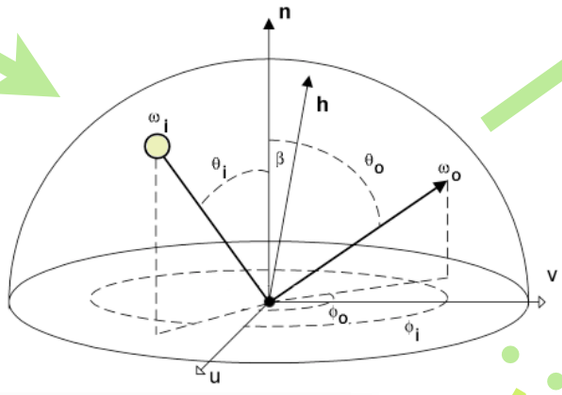
Shading

Part I: What is Light?

Part IV: Lighting

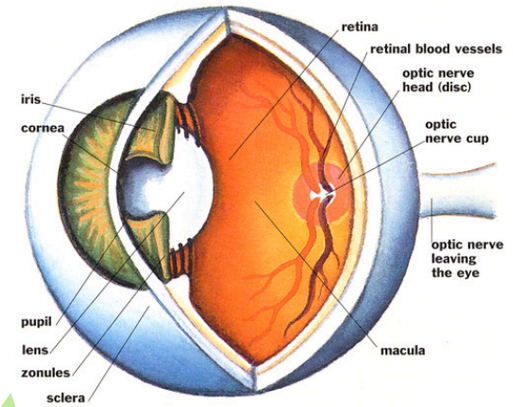
Light



Part III: Reflectance

No Light

Light



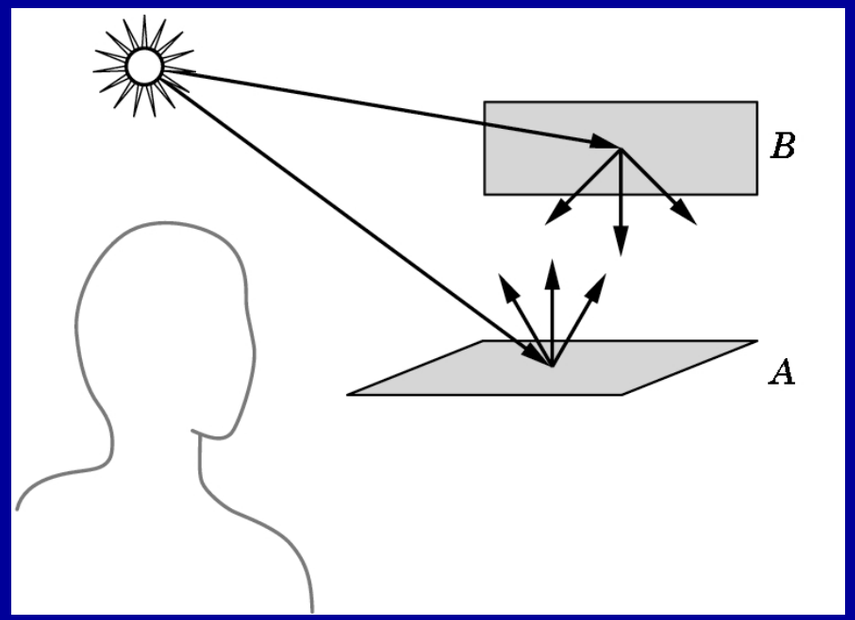
Light

Part II: The Eye

Light

Part V: Shadows

Shading: Illumination

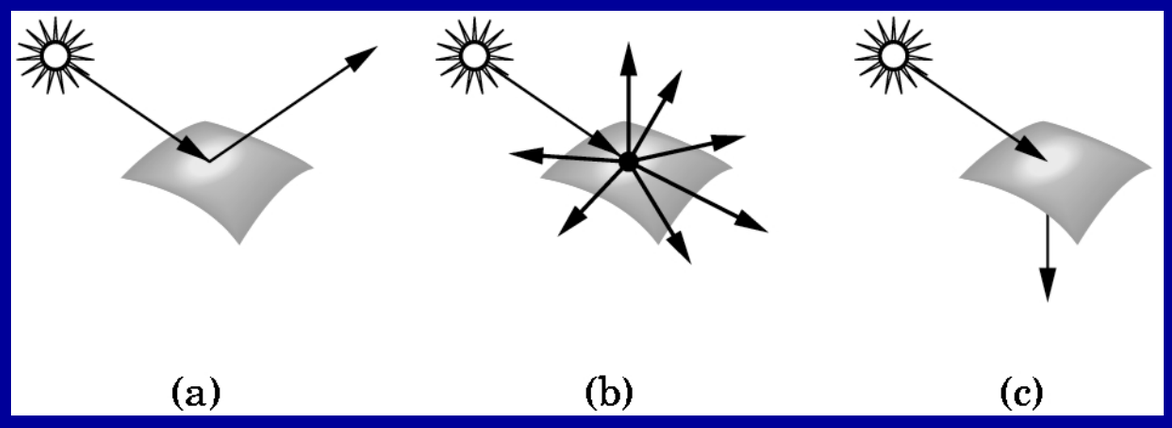


Light Sources emit light

- EM spectrum
- Position and direction

Surfaces reflect light

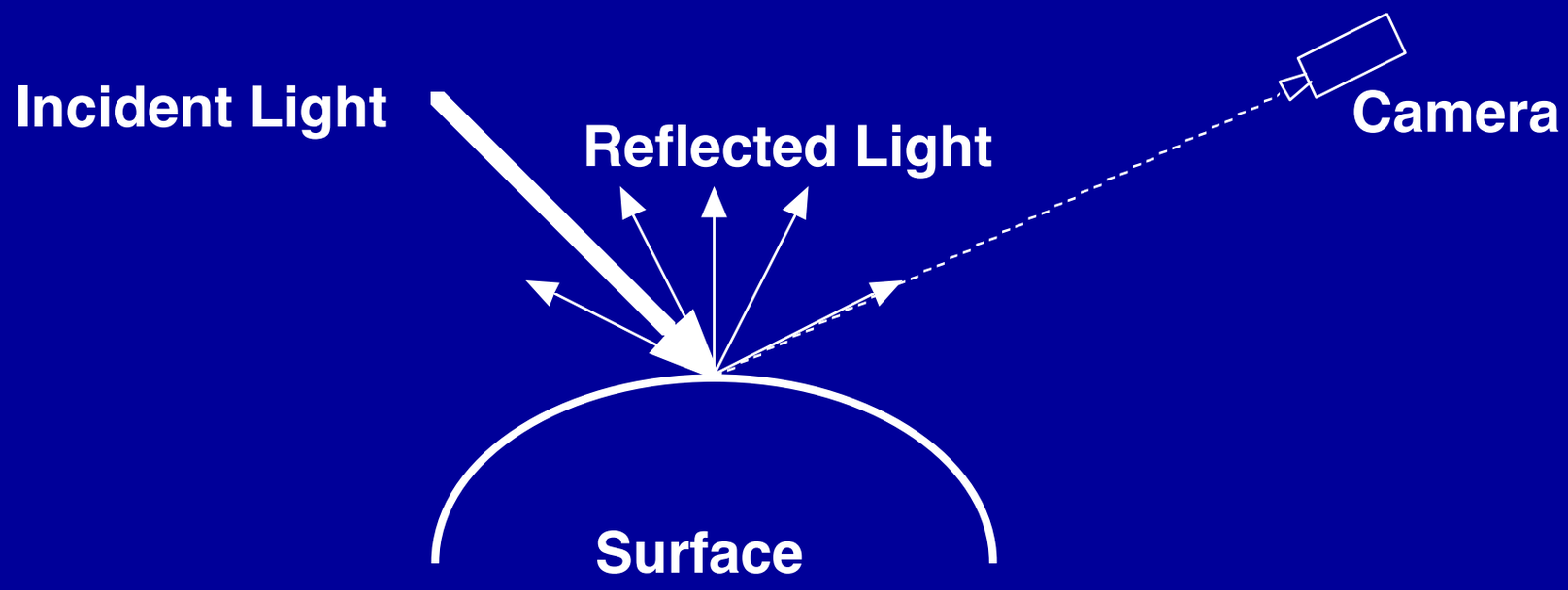
- Reflectance
- Geometry (position, orientation, micro-structure)
- Absorption
- Transmission



Illumination determined by the interactions between light sources and surfaces

Surface Reflection

- When light hits an opaque surface some is absorbed, the rest is reflected (some can be transmitted too--but ignore that for now)
- The reflected light is what we see
- Reflection is not simple and varies with material
  - the surface's micro structure define the details of reflection
  - variations produce anything from bright specular reflection (mirrors) to dull matte finish (chalk)



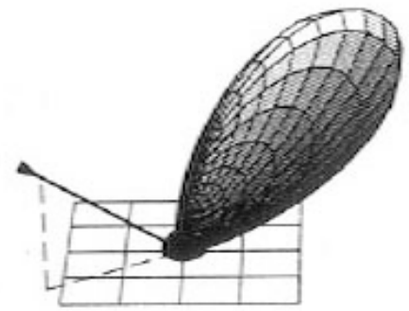
