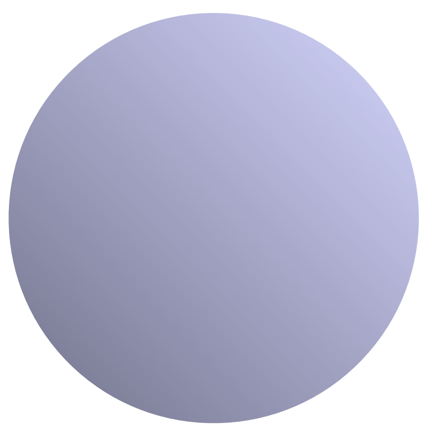
CS230 : Computer Graphics
Lecture 3: Lighting and Shading

Tamar Shinar
Computer Science & Engineering
UC Riverside

Why we need shading

- Suppose we build a model of a sphere using many polygons and color each the same color. We get something like

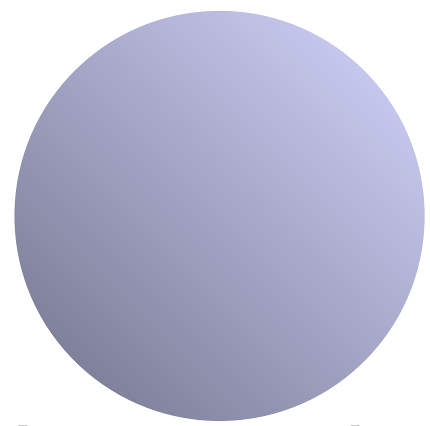
- But we want



The more realistically lit sphere has gradations in its color that give us a sense of its three-dimensionality

Shading

- Why does the image of a real sphere look like

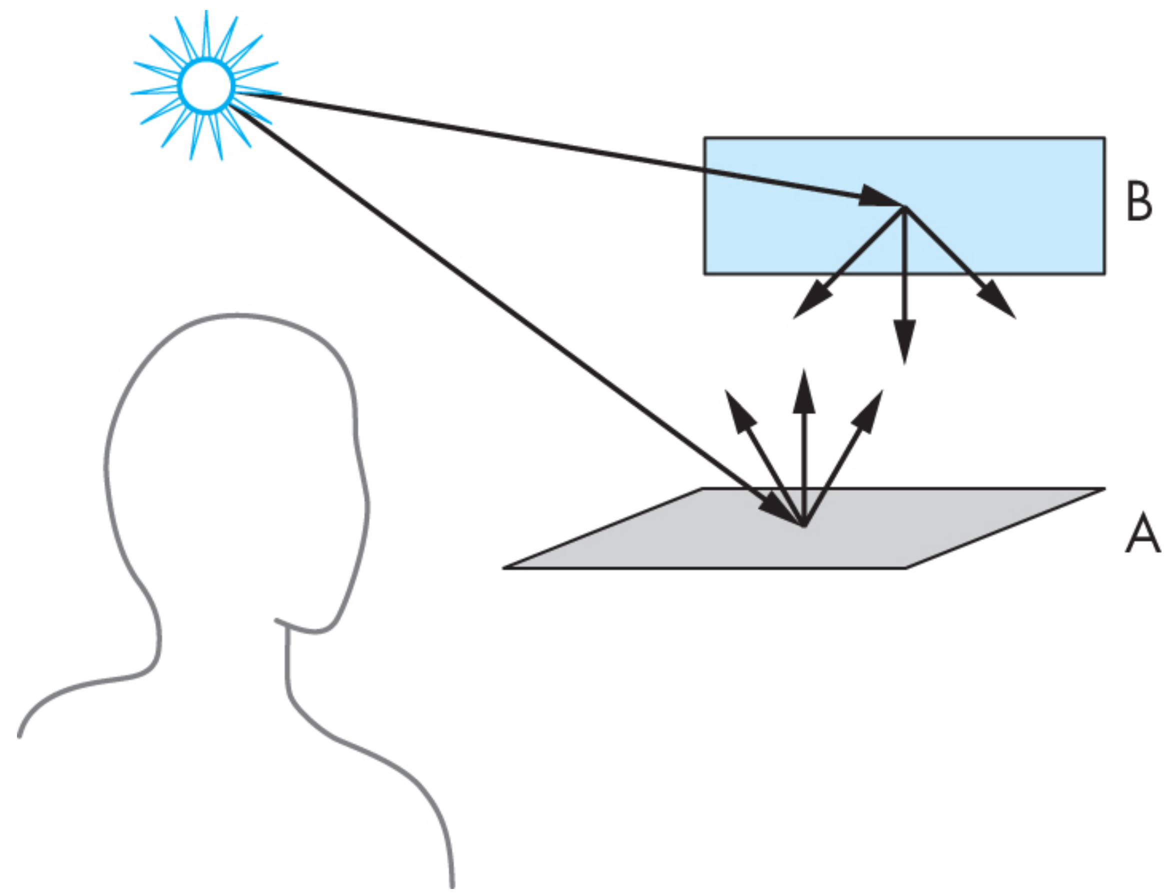


- Light-material interactions cause each point to have a different color or shade
- Need to consider
  - Light sources
  - Material properties
  - Location of viewer
  - Surface orientation (normal)

We are going to develop a **local** lighting model by which we can shade a point independently of the other surfaces in the scene
our **goal** is to use this in our first ray tracer and later to add this to a fast graphics pipeline architecture

General rendering

- The most general approach is based on physics - using principles such as conservation of energy
- a surface either **emits** light (e.g., light bulb) or **reflects** light for other illumination sources, or both
- light interaction with materials is **recursive**
- the **rendering equation** is an integral equation describing the limit of this recursive process



[Angel and Shreiner]

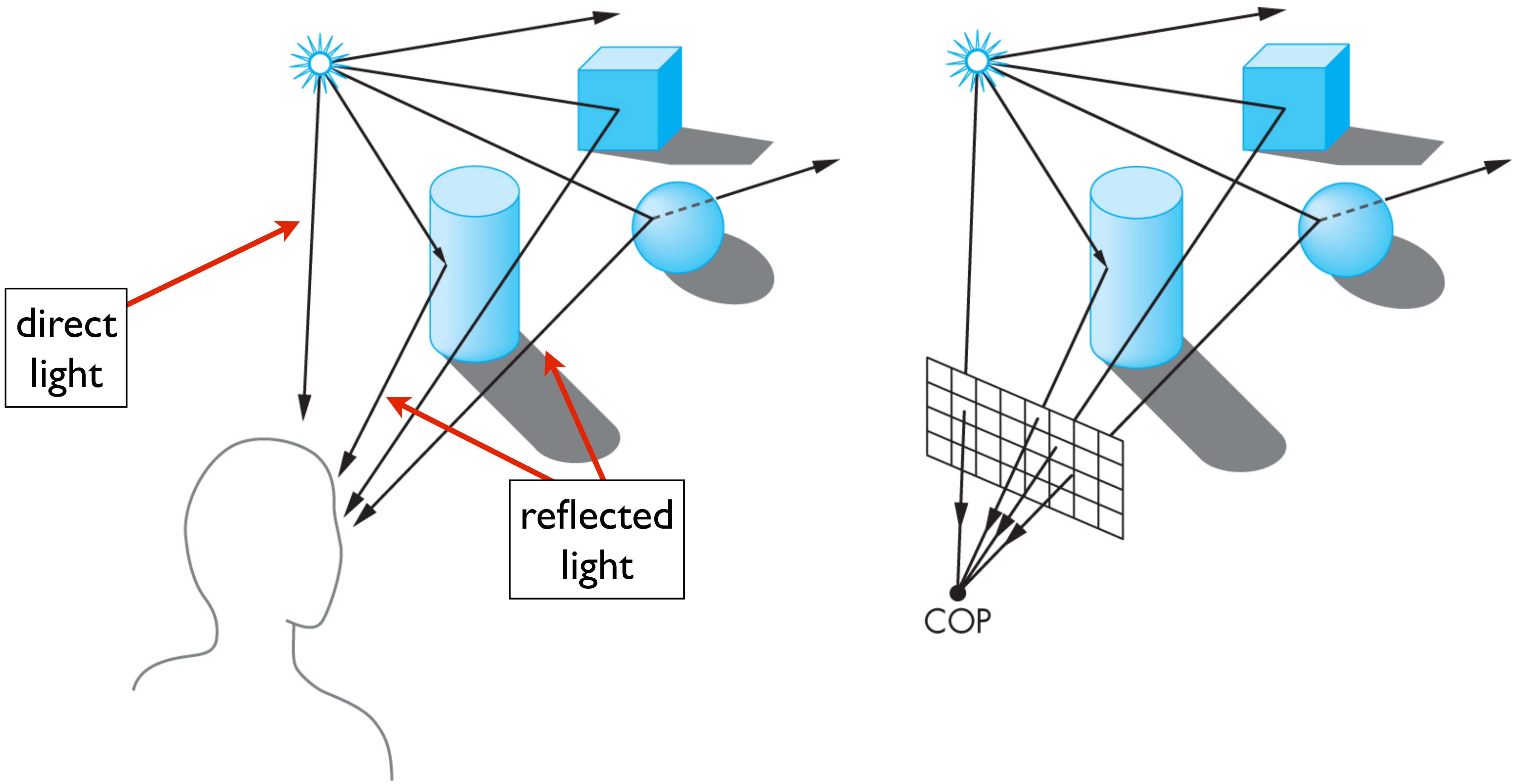
http://en.wikipedia.org/wiki/Rendering_equation

Fast local shading models

- the rendering equation can’t be solved analytically
- numerical methods aren’t fast enough for real-time
- we’ll use a **local** model where shade at a point is independent of other surfaces
- use **Phong reflection model**
    - shading based on local light-material interactions

some approximations to the rendering equation include **radiosity** and **ray tracing**, but they are still not as fast as the local model in the pipeline architecture

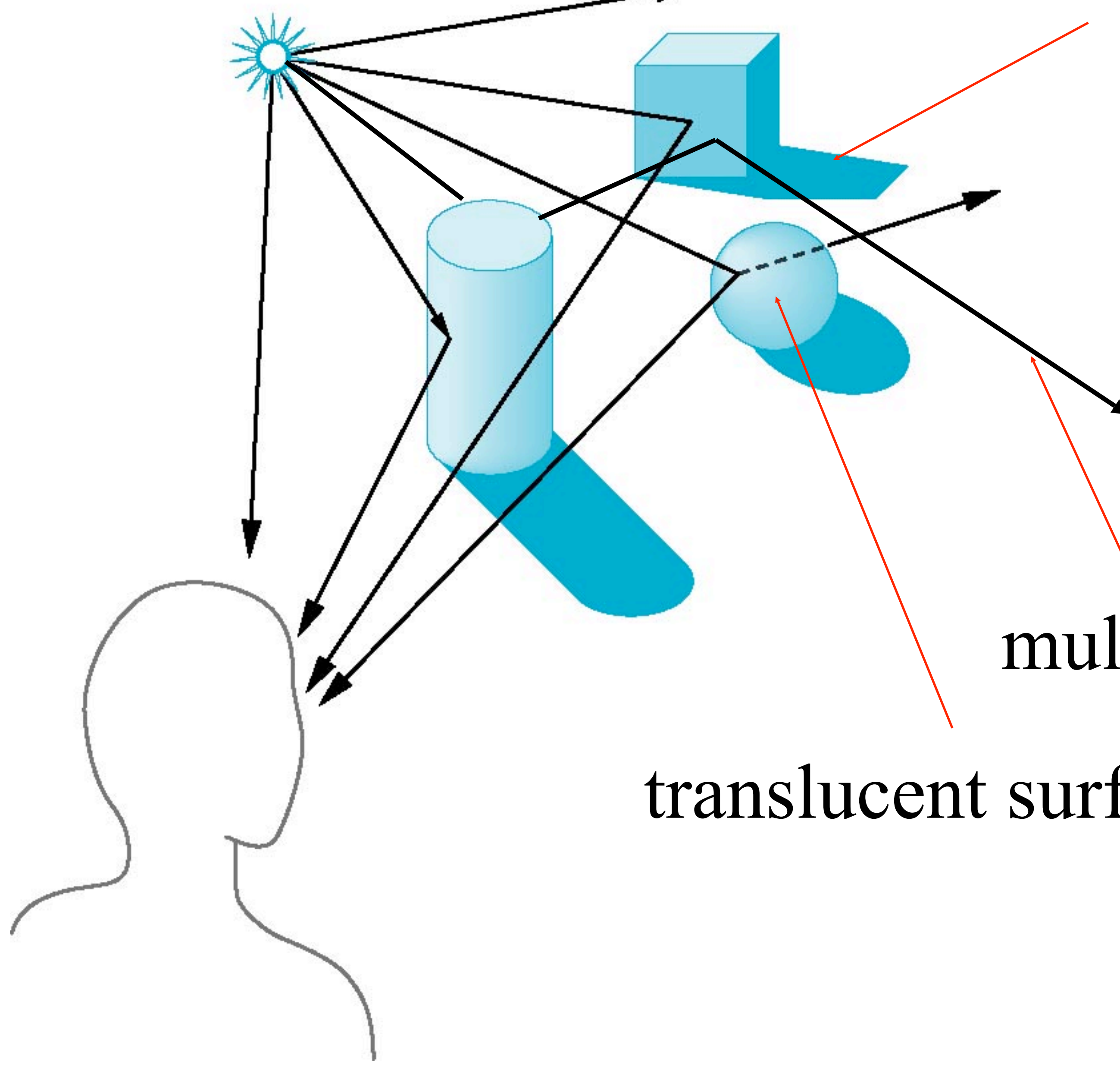
Local shading model



[Angel and Shreiner]

direct light is the color of the light source
reflected light is the color of the light reflected from the object surface
for rendering, color of light source and reflected light determines the colors of pixels in the frame buffer
only need to consider the rays that leave the source and reach the viewer's eye

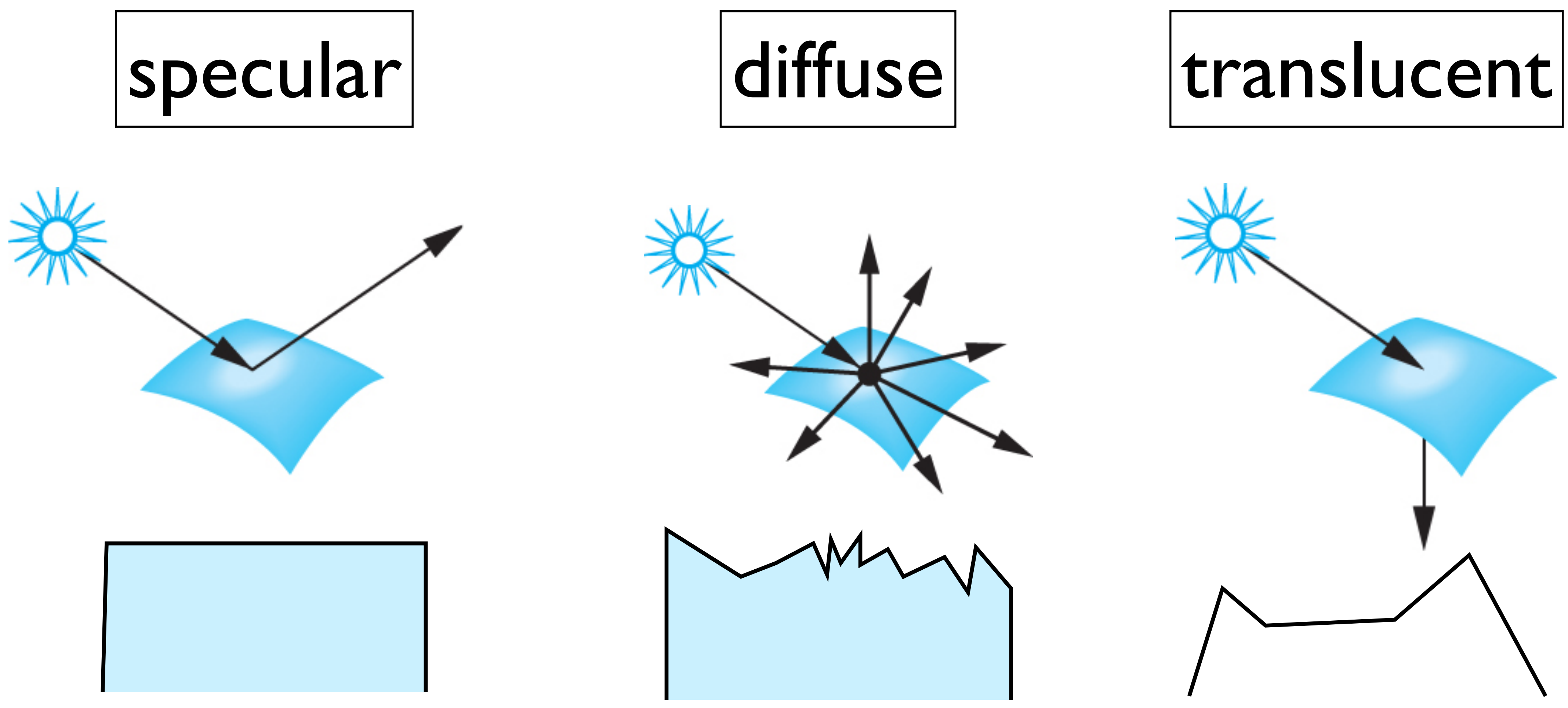
Global Effects



[Angel and Shreiner]

Light-material interactions

at a surface, light is absorbed, reflected, or transmitted



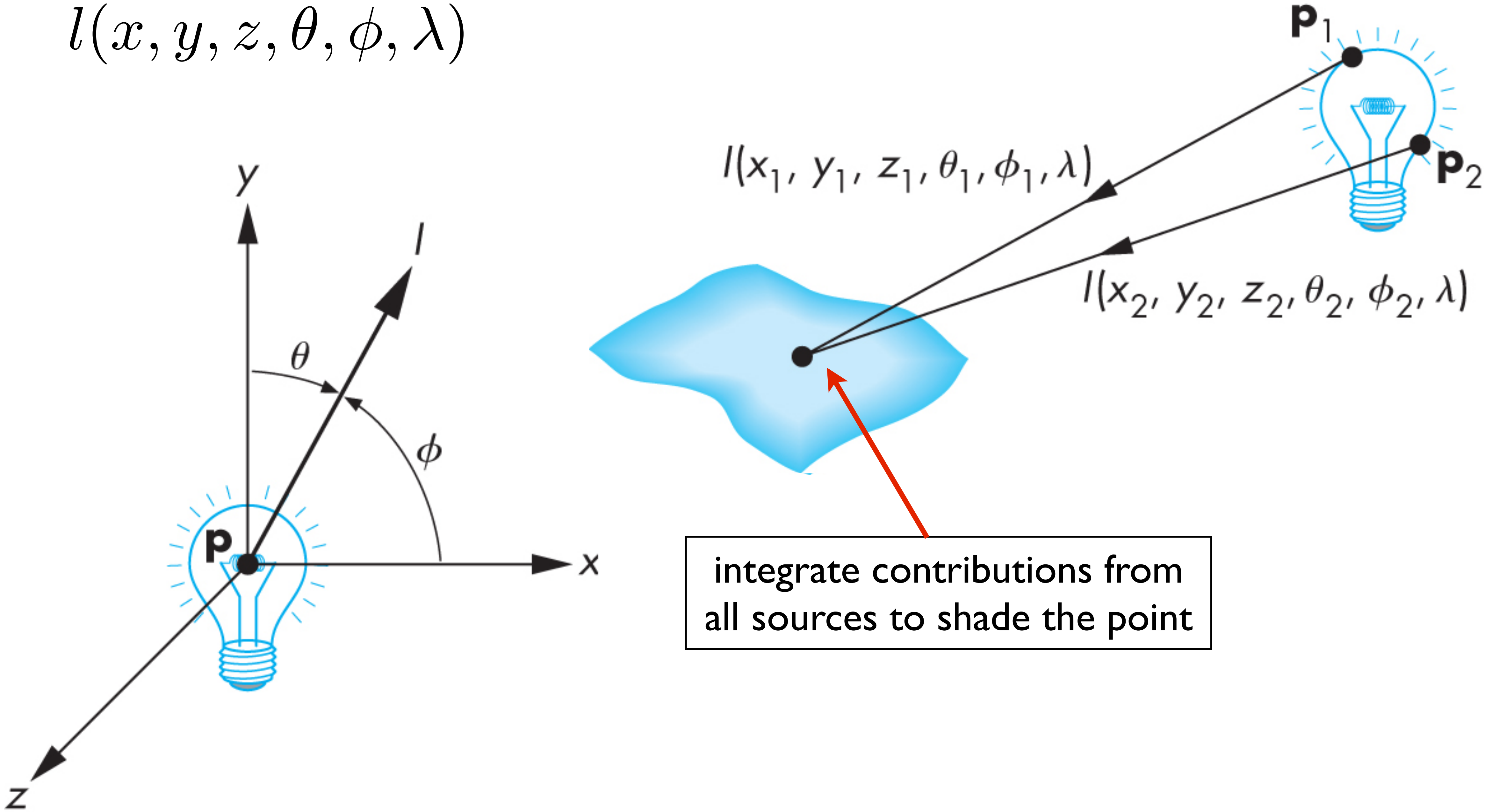
[Angel and Shreiner]

specular: shiny, smooth surface. light scattered in narrow range close to angle of reflection
e.g., mirror is perfectly specular
diffuse: matte, rough surface. light scattered in all directions
translucent: allows some light to pass through object. refraction: e.g., glass or water

