Introduction to Computer Graphics

2016 Spring

National Cheng Kung University

Instructors: Min-Chun Hu 胡敏君

Shih-Chin Weng 翁士欽 (西基電腦動畫)

Shading

Why Do We Need Shading?

- We color a sphere model with a constant color and get something like

- But we want 3D appearance like

Shading

- Why does the image of a real sphere look like

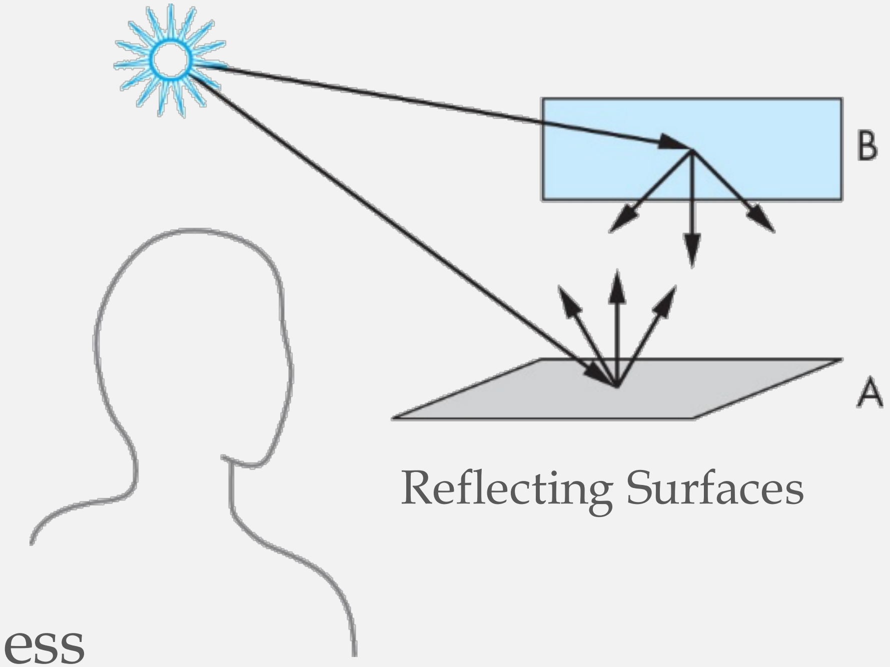
- Light-material interactions cause each point to have a different color or shade
- Need to consider the following factors:
  - Location and properties of the light sources
  - Material properties
  - Local geometry of the surface (surface orientation)
  - Location of the viewer

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

Light-material Interaction

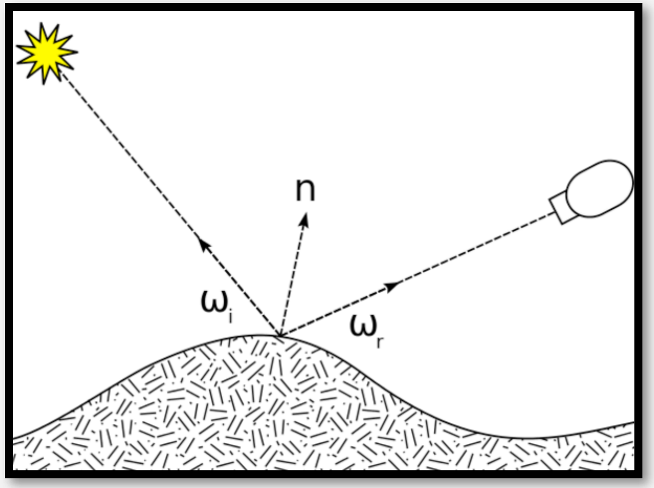
- The color we see is determined by multiple interactions among light sources and reflective surfaces
  - Light strikes A
    - Some scattered
    - Some absorbed
  - Some of scattered light strikes B
    - Some scattered
    - Some absorbed
  - Some of this scattered light strikes A and so on
- These interactions can be seen as a recursive process

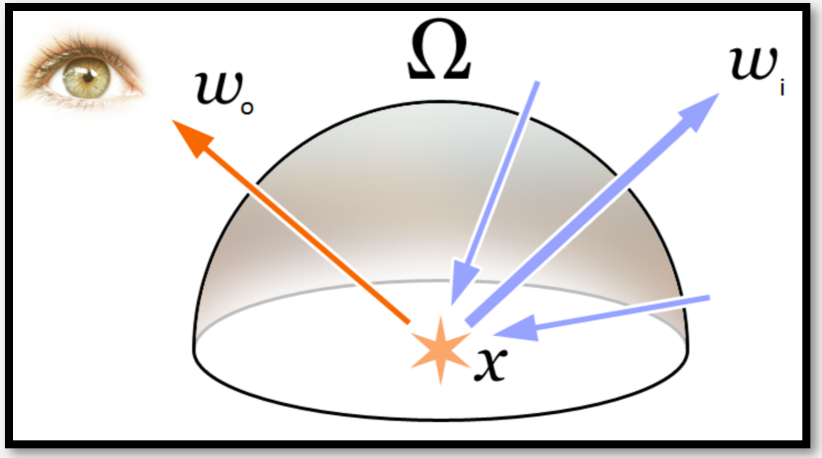


Reflecting Surfaces

Light-material Interaction (Cont.)

- BRDF (Bidirectional reflectance distribution function)
  - $f_r(\omega_i, \omega_r)$: A function that defines how light is reflected at an opaque surface
- Rendering Equation:
  - Describes the total amount of light emitted from a point x along a particular viewing direction, given a function of incoming light and a BRDF.