BRDF

Surface scattering can be modeled as a function that maps light from incoming (light) directions ω_{in} to outgoing (viewing) directions ω_{out}:

$$f_r(\omega_{in}, \omega_{out})$$

This function is called the **Bi-directional Reflectance Distribution Function** (**BRDF**).

Here's a plot with ω_{in} held constant:



ω_{in}

$f_r(\omega_{in}, \omega_{out})$

BRDF's can be quite sophisticated…

BRDFs (Continued)

BRDF Model:

$$L_o(\omega_o) = \int_\Omega L_i(\omega_i) f(\omega_i, \omega_o) d\omega$$

Constraint:

$$\int_\Omega f(\omega_i, \omega_o) d\omega \leq 1$$

What we will learn about today

Light (with color)

Specular highlights

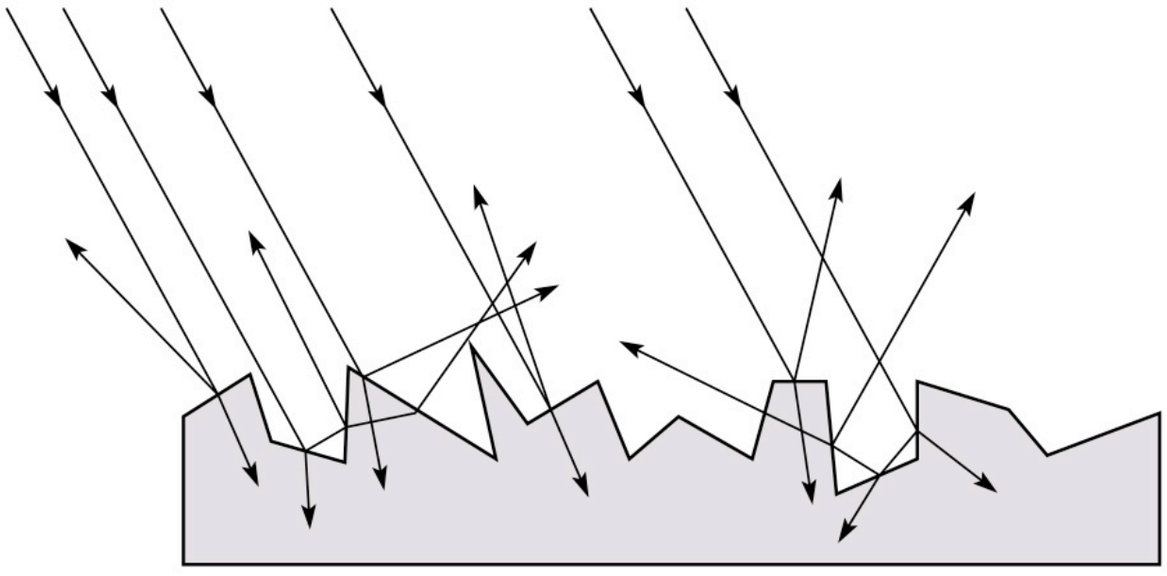
Shadows

No transmission of light through surfaces

No reflections from other surfaces

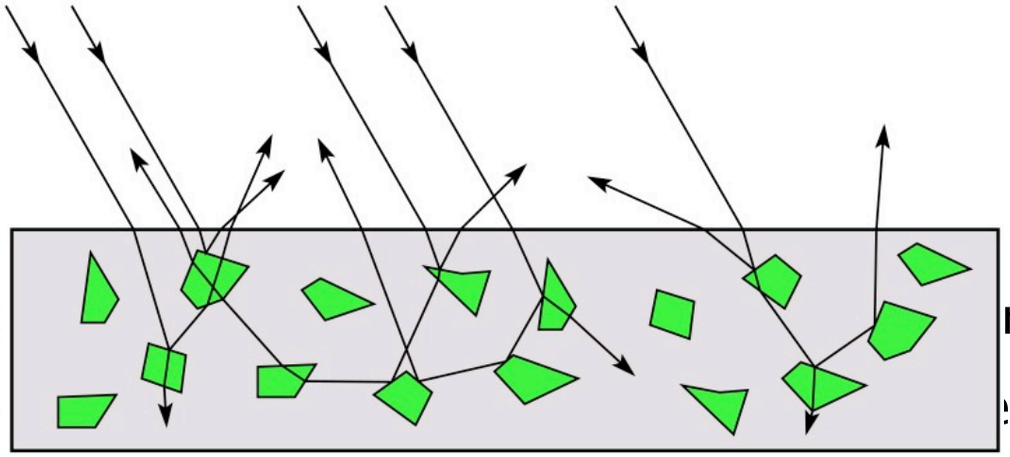
Diffuse reflectors

Diffuse reflection occurs from dull, matte surfaces, like latex paint, or chalk.
These **diffuse** or **Lambertian** reflectors reradiate light equally in all directions.
Picture a rough surface with lots of tiny **microfacets**.



