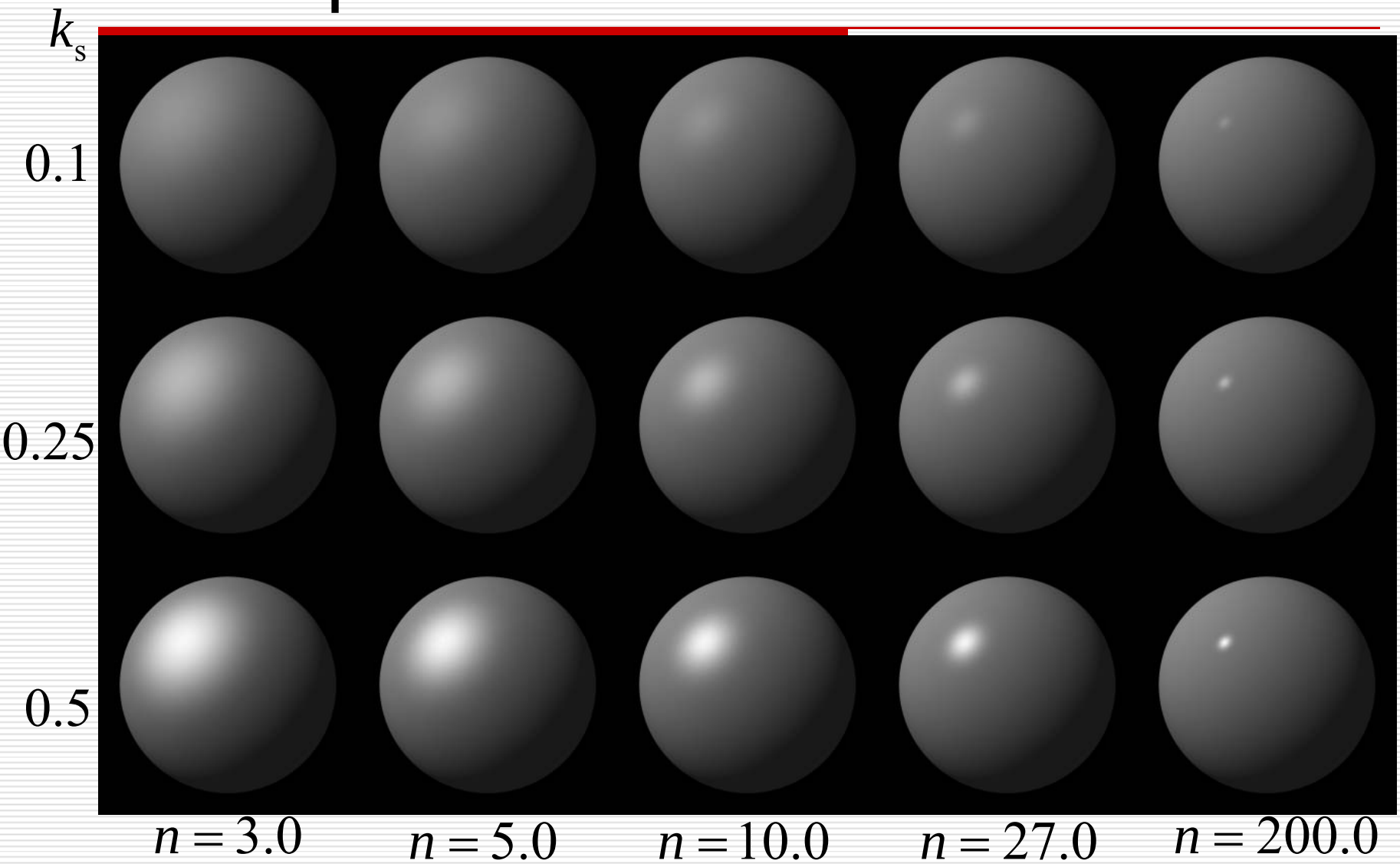
The Phong Illumination Model

- $I_\lambda = I_{a\lambda} k_a O_{d\lambda} + f_{att} I_{p\lambda} [k_d O_{d\lambda} \cos\theta + W(\theta) \cos^n \alpha]$
  - $W(\theta) = k_s$: 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**