General light source

Illumination function:

$$l(x, y, z, \theta, \phi, \lambda)$$

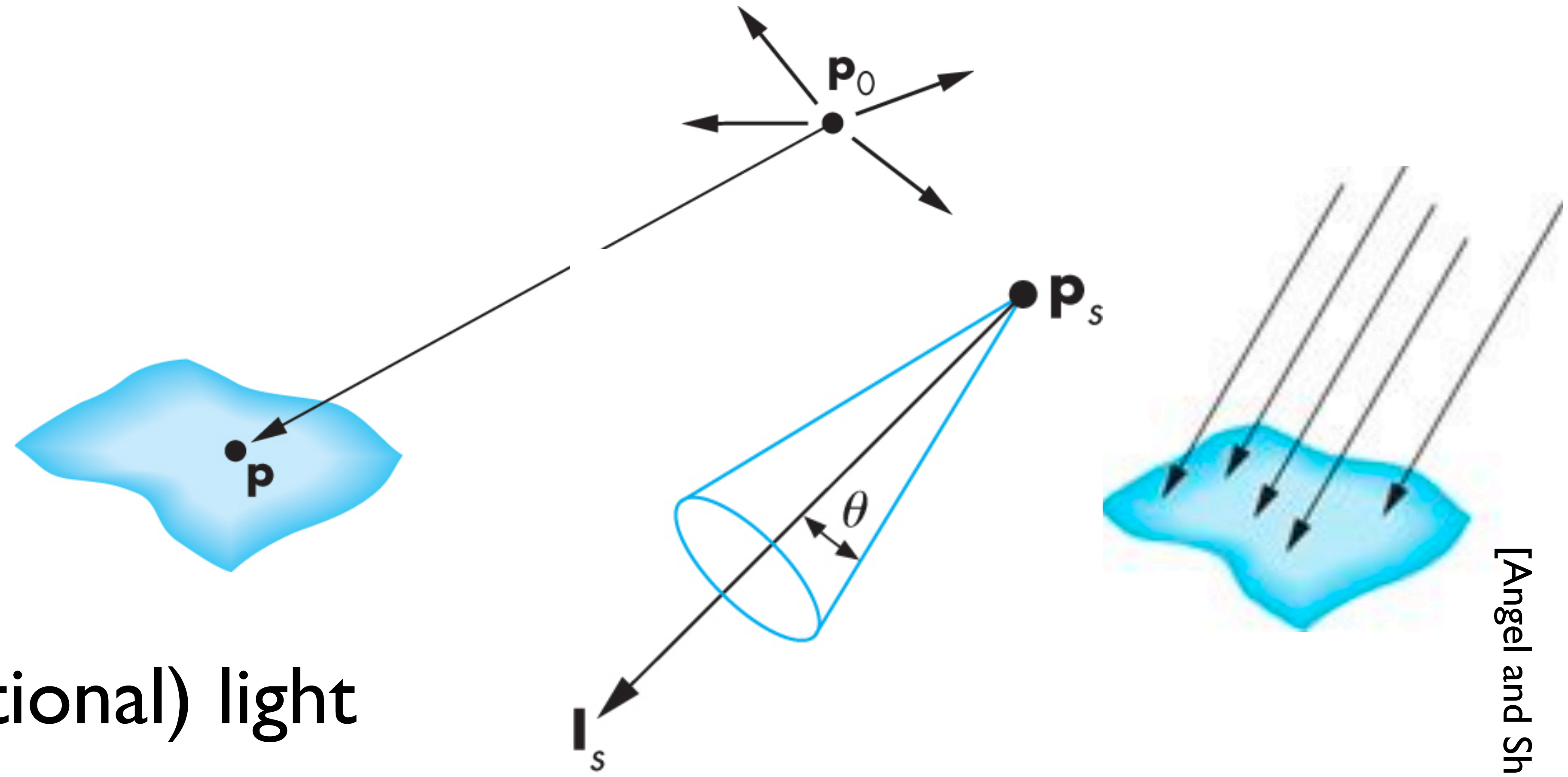


[Angel and Shreiner]

\vec{x} = (x,y,z)
\vec{omega} = theta, phi

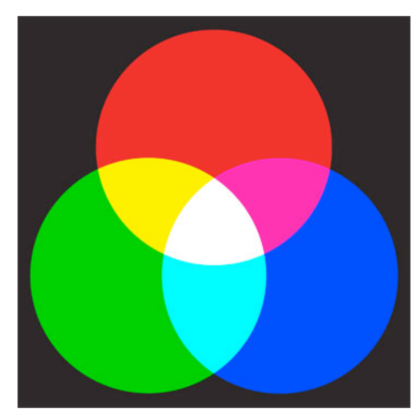
Idealized light sources

- Ambient light
- Point light
- Spotlight
- distant (directional) light



[Angel and Shreiner]

luminance: $\mathbf{L} = \begin{bmatrix} L_r \\ L_g \\ L_b \end{bmatrix}$



source will be described through three component intensity or **luminance**
decompose into red, green, blue channels
e.g., use the red component of source to calculate red component of image
use a single scalar equations – each equation applied independently to each channel

Ambient light source

- achieve a uniform light level
- no black shadows
- ambient light intensity at each point in the scene

$$\mathbf{L}_a = \left[\begin{array}{c} L_{ar} \\ L_{ag} \\ L_{ab} \end{array} \right]$$

$$L_a$$

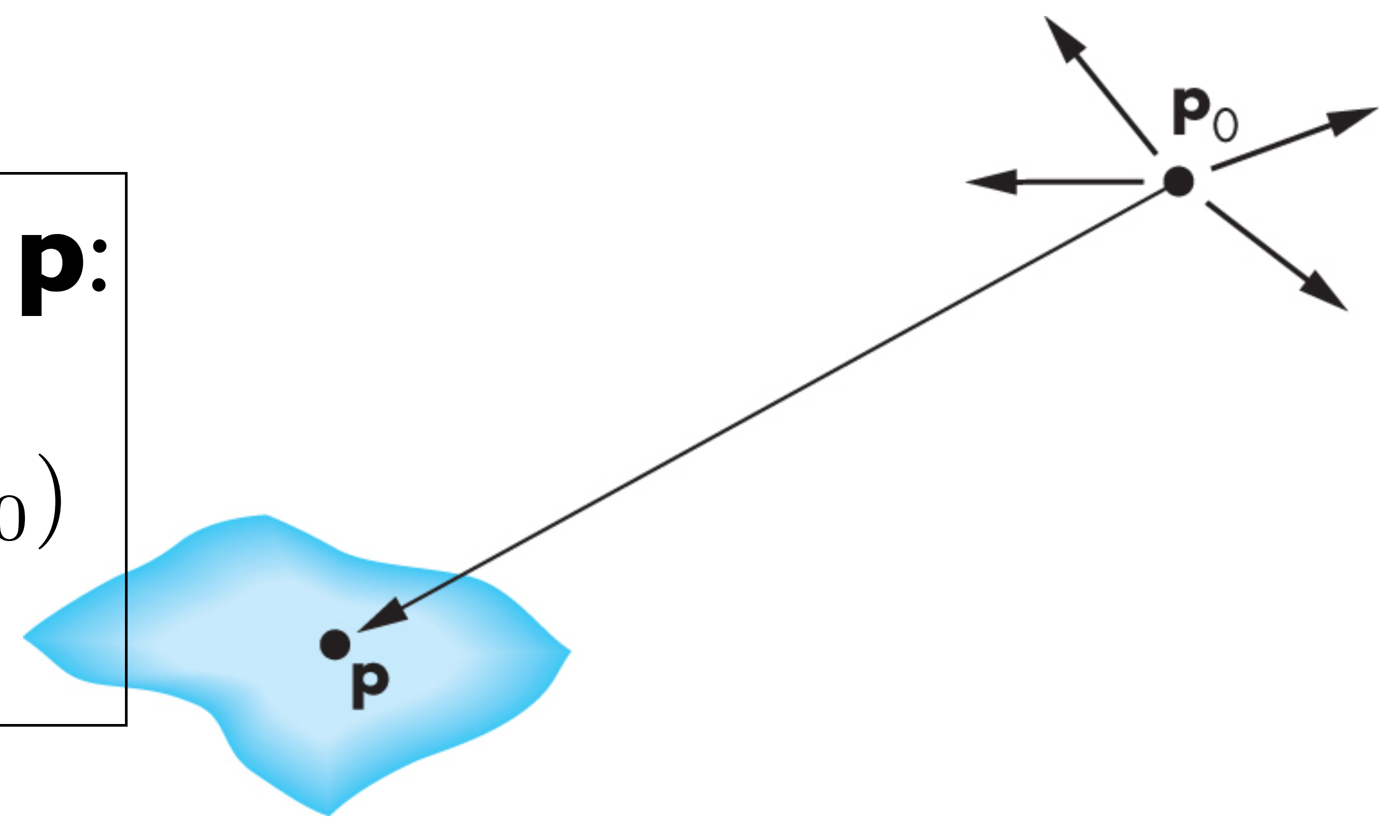
use scalar I_a to denote any component of \vec{I}_a
ambient light is the same everywhere
but different surfaces will **reflect** it differently

Point light source

$$\mathbf{L}(\mathbf{p}_0) = \begin{bmatrix} L_r(\mathbf{p}_0) \\ L_g(\mathbf{p}_0) \\ L_b(\mathbf{p}_0) \end{bmatrix} \qquad L(\mathbf{p}_0)$$

illumination intensity at **p**:

$$l(\mathbf{p}, \mathbf{p}_0) = \frac{1}{|\mathbf{p} - \mathbf{p}_0|^2} \mathbf{L}(\mathbf{p}_0)$$



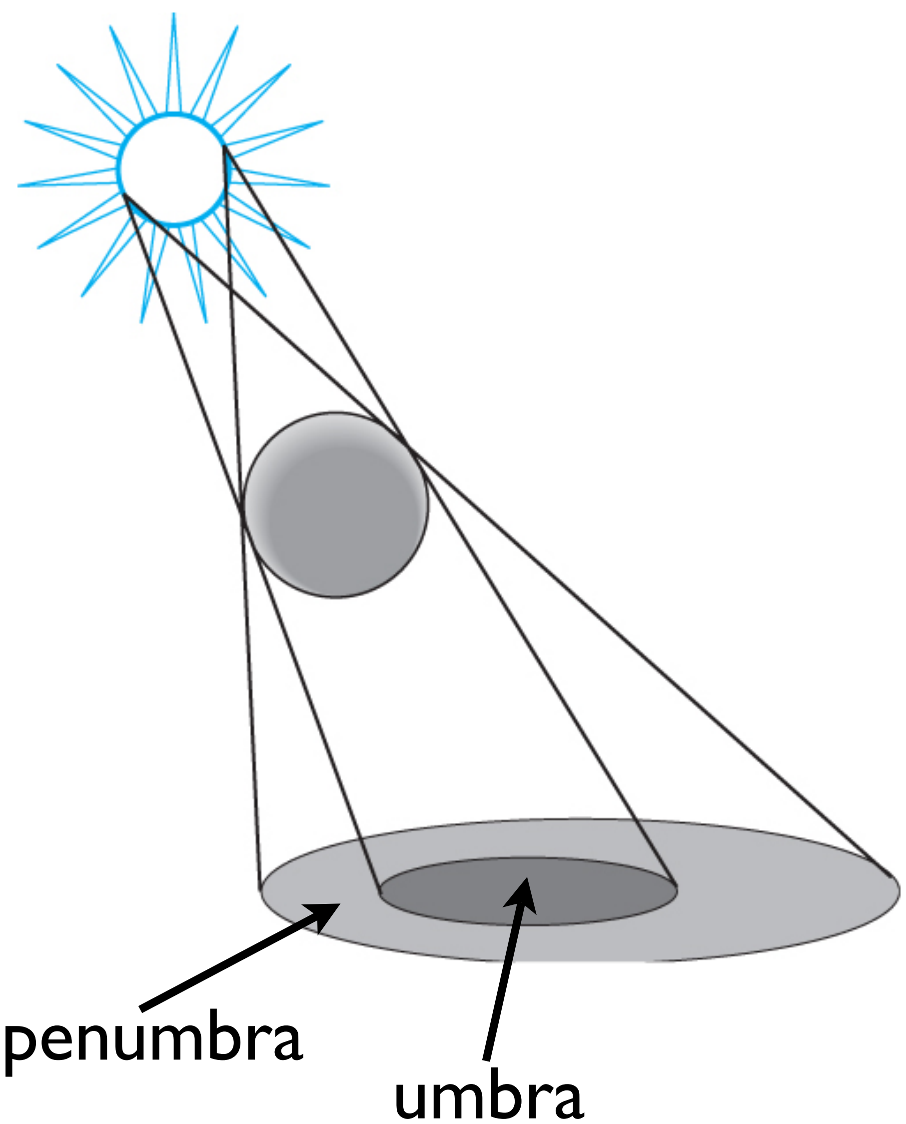
[Angel and Shreiner]

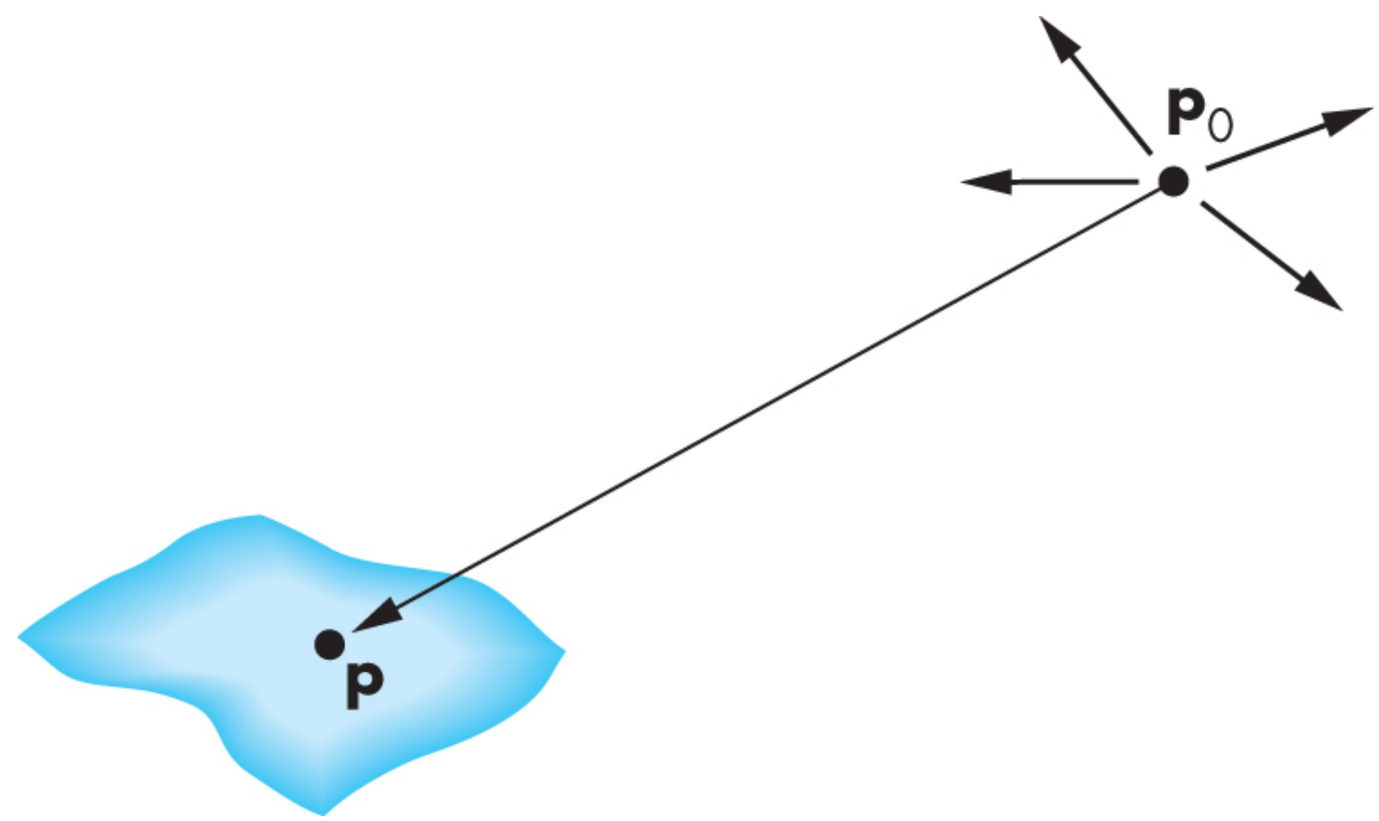
- use scalar I(\vec{p}_0) to denote any of three components
- points sources alone aren't too realistic looking -- tend to be high contrast
- most real-world scenes have large light sources
- add ambient light to mitigate high contrast

Point light source

Most real-world scenes have large light sources

Point light sources alone aren't too realistic
- add ambient light to mitigate high contrast



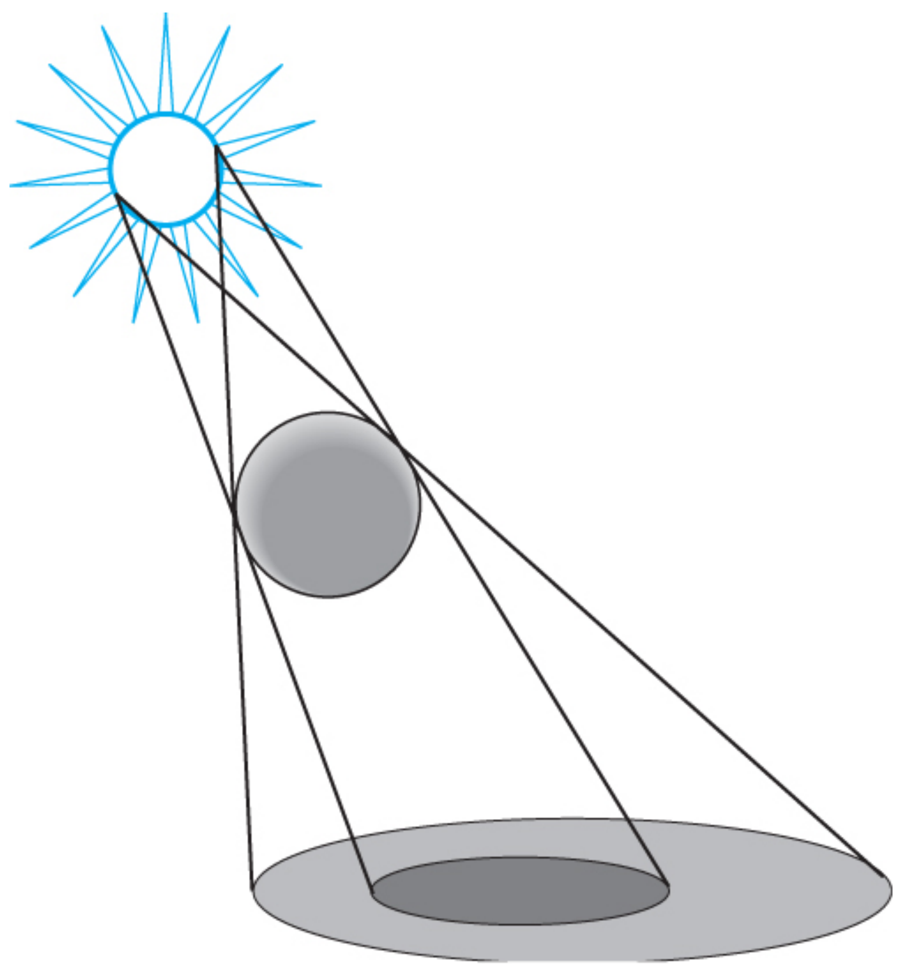


[Angel and Shreiner]