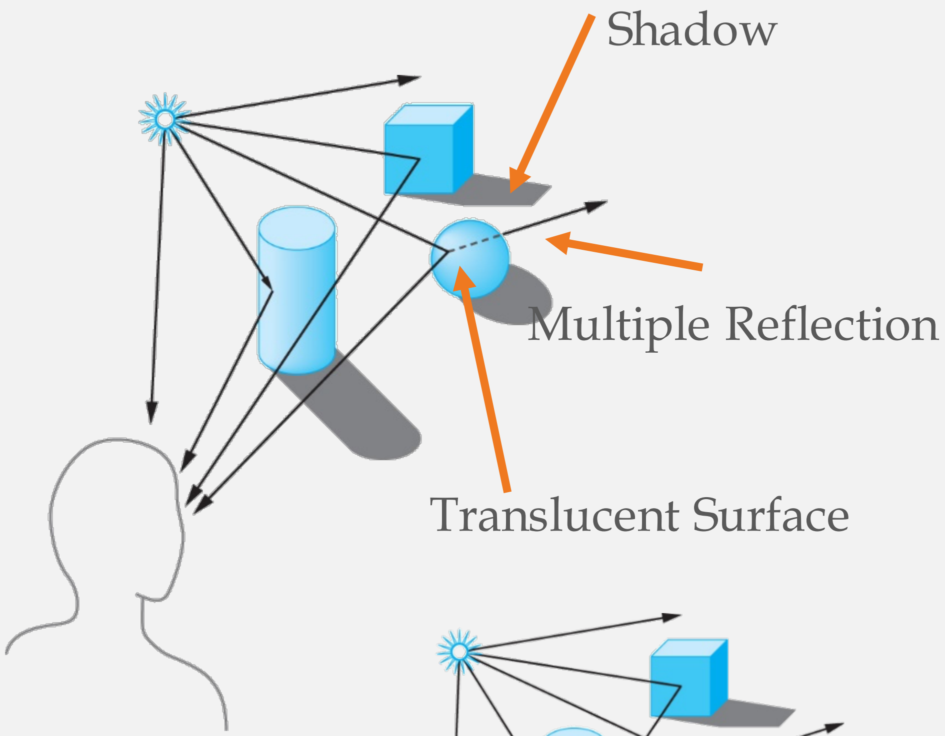


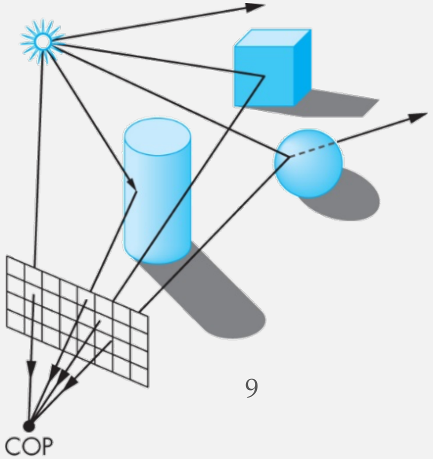
Light-material Interaction (Cont.)

- Radiant Energy : (J)
  - The energy transported by electromagnetic radiation
- Radiant Flux : (W=J/sec)
  - Radiant energy per unit time
- Irradiance : (W/m^2)
  - Total amount of radiant flux incident upon a point on a surface from all directions above the surface

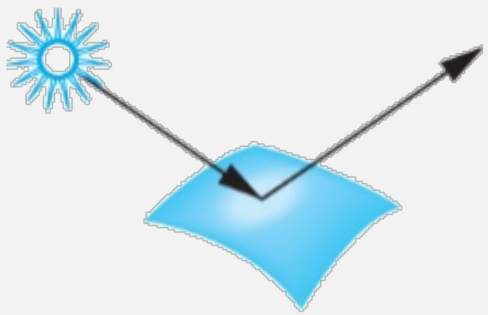
Light-material Interaction (Cont.)

- Rather than looking at a global energy balance, we only consider the lighting rays leaving the source and reaching the viewer's eye.
  - Only consider single interaction between light sources and surfaces
  - Must model the light sources in the scene
  - Must build a reflection model that deal with the interactions between material and light

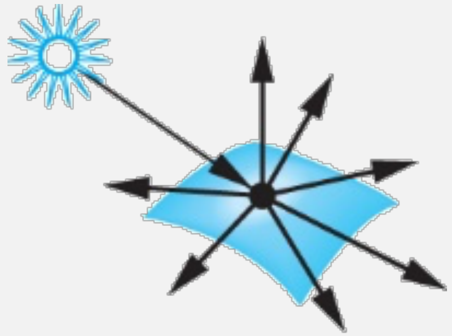




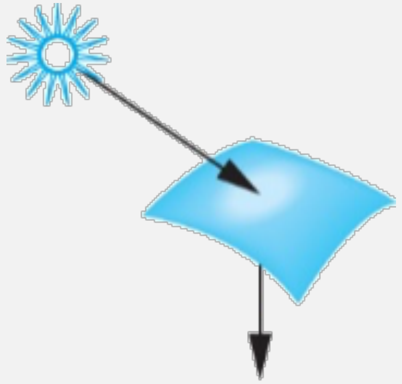
Light-material Interaction (Cont.)