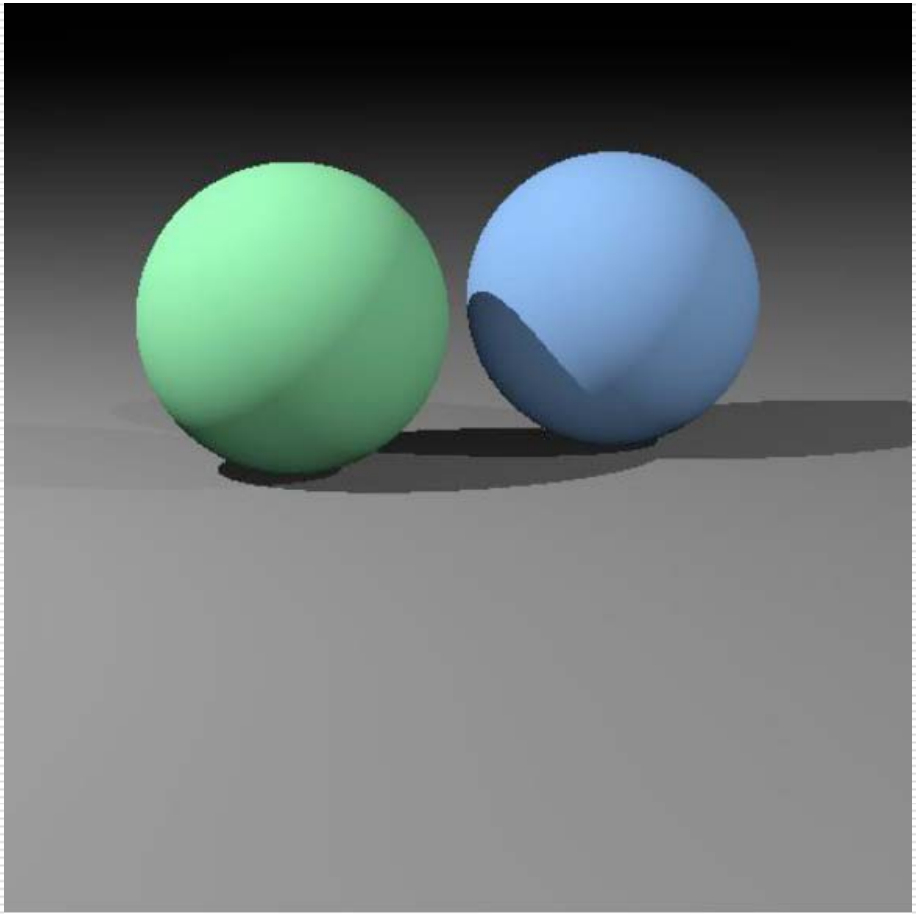
$$I_\lambda = I_{a\lambda} k_a O_{d\lambda} + f_{att} I_{p\lambda} [k_d O_{d\lambda} (\vec{N} \bullet \vec{L}) + k_s O_{s\lambda} (\vec{R} \bullet \vec{V})^n]$$