– **umbra** is fully in shadow, **penumbra** is partially in shadow

Point light source

Most real-world scenes have large light sources



Point light sources alone aren't too realistic
- drop off intensity more slowly

$$l(\mathbf{p}, \mathbf{p}_0) = \frac{1}{d^2}\mathbf{L}(\mathbf{p}_0)$$

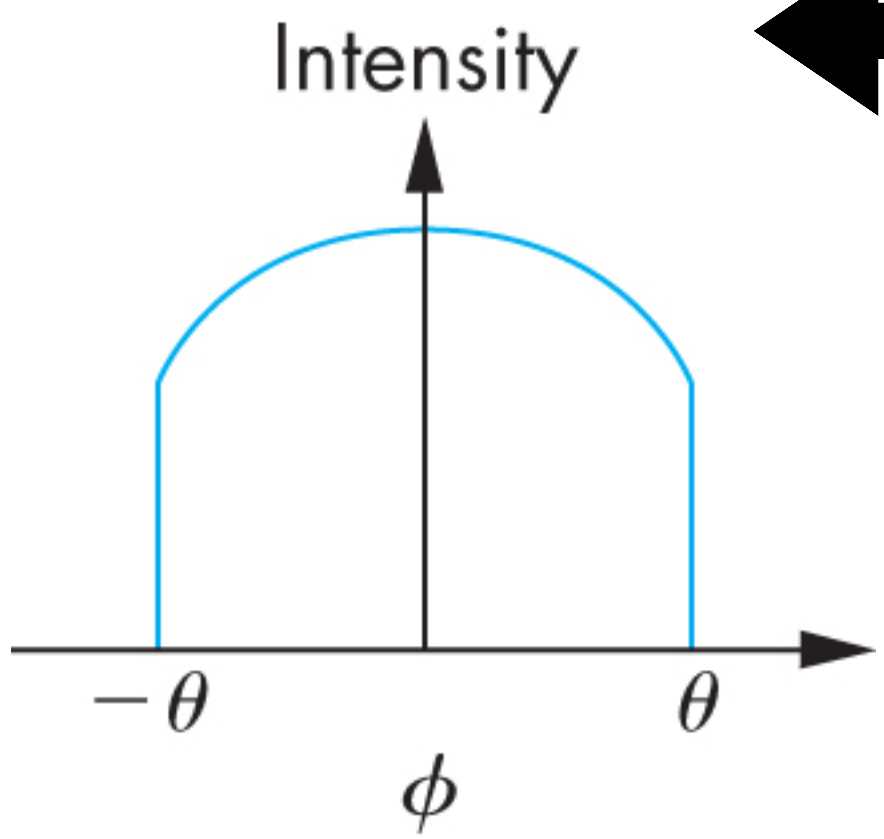
$$\Downarrow$$

$$l(\mathbf{p}, \mathbf{p}_0) = \frac{1}{a + bd + cd^2}\mathbf{L}(\mathbf{p}_0)$$

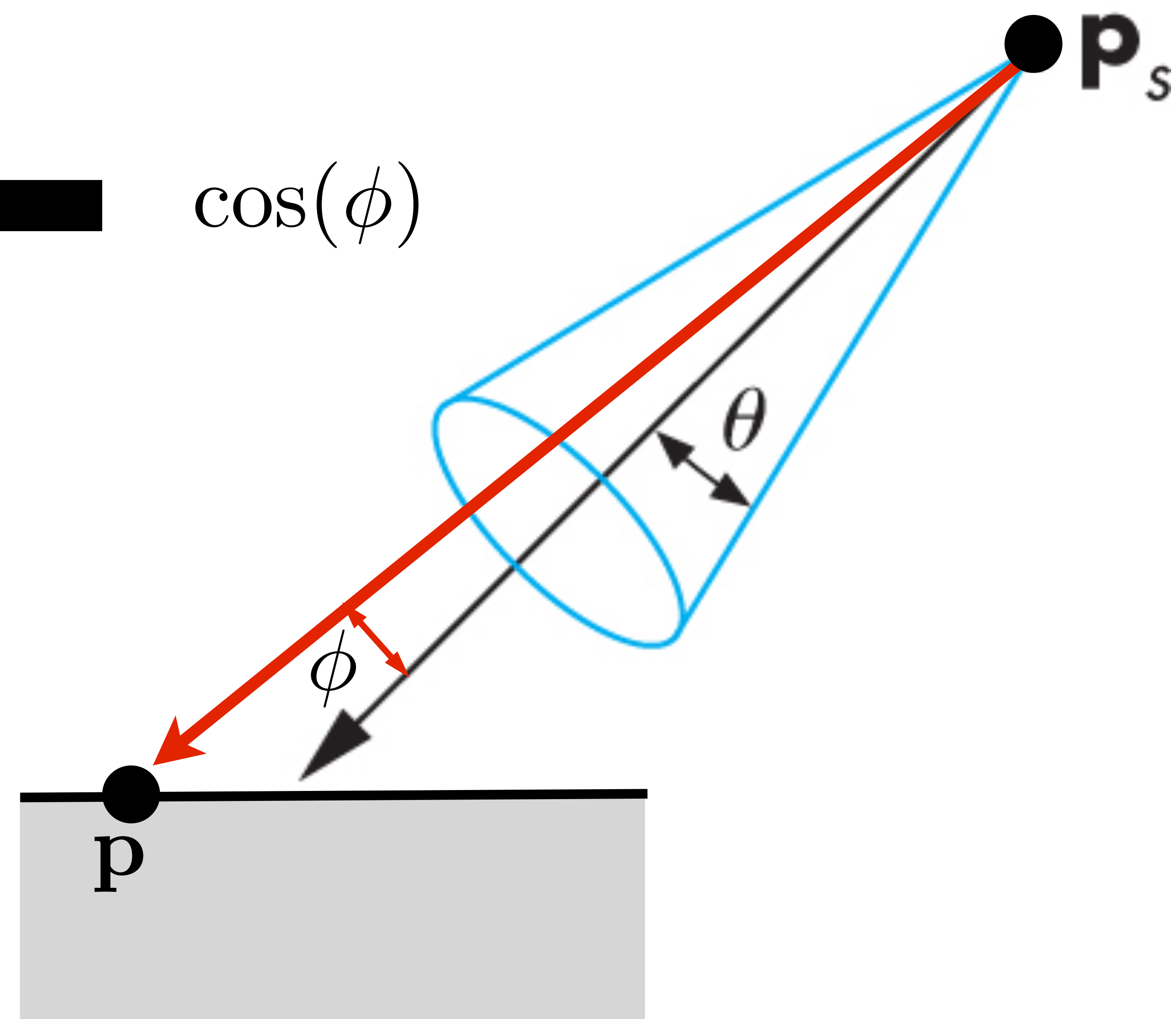
[Angel and Shreiner]

In practice, we also replace the 1/d^2 term by something that falls off more slowly

Spotlights

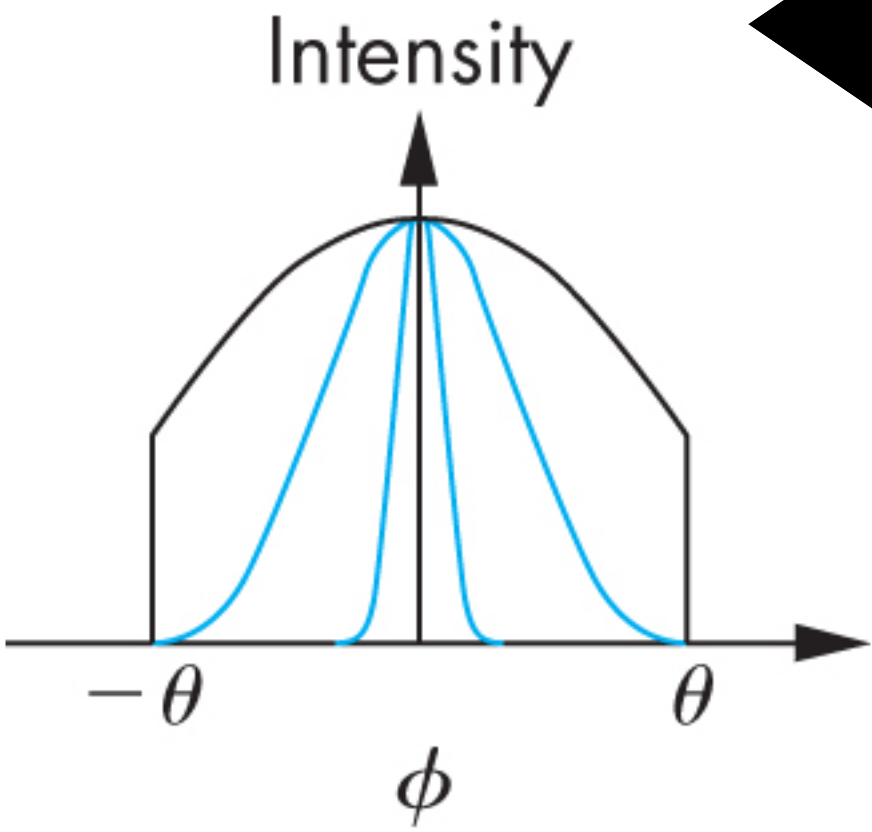


$\cos(\phi)$

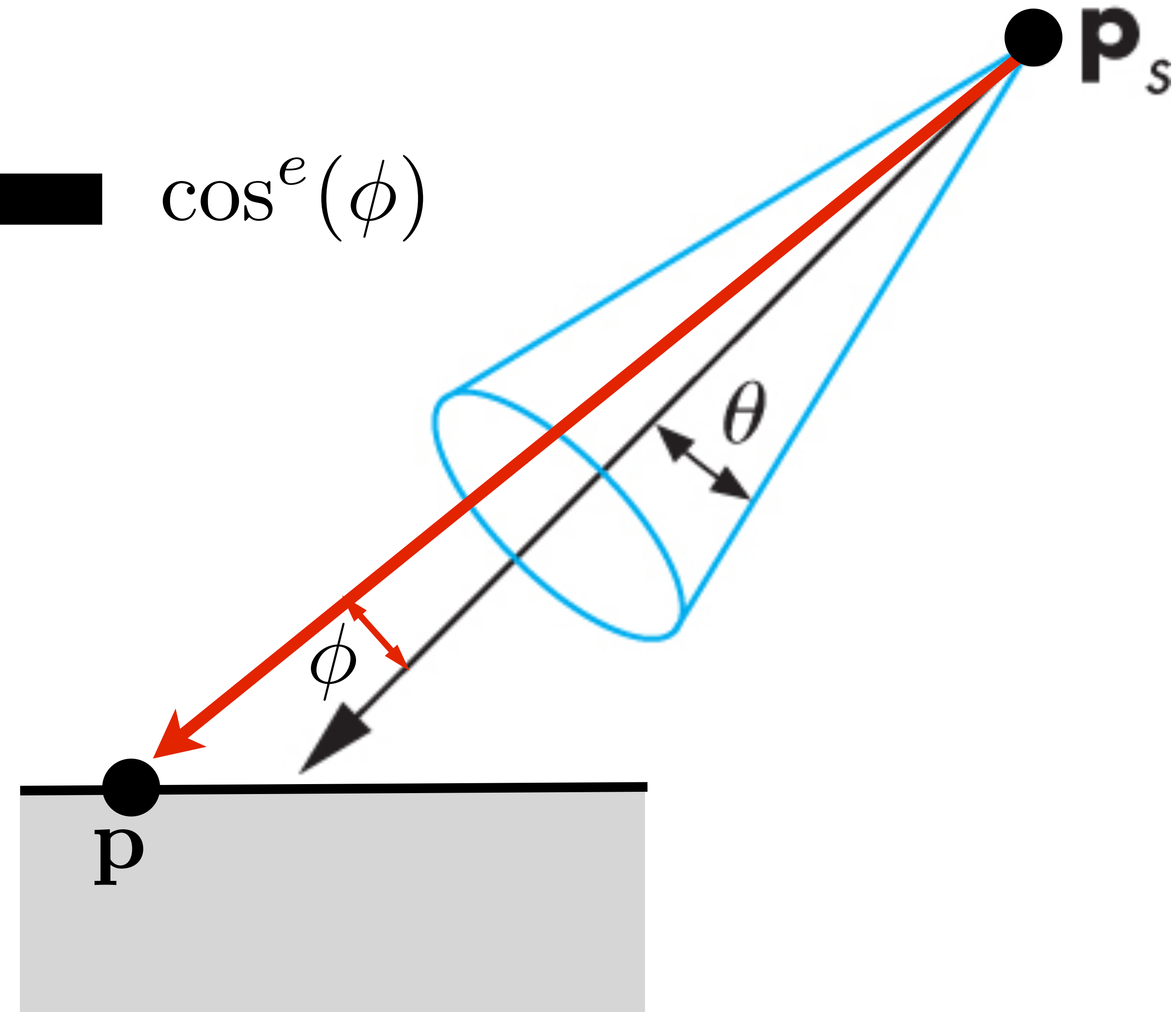


[Angel and Shreiner]

Spotlights



$\cos^e(\phi)$



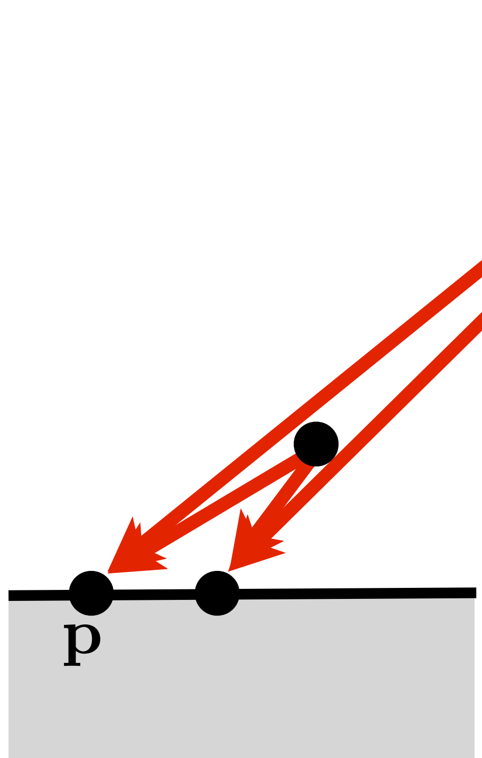
$$\cos^e(\phi)\ l(\mathbf{p}, \mathbf{p}_s)$$

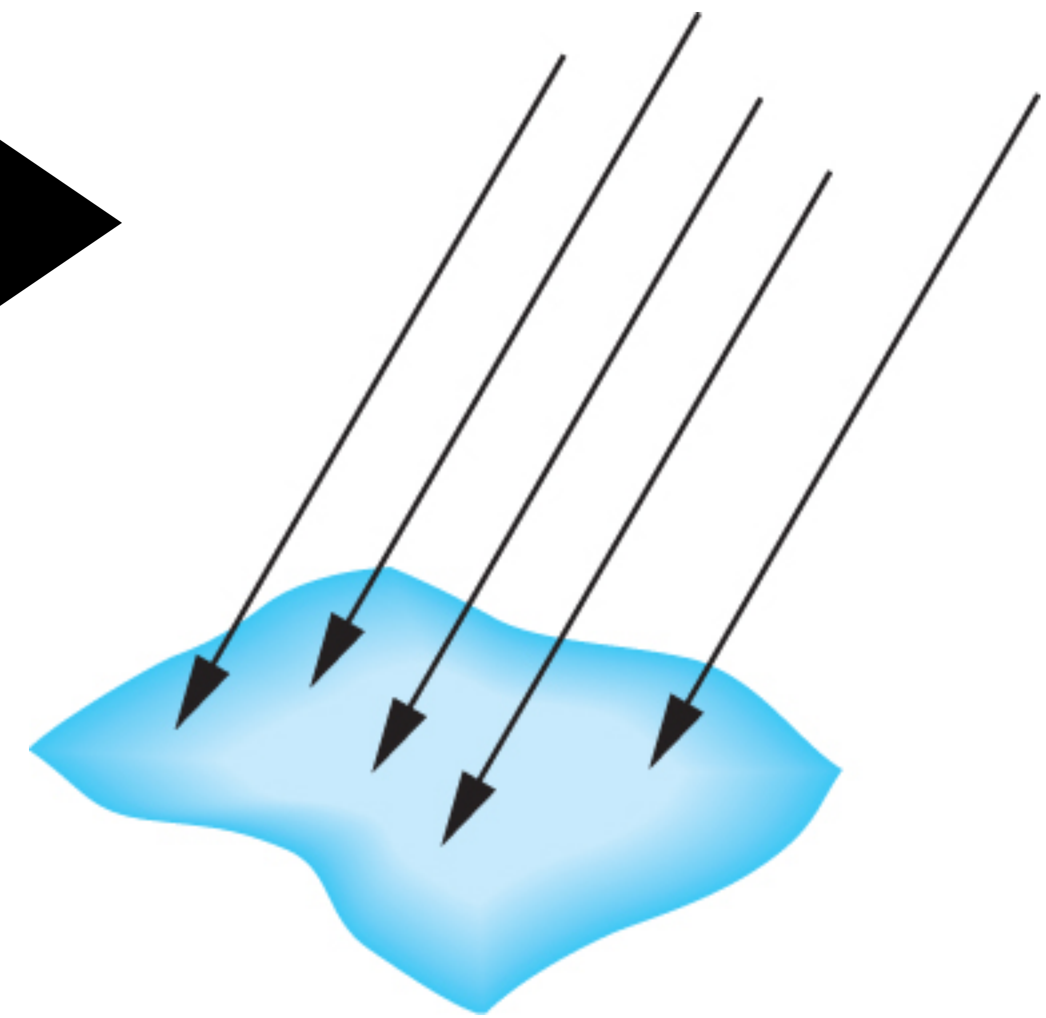
[Angel and Shreiner]

add an exponent for greater control
final result is like point light but modified by this cone

