Comp 410/510

Computer Graphics

Spring 2023

Shading

Why do we need shading?

- Suppose we build a model of a sphere using many polygons and then color it using a fixed color. We get something like

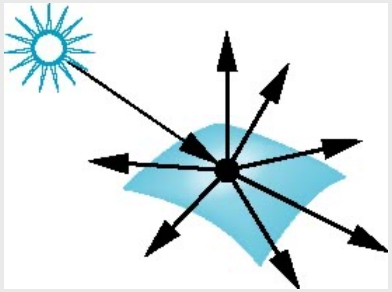
- But we rather want

Shading

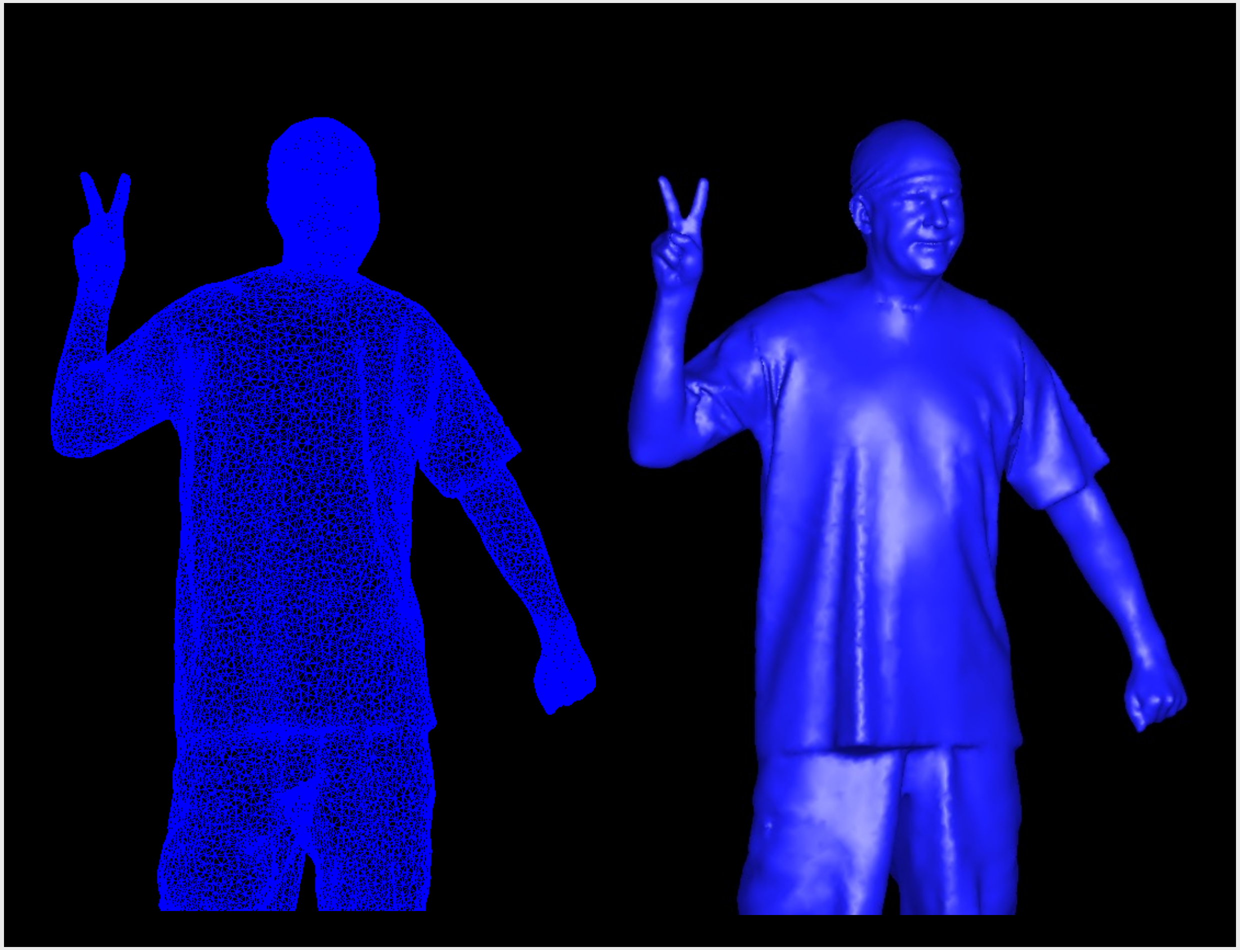
- Why does the image of a real sphere look like

?