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

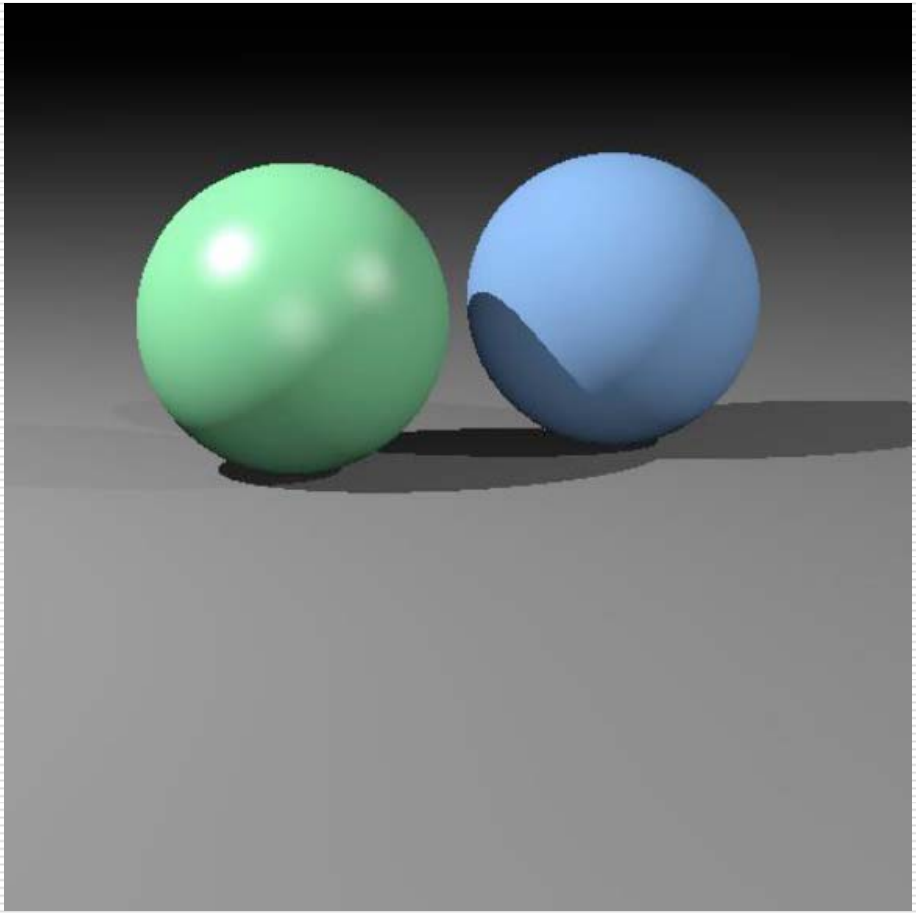
Examples



Specular Shading



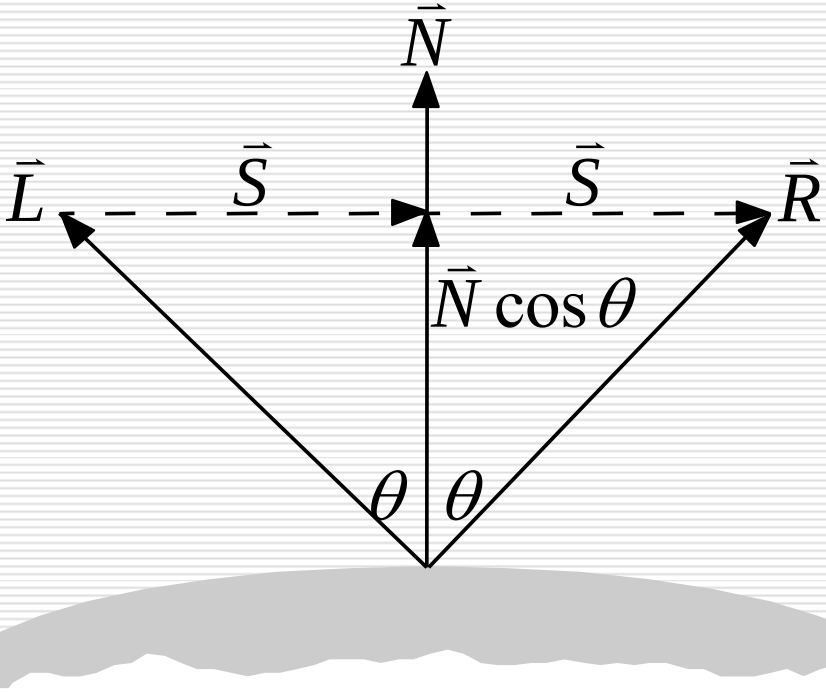
diffuse



diffuse + specular

Calculating the Reflection Vector

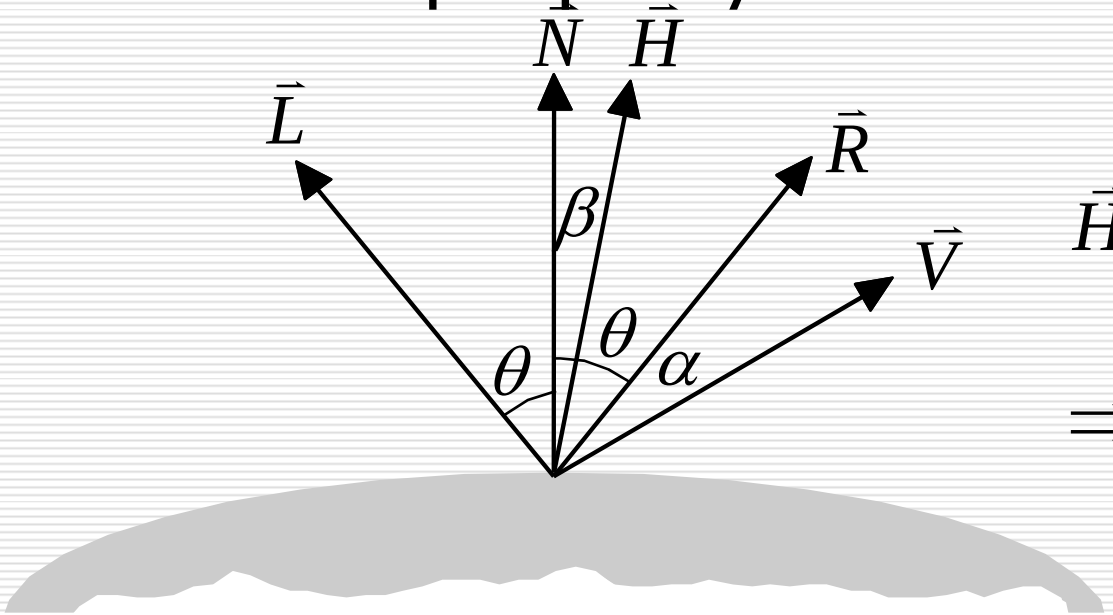
- Fall off gradually from the perfect reflection direction