Specular Surface



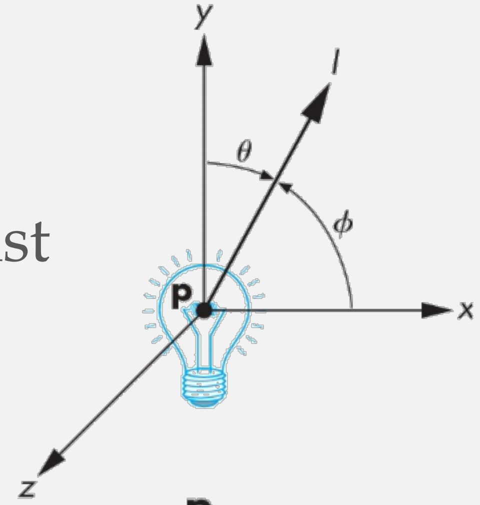
Diffuse Surface

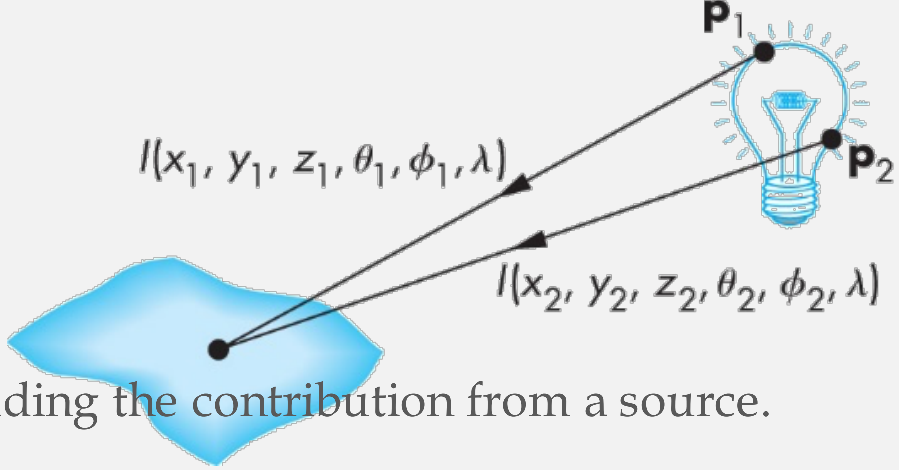


Translucent Surface

Light Sources

- A general light source can be characterized by a six-variable illumination function $I(x, y, z, \theta, \phi, \lambda)$
- General light sources are difficult to simulate because we must integrate light coming from all points on the source.
- The luminance of the color source: $I = \begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix}$





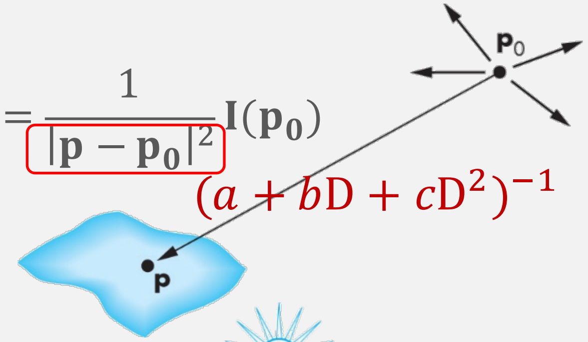
Adding the contribution from a source.

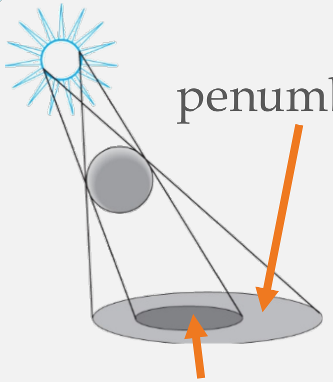
Simple Light Sources

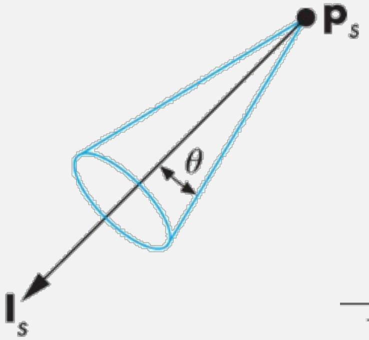
- Ambient light
  - Same amount of light everywhere in scene
  - Can model contribution of many sources and reflecting surfaces
- Point source
  - Model with position and color
  - Distant source = infinite distance away (parallel)
  - Scenes rendered with only point sources tend to have high contrast
    - Can be solved by adding ambient light
- Spotlight
  - Restrict light from ideal point source
- Distant Light

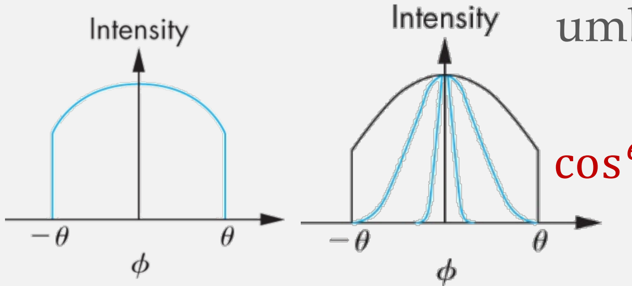
$$\mathbf{i}(\mathbf{p},\mathbf{p_0}) = \frac{1}{\boxed{|\mathbf{p}-\mathbf{p_0}|^2}}\mathbf{I}(\mathbf{p_0})$$

$$(a + bD + cD^2)^{-1}$$



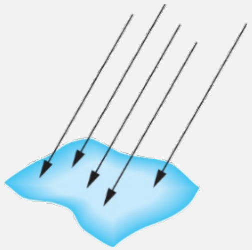






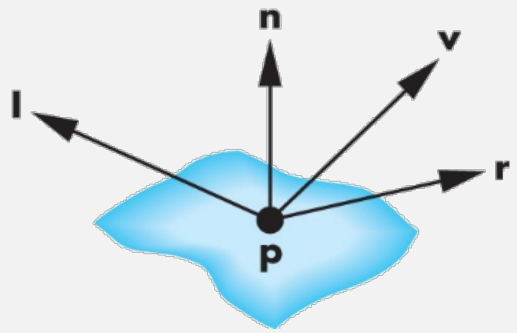
$\cos^e \phi$

Attenuation of a spotlight



The Phong Reflection Model

- A simple model that can be computed efficiently and is a close-enough approximation to physical reality
- Uses four vectors to calculate a color for a point **p**
  - The normal at **p**: **n**
  - The vector from **p** to the viewer (COP): **v**
  - The vector from **p** to the light source: **l**
  - Perfect reflector ray that **l** would take: **r**



- Support three types of material-light interactions
  - Ambient
  - Diffuse
  - Specular

The Phong Reflection Model (Cont.)