Distant light source

characterized by direction

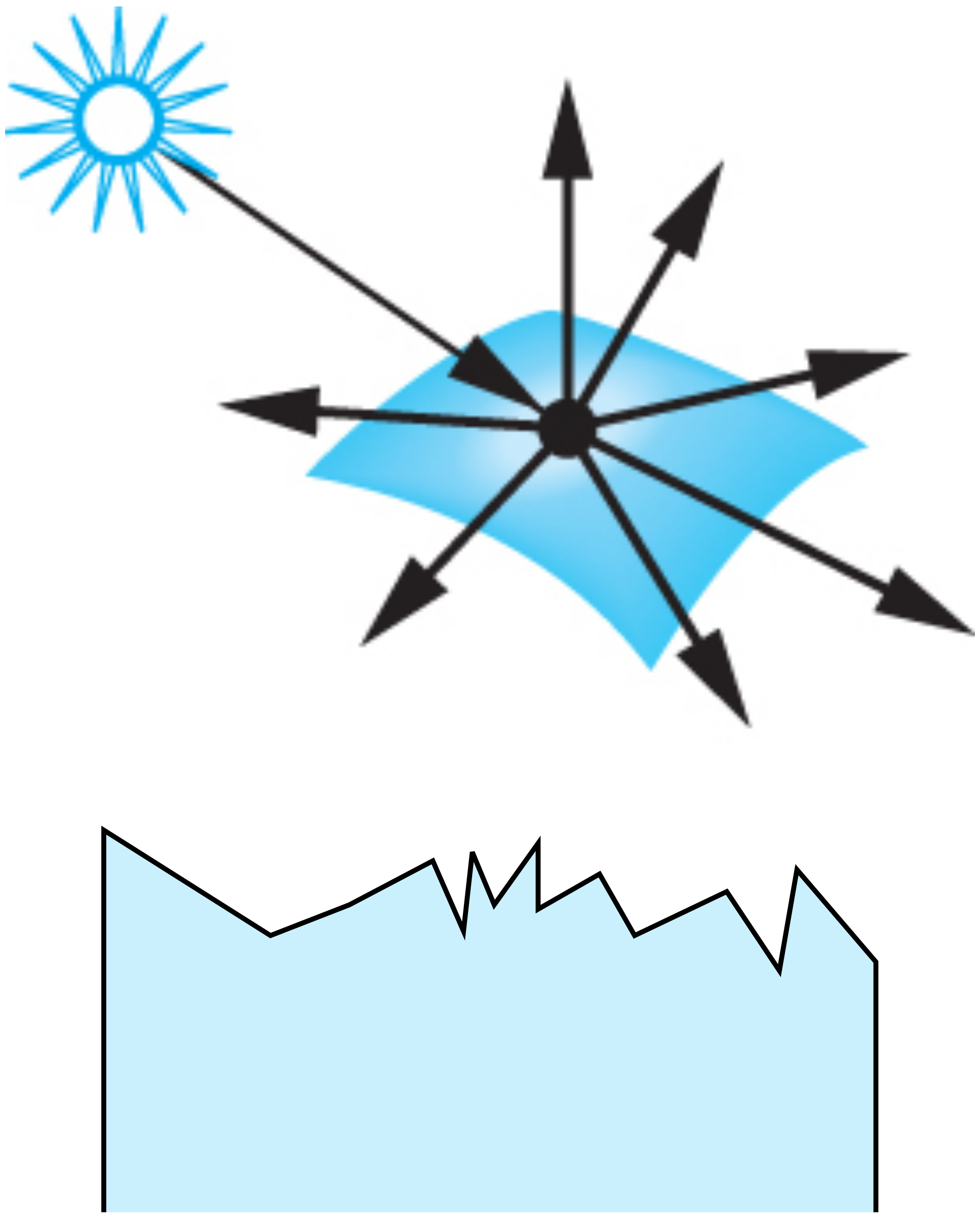




[Angel and Shreiner]

most shading calculations require direction from the surface point to the light source position
if the light source is very far, the direction vectors don't change
e.g., sun
characterized by direction rather than position

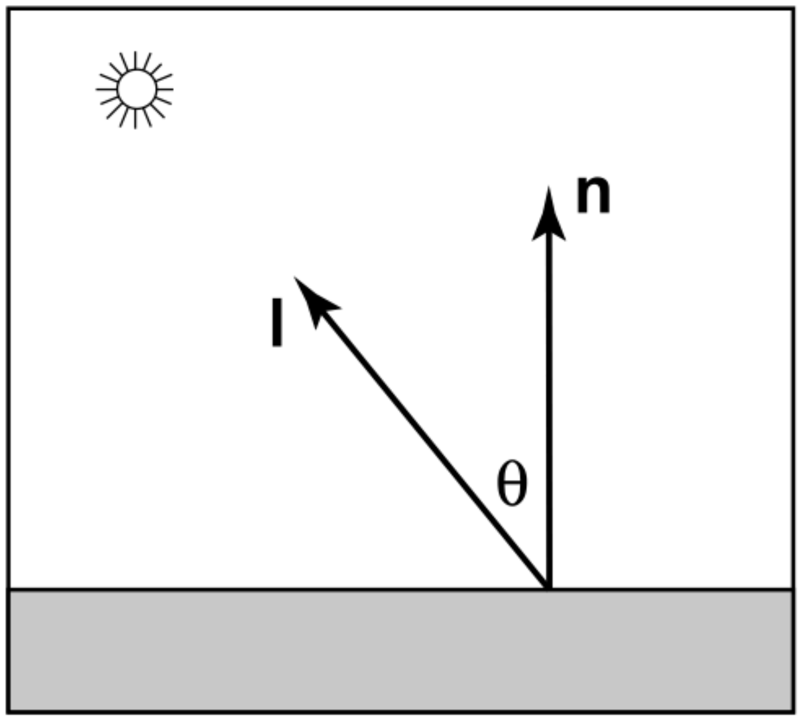
Lambertian Reflection Model

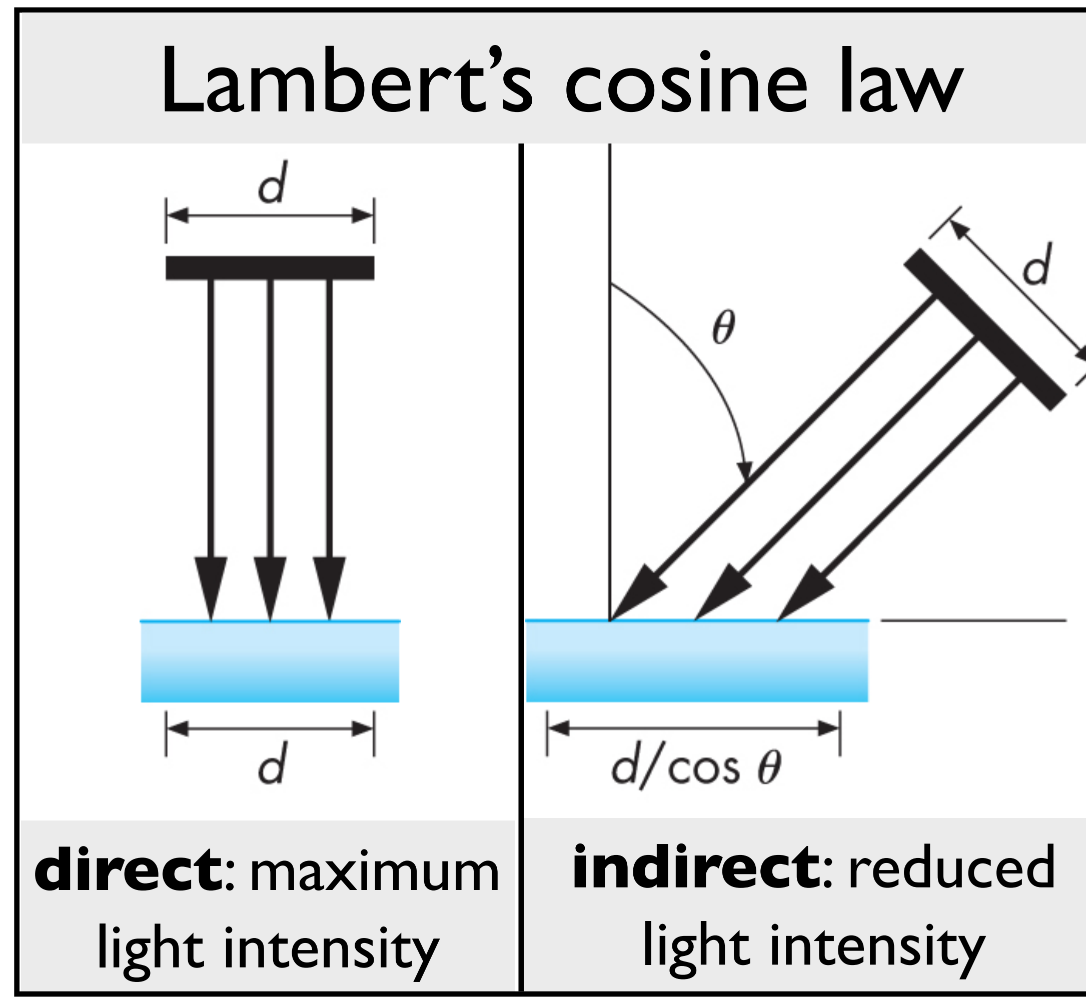


http://www.andreafisherpottery.com/cgi-bin/artistlnk.cgi?Jody_Folwell

The **Lambertian reflection model** is good for **diffuse** surfaces (those with a rough surface). The bottom part of the vase could be rendered with the Lambertian reflection model, since it is matte in appearance. The top part of the vase is reflective and has specular highlights.

Lambertian Reflection Model



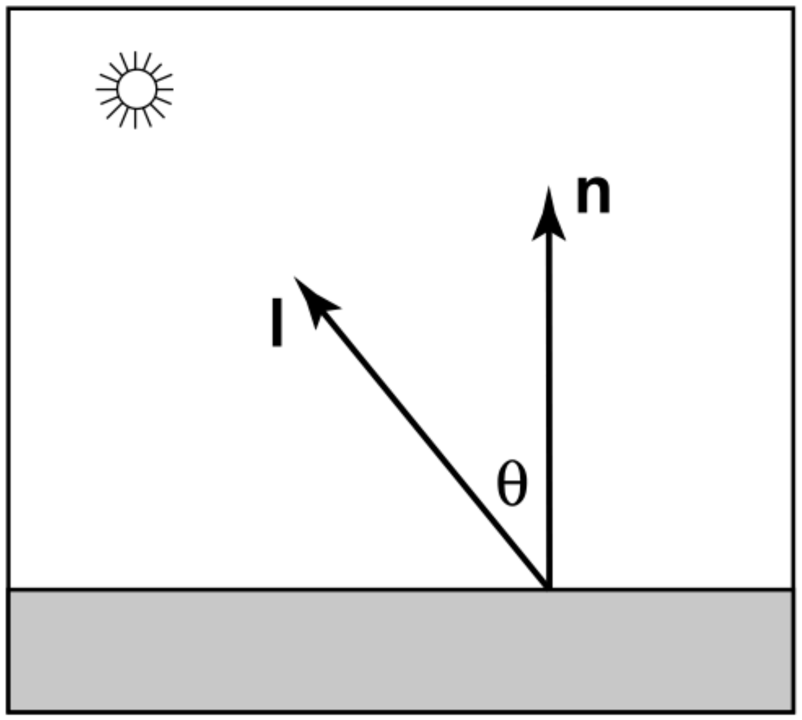


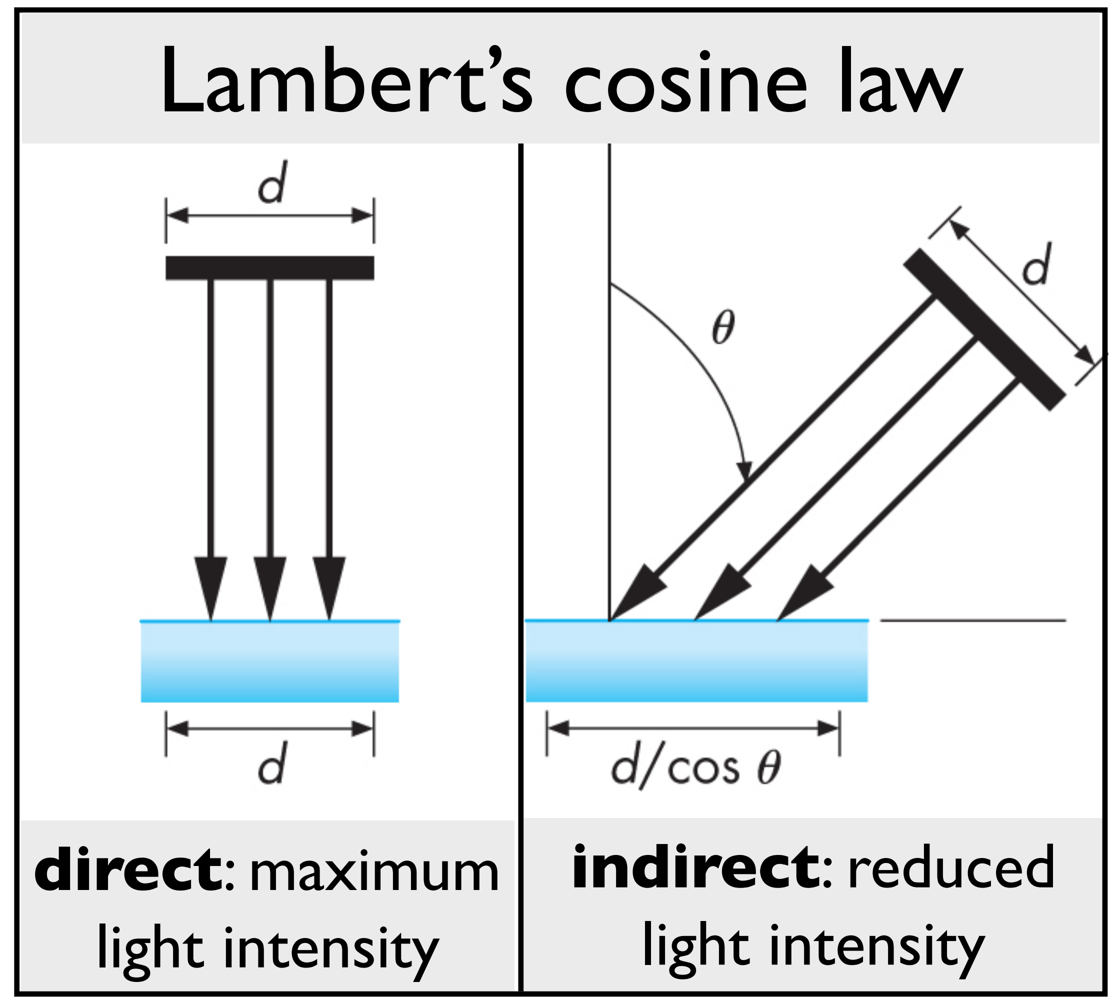
$$I \propto \cos\theta$$

color intensity

Lambert's cosine law says that the col or intensity should be proportional to the cosine of the angle between l and n. The light source with length d has a certain amount of light energy associated with it. If the light is tilted relative to the surface, the same amount of light energy shines on **more** surface area. Therefore, the intensity of the light is **less** per unit surface area.

Lambertian Reflection Model



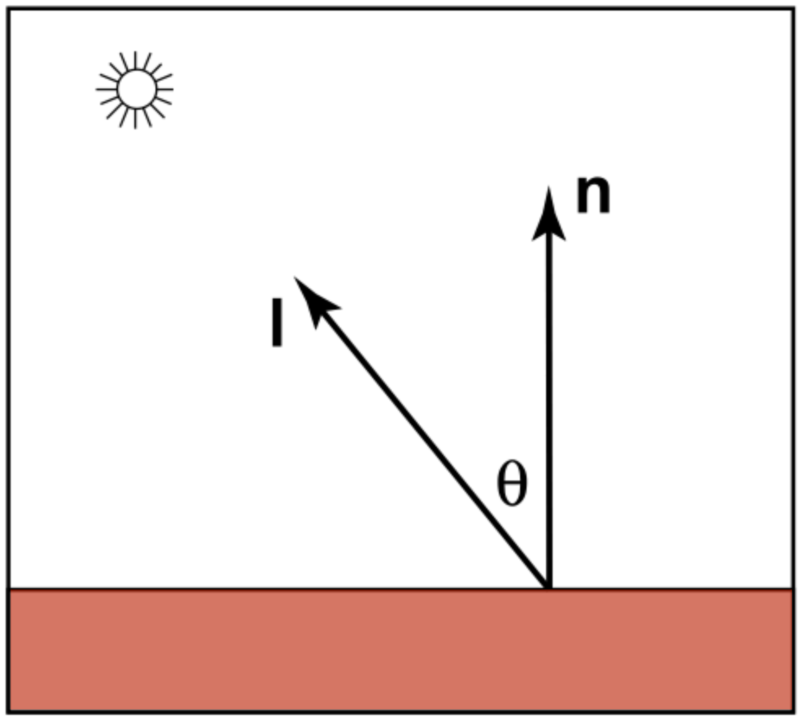


$$I \propto \mathbf{n} \cdot \mathbf{l}$$

color intensity

cos theta = **n . l**

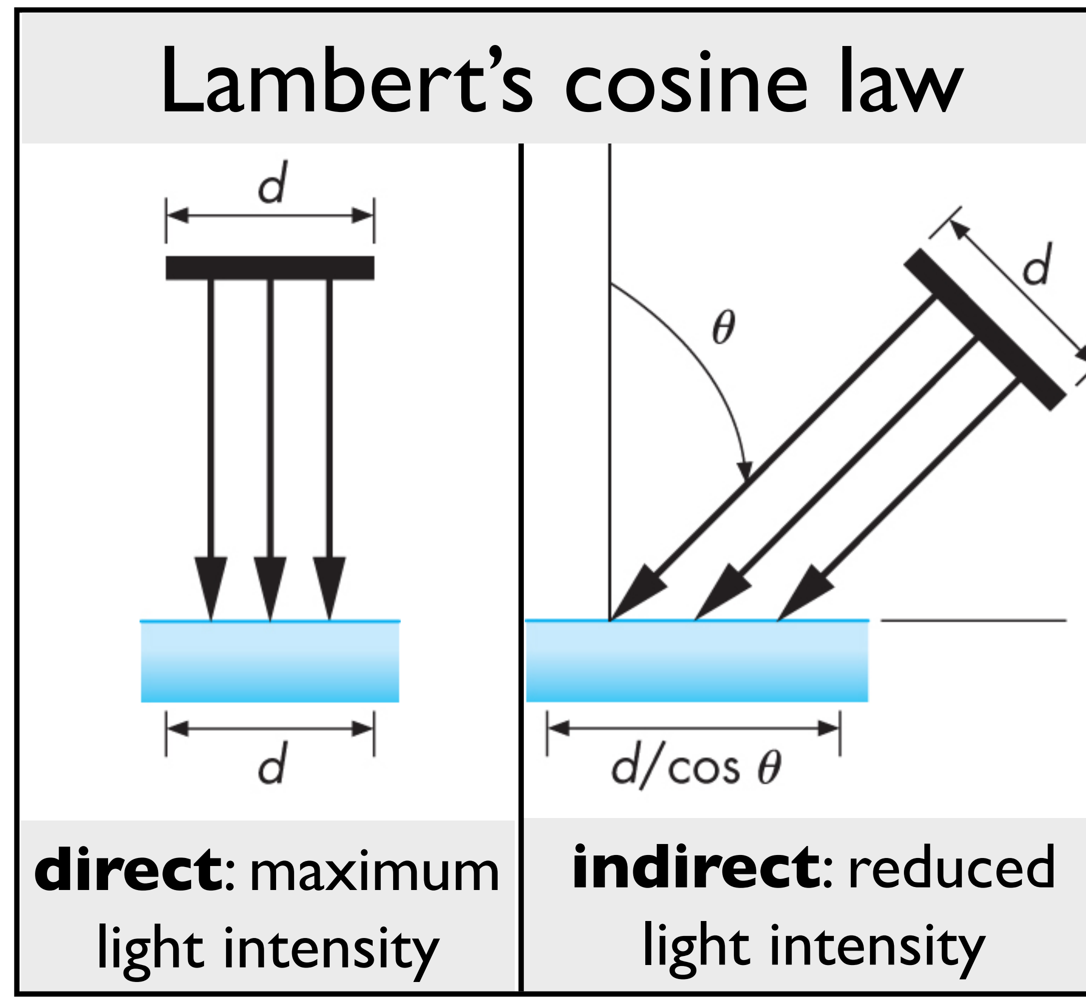
Lambertian Reflection Model



$$I \propto R\mathbf{n} \cdot \mathbf{l}$$

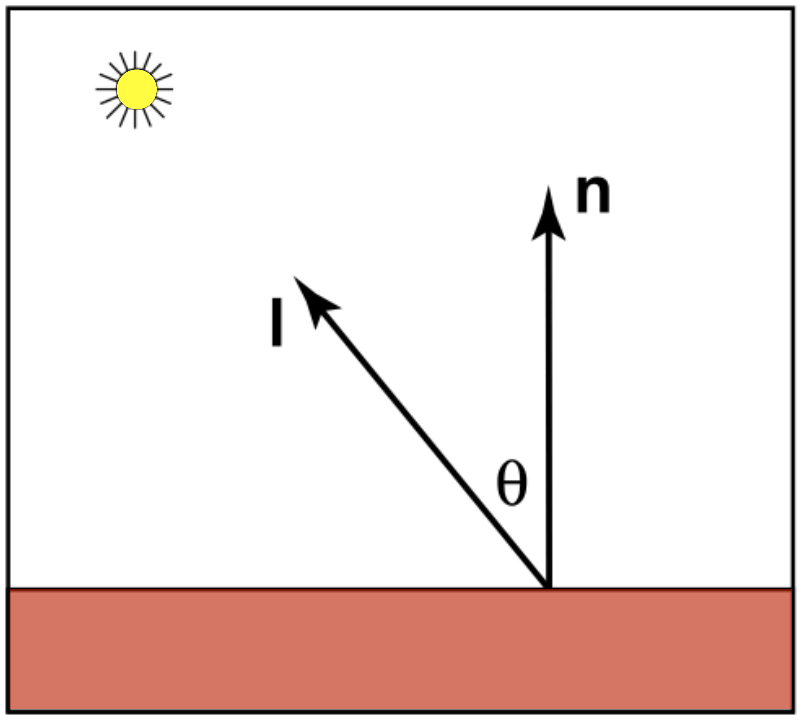
color intensity

reflectance



the color intensity is also going to be proportional to the reflectance of the object in that color channel