$$\vec{R} = \vec{N}\cos\theta + \vec{S}$$
$$= \vec{N}\cos\theta + \vec{N}\cos\theta - \vec{L}$$
$$= 2\vec{N}\cos\theta - \vec{L}$$
$$= 2\vec{N}(\vec{N} \bullet \vec{L}) - \vec{L}$$

The Halfway Vector (Blinn-Phong)

- Rather than computing reflection directly; just compare to normal bisection property.



$$\vec{H} = \frac{\vec{L}+\vec{V}}{\left|\vec{L}+\vec{V}\right|}$$

$$\Rightarrow \cos\alpha \approx \vec{N} \bullet \vec{H}$$

Multiple Light Sources

- If there are m light sources, then

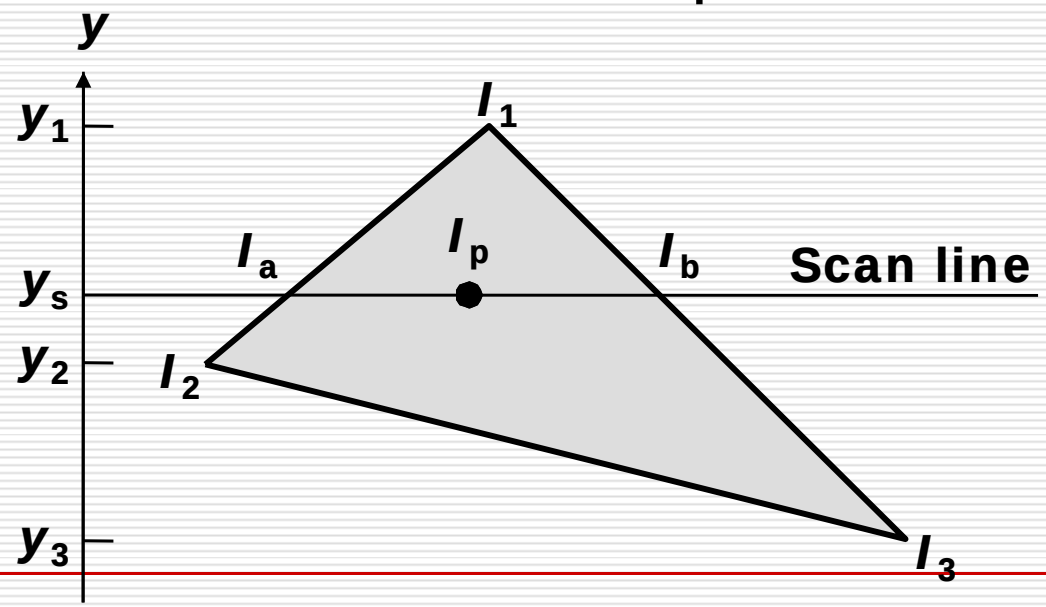
$$I_\lambda = I_{a\lambda} k_a O_{d\lambda} + \sum_{1 \le i \le m} f_{att_i} I_{p\lambda_i} [k_d O_{d\lambda} (\vec{N} \bullet \vec{L}_i) + k_s O_{s\lambda} \cos^n \alpha_i]$$