Diffuse reflectors

…or picture a surface with little pigment particles embedded beneath the surface (neglect reflection at the surface for the moment):



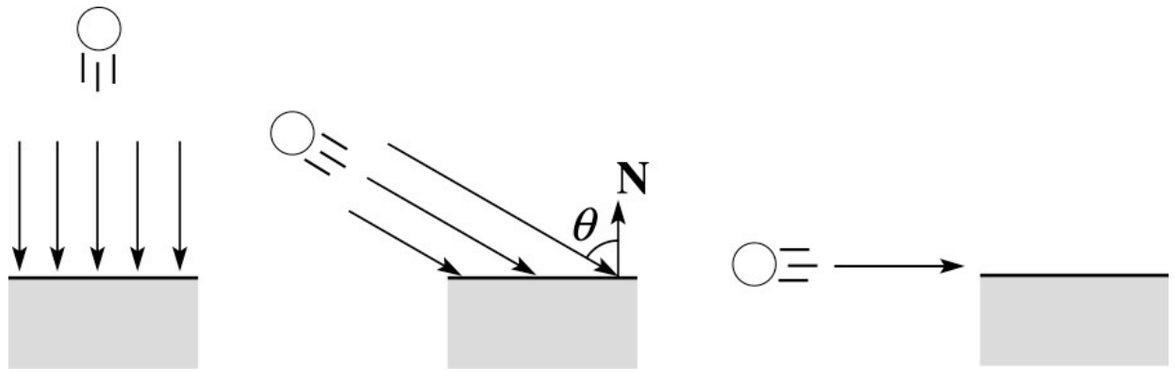
The microfacets and pi ns.

Embedded pigments ar ly reflected light in plastics and paints.

Note: the figures above are intuitive, but not strictly (physically) correct.

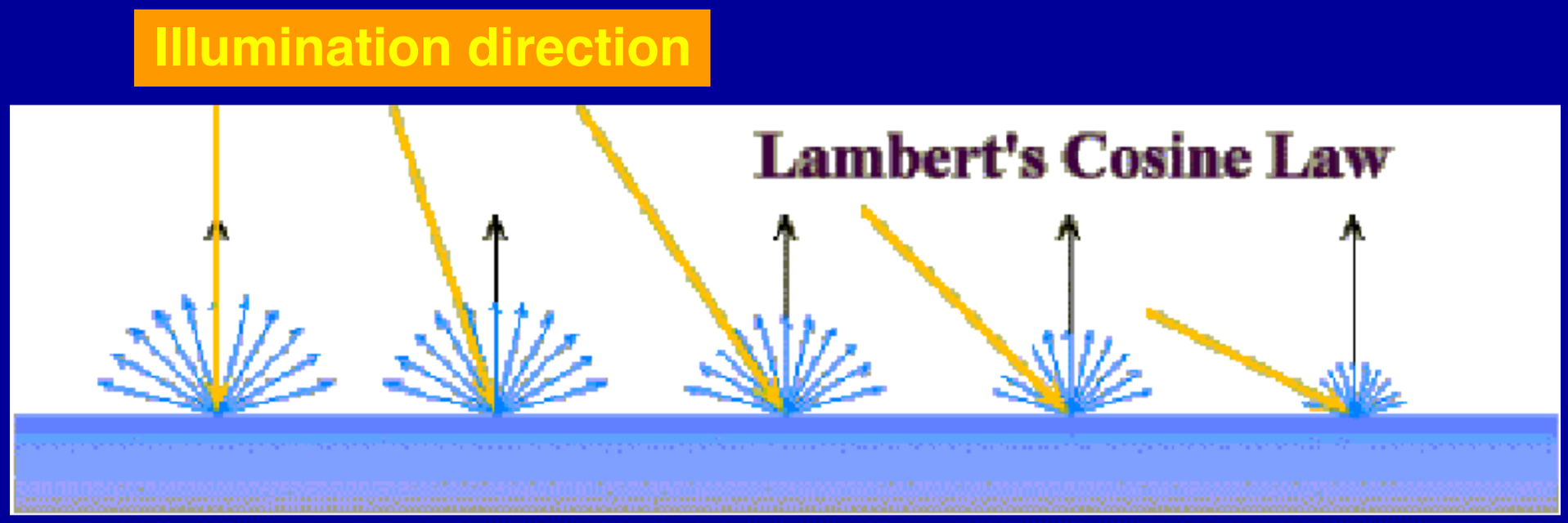
Diffuse reflectors, cont.

The reflected intensity from a diffuse surface does not depend on the direction of the viewer. The incoming light, though, does depend on the direction of the light source:



Diffuse Reflection

- Simplest kind of reflector (also known as *Lambertian Reflection*)
- Models a matte surface -- rough at the microscopic level
- Ideal diffuse reflector
  - incoming light is scattered equally in all directions
  - viewed brightness does not depend on viewing direction
  - brightness *does* depend on direction of illumination



