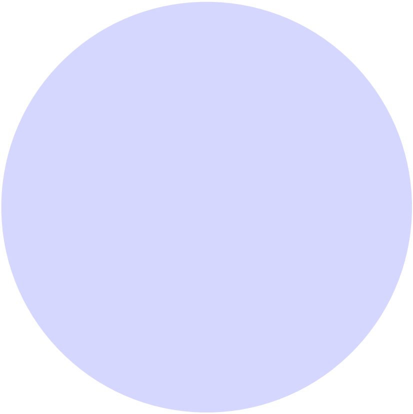
CS130 : Computer Graphics

Lecture 12: Lighting and Shading

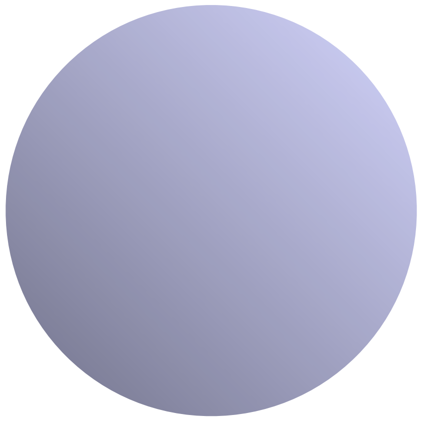
Tamar Shinar
Computer Science & Engineering
UC Riverside

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



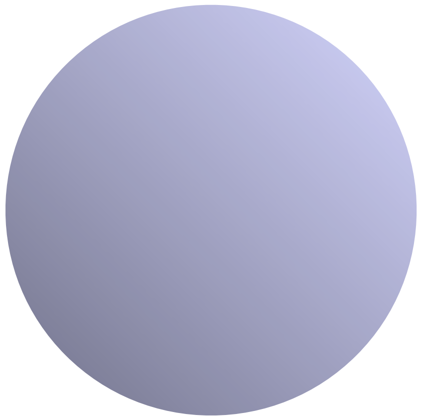
- But we want



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

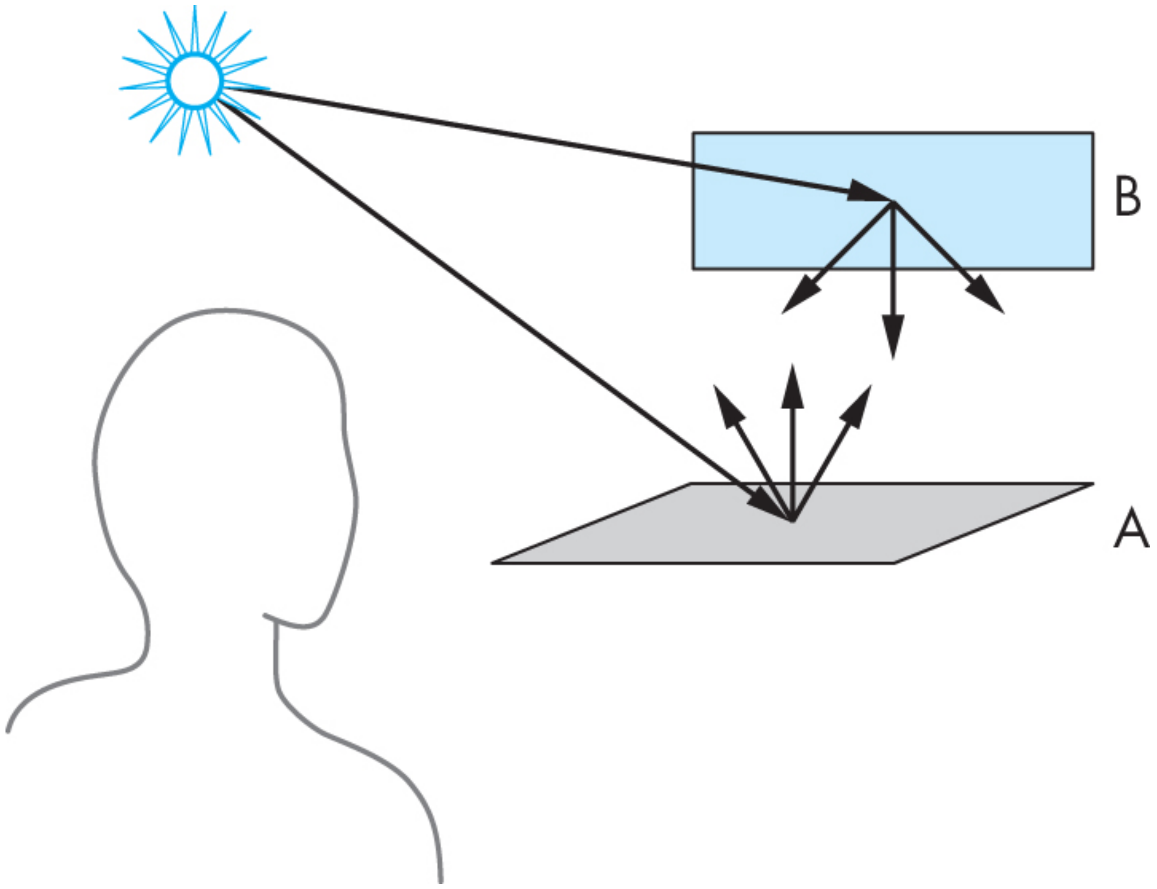


- Light-material interactions cause each point to have a different color or shade
- Need to consider
  - Light sources
  - Material properties
  - Location of viewer
  - Surface orientation (normal)

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 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]