Lambertian Reflection Model

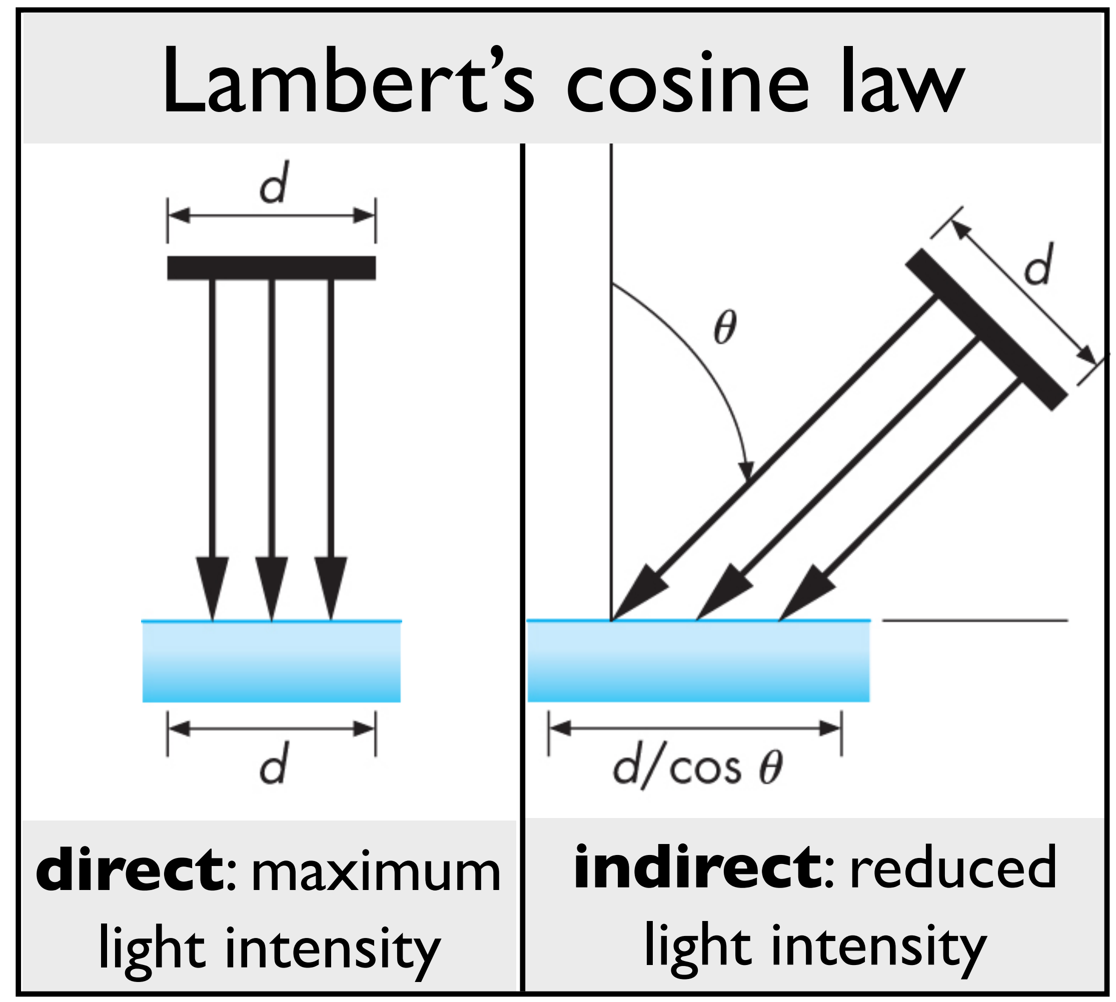


illumination

$$I = LR\mathbf{n} \cdot \mathbf{l}$$

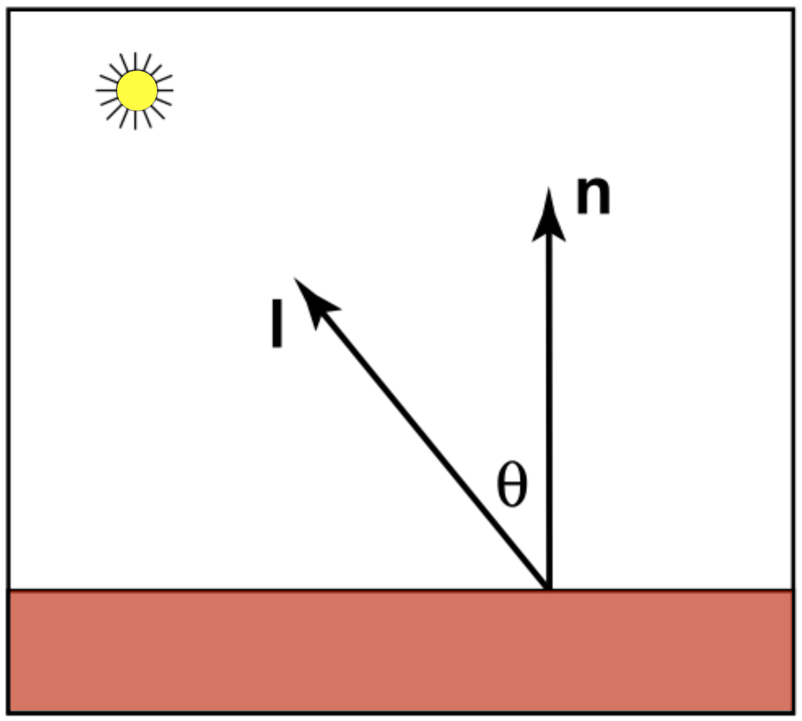
color intensity

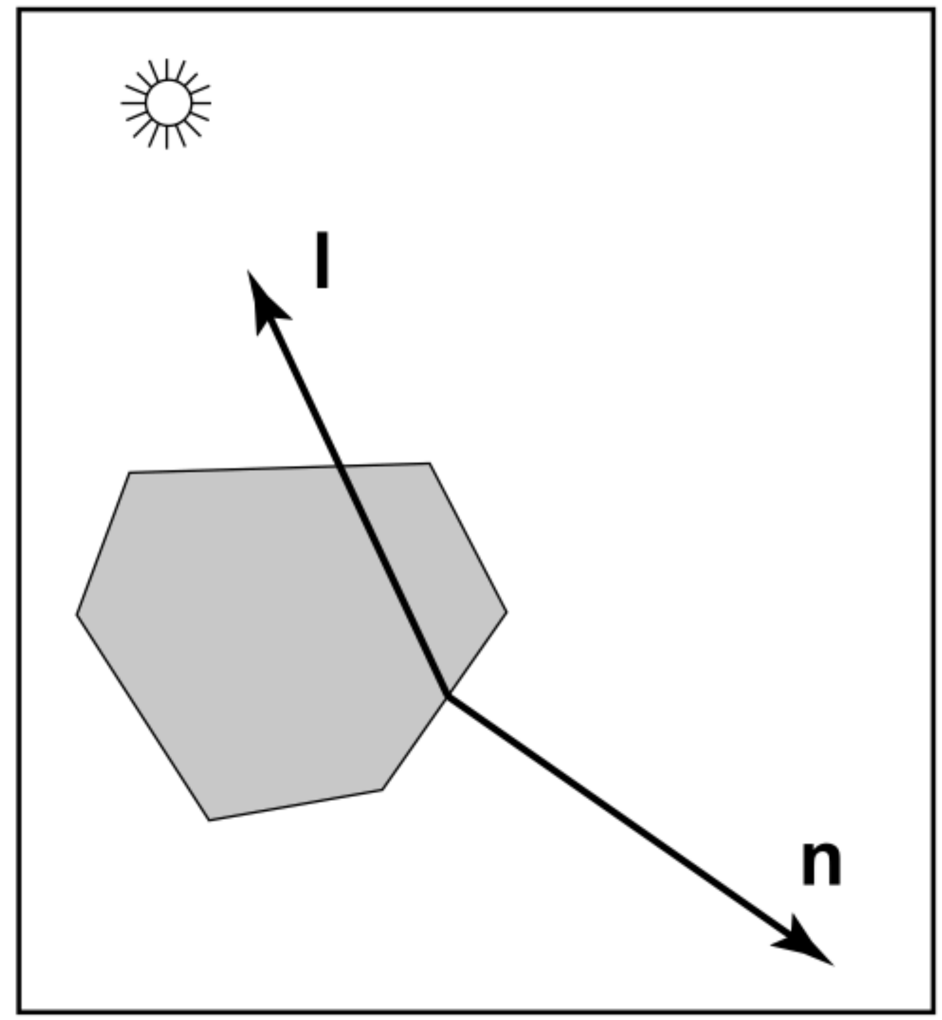
reflectance



and it will be proportional to the light intensity

Lambertian Reflection Model

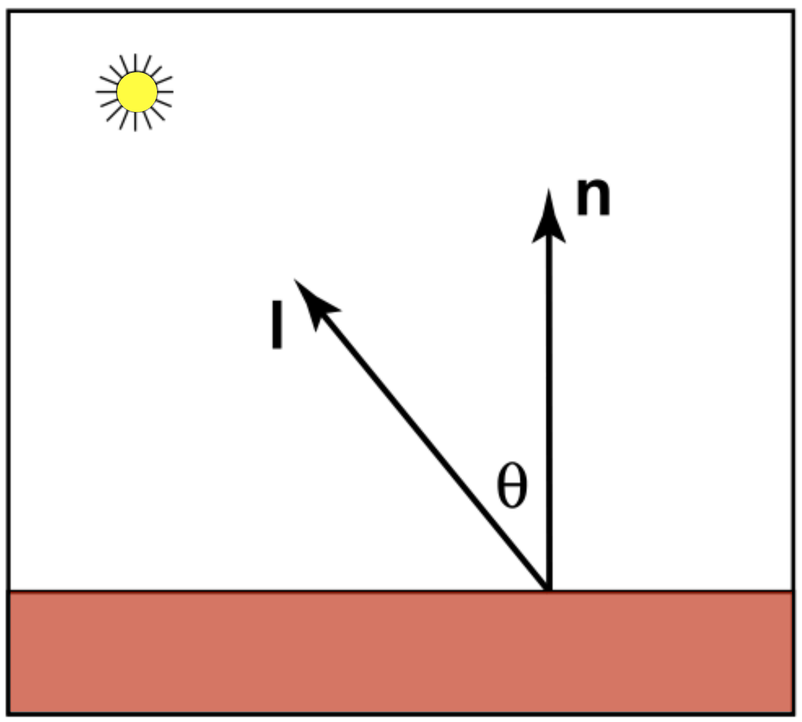




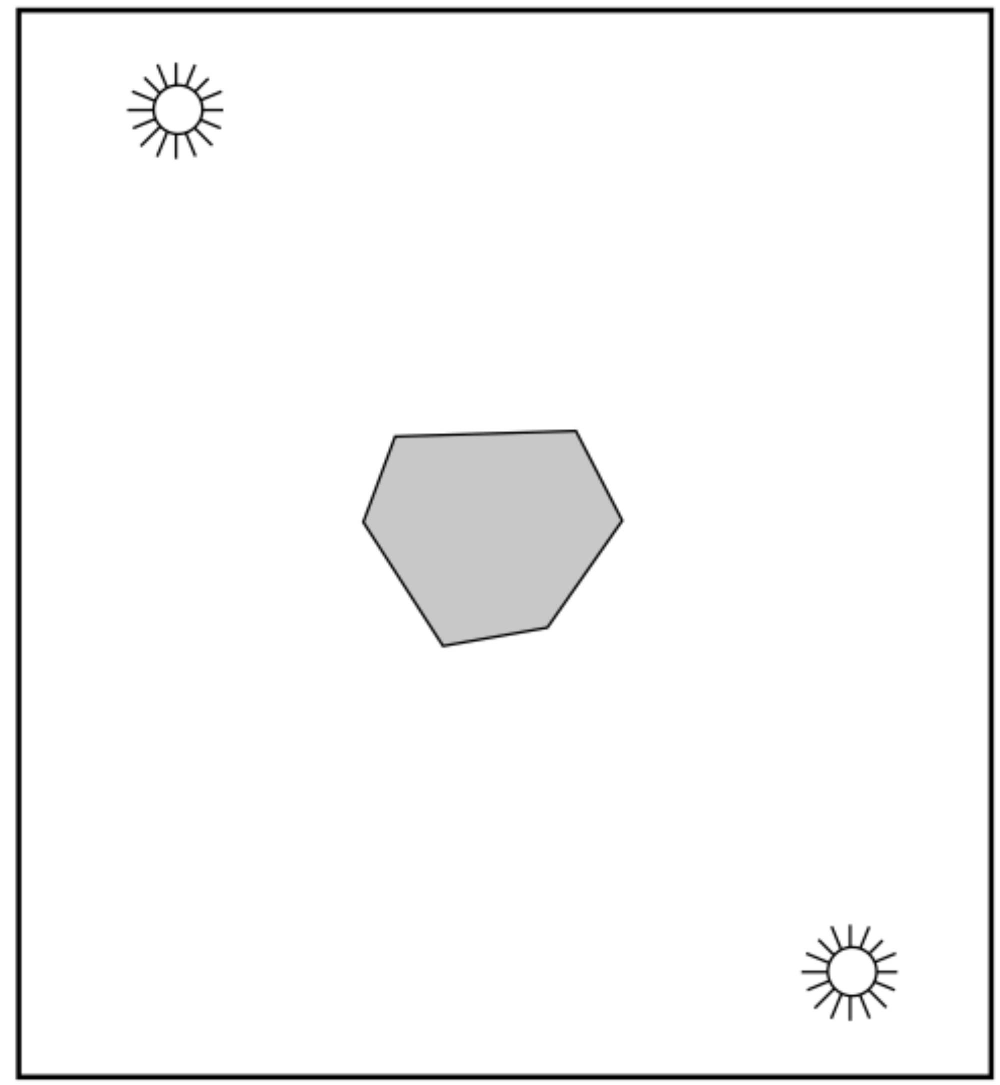
$$I = LR\max(0, \mathbf{n} \cdot \mathbf{l})$$

the cosine is negative if the angle is more than 90 degrees. In this case, the face points away from the light. If we don't modify the formula we'll get a negative intensity. We can put in the max to ensure that if the face points away, it won't be lit by the light.

Lambertian Reflection Model



$$I = LR|\mathbf{n} \cdot \mathbf{l}|$$



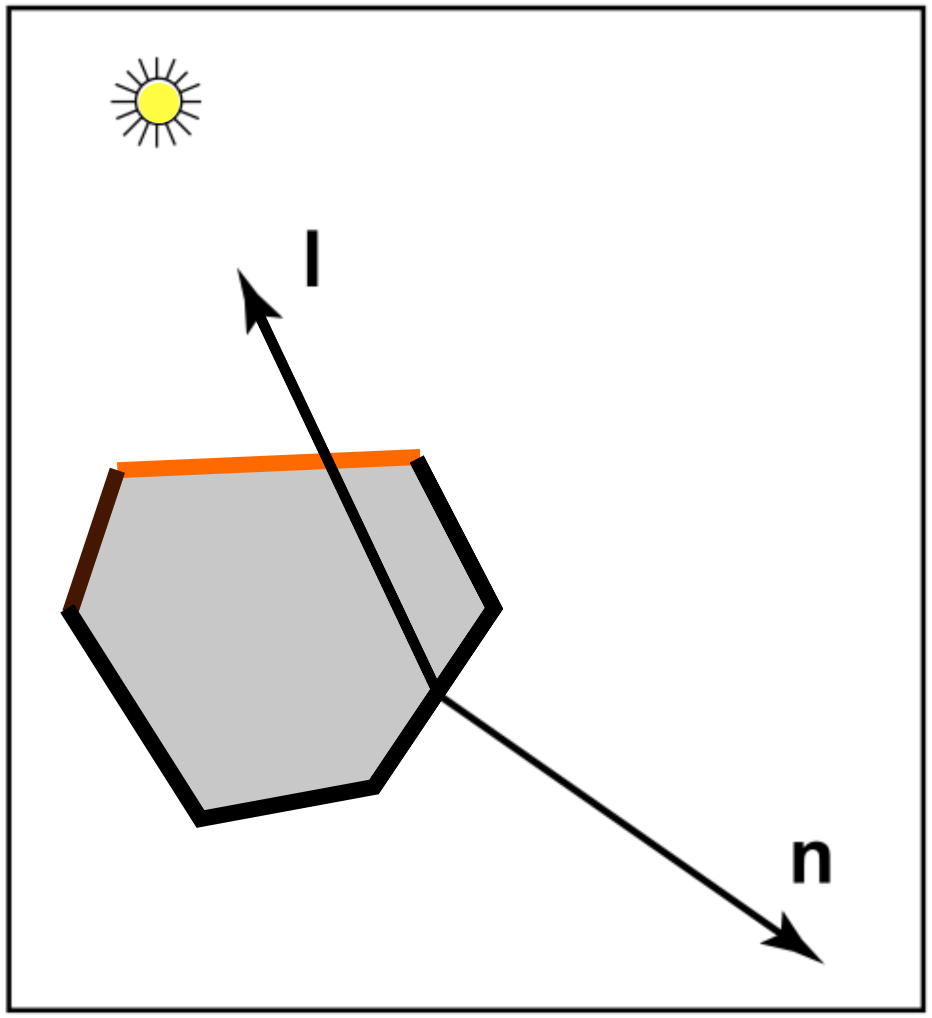
two-sided lighting

An alternative is to take the absolute value. This is equivalent to having another light on the other side of the object exactly opposite the first.