Lambertian Shading Model

$$c \propto \cos\theta$$

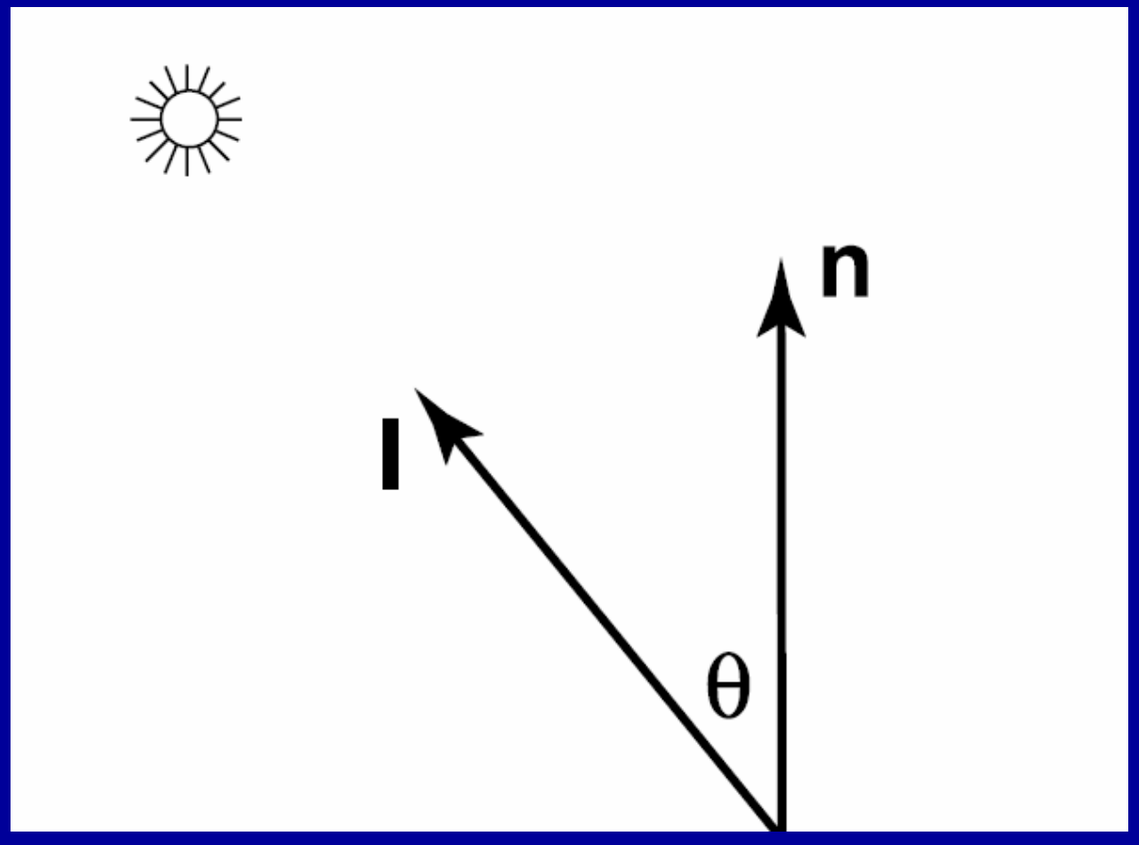
$$c \propto n \bullet l$$

n : surface normal

l : direction to light

θ : Light/Normal angle

$$\cos\theta = \frac{n \bullet l}{|n||l|}$$



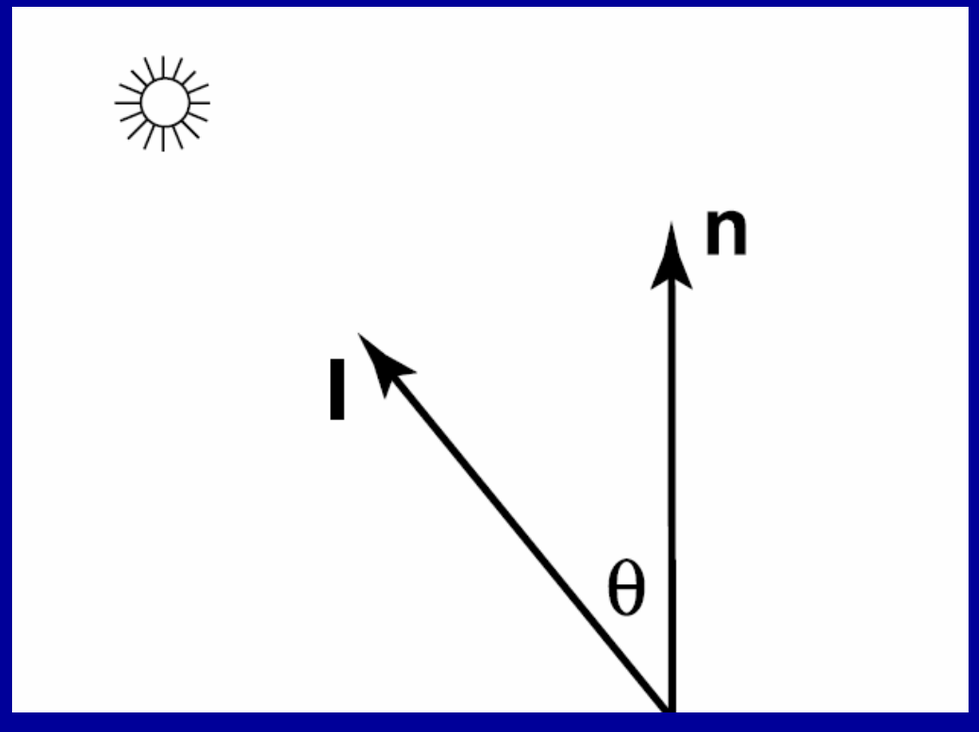
Lambert's Law

$$I_{diffuse} = k_d I_{light} \cos\theta$$
$$= k_d I_{light} (n \bullet l)$$

I_{light} : Light Source Intensity

k_d : Surface reflectance coefficient in [0,1]