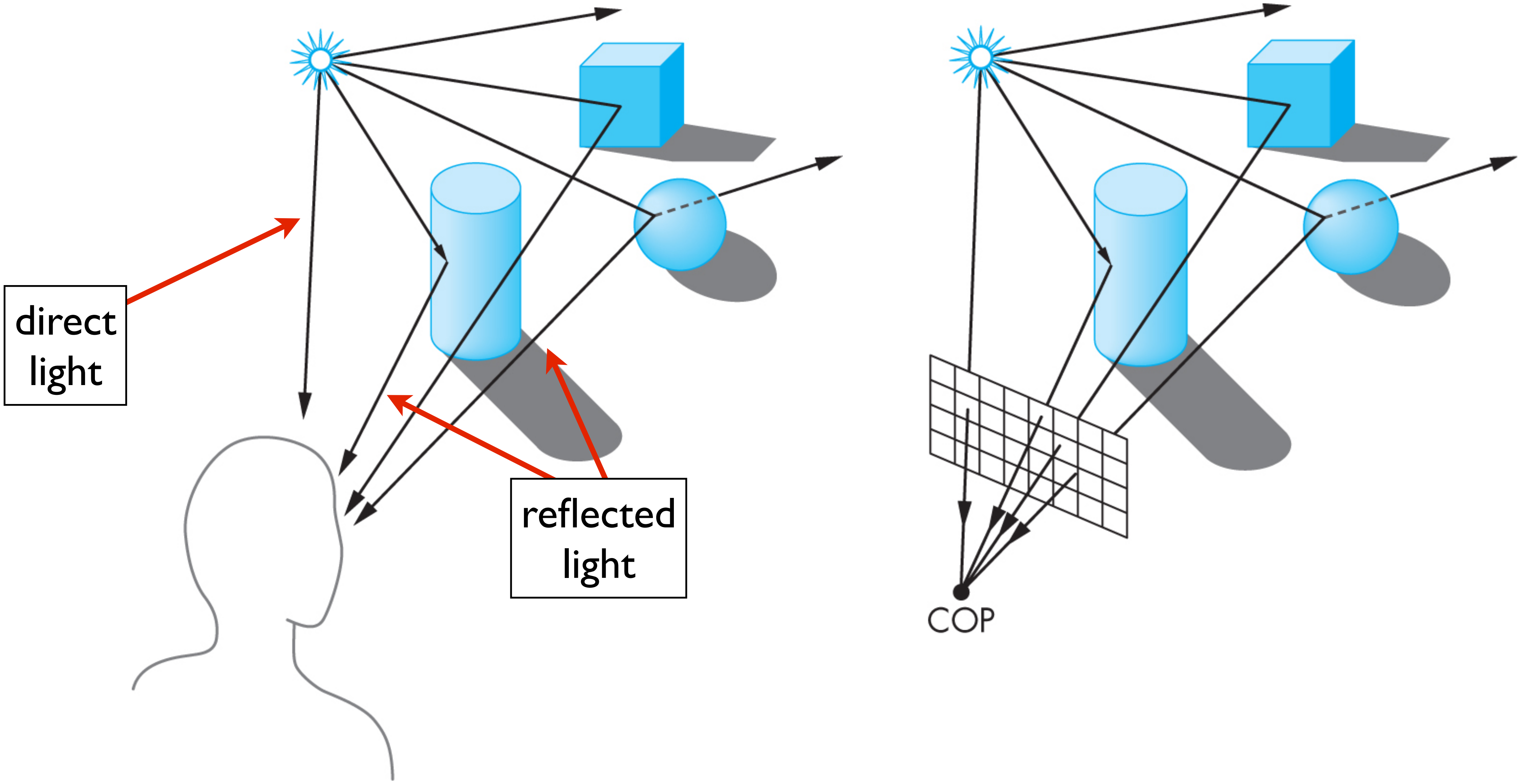
http://en.wikipedia.org/wiki/Rendering_equation

Fast local shading models

- the rendering equation can’t be solved analytically
- numerical methods aren’t fast enough for real-time
- for our fast graphics rendering pipeline, 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