Ambient Reflection

$$I = LR \max(0, \mathbf{n} \cdot \mathbf{l})$$

Surfaces facing away from the light will be totally **black**

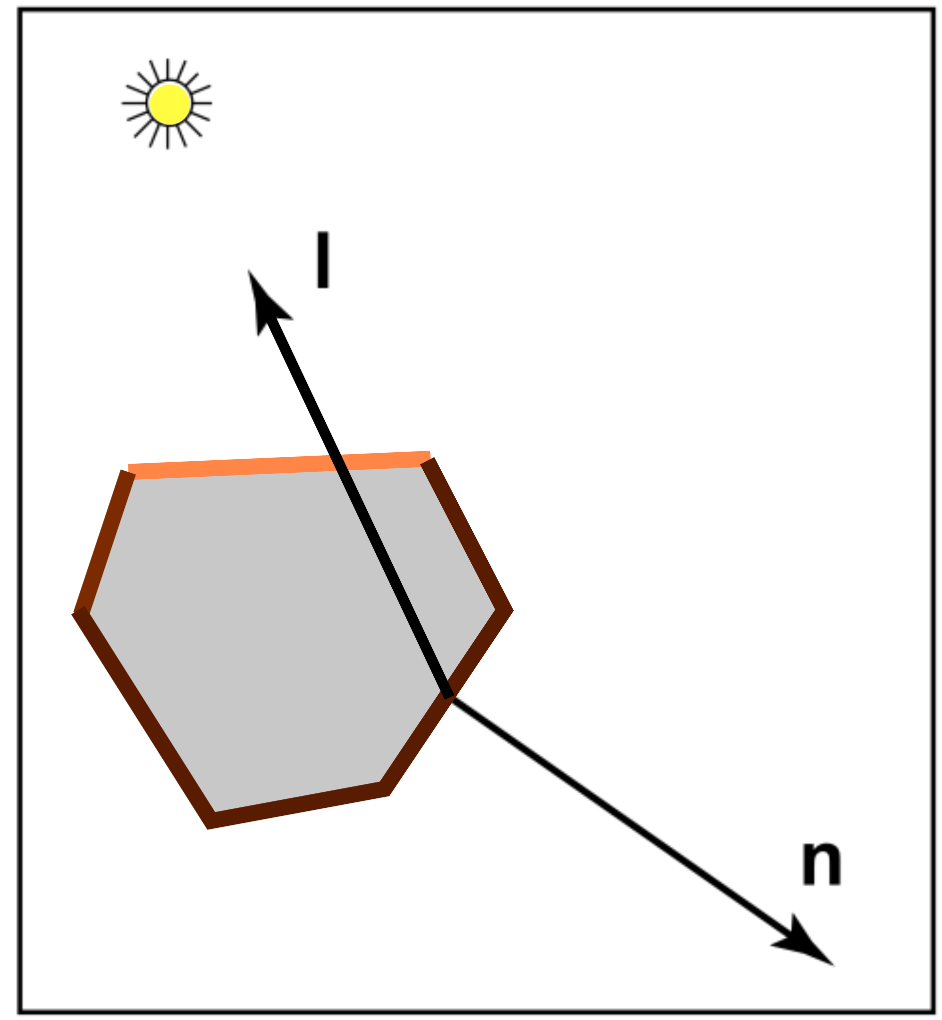


Problem: surfaces facing away from the light will be totally black.

Ambient Reflection

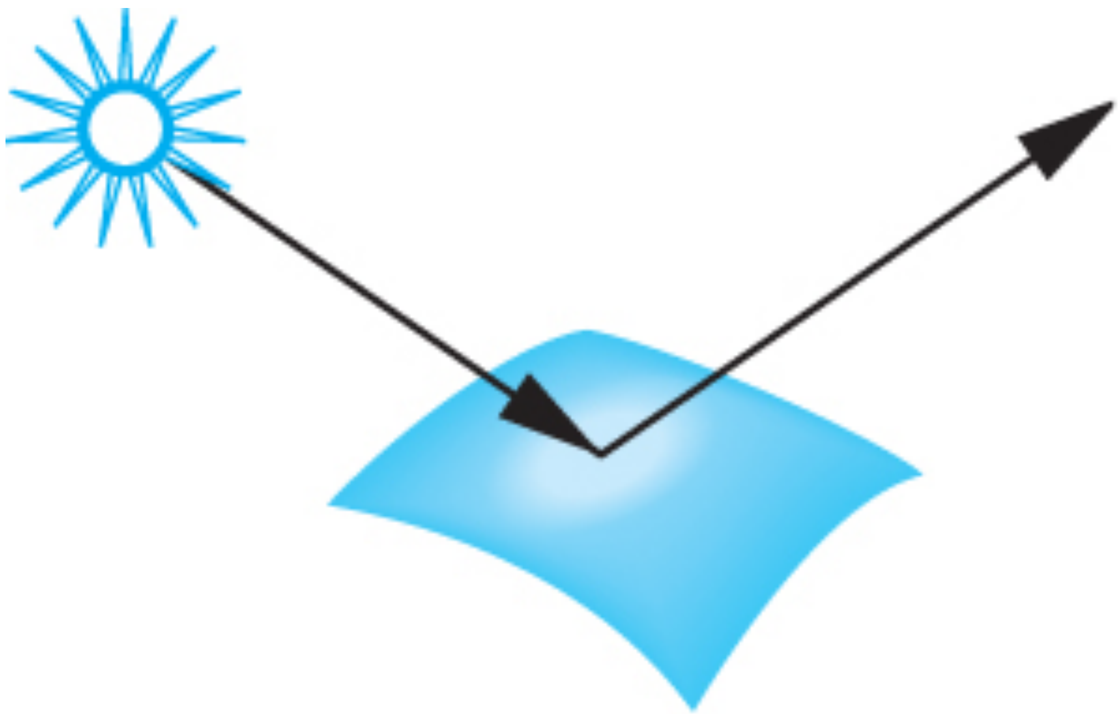
$$I = L_a R_a + L_d R_d \max(0, \mathbf{n} \cdot \mathbf{l})$$

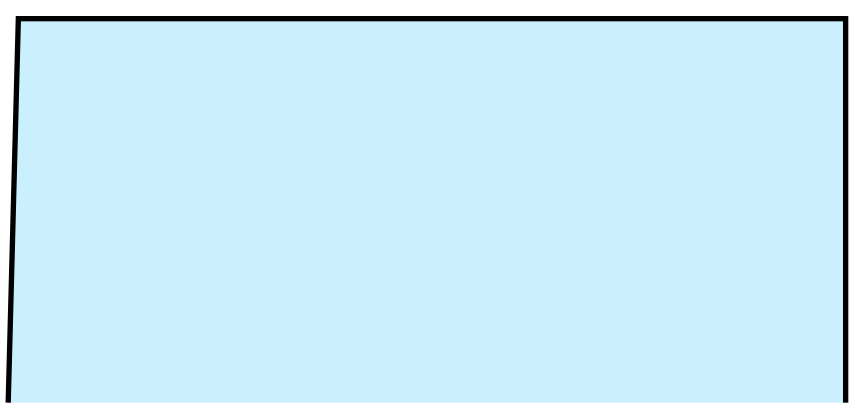
All surfaces get same amount of ambient light



Problem: surfaces facing away from the light will be totally black – ambient light mitigates this by adding some light everywhere

Phong Reflection Model



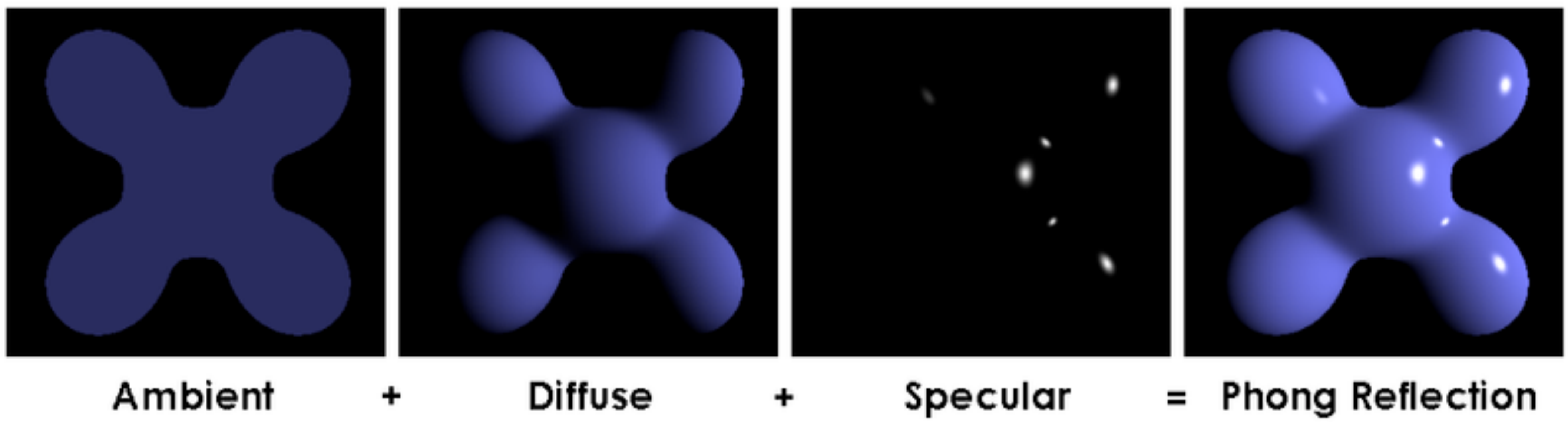


http://www.andreafisherpottery.com/cgi-bin/artistlnk.cgi?Jody_Folwell

The **Phong reflection model** combines the Ambient and Lambertian reflections with a **specular** reflection to capture highlights such as the white highlight seen on the shiny part of the vase

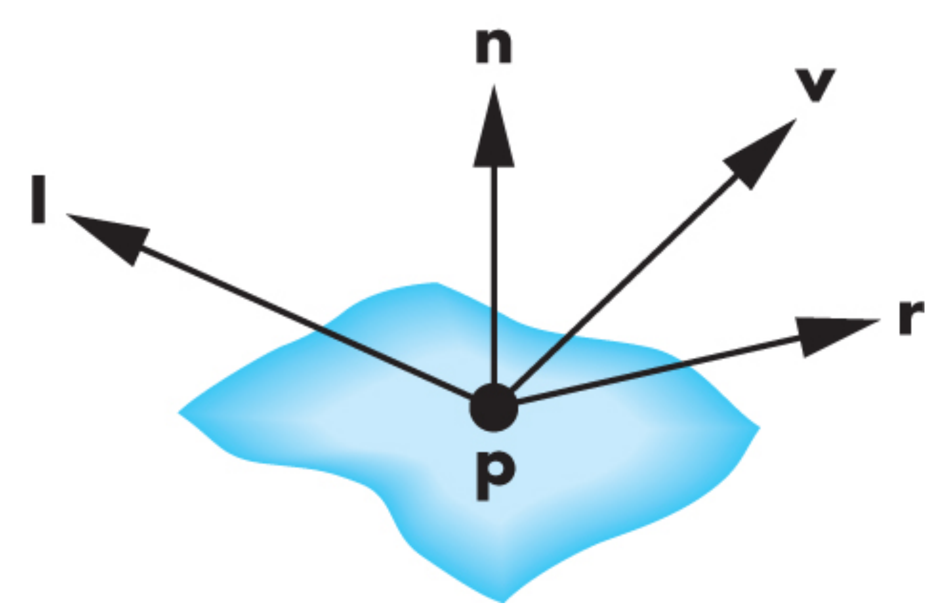
The highlight is a reflection of the light and it is the color of the light.

Phong Reflection Model



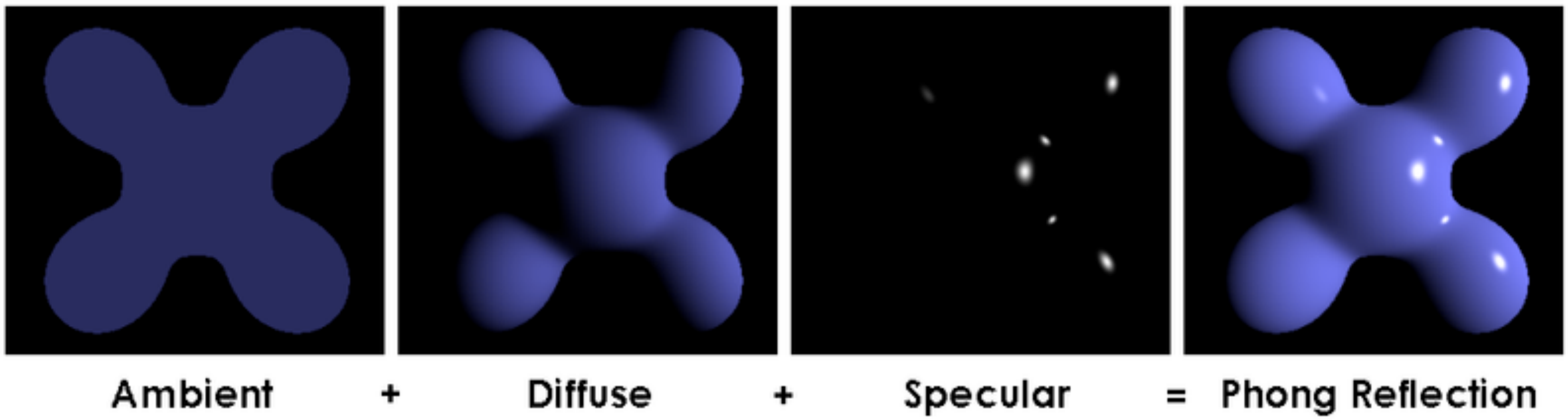
[Brad Smith, Wikimedia Commons]

- efficient, reasonably realistic
- 3 components
- 4 vectors



- **l** to light source
- **n** surface normal
- **v** to viewer
- **r** perfect reflector (function of **n** and **l**)

Phong Reflection Model



[Brad Smith, Wikimedia Commons]

$$
\begin{aligned}
I &= I_a + I_d + I_s \\
&= R_a L_a + R_d L_d \max(0, \mathbf{l} \cdot \mathbf{n}) + R_s L_s \max(0, \cos\phi)^{\alpha}
\end{aligned}
$$

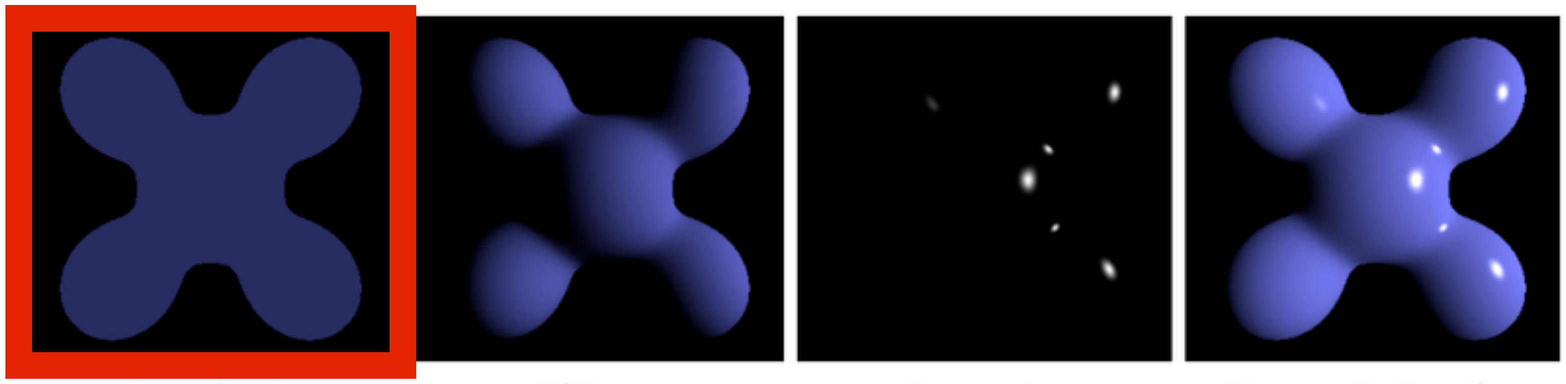
color intensity

reflectance

illumination

This formula will be applied for each of the three color channels independently.

Ambient reflection



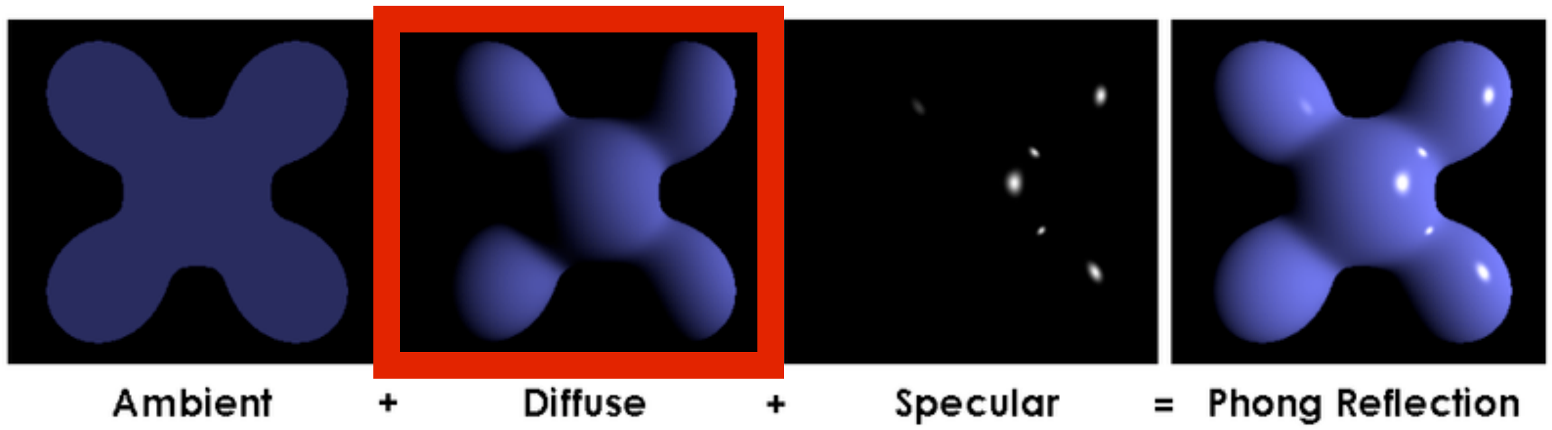
different ambient coefficients for different colors

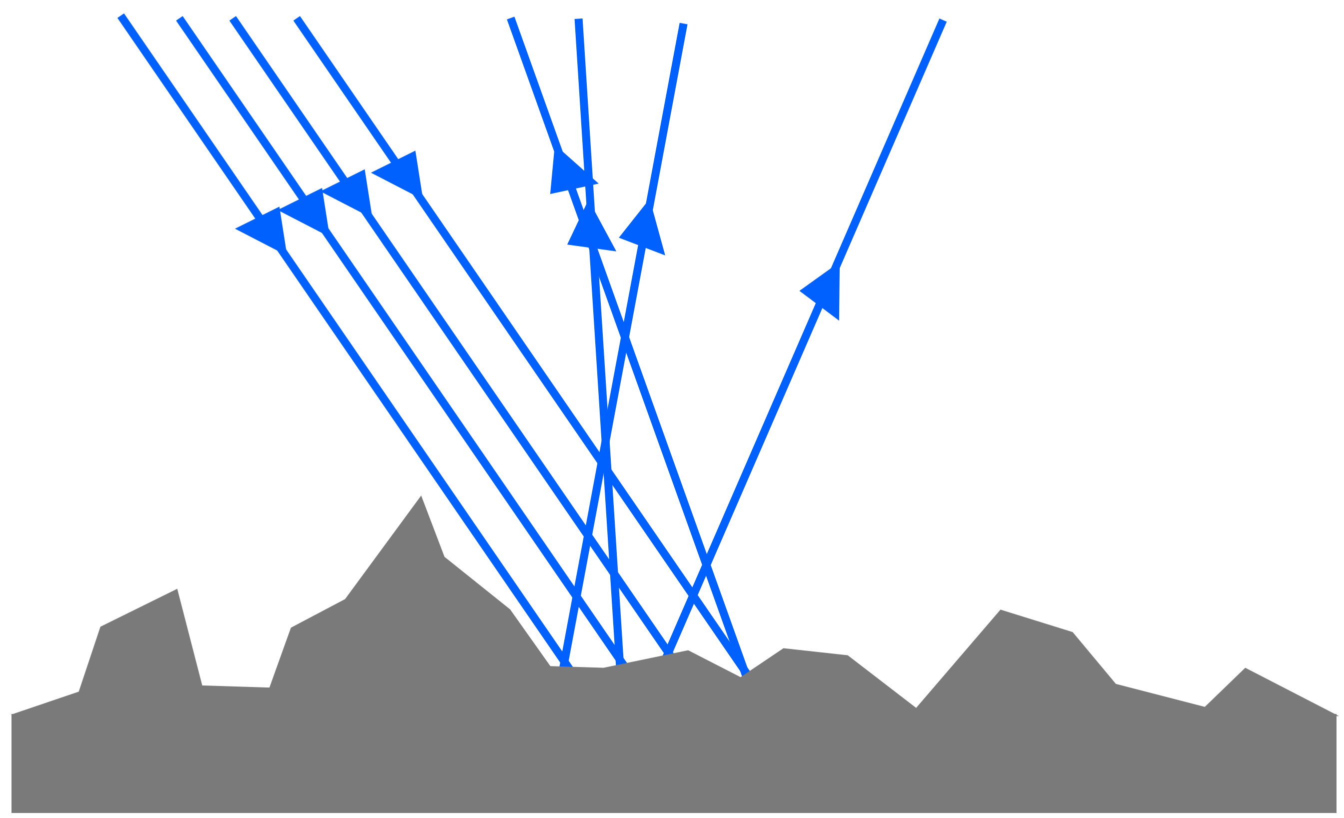
$$I_a = R_a L_a, \qquad 0 \leq R_a \leq 1$$

ambient reflection coefficient

e.g., white light shining on the object will be reflected differently in red, green, blue channels
e.g., more red and blue reflection here