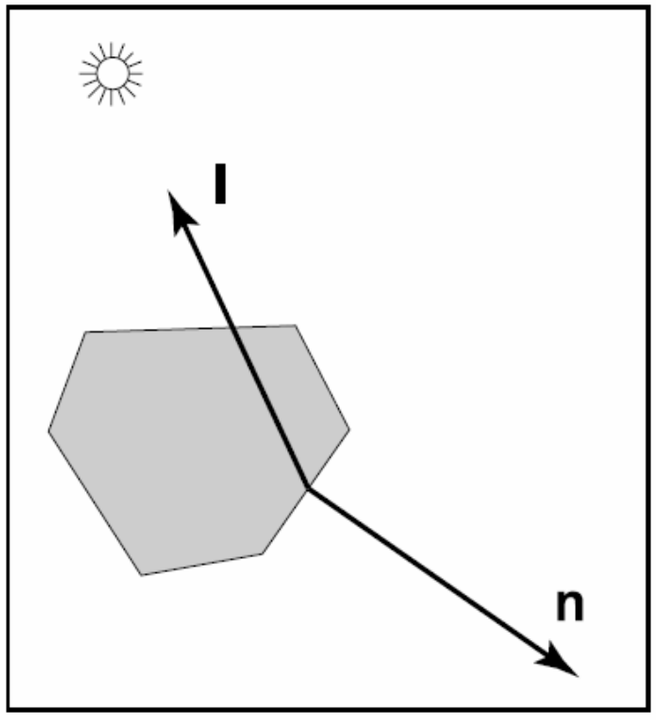
θ : Light/Normal angle

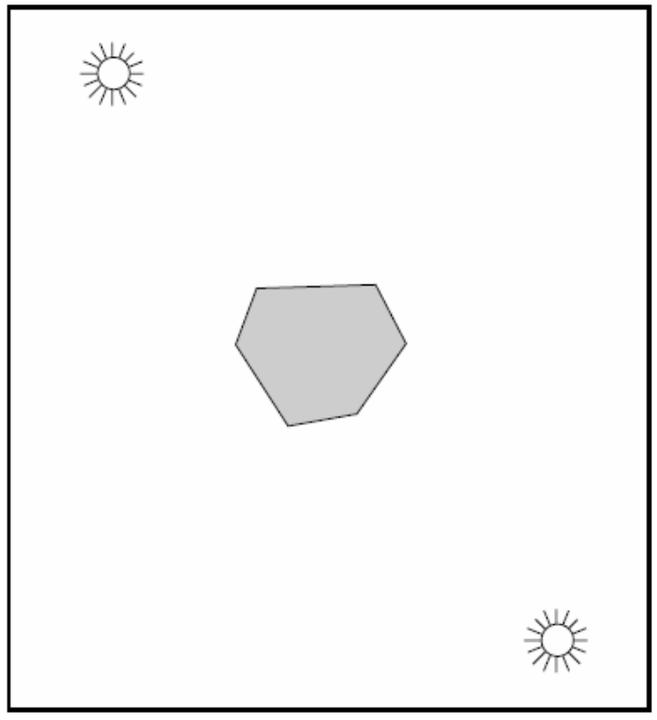


What happens for surfaces facing away from the light?

$$c = c_r c_l \max(0, n \bullet l)$$

$$c = c_r c_l \left| n \bullet l \right|$$ Two-sided light





Wavelength dependence

Really, k_e, k_a, and L_a are functions over all wavelengths λ.

Ideally, we would do the calculation on these functions. For the ambient shading equation, we would start with:

$$I(\lambda) = k_a(\lambda) L_a(\lambda)$$

then we would find good RGB values to represent the spectrum $I(\lambda)$.

Traditionally, though, k_a and I_a are represented as RGB triples, and the computation is performed on each color channel separately:

$$I_R = k_{a,R}\ L_{a,R}$$
$$I_G = k_{a,G}\ L_{a,G}$$
$$I_B = k_{a,B}\ L_{a,B}$$

Examples of Diffuse Illumination

Same sphere lit diffusely from different lighting angles

What happens with surfaces facing away from the light?

- Pitch black—not exactly realistic

How to solve?

- Several light sources—dim light source at eye, for example
- Ambient light

Ambient + Diffuse Reflection

$$I_{d+a} = k_a I_a + k_d I_{light}(n \bullet l)$$

$$c = c_r (c_a + c_l \max(0, n \bullet l)$$

I_a : Ambient light intensity (global)

k_a : Ambient reflectance (local)

Diffuse illumination plus a simple ambient light term

a TRICK to account for a background light level caused by multiple reflections from all objects in the scene (less harsh appearance)

Further Simple Illumination Effects

- Light attenuation:
  - light intensity falls off with the square of the distance from the source - so we add an extra term for this

$$I_{d+a} = k_a I_a + f_{att} k_d I_{light} (n \bullet l) \qquad \text{where} \quad f_{att} = \frac{1}{d^2}$$

with d the light source to surface distance—more complicated formulae are possible

- Colored lights and surfaces:
  - just have three separate equations for RGB
- Atmospheric attenuation:
  - use viewer-to-surface distance to give extra effects
  - the distance is used to blend the object's radiant color with a "far" color (e.g., a nice hazy gray)

Specular reflection

Specular reflection accounts for the highlight that you see on some objects.

It is particularly important for smooth, shiny surfaces, such as:

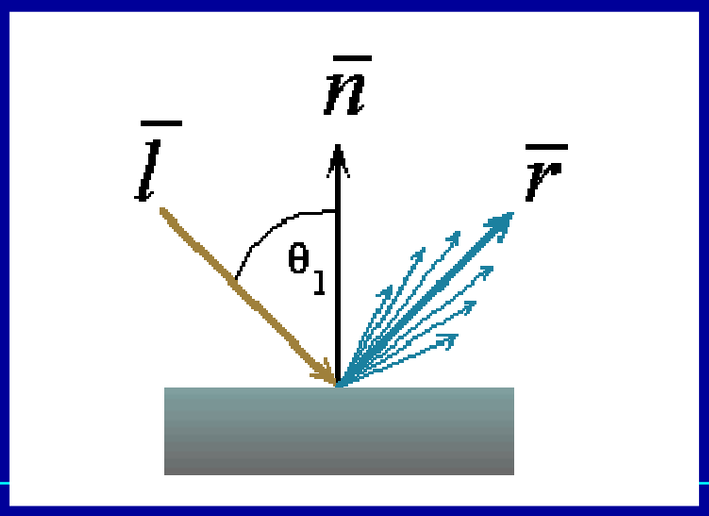
- metal
- polished stone
- plastics
- apples
- skin

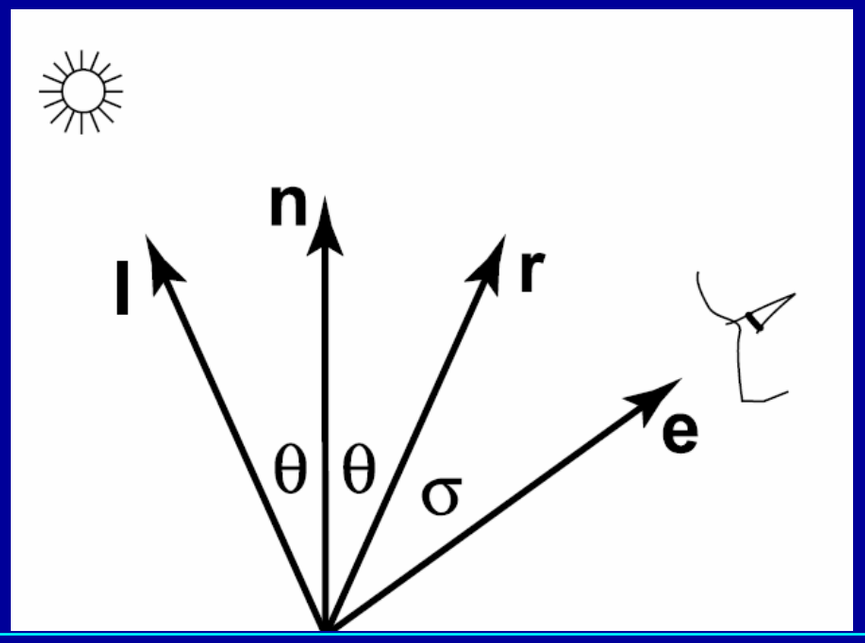
Properties:

- Specular reflection depends on the viewing direction **V**.
- For non-metals, the color is determined solely by the color of the light.
- For metals, the color may be altered (e.g., brass)

Specular Reflection (Phong Shading)