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

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

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

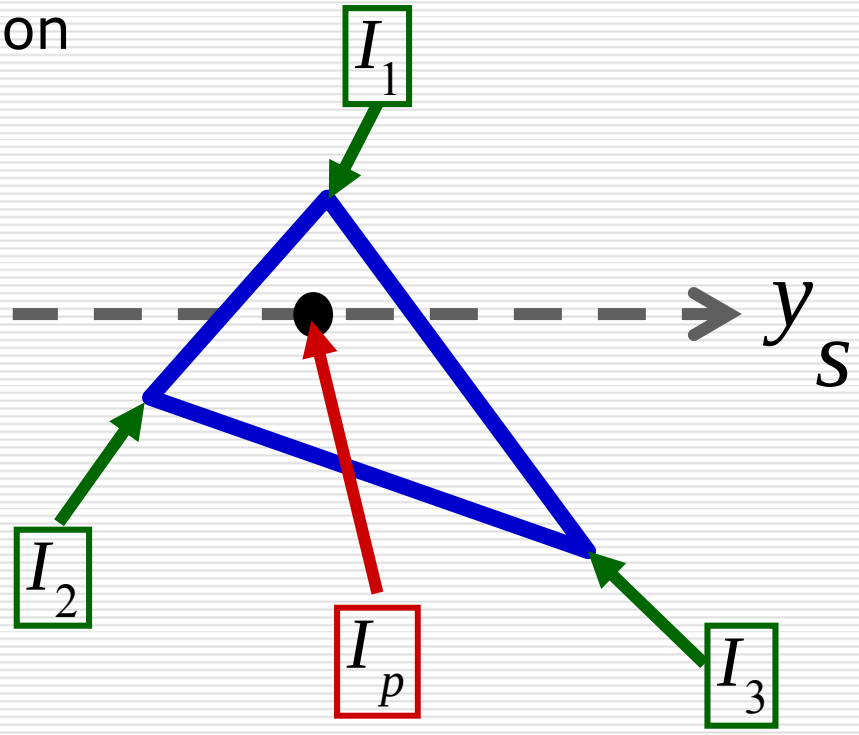
Flat Shading

- 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

$$I_p = I$$

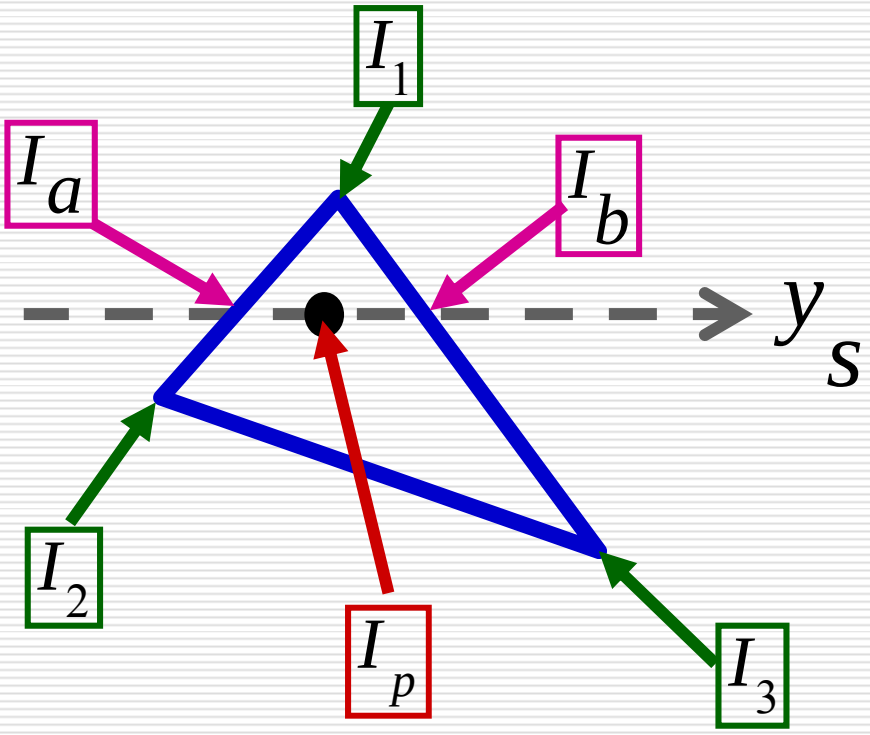


Intensity Interpolation (Gouraud)