- Light-material interactions cause each point to have a different color, referred to as *shade*.
- Need to consider
  - Light sources
  - Material properties
  - Location of viewer
  - Surface orientation

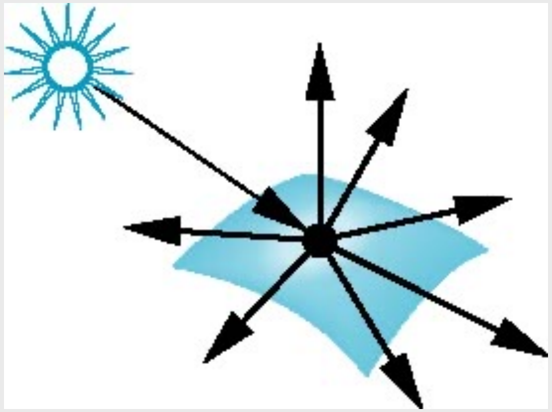


Shading

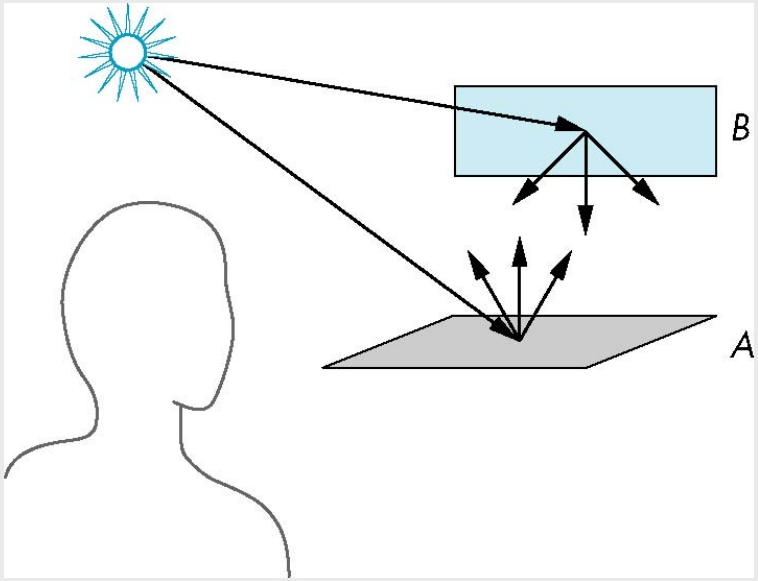


Light-Material Interaction

- Light that strikes an object is partially absorbed and partially scattered (reflected)
- The reflected light is scattered in a manner that depends on the smoothness and orientation of the surface
- The amount of light 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