- Shiny surfaces change appearance when viewpoint is varied
  - specularities (highlights) are view-dependent
  - caused by surfaces that are microscopically smooth (tile floors, gloss paint, whiteboards)
- For shiny surfaces part of the incident light reflects coherently
  - an incoming ray is reflected in a single direction (or narrow beam)
  - direction is defined by the incoming direction and the surface normal
  - when σ is near zero, viewer sees reflection





Phong Illumination

- One function that approximates specular falloff is called the *Phong Illumination* model

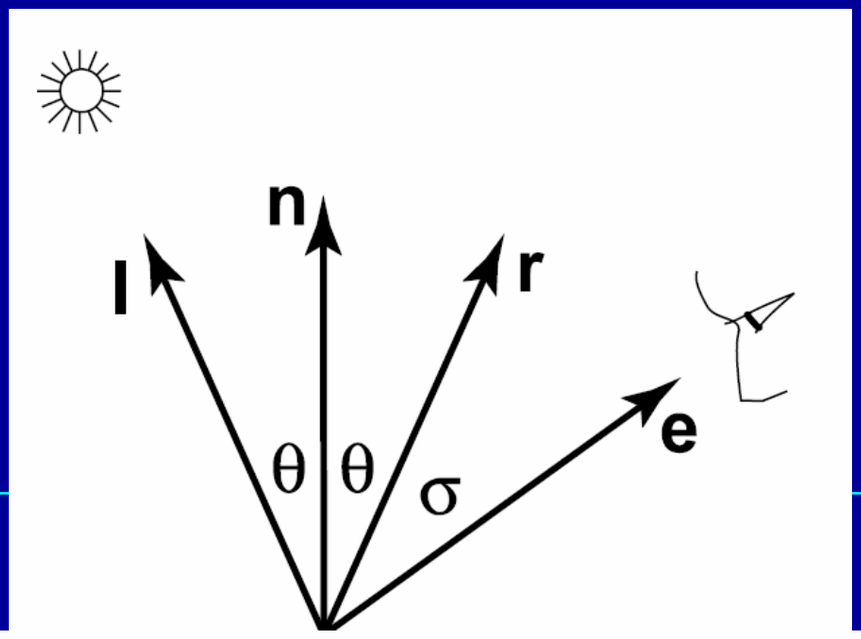
$$c = c_l (e \bullet r)$$

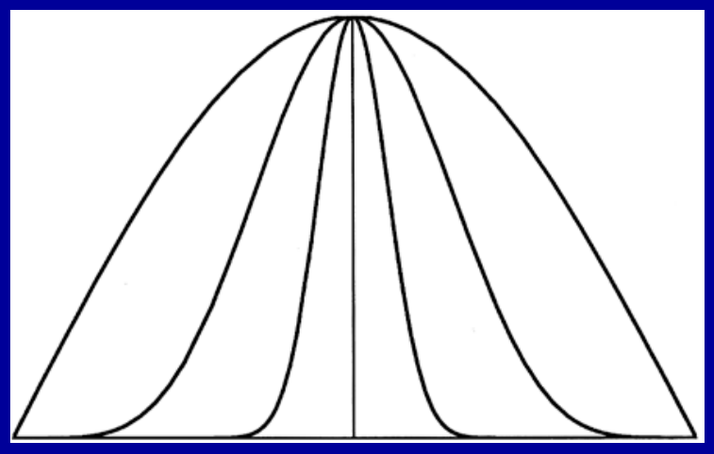
$$c = c_l \max(0, e \bullet r)^p$$

$$I_{specular} = k_s I_{light} (e \bullet r)^p$$

k_s : Specular reflectance

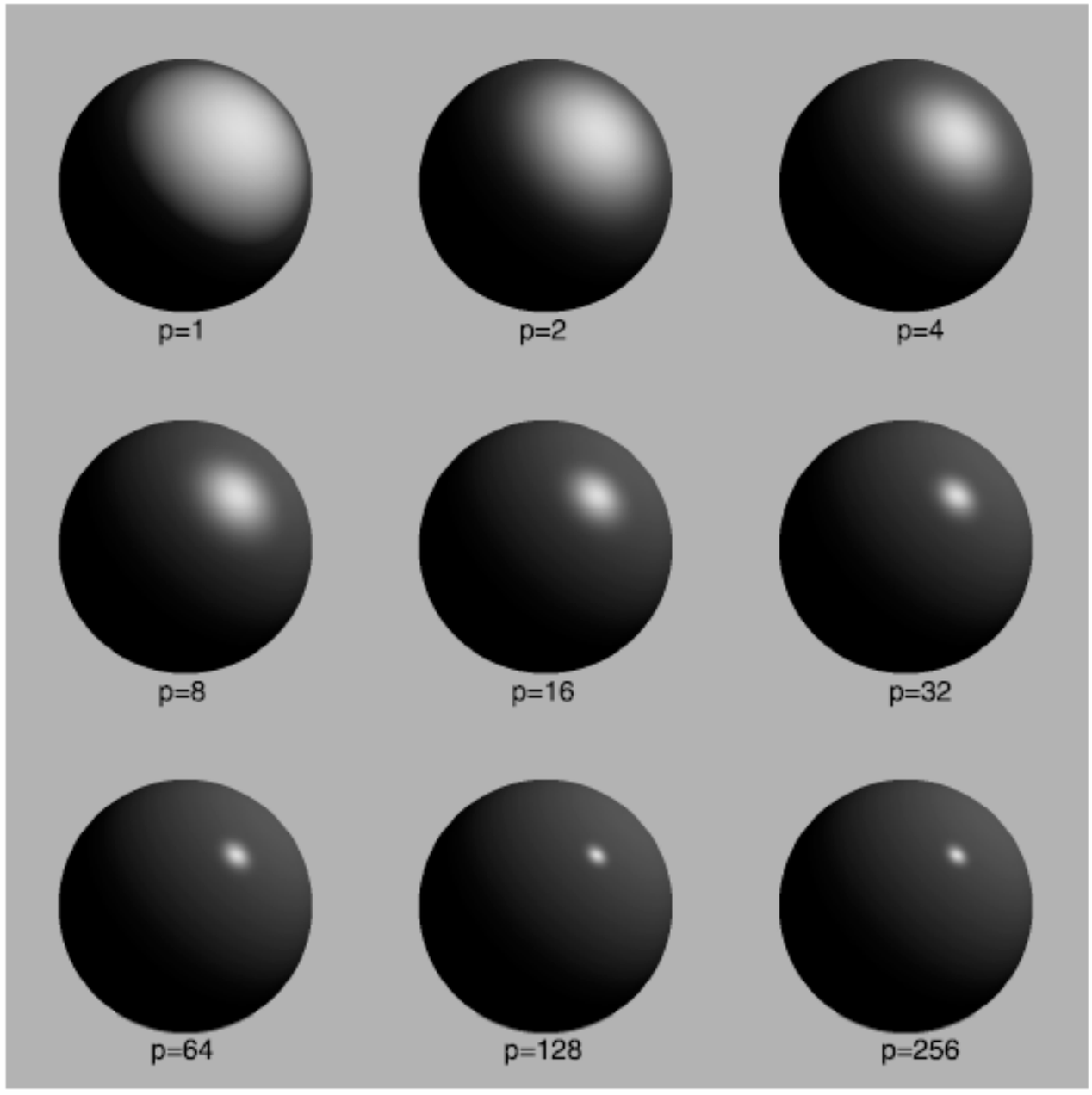
p : Rate of specular falloff (phong exponent)