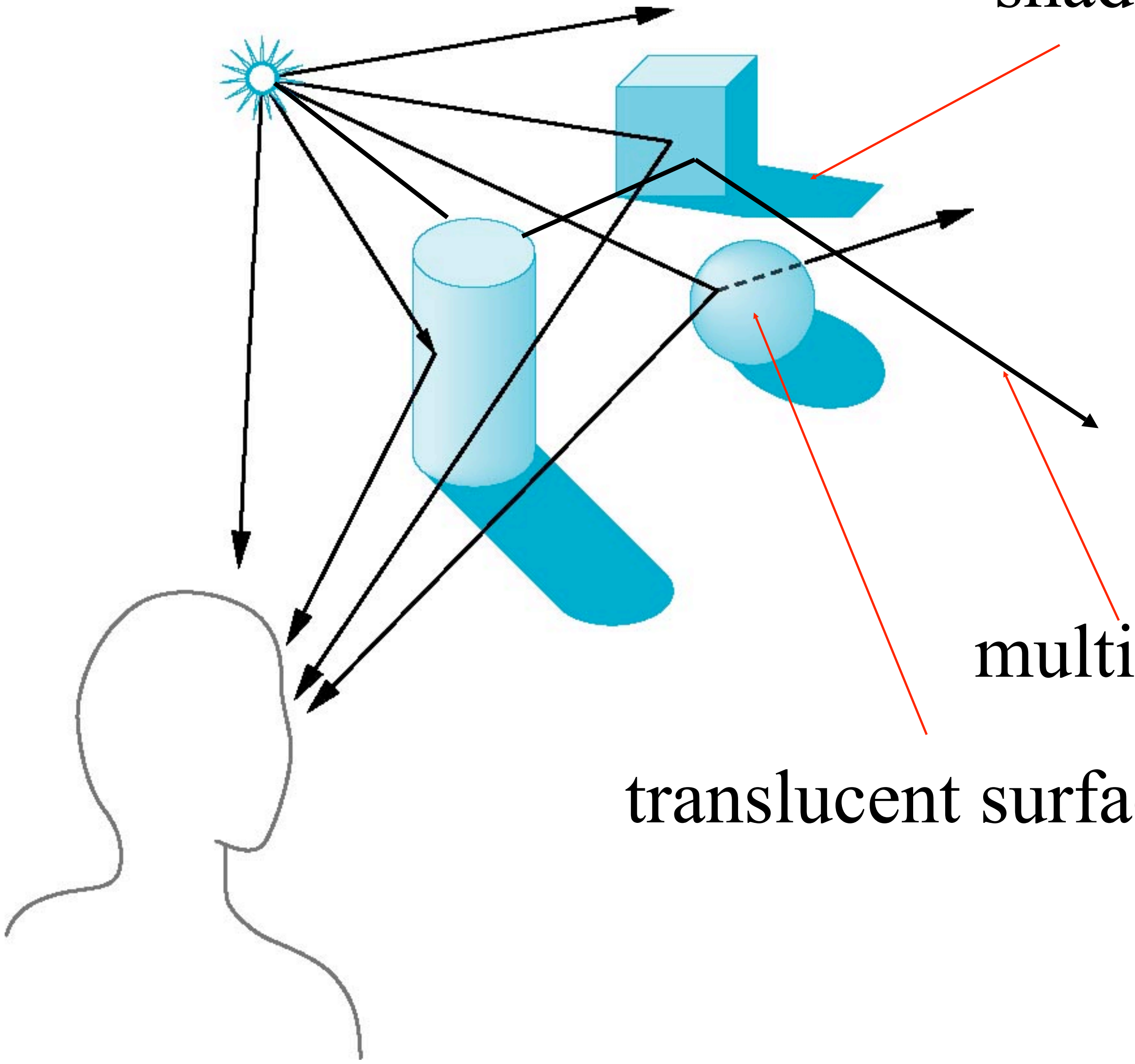
Local shading model



[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



shadow

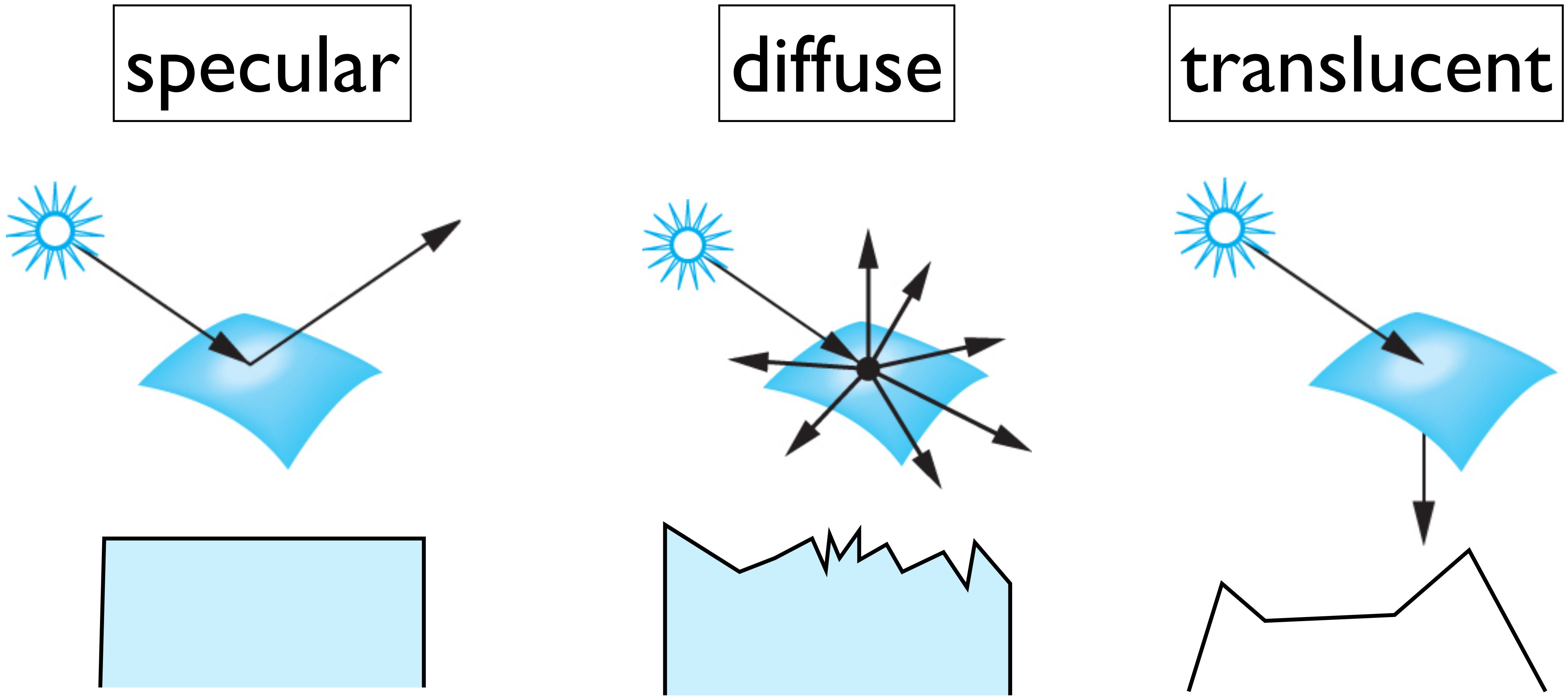
multiple reflection

translucent surface

[Angel and Shreiner]

Light-material interactions

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



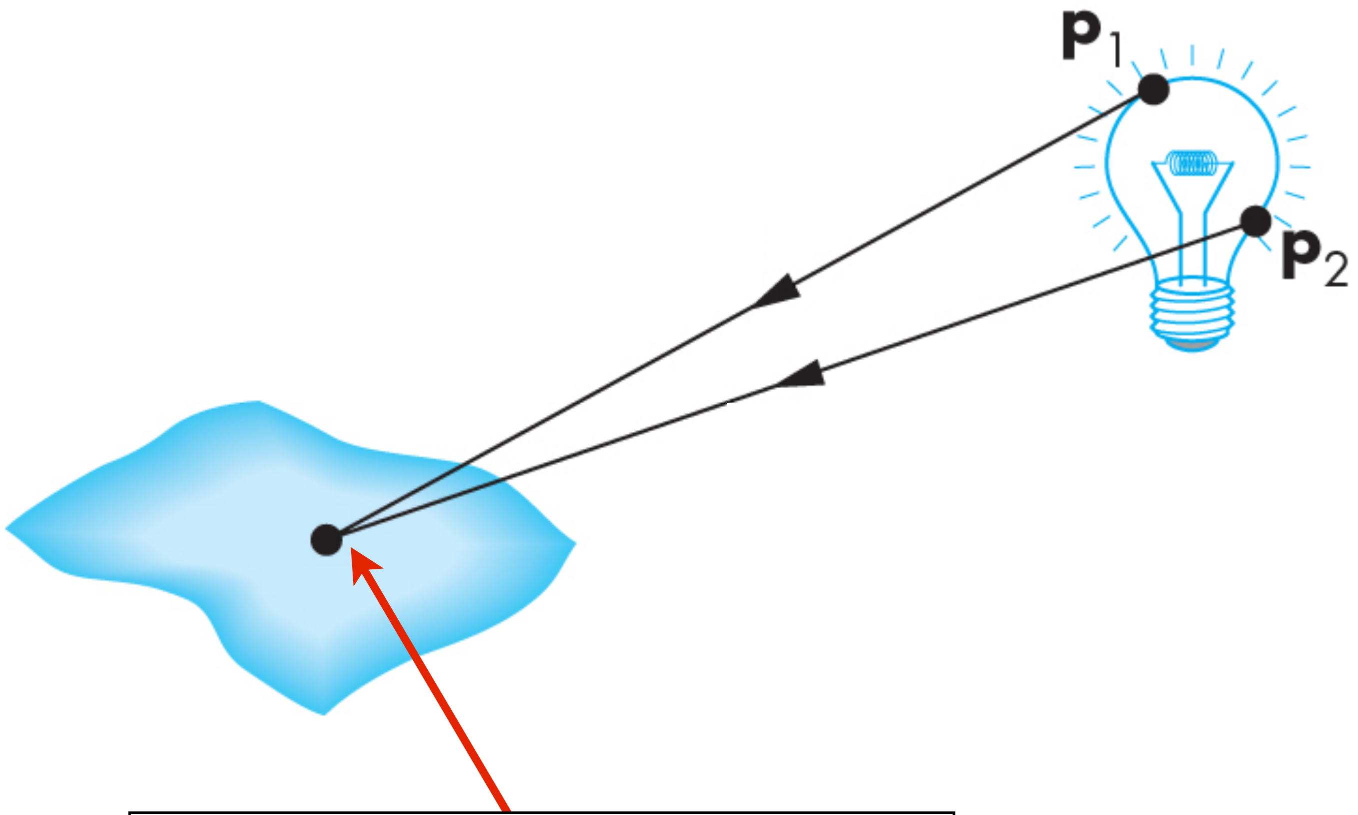
[Angel and Shreiner]