- Assume each source can have separate ambient, diffuse, and specular components for each of the three primary colors:

$$\text{The i-th light source } L_i = \begin{bmatrix} L_{ira} & L_{iga} & L_{iba} \\ L_{ird} & L_{igd} & L_{ibd} \\ L_{irs} & L_{igs} & L_{ibs} \end{bmatrix}$$

- The reflection term $R_i = \begin{bmatrix} R_{ira} & R_{iga} & R_{iba} \\ R_{ird} & R_{igd} & R_{ibd} \\ R_{irs} & R_{igs} & R_{ibs} \end{bmatrix}$
- The intensity we see at **p** from source $\boldsymbol{i}$ is :

$$\boldsymbol{I_{ic}} = \boldsymbol{R_{ica}L_{ica}} + \boldsymbol{R_{icd}L_{icd}} + \boldsymbol{R_{ics}L_{ics}} = \boldsymbol{I_{ica}} + \boldsymbol{I_{icd}} + \boldsymbol{I_{ics}}$$

- Obtain the total intensity of the r/g/b color component by adding contributions of all sources:

$$I_c = \sum_i (\boldsymbol{I_{ica}} + \boldsymbol{I_{icd}} + \boldsymbol{I_{ics}}) + \boldsymbol{I_{ac}}$$

r/g/b component of the global ambient light

Ambient Reflection

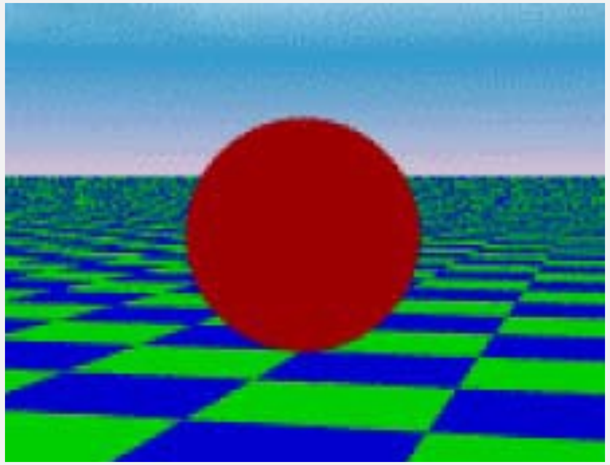
- The intensity of ambient light (I_a) is the same at every point on the surface
  - Some of L_a is absorbed and some is reflected according to the reflection coefficient

$$I_{ra} = k_{ra} \cdot L_{ra}$$

$$I_{ga} = k_{ga} \cdot L_{ga}$$

$$I_{ba} = k_{ba} \cdot L_{ba}$$

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



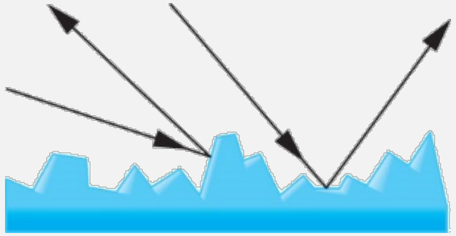
Diffuse Reflection

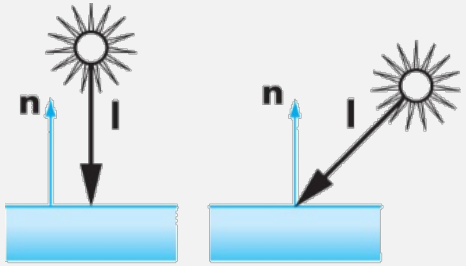
- Light scattered equally in all directions, hence appears the same to all viewers
- Reflected intensities depend on both the material (some would be absorbed) and the direction of the light
- Perfect diffuse surfaces, named Lambertian Surface, are so rough that there is no preferred angle of reflection.
- Lambert's law: we see only the vertical component of the incoming light

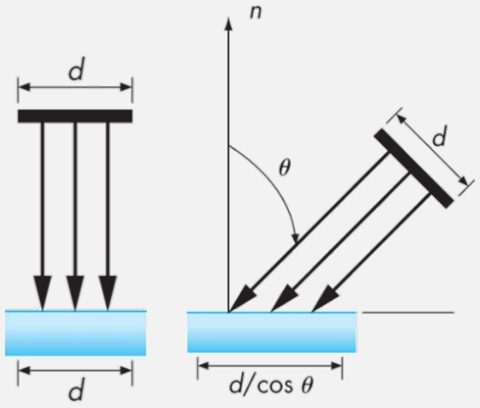
$$I_d = k_d L_d \cos\theta = k_d L_d (\mathbf{l} \cdot \mathbf{n})$$

- Consider the distance term:

$$I_d = \frac{k_d}{a + bD + cD^2} L_d (\mathbf{l} \cdot \mathbf{n})$$

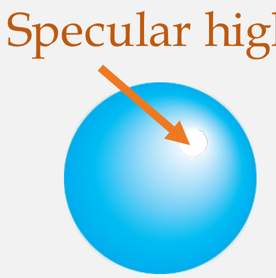


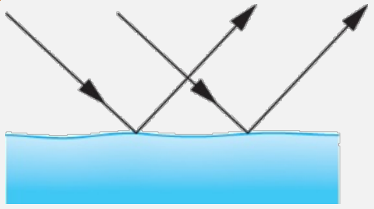




Specular Reflection

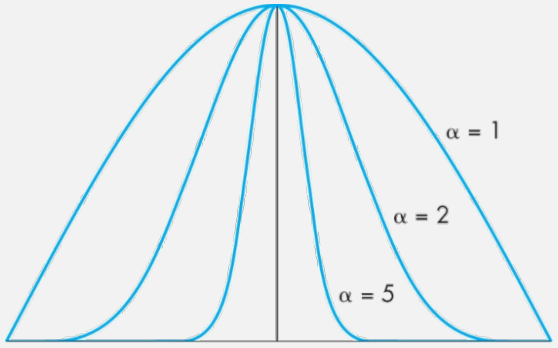
- Most surfaces are neither ideal diffusers nor perfectly specular (ideal reflectors)
- Incoming light is reflected in directions concentrated close to the perfect reflection direction
- The amount of light the viewer sees depends on the angle between **r** and **v** :





$$I_s = k_s L_s \cos^{\alpha} \phi = k_s L_s (\mathbf{r} \cdot \mathbf{v})^{\alpha}$$

shininess coefficient



- Consider the distance term:

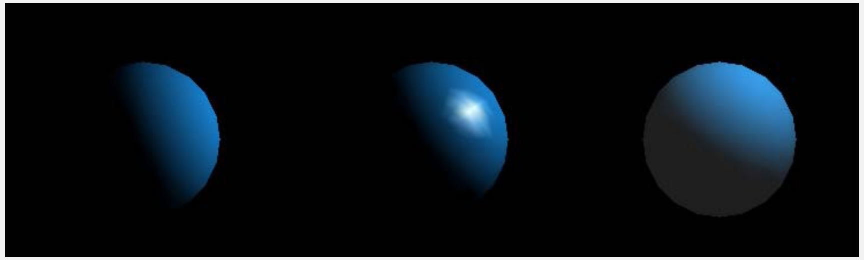
$$I_s = \frac{1}{a + bD + cD^2} k_s L_s (\mathbf{r} \cdot \mathbf{v})^{\alpha}$$

Phong Model

- For each light source and each primary component:

$$I = \frac{1}{a + bD + cD^2}\left(k_d L_d \max\left((\mathbf{l} \cdot \mathbf{n}), 0\right) + k_s L_s \max\left((\mathbf{r} \cdot \mathbf{v})^{\alpha}, 0\right)\right) + k_a L_a$$

- Coefficients:
  - 9 coefficients for each point light source
  - 9 absorption coefficients
  - 1 shininess coefficient



Diffuse | Diffuse + Specular | Diffuse + Ambient

Modified Phong Model (Blinn-Phong Model)

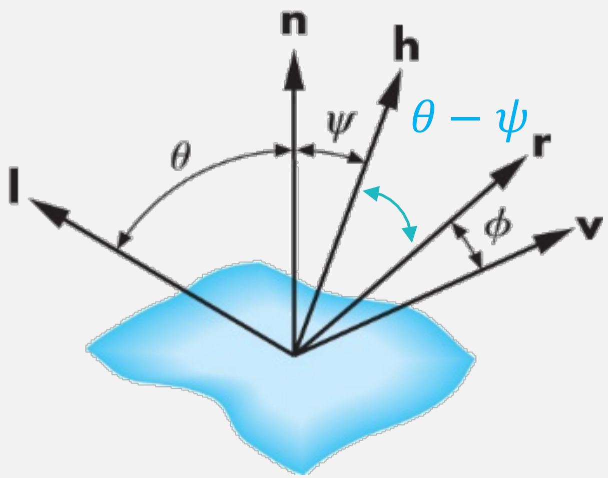
- Problem:
  - In the specular component of the Phong model, it requires the calculation of a new reflection vector **r** and view vector **v** for each vertex

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

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

$$\mathbf{h} = \frac{\mathbf{l}+\mathbf{v}}{|\mathbf{l}+\mathbf{v}|}$$

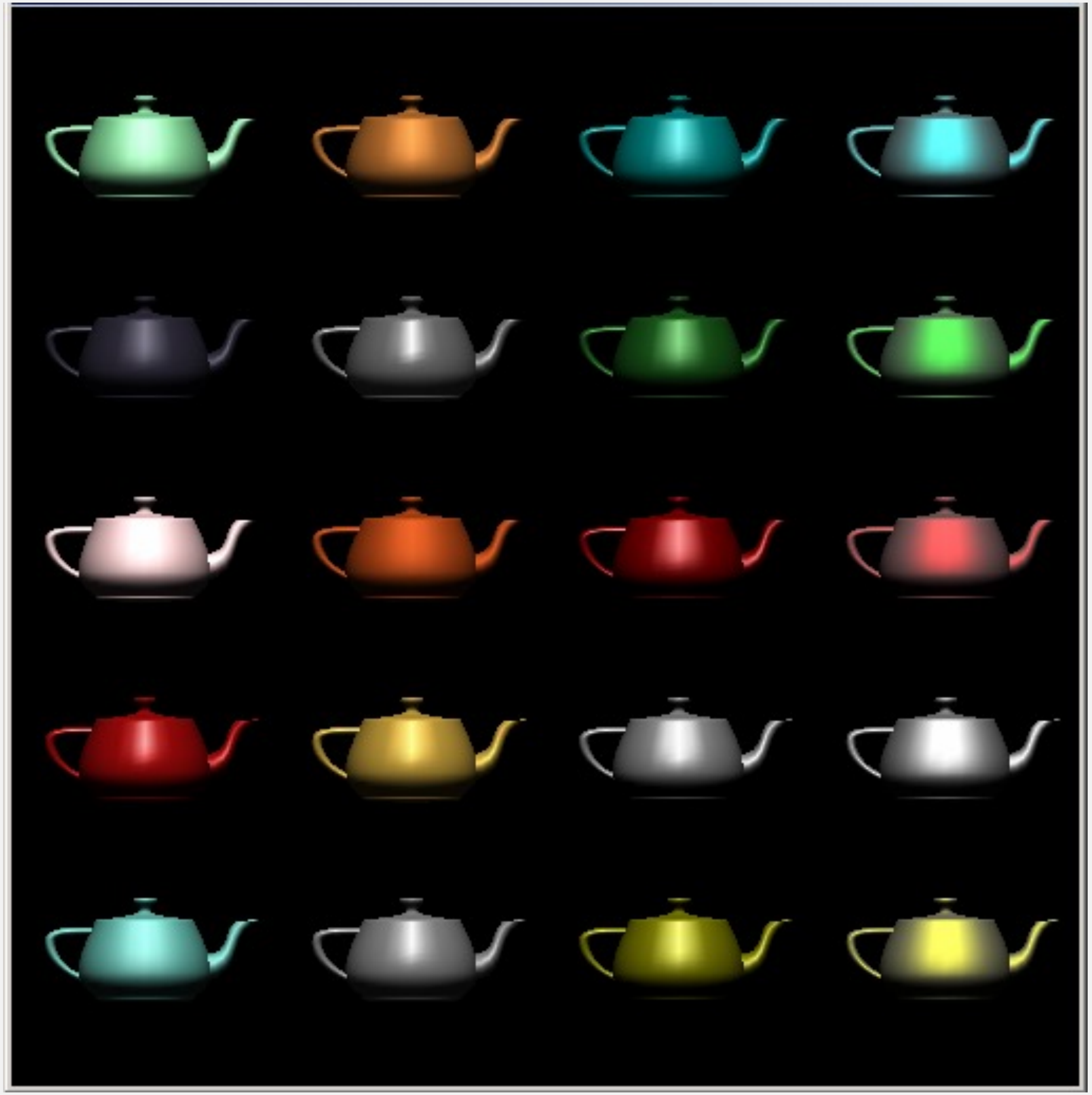
  - Replace $(\mathbf{r}\cdot\mathbf{v})^{\alpha}$ by $(\mathbf{n}\cdot\mathbf{h})^{\alpha'}$



if vectors are coplanar:

$$\theta + \psi = \theta - \psi + \phi$$
$$\Rightarrow 2\psi = \phi$$

Examples (Modified Phong Model)