Diffuse reflection

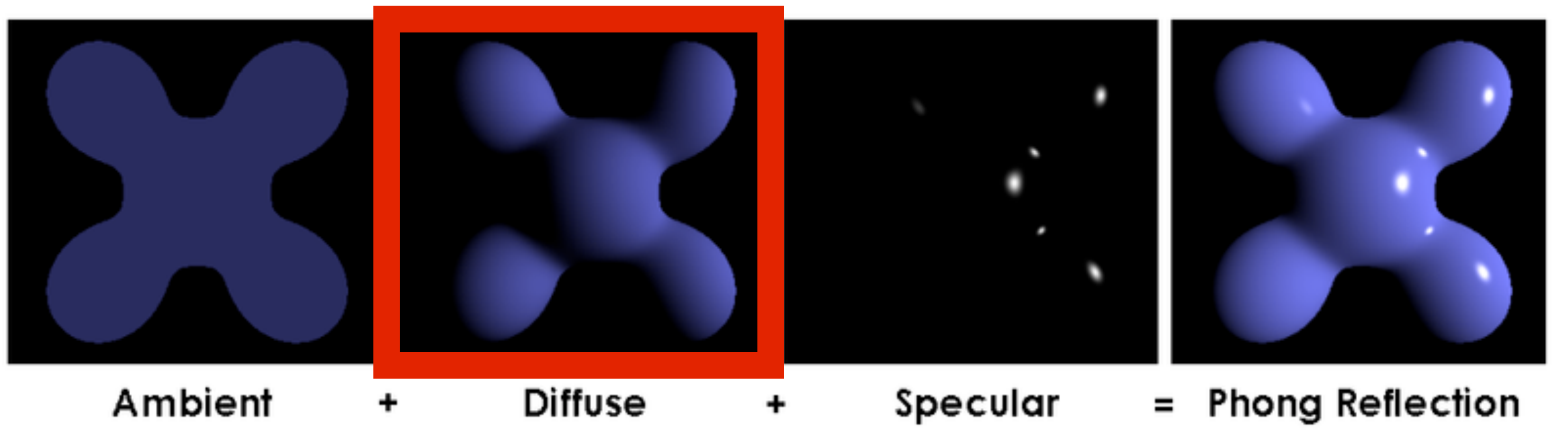




e.g., paper, unfinished wood, unpolished stone

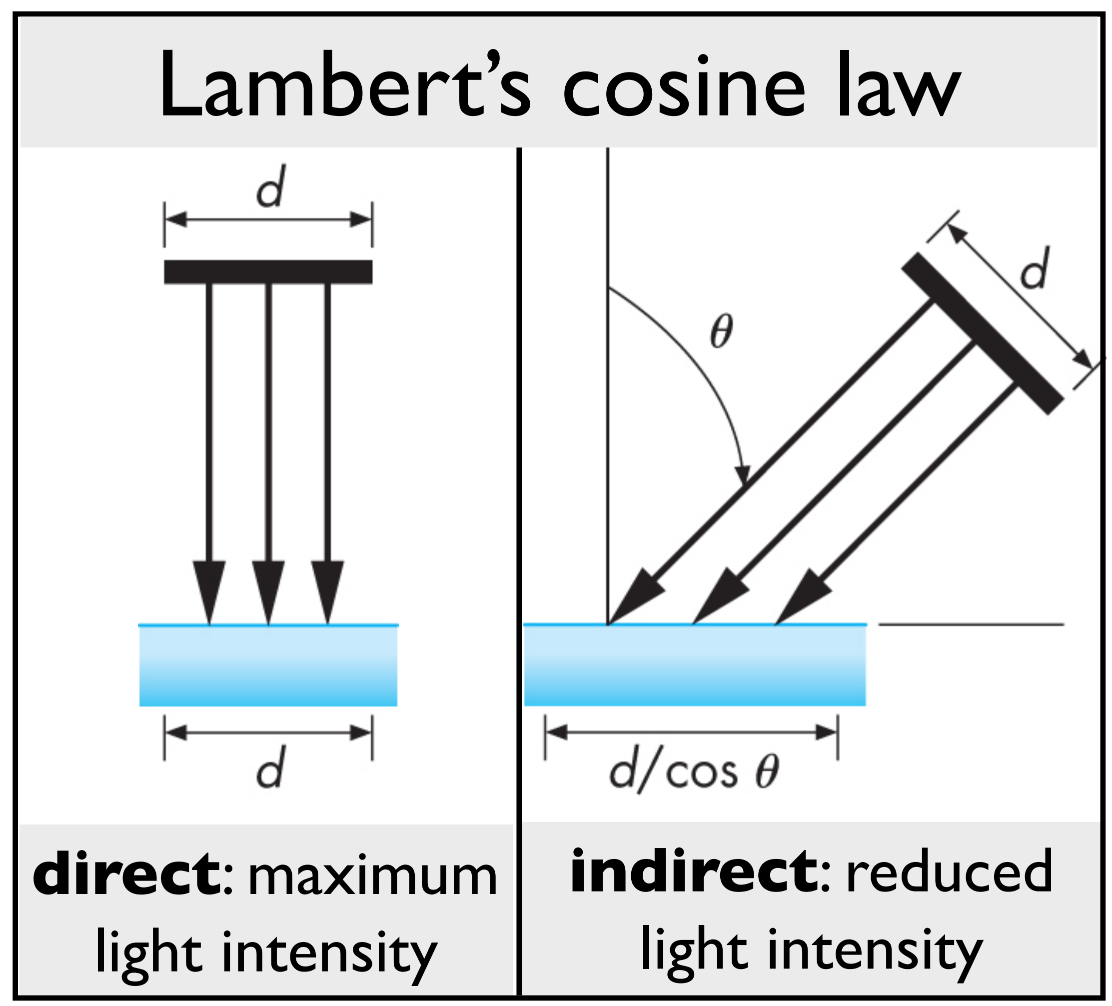
The diffuse component of the Phong reflectance model is the same as the Lambertian reflectance model

Diffuse reflection



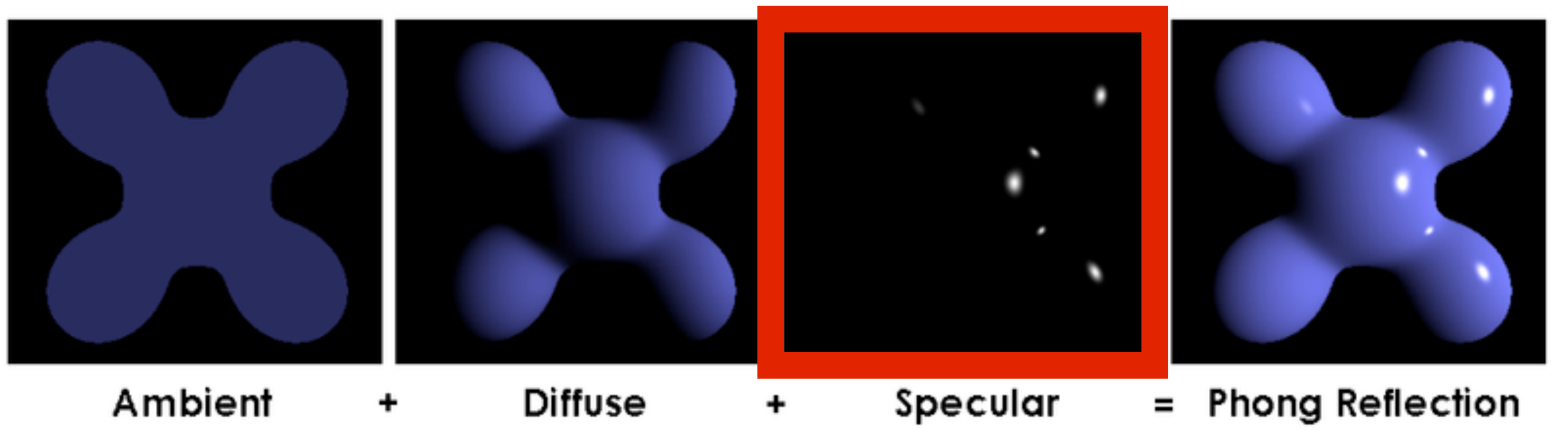
$$I_d = R_d L_d \max(0, \mathbf{l} \cdot \mathbf{n})$$

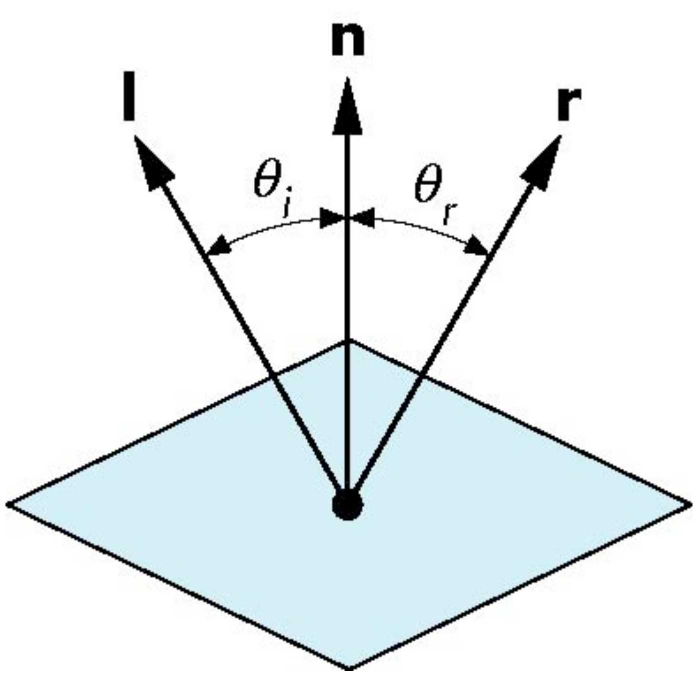
diffuse reflection coefficient



- the light is reduced by cos of angle
  - this is because same amount of light is spread over larger area when light comes in at an angle

Specular reflection





Ideal reflector

$$\theta_i = \theta_r$$

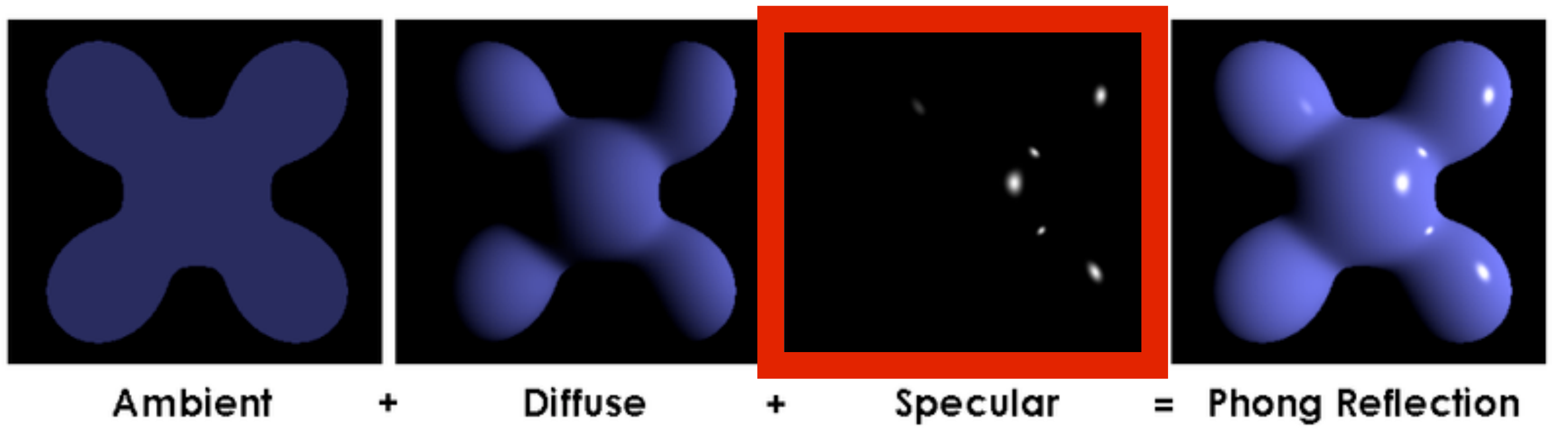
angle of incidence

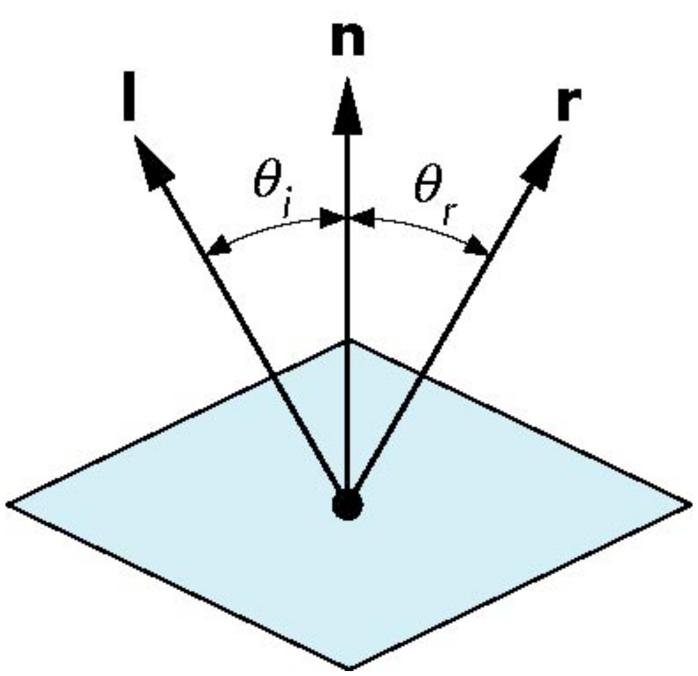
angle of reflection

r is the mirror reflection direction

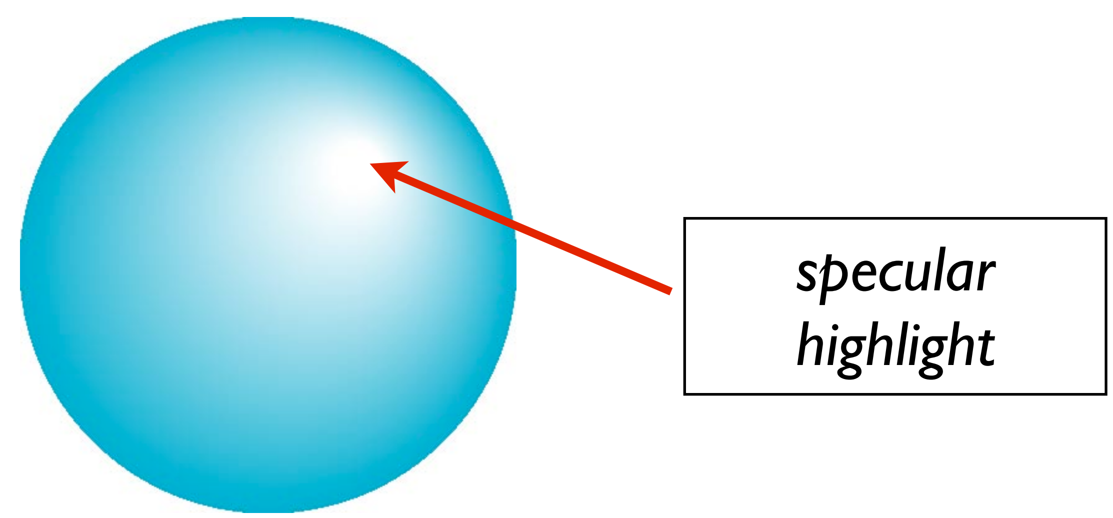
The new thing in the Phong reflection model is the specular component