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:

$$L(\mathbf{x}, \omega, \lambda)$$



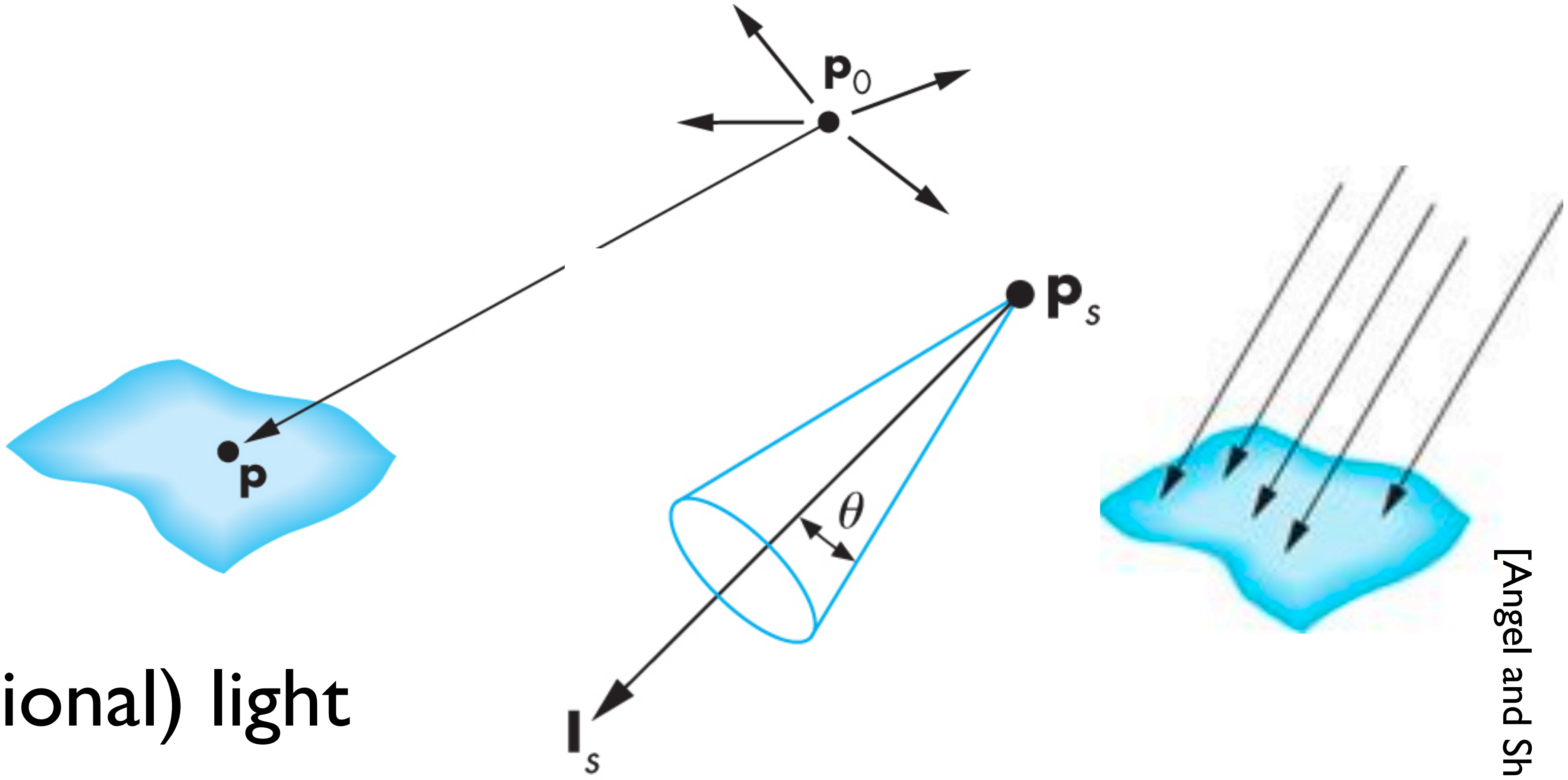
integrate contributions from all sources to shade the point

[Angel and Shreiner]