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

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

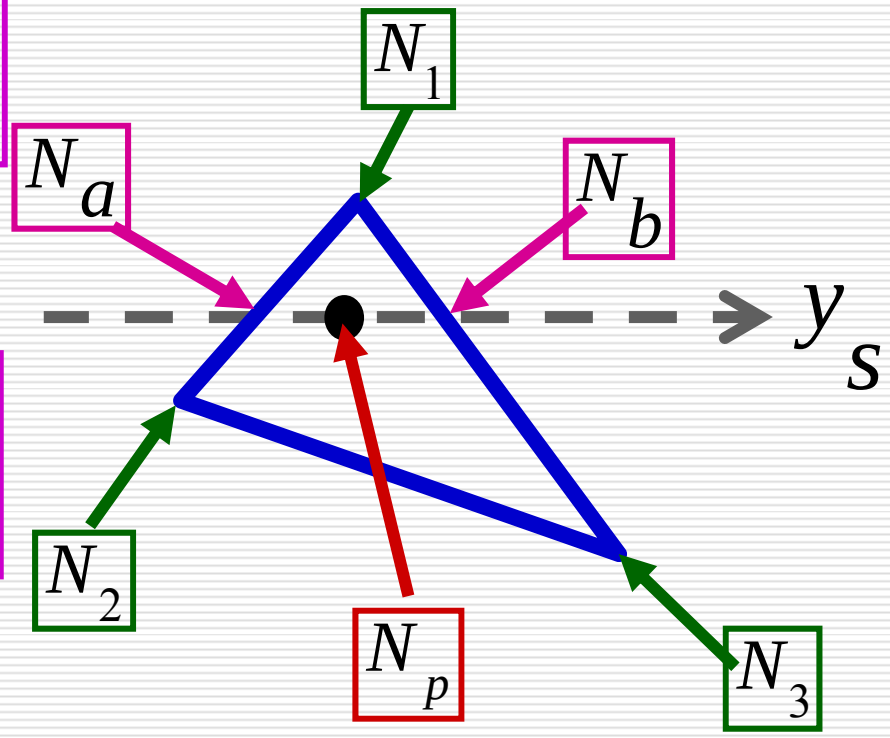
$$I_p = I_a \frac{x_b - x_p}{x_b - x_a} + I_b \frac{x_p - x_a}{x_b - x_a}$$



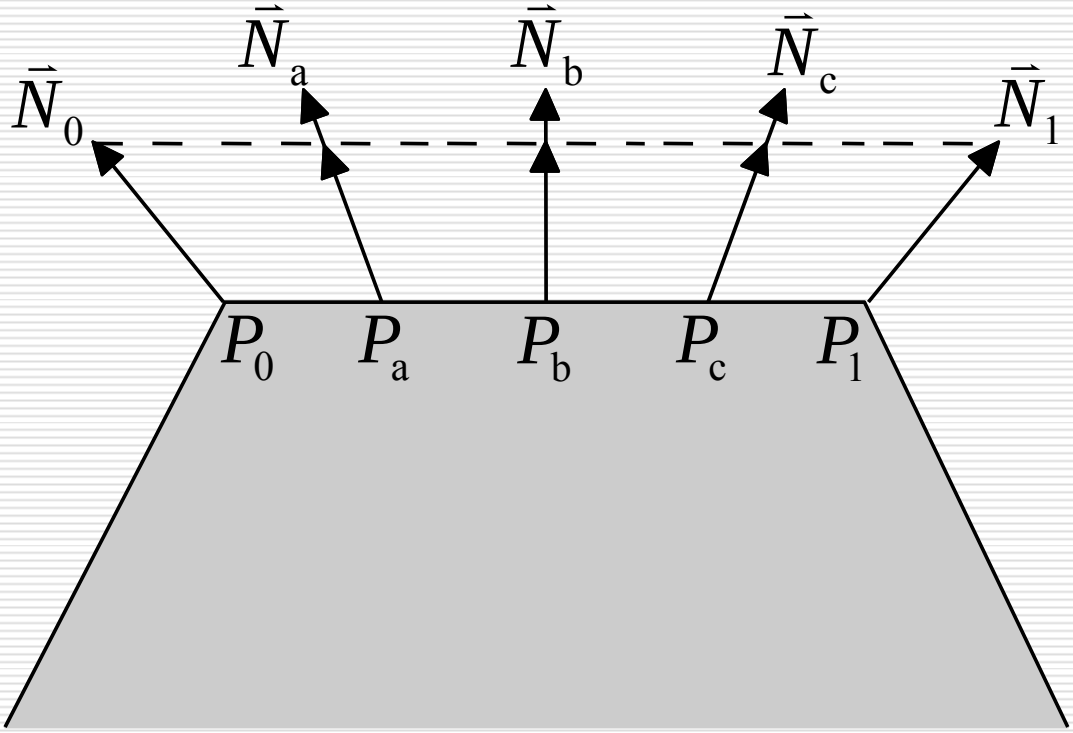
Normal Interpolation (Phong)

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

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



Normal Interpolation (Phong)