Specular reflection





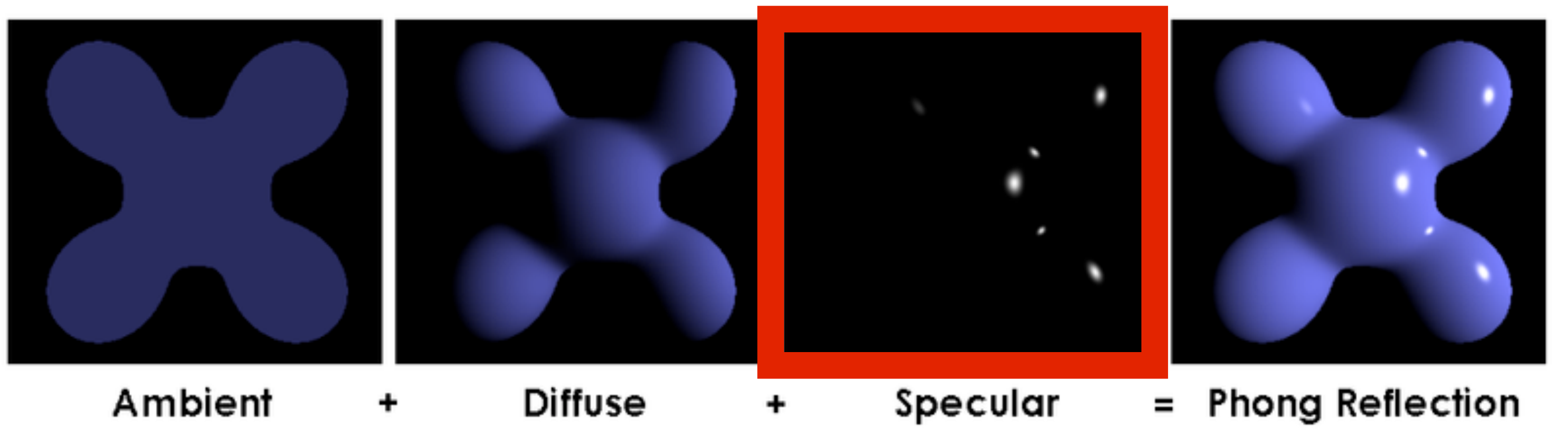
Specular surface

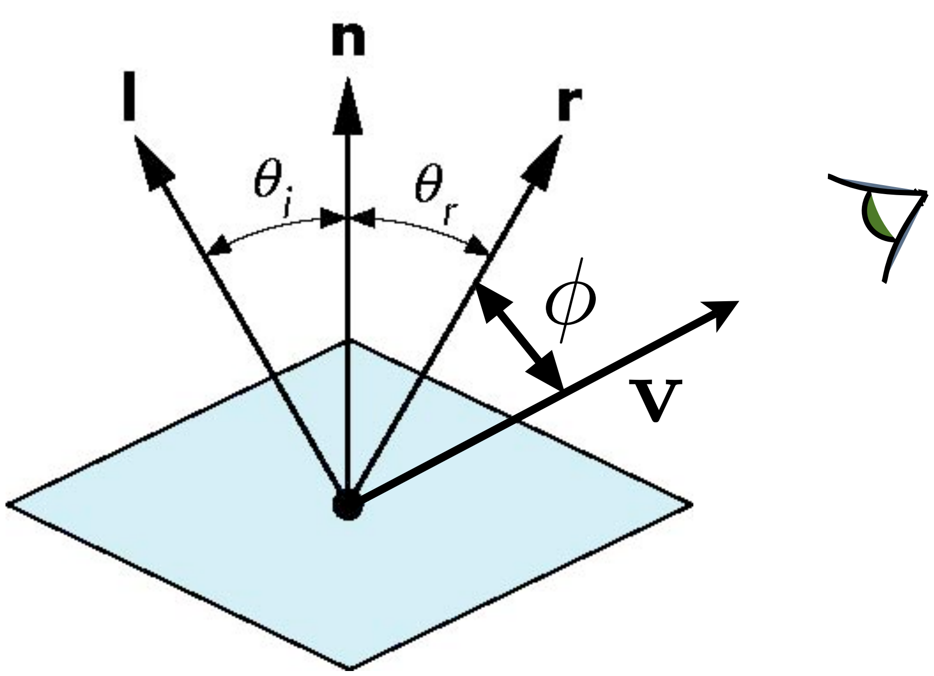


specular reflection is strongest in mirror reflection direction

area of specular highlight depends on how smooth the surface is

Specular reflection





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

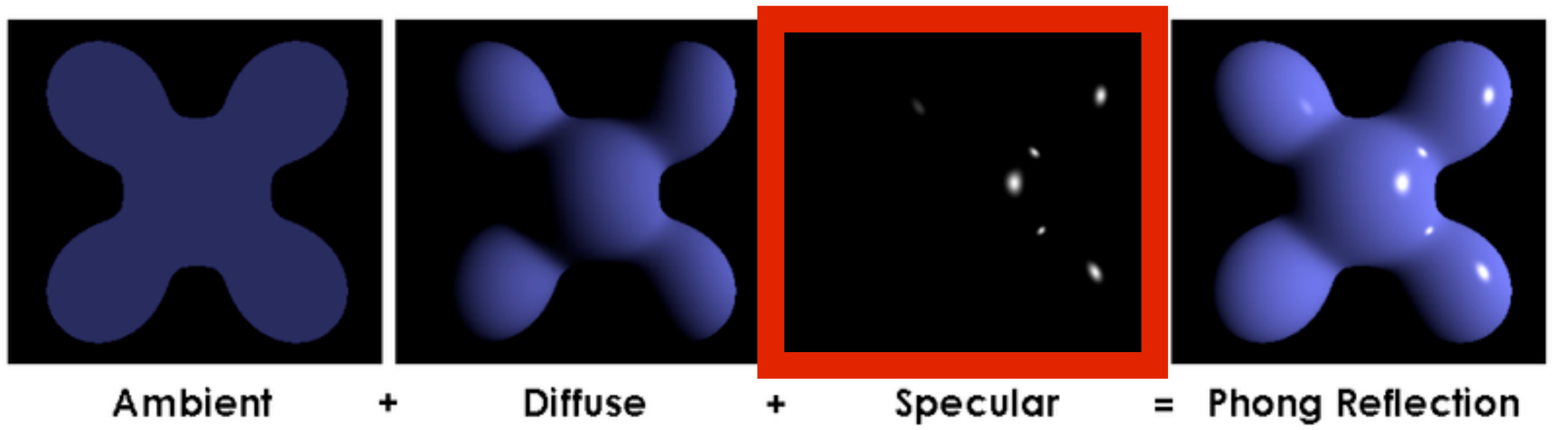
specular reflection coefficient

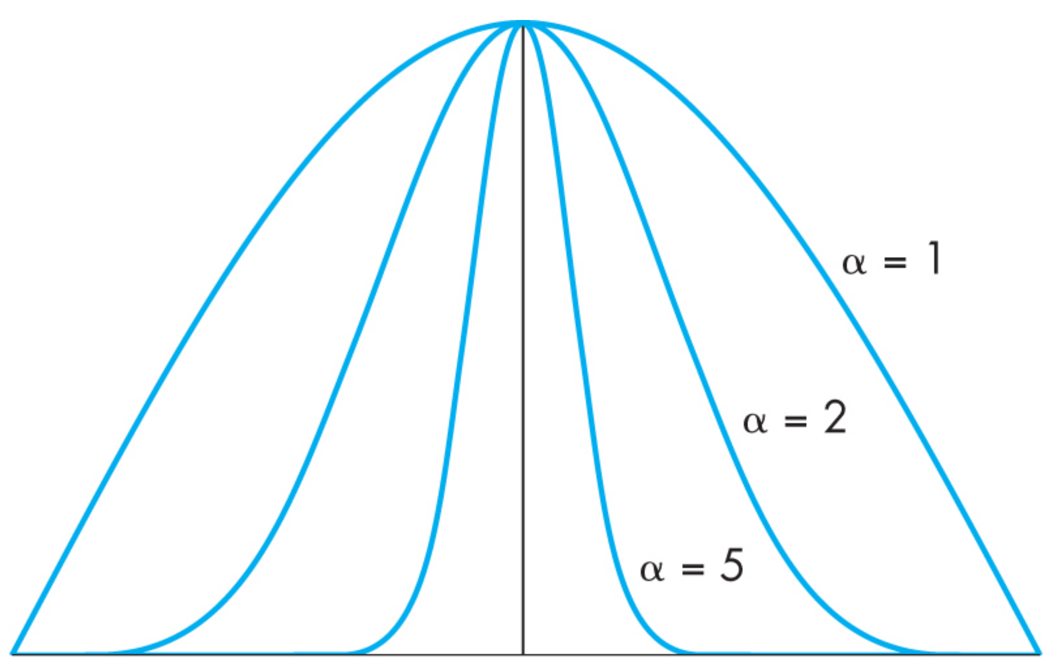
Phong exponent

specular reflection drops off with increasing angle ϕ

Phong proposed this model

Specular reflection





$$I_s = R_s L_s \max(0, \cos\phi)^{\alpha}$$

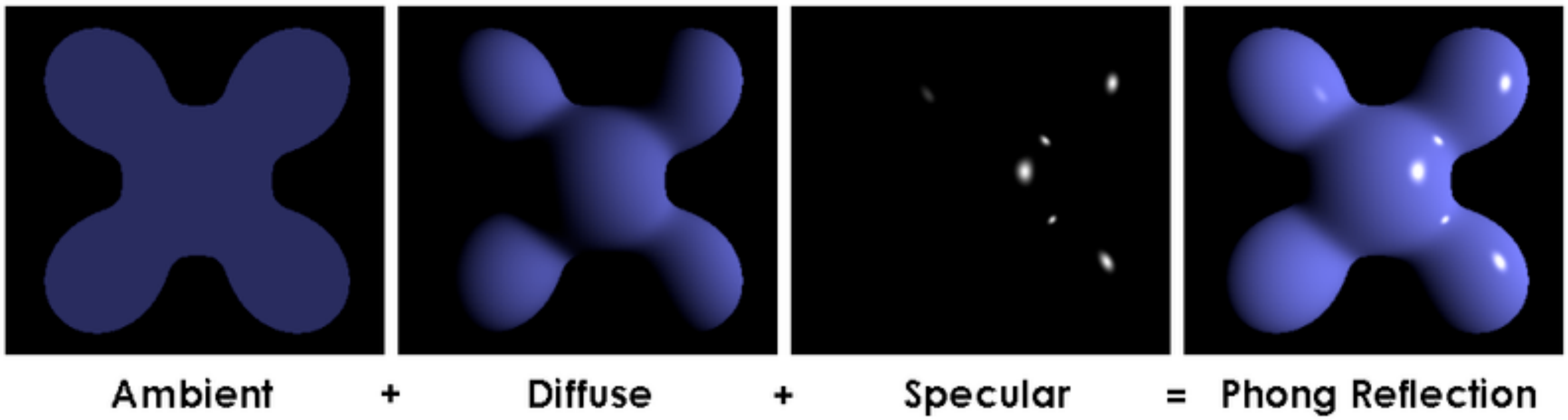
Phong exponent

$\alpha = 5..10$ plastic

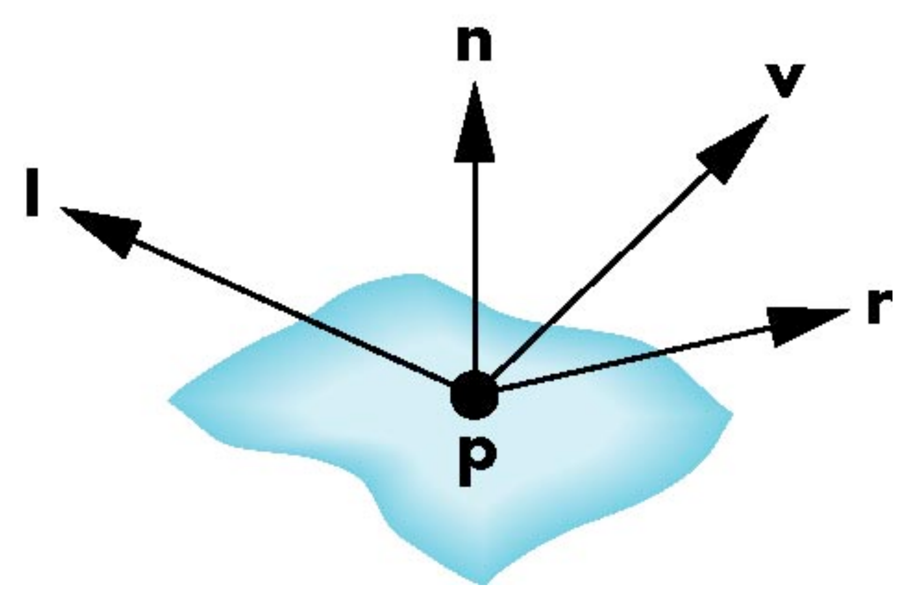
$\alpha = 100..200$ metal

Phong proposed this model
clamp to 0 -- avoid negative values
the fuzzy highlight was too big without an exponent

Phong Reflection Model

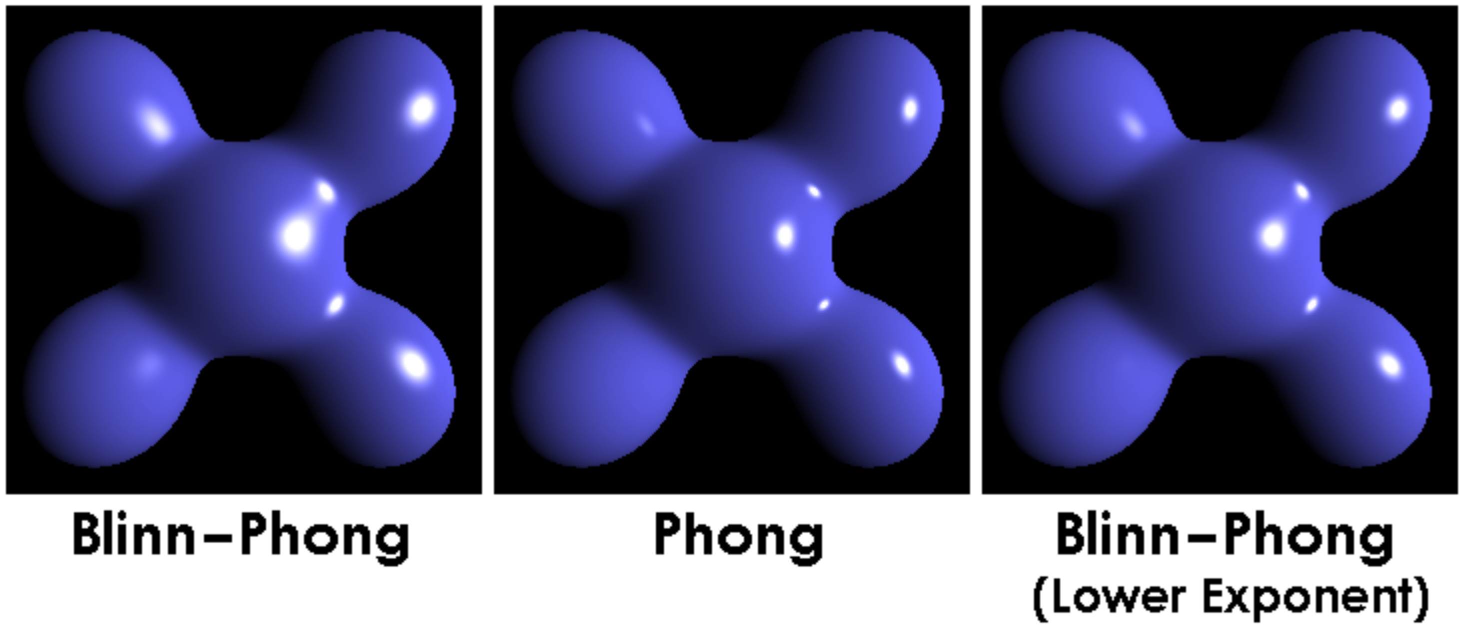


[Brad Smith, Wikimedia Commons]



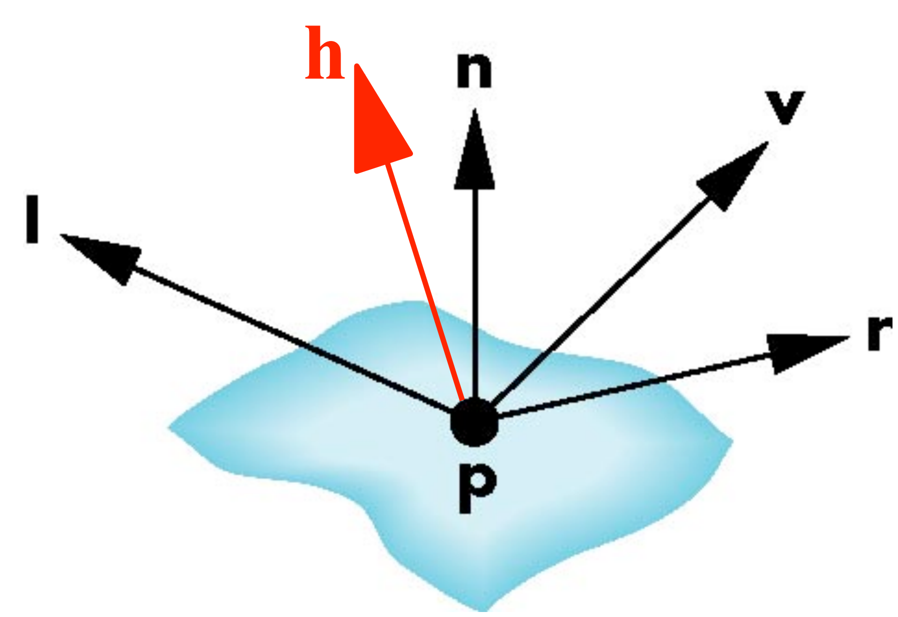
$$\begin{aligned} I &= I_a + I_d + I_s \\ &= \underbrace{R_a L_a}_{\text{Ambient}} + \underbrace{R_d L_d \max(0, \mathbf{l} \cdot \mathbf{n})}_{\text{Diffuse}} + \underbrace{R_s L_s \max(0, \mathbf{v} \cdot \mathbf{r})^\alpha}_{\text{Specular}} \end{aligned}$$

Alternative: Blinn-Phong Model



[Brad Smith, Wikimedia Commons]

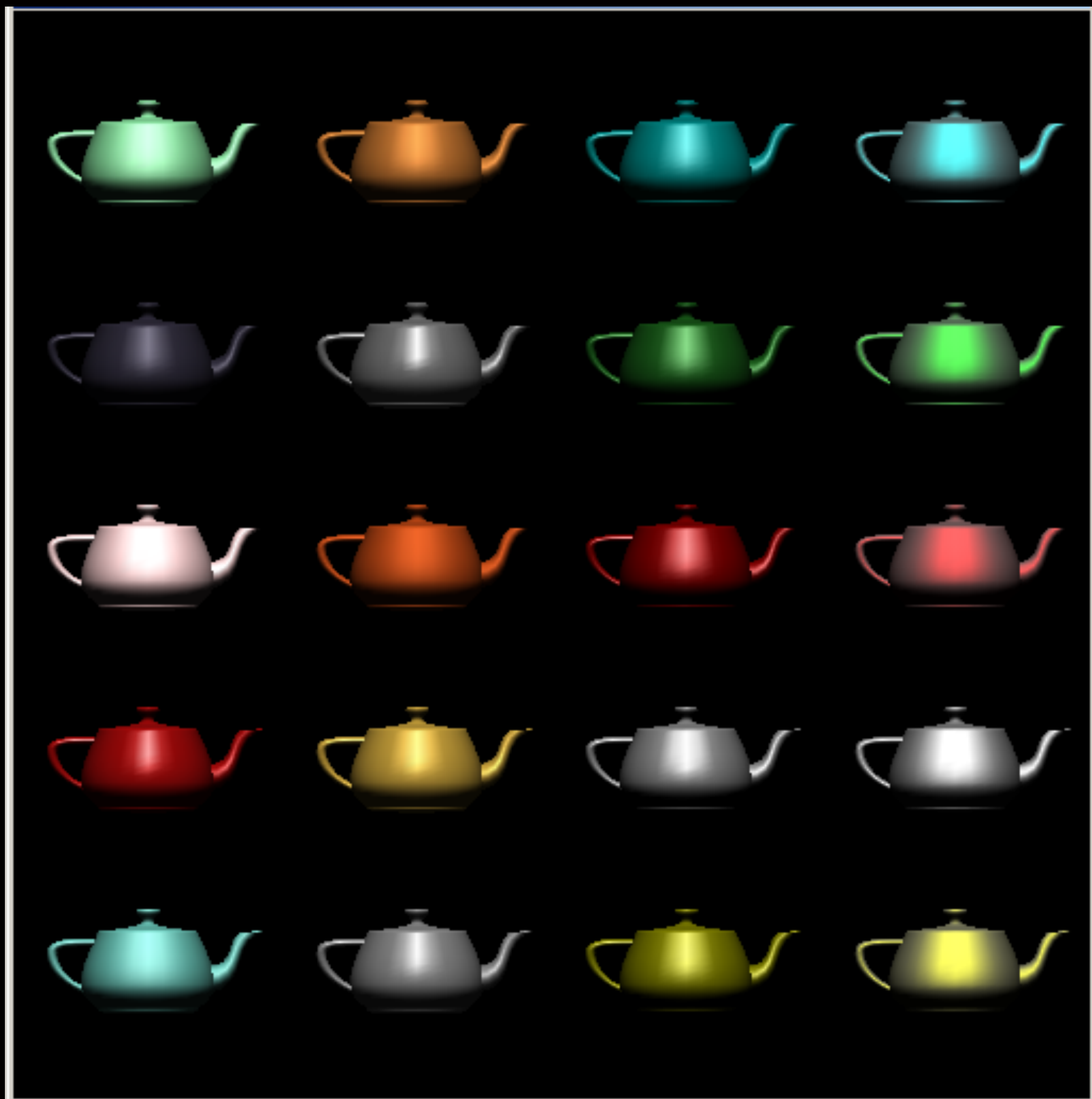
halfway vector $$\boxed{\mathbf{h} = \frac{\mathbf{l} + \mathbf{v}}{|\mathbf{l} + \mathbf{v}|}}$$



$$\begin{aligned} I &= I_a + I_d + I_s \\ &= R_a L_a + R_d L_d \max(0, \mathbf{l} \cdot \mathbf{n}) + R_s L_s \max(0, \mathbf{h} \cdot \mathbf{n})^{\alpha} \end{aligned}$$

Ambient Diffuse Specular

replace **v.r** with **h.n**
this way we don't have to recompute **r**, which depends on **n**
h does not depend on **n**
saves a lot especially for directional lights and constant viewing direction



α

10: eggshell
100: shiny
1000: glossy
10000: mirror-like