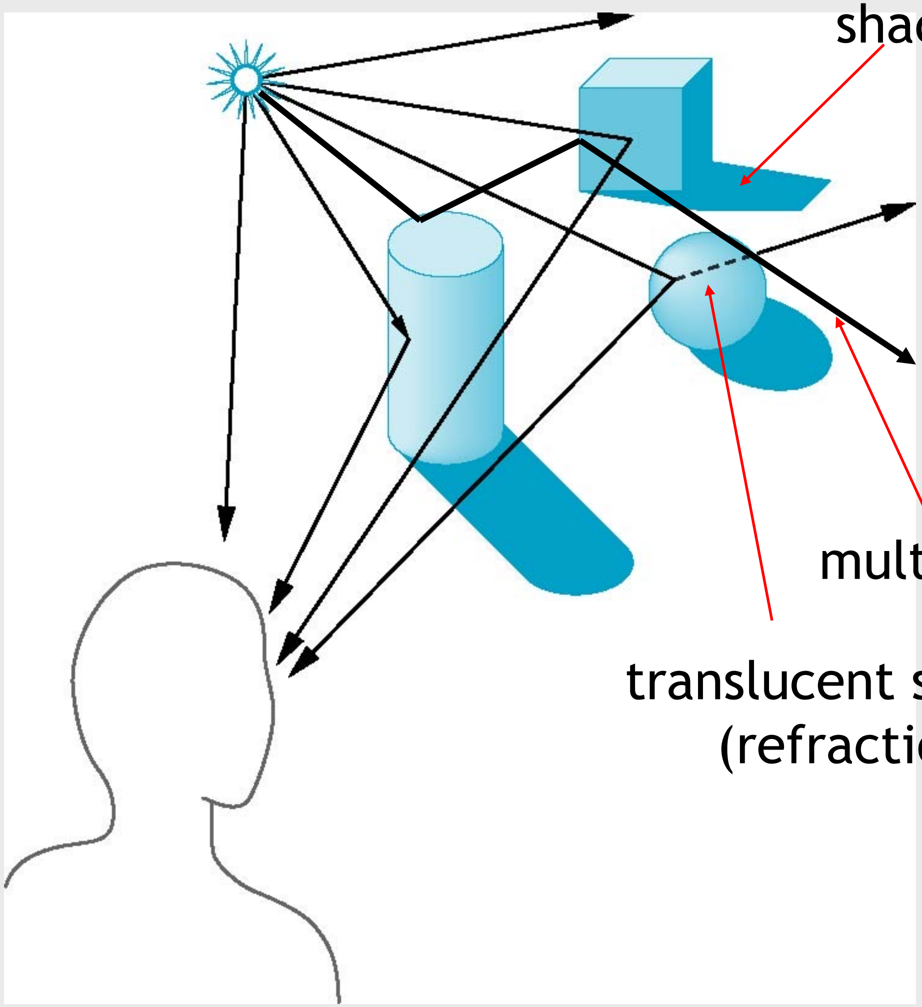


Scattering



- Light strikes A
  - Some scattered
  - Some absorbed
- Some of scattered light strikes B
  - Some scattered
  - Some absorbed
- Some of this scattered light strikes again A and so on

Global Effects



Rendering Equation

- The infinite scattering and absorption of light can be described by the **rendering equation** (see Chapter 11.4 from textbook)
- Rendering equation is global and includes
  - Shadows
  - Multiple scattering from object to object
  - Refractions
- Too complex for a practical solution
- **Ray tracing** is a special case of rendering equation for perfectly reflecting surfaces, and **radiosity technique** for perfectly diffuse surfaces.

Local vs Global Rendering

- Correct shading requires a global calculation involving all objects and light sources
  - i.e., solve rendering equation
  - which is incompatible with pipeline model that shades each polygon independently (local illumination)
- In computer graphics, especially in real time graphics, we are content if things "look right"
  - There exist many techniques for approximating global effects

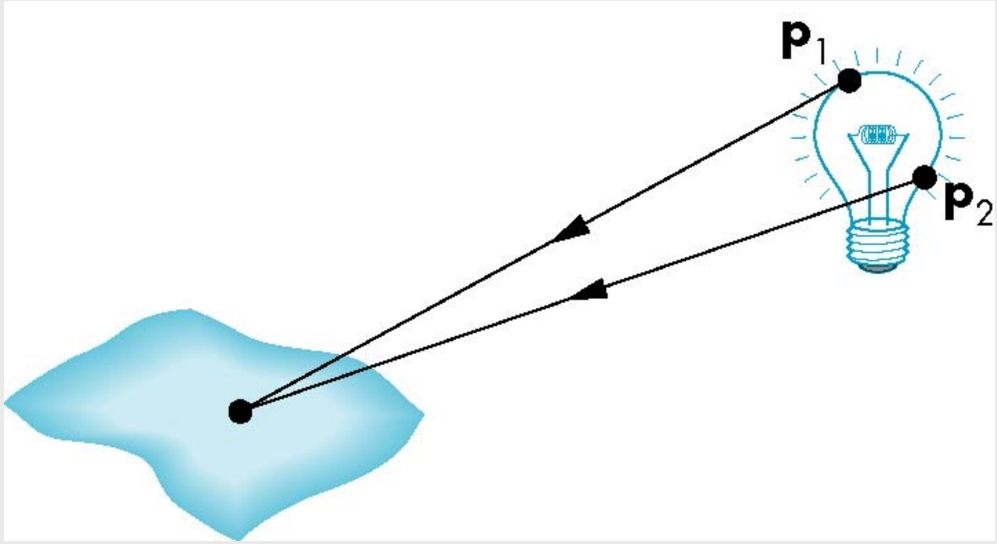
Local illumination

- So we need a local illumination model that will work with the pipeline approach:

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

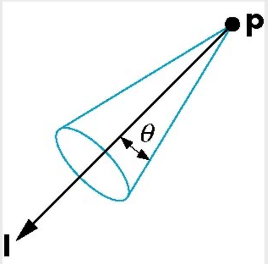
Light Sources

Realistic light sources are difficult to work with because we must integrate light coming from all points on the source



Simple Light Sources

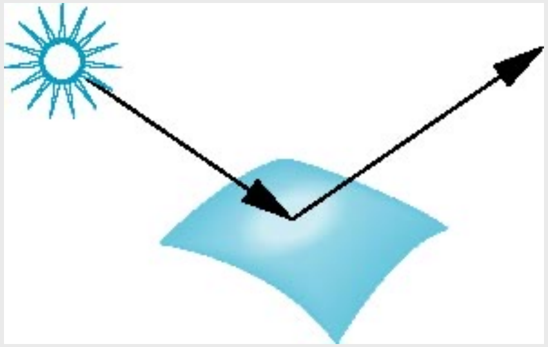
- Point source
  - Model with position and color
  - Distant source = infinite distance away (parallel)
- Spotlight
  - Restrict light from ideal point source



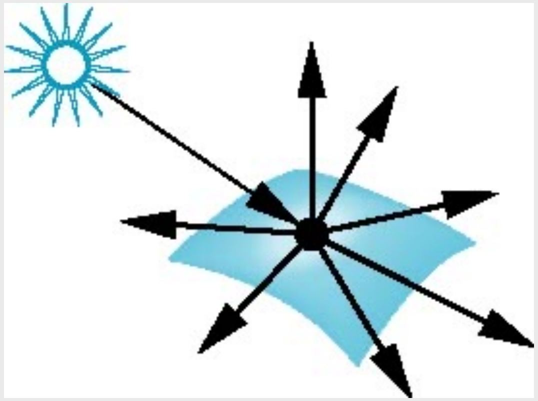
- Ambient light
  - Same amount of light everywhere in the scene
  - Can model contribution of many sources and scattering

Surface Types

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



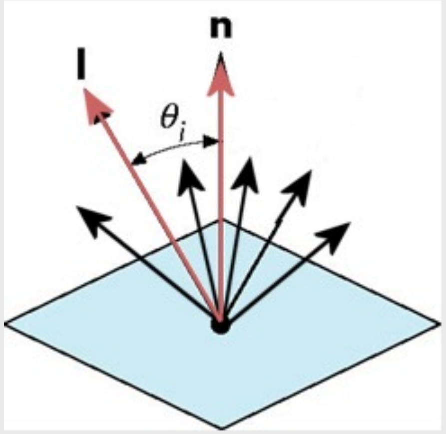
smooth surface



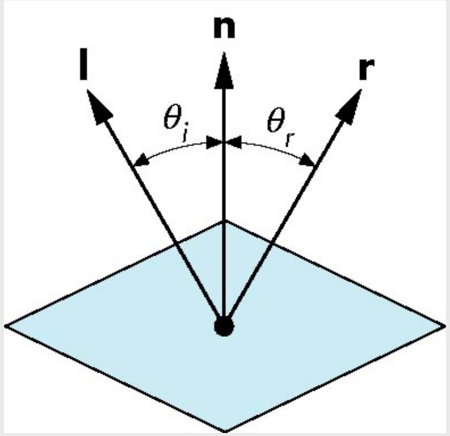
rough surface

Two extremes (Diffuse vs Specular)

- Perfectly diffuse (Lambertian) surfaces
  - Light scattered equally in all directions
  - Amount of light reflected is proportional to the vertical component of incoming light
- Perfectly specular surfaces
  - Ideal reflectors (mirror-like)
  - Angle of incidence = Angle of reflection