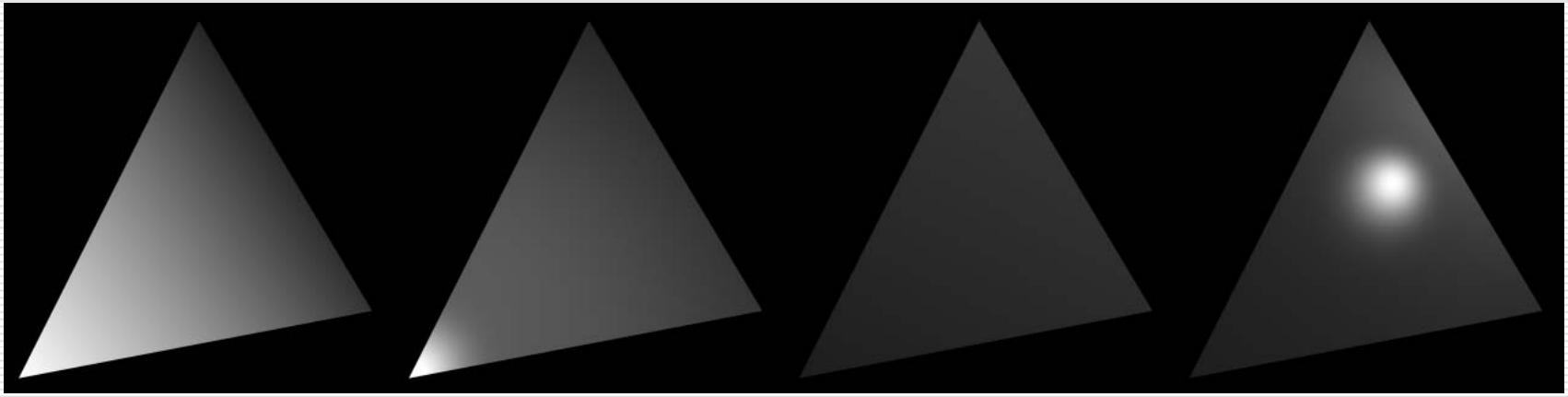
Normal Interpolation (Phong)

$$\tilde{N}_p = \frac{N_a}{\|N_a\|}\left[\frac{x_b - x_p}{x_b - x_a}\right] + \frac{N_b}{\|N_b\|}\left[\frac{x_p - x_a}{x_b - x_a}\right]$$