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

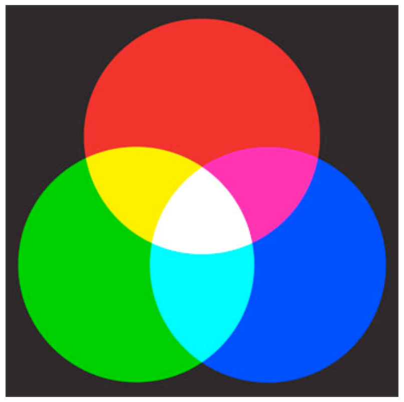
Idealized light sources

- 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

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

$$L_a$$

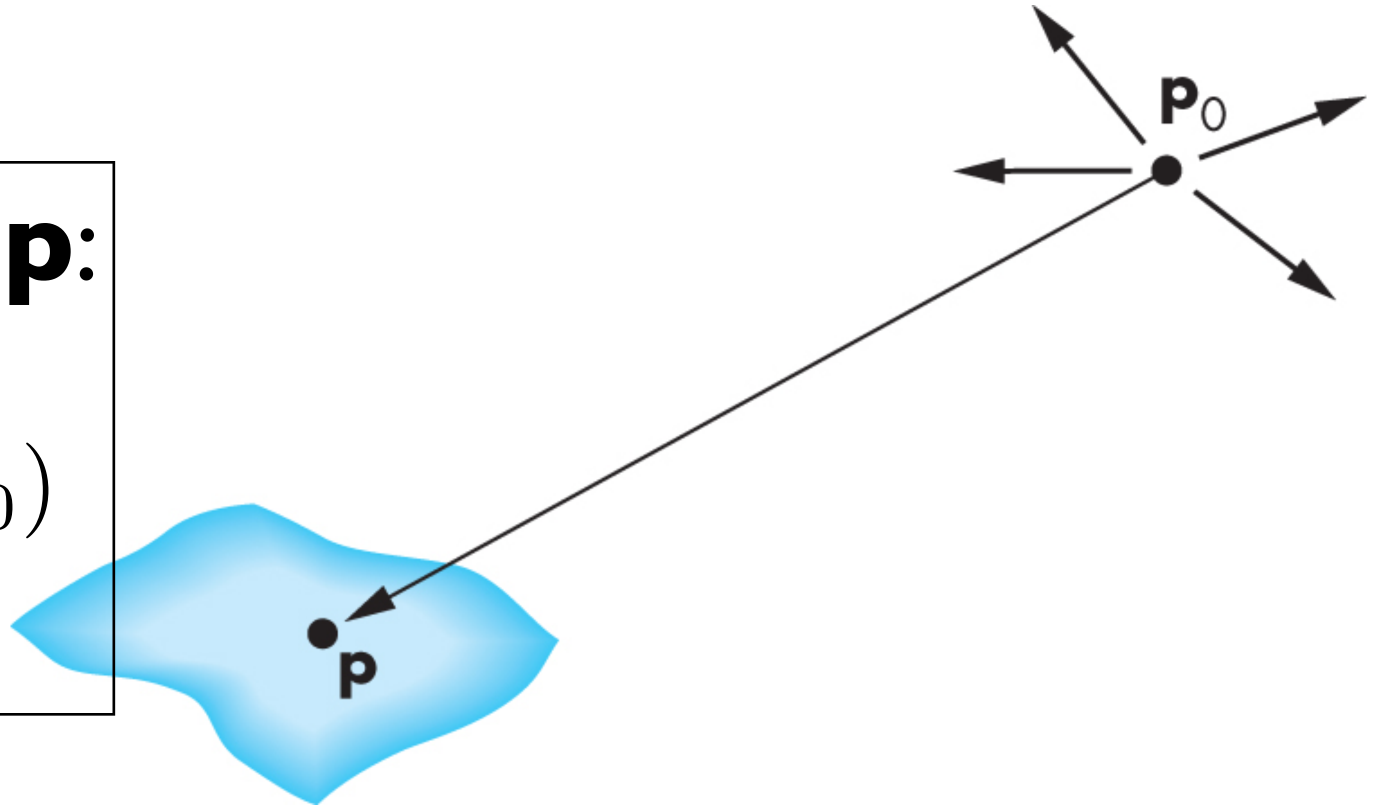
use scalar I_a to denote any component of \vec{I}_a
ambient light is the same everywhere
but different surfaces will **reflect** it differently

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**:

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

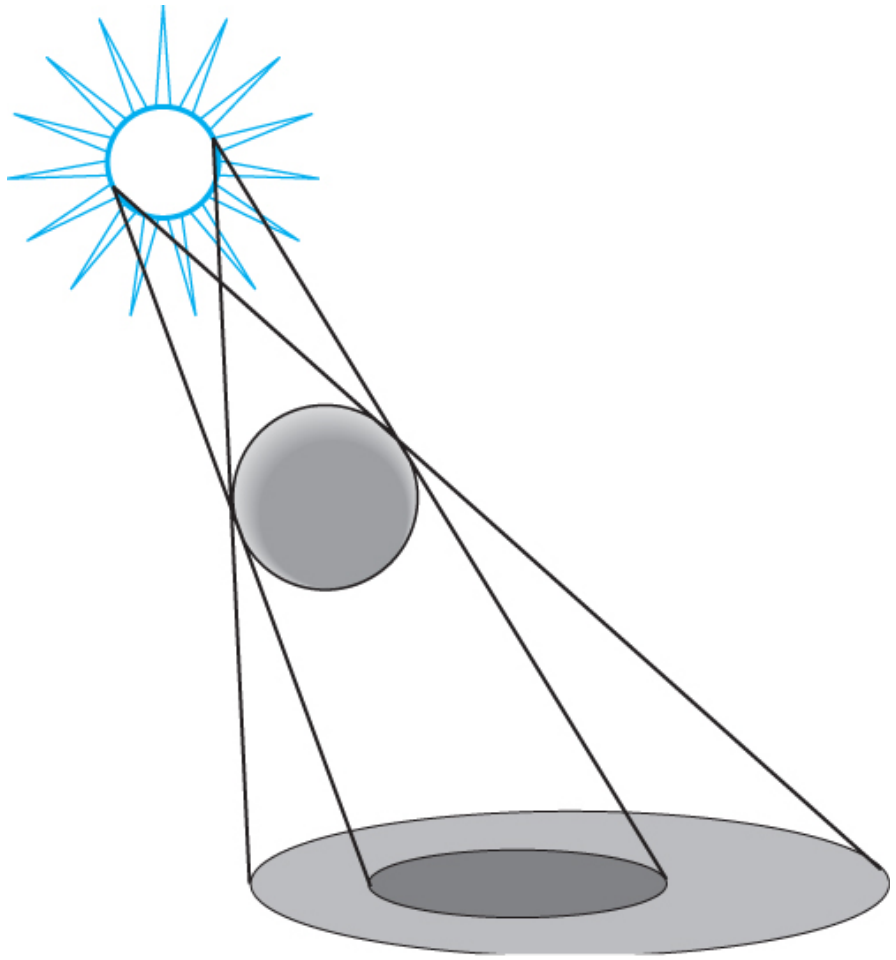


[Angel and Shreiner]