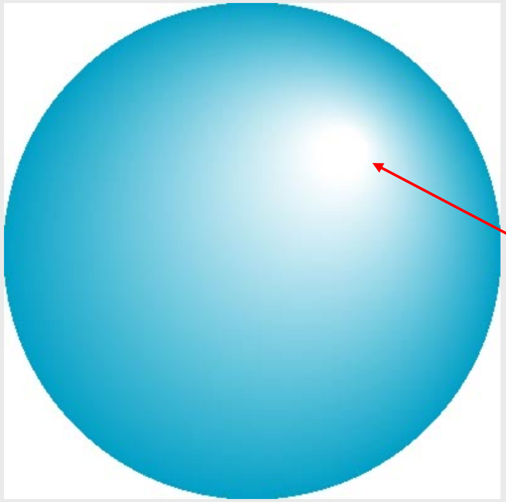
perfectly diffuse



perfectly specular

Real Surfaces

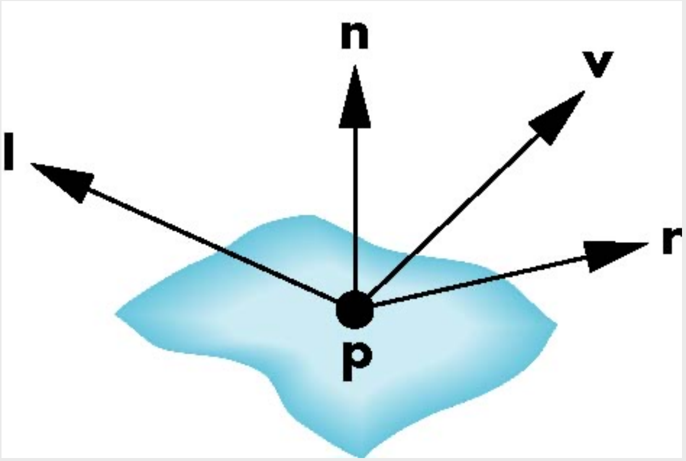
- Most surfaces are neither ideal diffusers nor perfectly specular
- Real surfaces show specular highlights due to incoming light being reflected in directions concentrated close to the direction of a perfect reflection



specular highlight

Phong Model

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

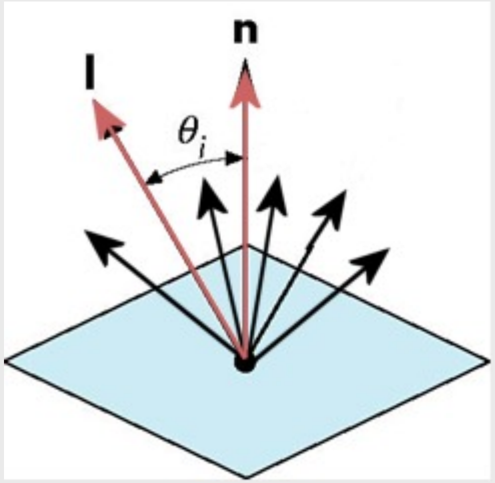


Perfectly Diffuse Surfaces

- Light scattered equally in all directions
- Amount of light reflected is proportional to the perpendicular component of incoming light
- Normal (perpendicular) vector $\boldsymbol{n}$ is determined by local orientation
  - reflected light ~ $\cos\theta_i$
  - $\cos\theta_i = \boldsymbol{l} \cdot \boldsymbol{n}$ (if vectors are normalized to be unity)
  - There are also three coefficients, k_{rd}, k_{bd}, k_{gd}, that show how much of each color component is reflected

$$I_d = k_d L_d \, \boldsymbol{l} \cdot \boldsymbol{n}$$

$$(0 \le k_d \le 1)$$



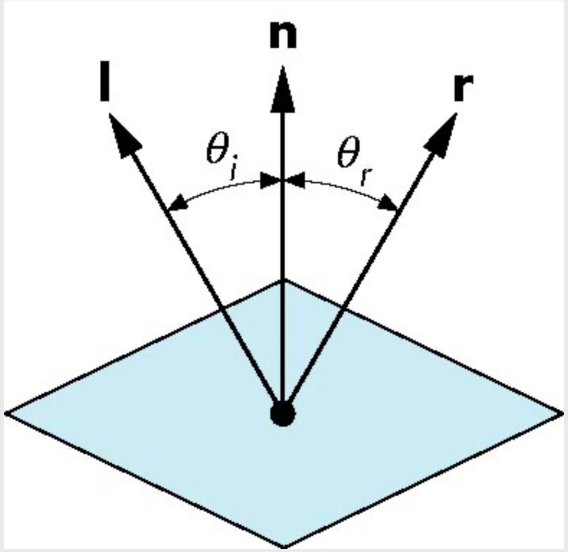
I_d : diffuse component of the shade

Perfectly Specular Surfaces

- Ideal reflectors
- Angle of incidence = Angle of reflection
- The three vectors $\boldsymbol{n}$, $\boldsymbol{l}$ and $\boldsymbol{r}$ must be coplanar:

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

(see page 275 from textbook for derivation)



Modeling Specular Reflections

- Phong proposed using a term that drops off as the angle between the viewer and the ideal reflection increases

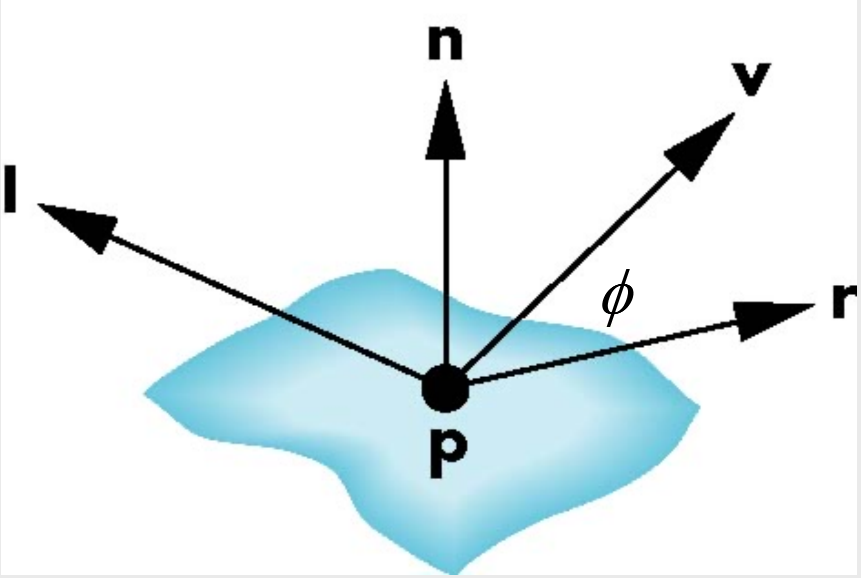
$$I_s = k_s \, L_s \cos^{\alpha}\phi$$

reflected intensity

reflection coef

incoming intensity

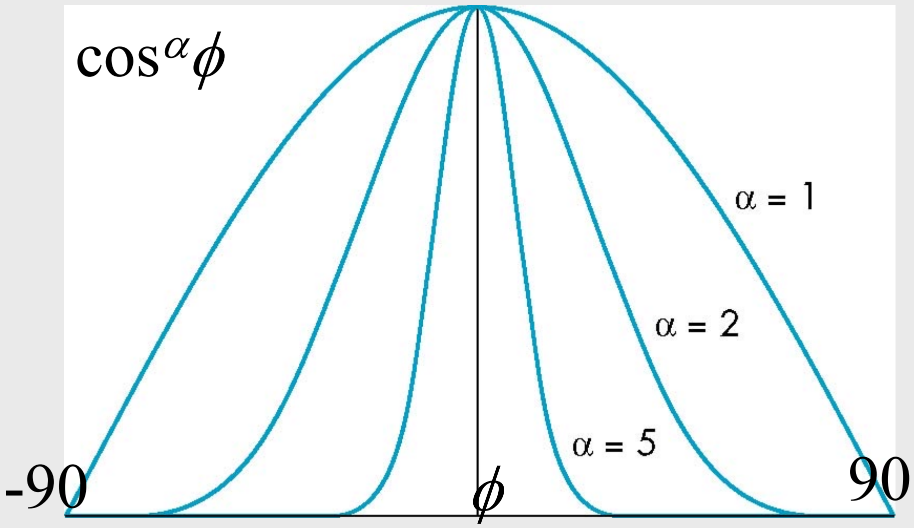
shininess coef



I_s : specular component of the shade

The Shininess Coefficient

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



$$I_s = k_s L_s \cos^{\alpha}\phi$$

Ambient Light

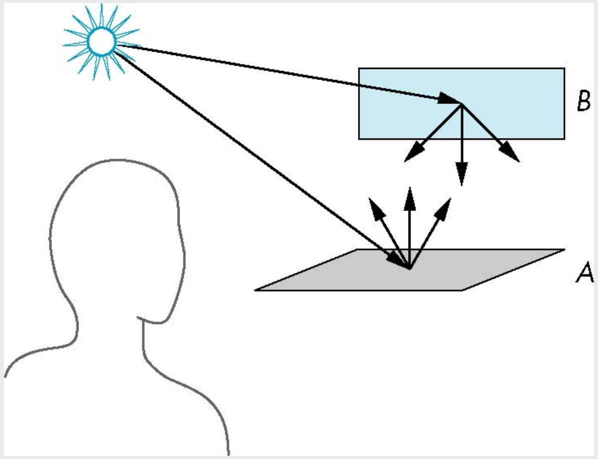
- Ambient light is the result of multiple interactions between light sources and the objects in the environment
- Amount and color of ambient light depend on both color of the light(s) and material properties of the object
- Add $k_a L_a$ to diffuse and specular terms

reflection coef

intensity of ambient light

Distance Terms

- The light from a point source that reaches a surface is inversely proportional to the square of the distance between them.