$$N_p = \frac{\tilde{N}_p}{\left\|\tilde{N}_p\right\|}$$

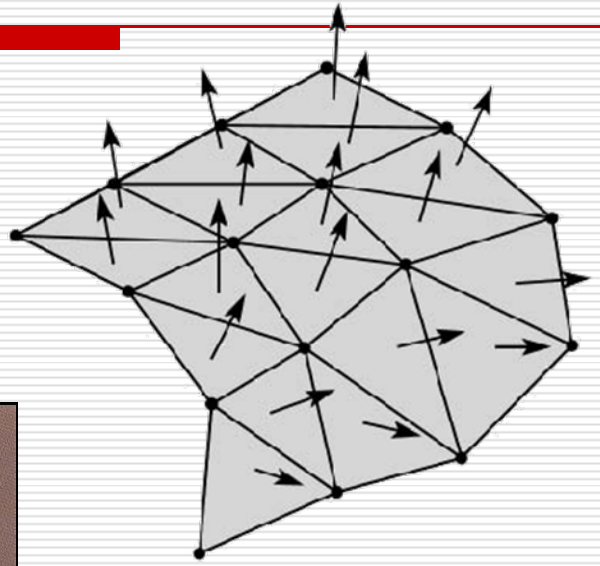
Normalizing makes this a unit vector

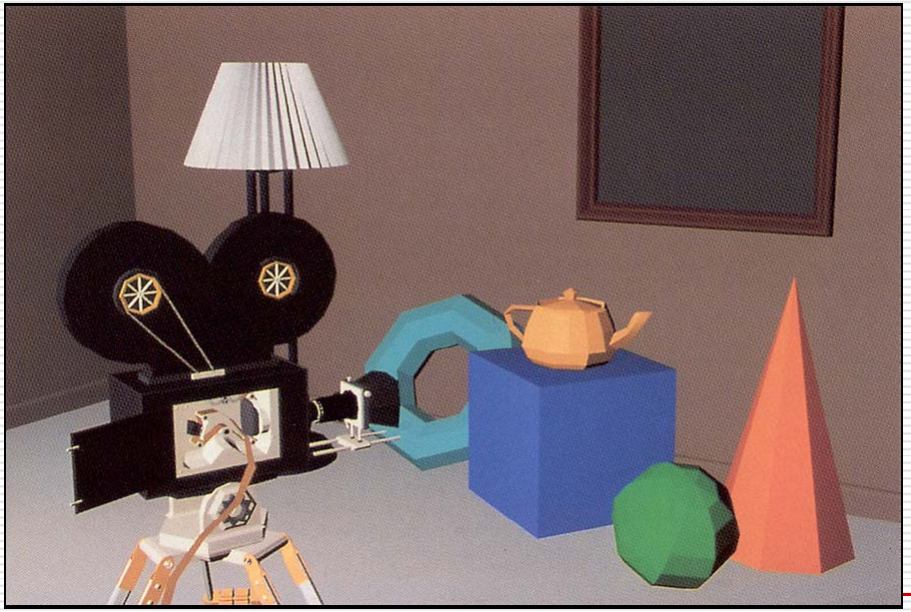
Gouraud v.s. Phong Shading



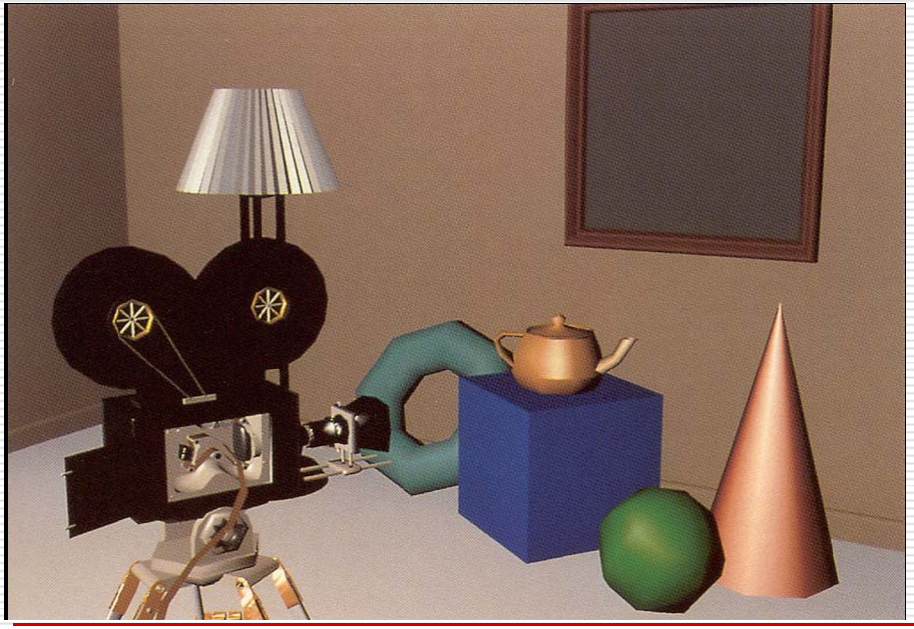
Gouraud Phong Gouraud Phong

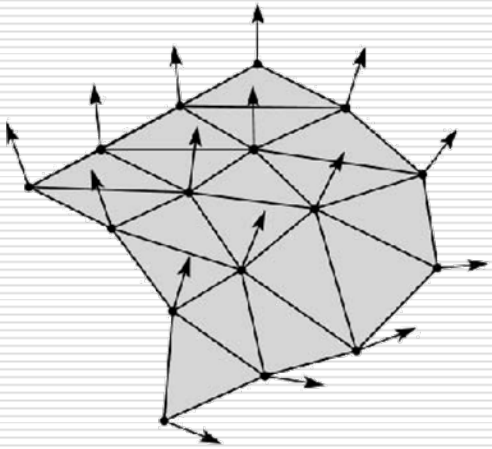
Flat Shading

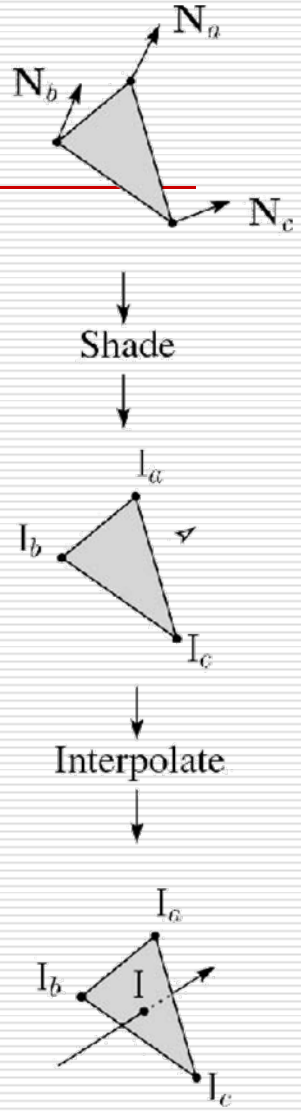




Gouraud Shading







Phong Shading