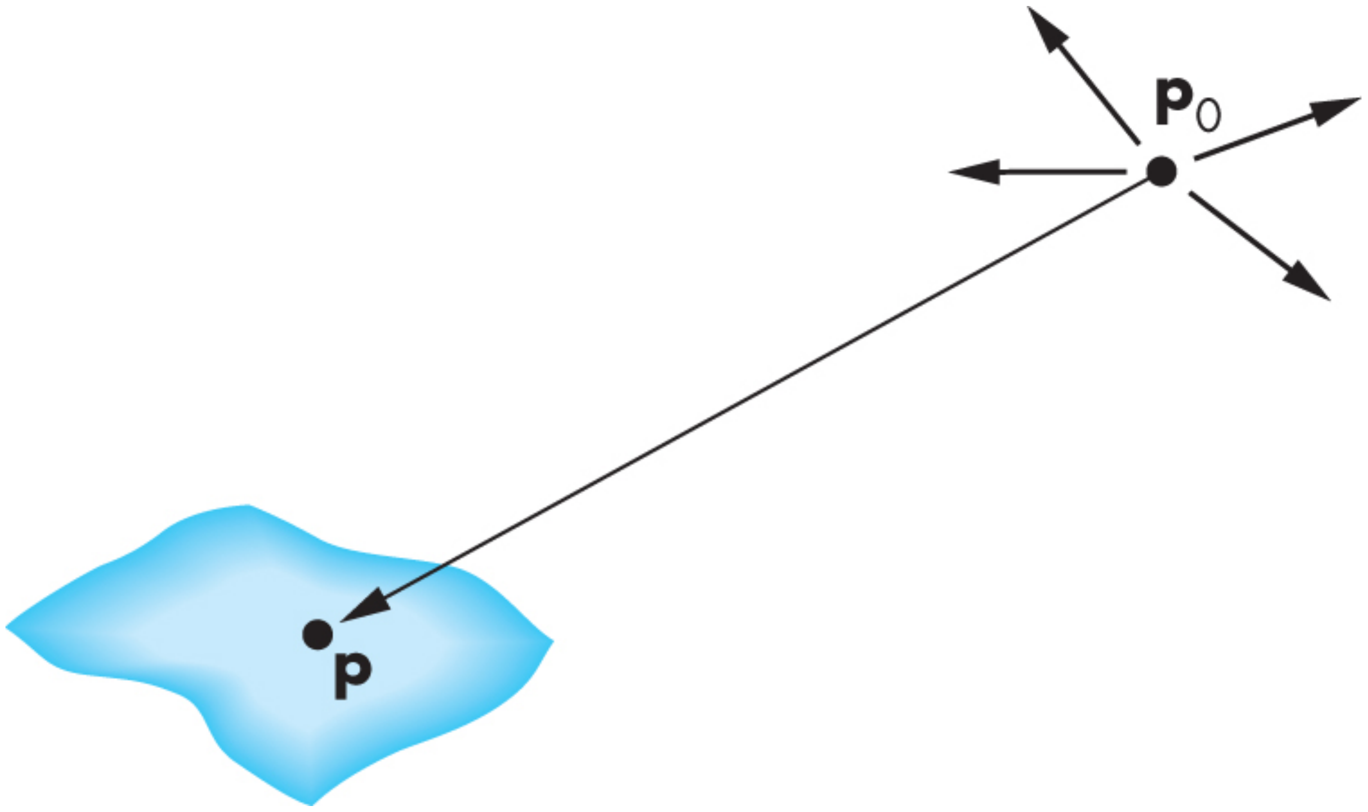
- use scalar I(\vec{p}_0) to denote any of three components
- points sources alone aren’t too realistic looking -- tend to be high contrast
- most real-world scenes have large light sources
- add ambient light to mitigate high contrast

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

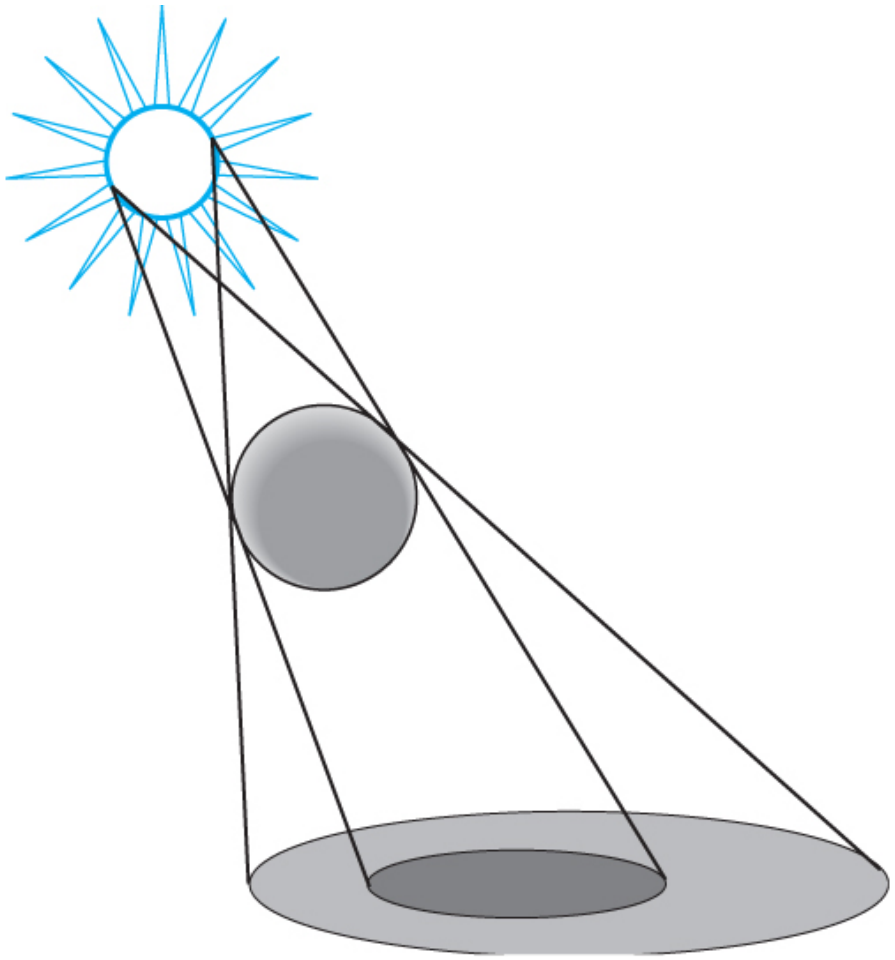


[Angel and Shreiner]