- We can 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.



Light Sources

- In the Phong Model, we add the contributions from each light source.
- Each light source has separate diffuse, specular, and ambient terms to allow for maximum flexibility, even though this form does not have a physical justification.
- Separate red, green and blue components.
- Hence, 9 coefficients for each point light source
  - $L_{dr}, L_{dg}, L_{db}, L_{sr}, L_{sg}, L_{sb}, L_{ar}, L_{ag}, L_{ab}$

Material Properties

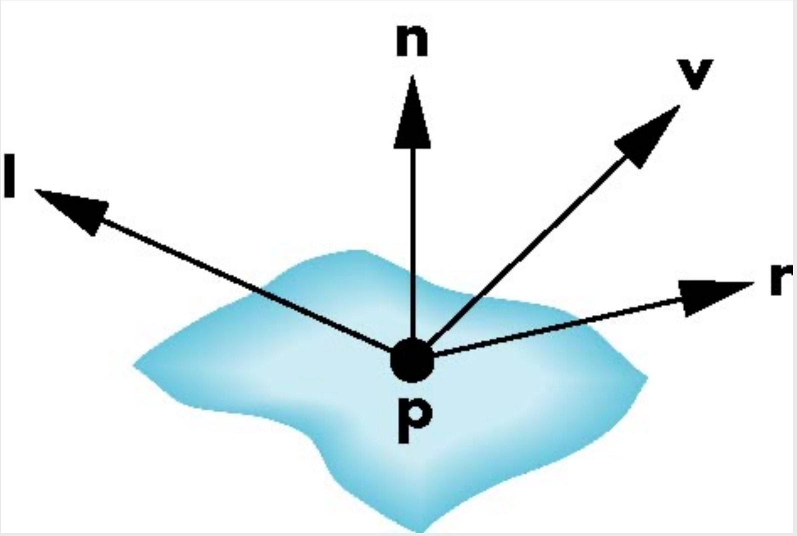
- Material properties match light source properties
  - Nine reflection coefficients
    - $k_{dr}, k_{dg}, k_{db}, k_{sr}, k_{sg}, k_{sb}, k_{ar}, k_{ag}, k_{ab}$
  - Shininess coefficient α

Adding up the Components

For each light source and each color component, the Phong model can then be written as

$$I = \frac{1}{a + bd + cd^2}\left(k_d L_d \, \boldsymbol{l} \cdot \boldsymbol{n} + k_s L_s \, (\boldsymbol{v} \cdot \boldsymbol{r})^{\alpha} \right) + k_a L_a$$

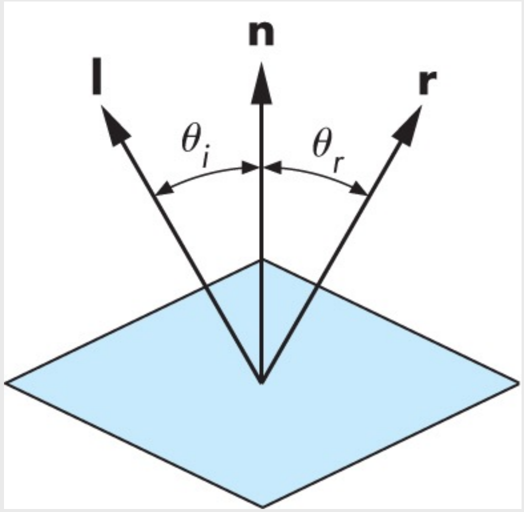
For each color component we add contributions from all light sources.