Teapot with different parameters

Computation of Vectors

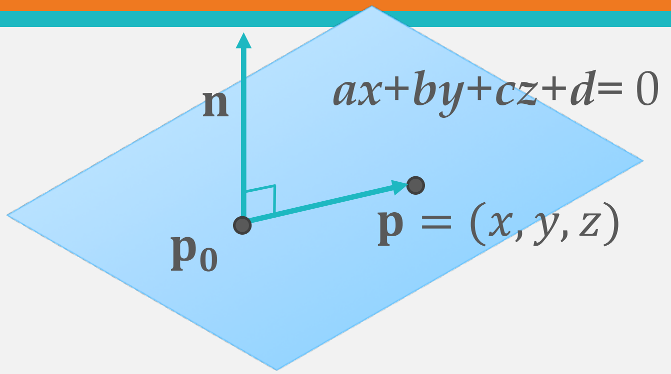
- **l** and **v** : specified by the application
- **r** : computed from **l** and **n**
- Determine **n**
  - OpenGL leaves determination of normal to application and put them in a vertex array buffer (VAB) just as we do for vertex positions
  - Exception for GLU quadrics and Bezier surfaces

Normals

- Equation of plane: $\boldsymbol{ax} + \boldsymbol{by} + \boldsymbol{cz} + \boldsymbol{d} = 0$
  - Plane 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 $f(x, y, z) = x^2 + y^2 + z^2 - 1 = 0$
  - Sphere normal is given by gradient: $\mathbf{n} = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right]^T$

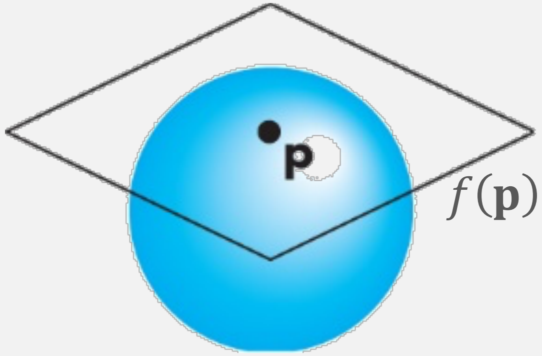
$$= [2x, 2y, 2z]^T$$

$$= 2\mathbf{p}^T$$

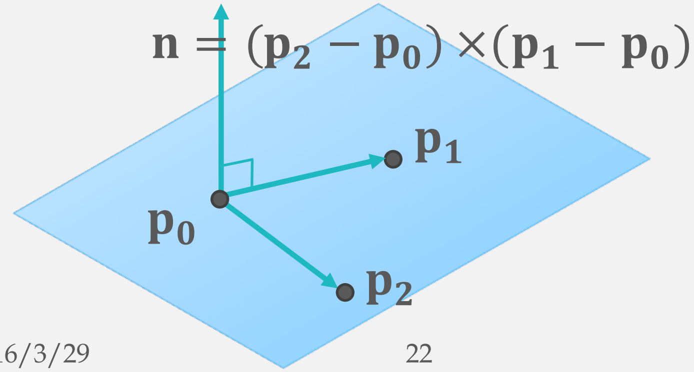


$$\mathbf{n} \cdot (\mathbf{p} - \mathbf{p_0}) = 0$$

$$\Rightarrow \mathbf{n} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \text{ or } \mathbf{n} = \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix}$$



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



Normal Vector Calculation

- Parametric form for sphere

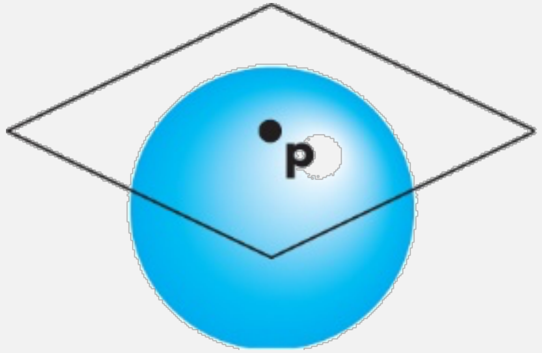
$$x = x(u,v) = \cos u \sin v$$

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

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

$$-\frac{\pi}{2} < u < \frac{\pi}{2}, \quad -\pi < v < \pi$$

- Tangent plane determined by vectors:



$$\frac{\partial \mathbf{p}}{\partial u} = \left[\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u}\right]^T \qquad \frac{\partial \mathbf{p}}{\partial v} = \left[\frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v}\right]^T$$

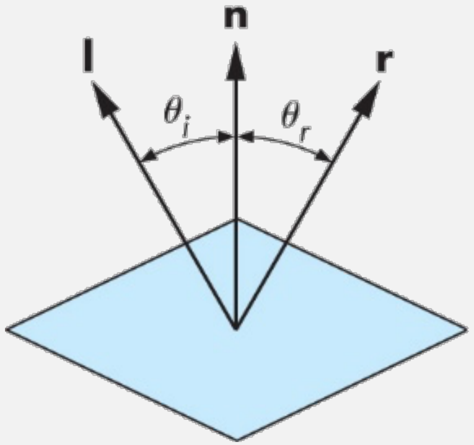
- Sphere normal is given by cross product

$$\mathbf{n} = \frac{\partial \mathbf{p}}{\partial u} \times \frac{\partial \mathbf{p}}{\partial v} = \cos u \begin{bmatrix} \cos u \sin v \\ \cos u \cos v \\ \sin u \end{bmatrix} = (\cos u)\mathbf{p} \qquad \Rightarrow \mathbf{n} = \mathbf{p}$$