Greater p , more focused beam

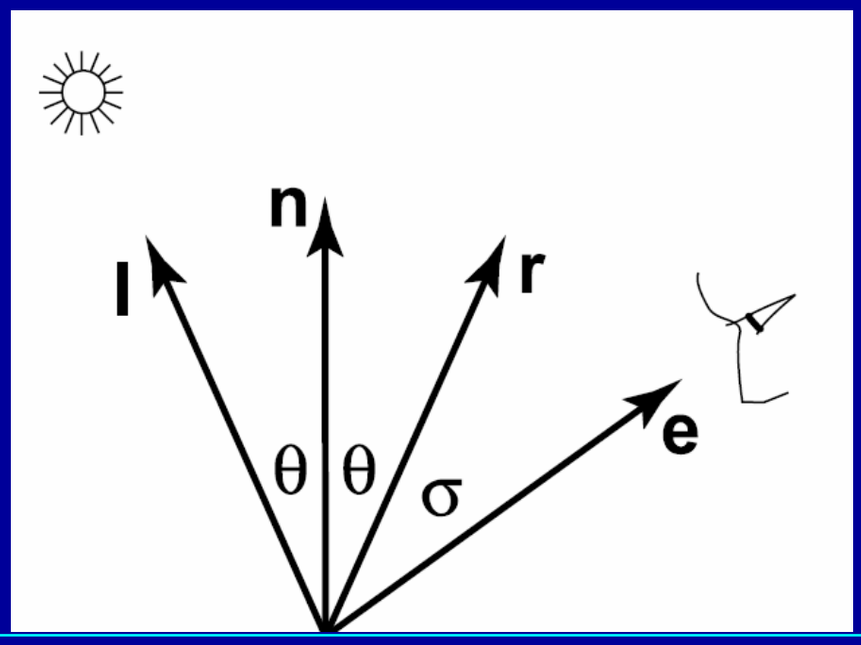
Phong Illumination



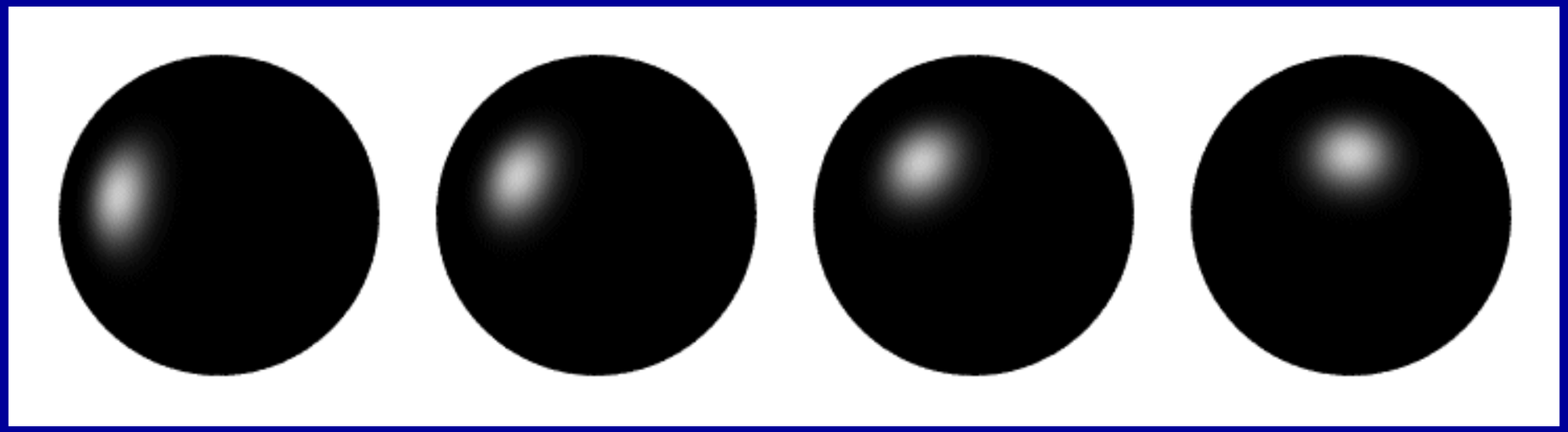
No real physical basis but provides approximately the right answer

Computing the Reflected Ray

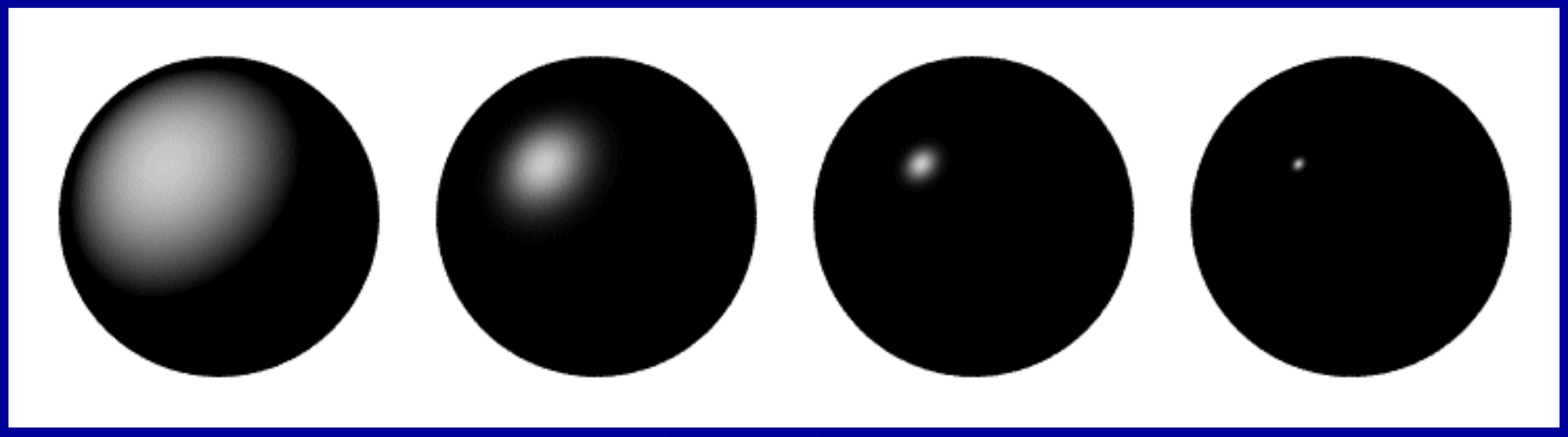
blackboard



Phong Illumination



Moving the light source



Changing p

$$I = k_e + k_a L_a +$$

$$\sum_j \frac{\left(\mathbf{L}_j \times \mathbf{S}_j\right)^{e_j}}{a_j + b_j d_j + c_j d_j^{\,2}} L_j \left[k_d \left(\mathbf{N} \times \mathbf{L}_j\right)_+ + k_s \left(\mathbf{V} \times \mathbf{R}_j\right)_+^{n_s} \right]$$