– **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

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

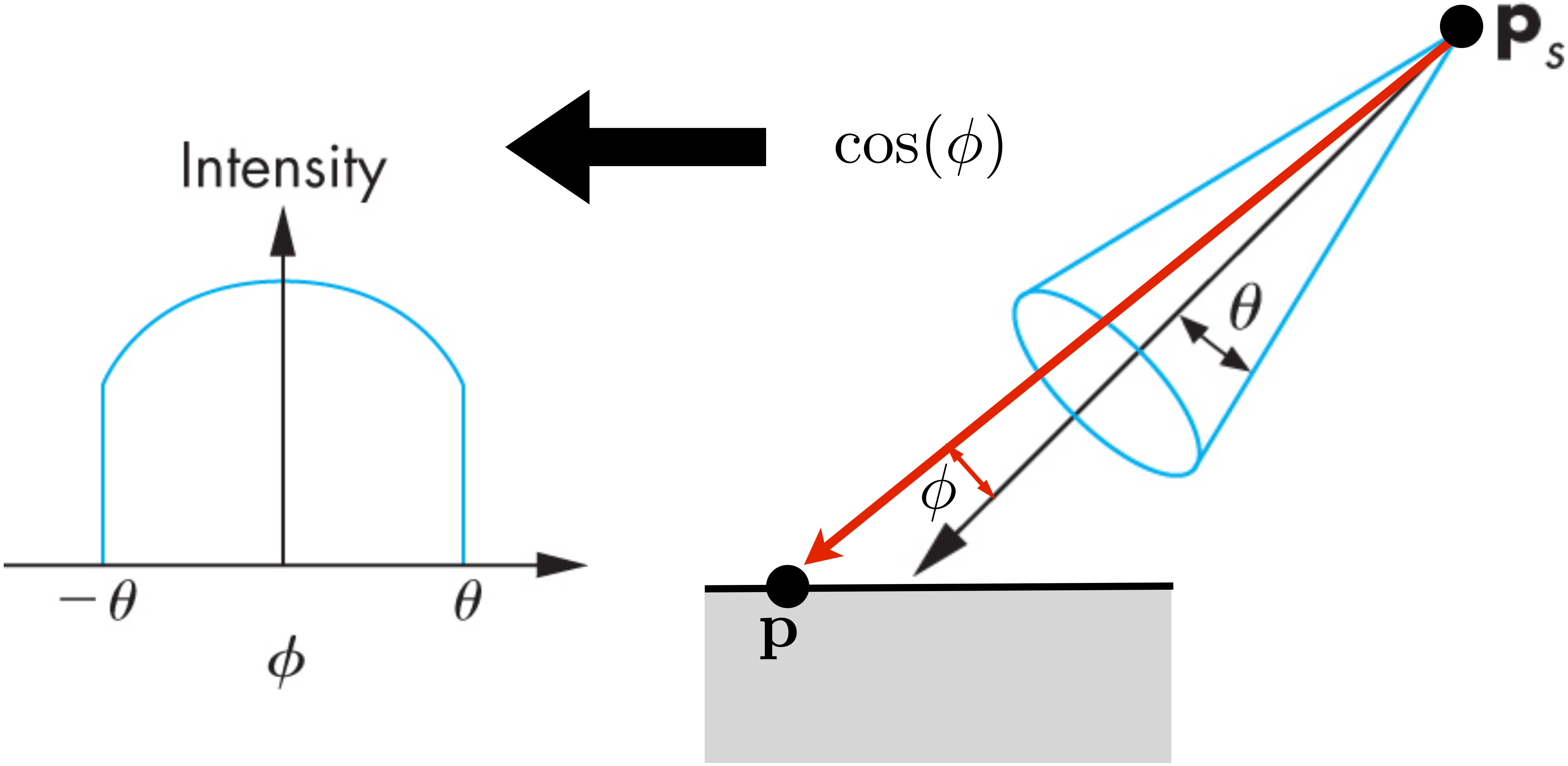
$$\Downarrow$$

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

[Angel and Shreiner]