Reflection Vector Calculation

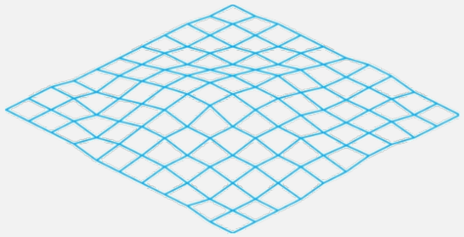
- Determine r from **l** and **n**
  - The angle of incidence is equal to the angle of reflection:
    $$\theta_i = \theta_r \Rightarrow \cos\theta_i = \underline{\mathbf{l}\cdot\mathbf{n}} = \cos\theta_r = \underline{\mathbf{n}\cdot\mathbf{r}}$$
  - The incoming light ray, the reflected light ray, and the normal at the point **p** all lie in the same plane (coplanar condition):
    $$① \quad \mathbf{r} = \alpha\mathbf{l} + \beta\mathbf{n} \Rightarrow \underline{\mathbf{n}\cdot\mathbf{r}} = \alpha\mathbf{l}\cdot\mathbf{n} + \beta\mathbf{n}\cdot\mathbf{n} = \alpha\mathbf{l}\cdot\mathbf{n} + \beta = \underline{\mathbf{l}\cdot\mathbf{n}}$$
  - Assume $|\mathbf{l}| = |\mathbf{n}| = |\mathbf{r}| = 1$:
    $$② \quad 1 = \mathbf{r}\cdot\mathbf{r} = (\alpha\mathbf{l} + \beta\mathbf{n})\cdot(\alpha\mathbf{l} + \beta\mathbf{n}) = \alpha^2 + 2\alpha\beta\mathbf{l}\cdot\mathbf{n} + \beta^2$$

By ① and ② : $\quad \mathbf{r} = \mathbf{2}(\mathbf{l}\cdot\mathbf{n})\mathbf{n} - \mathbf{l}$



Polygonal Shading

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

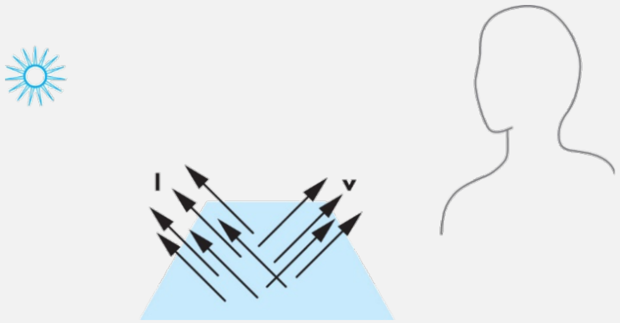


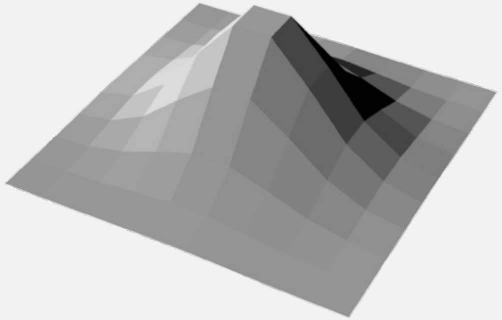
Polygonal Mesh

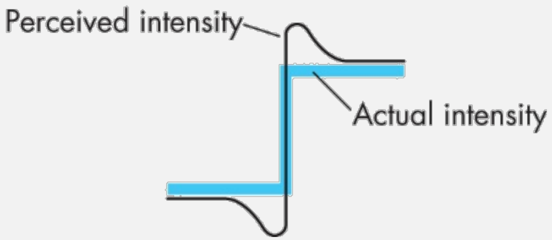
$$I = \frac{1}{a + bD + cD^2}\left(k_d L_d \max\left((\mathbf{l} \cdot \mathbf{n}), 0\right) + k_s L_s \max\left((\mathbf{r} \cdot \mathbf{v})^{\alpha}, 0\right)\right) + k_a L_a$$

Flat/Constant Shading

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







$$I = \frac{1}{a + bD + cD^2}\left(k_d L_d \max\left((\mathbf{l} \cdot \mathbf{n}), 0\right) + k_s L_s \max\left((\mathbf{r} \cdot \mathbf{v})^{\alpha}, 0\right)\right) + k_a L_a$$

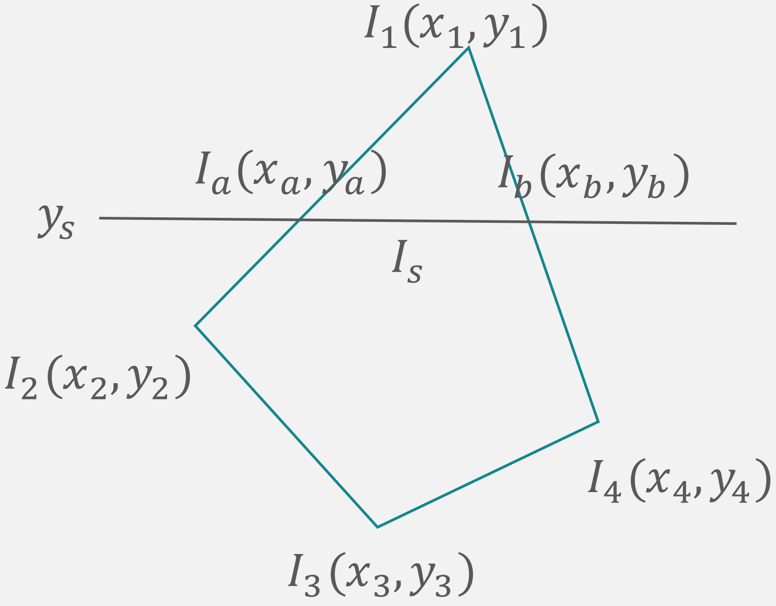
Smooth/Gouraud Shading

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

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

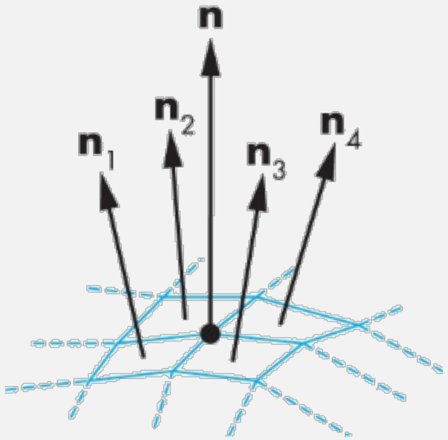
$$I_b = \frac{1}{y_1 - y_4}[I_1(y_s - y_4) + I_4(y_1 - y_s)]$$