Computation of Vectors

$$I = \frac{1}{a+bd+cd^2}\left(k_d L_d \, \boldsymbol{l} \cdot \boldsymbol{n} + k_s L_s \, (\boldsymbol{v} \cdot \boldsymbol{r})^{\alpha} \right) + k_a L_a$$

- $\boldsymbol{l}$ and $\boldsymbol{v}$ are specified by the application
- Can compute $\boldsymbol{r}$ from $\boldsymbol{l}$ and $\boldsymbol{n}$
- But how to compute $\boldsymbol{n}$?
- OpenGL leaves computation of $\boldsymbol{n}$ to application

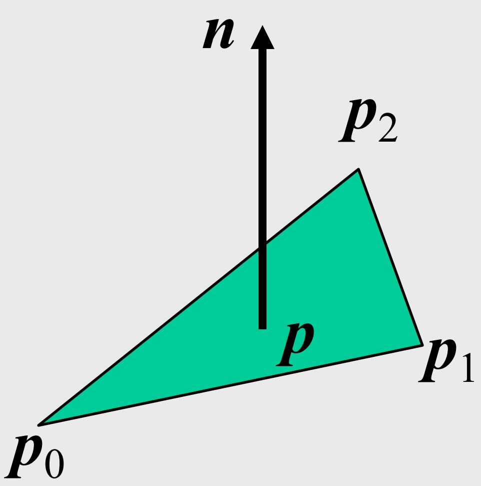


All vectors to be **unit** length

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

Computation of Normal for Triangles

- Right-hand rule determines outward face
- Normal: $\boldsymbol{n} = (\boldsymbol{p}_1 - \boldsymbol{p}_0) \times (\boldsymbol{p}_2 - \boldsymbol{p}_0)$
- Normalize by $\boldsymbol{n} \leftarrow \boldsymbol{n} / |\boldsymbol{n}|$



Modified Phong Model

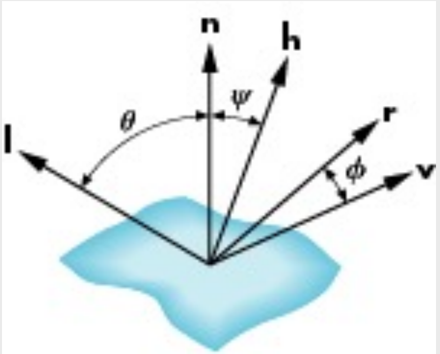
$$I = \left(k_d L_d \, \boldsymbol{l} \cdot \boldsymbol{n} + k_s L_s \, (\boldsymbol{v} \cdot \boldsymbol{r})^{\alpha} \right) + k_a L_a$$

- The specular term in the Phong model above is problematic because it requires for each vertex the calculation of a new reflection vector $\boldsymbol{r}$ (along with view vector $\boldsymbol{v}$)
- *Blinn* suggested an approximation using the halfway vector that is more efficient

The Halfway Vector

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

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



$$I = (\, k_d L_d \, \boldsymbol{l} \cdot \boldsymbol{n} \; + k_s L_s \boxed{(\boldsymbol{n} \cdot \boldsymbol{h})^{\beta}}) + k_a L_a$$

Hence we replace $(\boldsymbol{v} \cdot \boldsymbol{r})^{\alpha}$ by $(\boldsymbol{n} \cdot \boldsymbol{h})^{\beta}$

Note that $\boldsymbol{n} \cdot \boldsymbol{h}$ depends on the angle between normal vector $\boldsymbol{n}$ and halfway vector $\boldsymbol{h}$

Using the halfway vector

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

$$I = (k_d L_d \, \boldsymbol{l} \cdot \boldsymbol{n} + k_s L_s (\boldsymbol{n} \cdot \boldsymbol{h})^{\beta}) + k_a L_a$$

- β is chosen so as to match shininess α
- Resulting model is known as the modified Phong or Blinn lighting model

OpenGL Example

Differences in these teapots are only due to the parameters in the modified Phong model