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

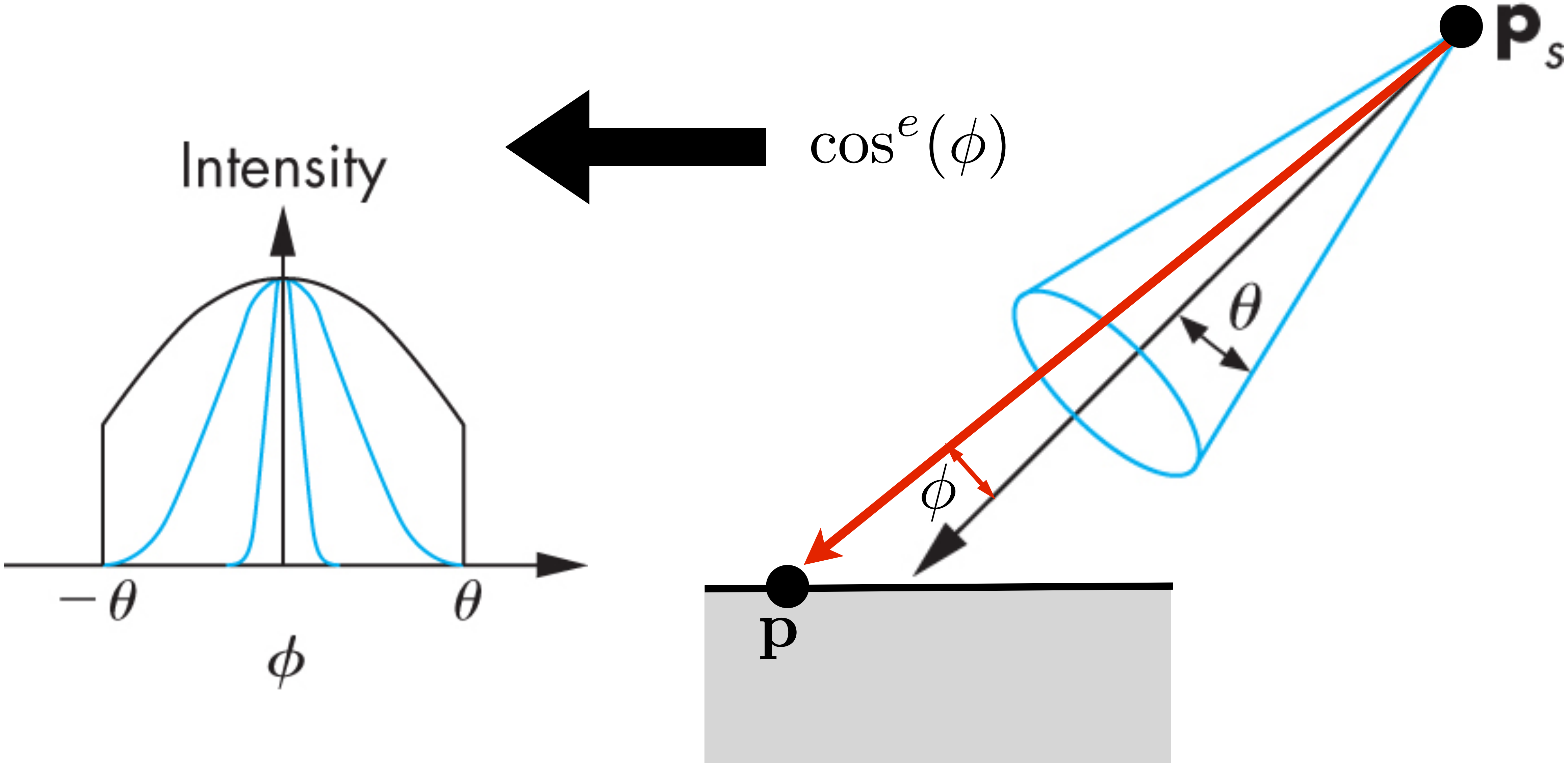
Spotlights



$\cos(\phi)$

[Angel and Shreiner]

Spotlights



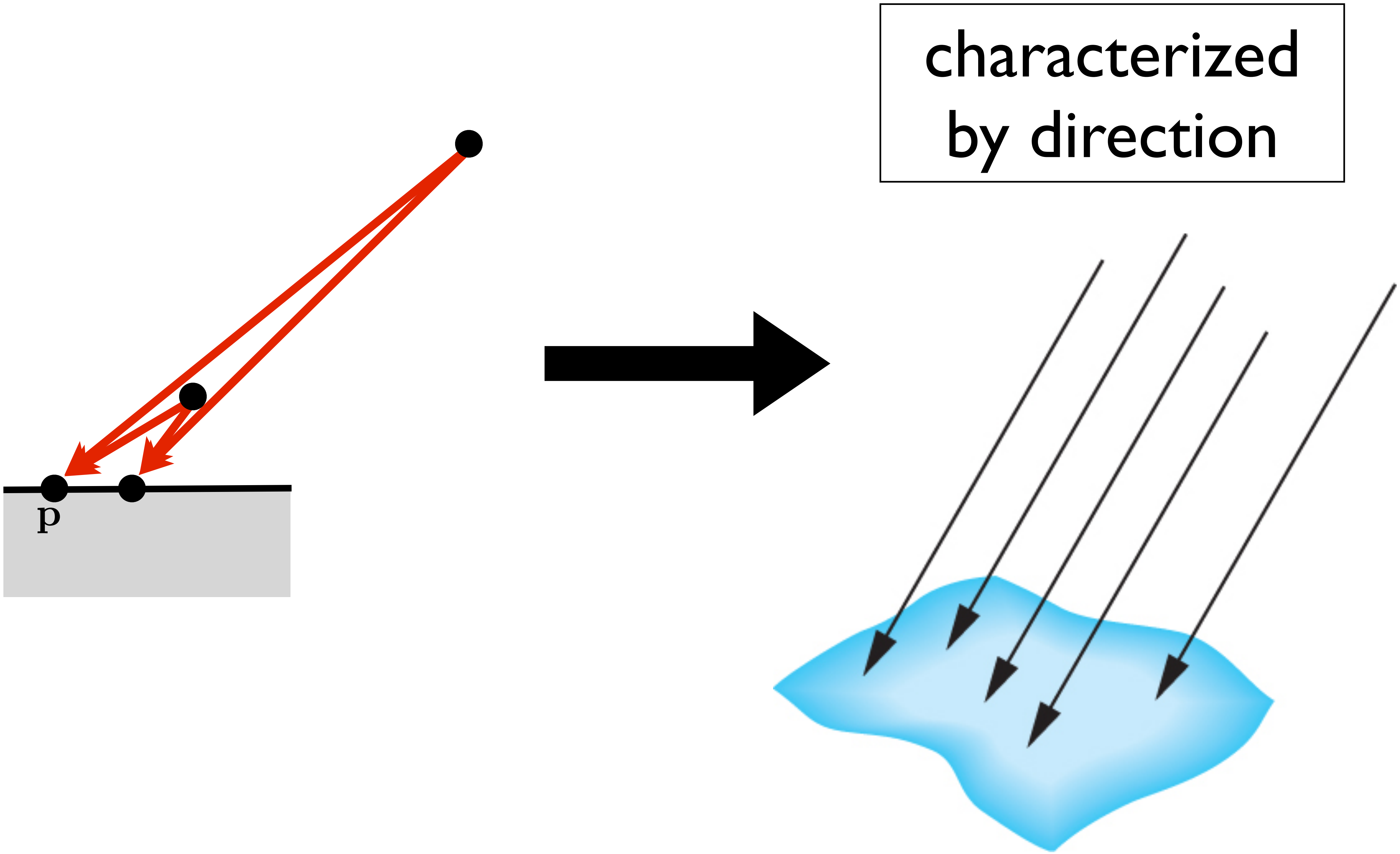
$\cos^e(\phi)$

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

[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