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

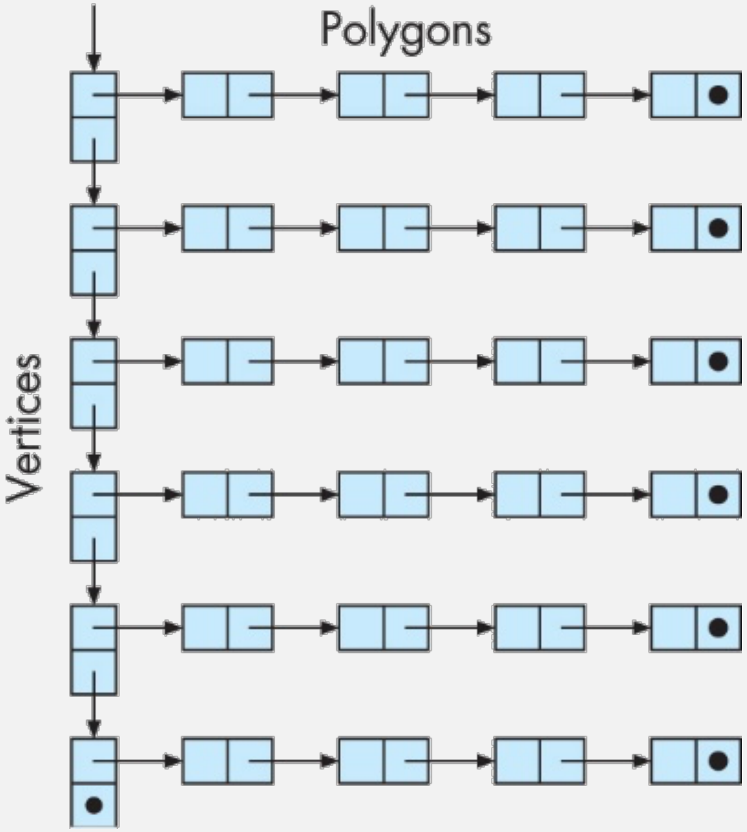


Normal at A Vertex

- $\mathbf{n} = \frac{\mathbf{n_1}+\mathbf{n_2}+\mathbf{n_3}+\mathbf{n_4}}{|\mathbf{n_1}+\mathbf{n_2}+\mathbf{n_3}+\mathbf{n_4}|}$

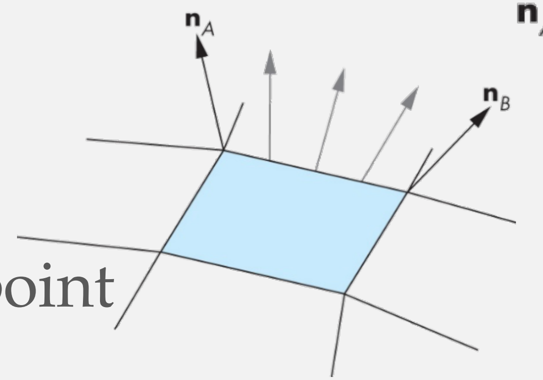


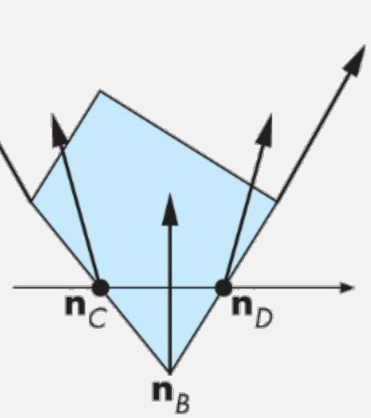
- How to find the normal that we should average together?
  - Maintain a data structure containing polygons, vertices, normal, and material properties.



Phong Shading (Per-fragment Shading)

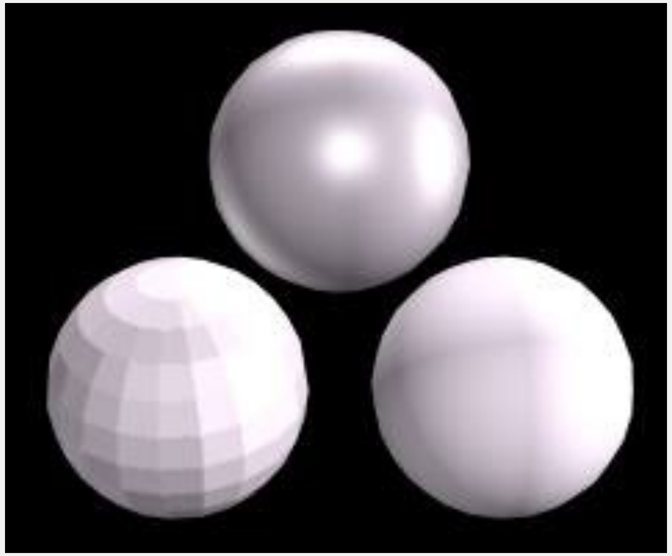
- Find vertex normals
- Interpolate vertex normal across edges
- Interpolate vertex normal of each interior point
- Calculate shade for each point





$$n_C(\alpha) = (1-\alpha)n_A + \alpha n_B$$

$$n_{in}(\alpha,\beta) = (1-\beta)n_C + \beta n_D$$



Problems with Interpolated Shading

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