Choosing the parameters

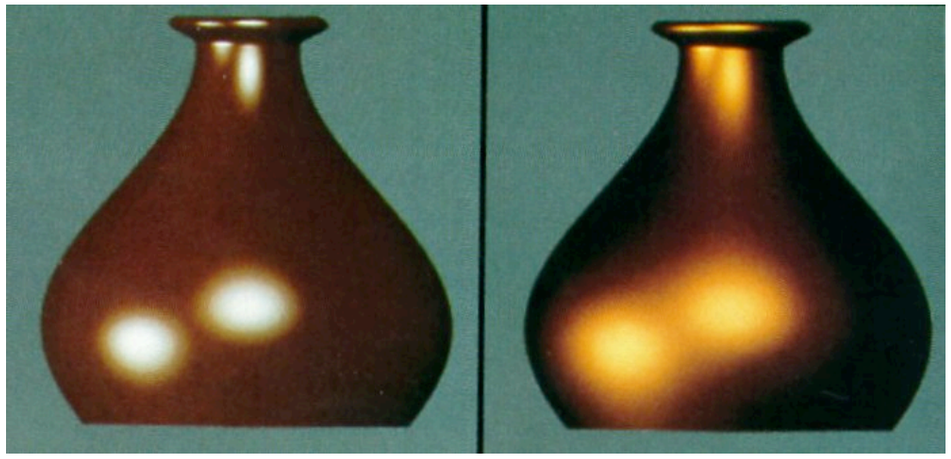
Experiment with different parameter settings. To get you started, here are a few suggestions:

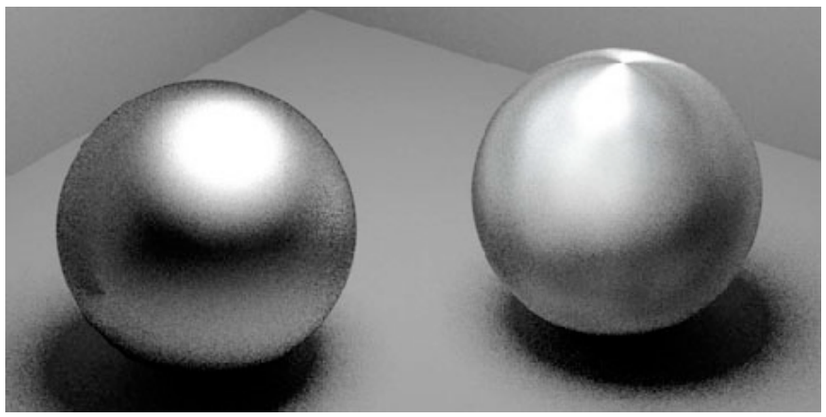
- Try n_s in the range [0,100]
- Try $k_a + k_d + k_s < 1$
- Use a small k_a (~0.1)

	n_s	k_d	k_s
Metal	large	Small, color of metal	Large, color of metal
Plastic	medium	Medium, color of plastic	Medium, white
Planet	0	varying	0

More sophisticated BRDF's

Cook and
Torrance, 1982

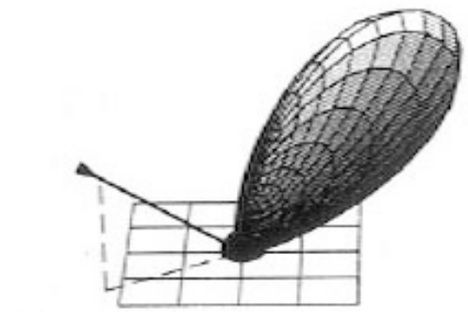




Westin, Arvo, Torrance 1992

BRDF

What assumptions does the BRDF model make?



ω_{in}

$f_r(\omega_{in}, \omega_{out})$

BSSRDF

source: http://graphics.stanford.edu/papers/bssrdf/

BRDF **BSSRDF**

$$dL_o(x_o, \vec{\omega}_o) = S(x_i, \vec{\omega}_i; x_o, \vec{\omega}_o)\, d\Phi_i(x_i, \vec{\omega}_i).$$

BRDF is an approximation that assumes $x_i = x_o$.

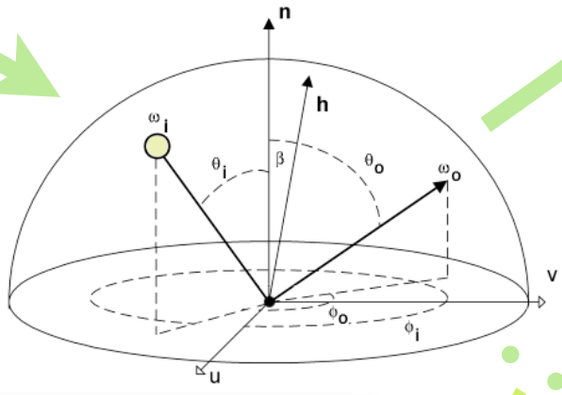
Snow

Shading

Part I: What is Light?

Part IV: Lighting

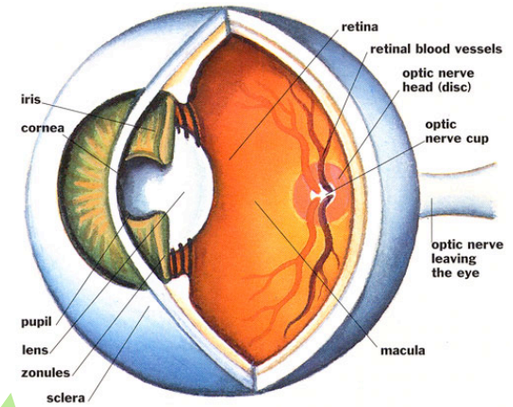
Light



Part III: Reflectance

No Light

Light



Light

Part II: The Eye

Light

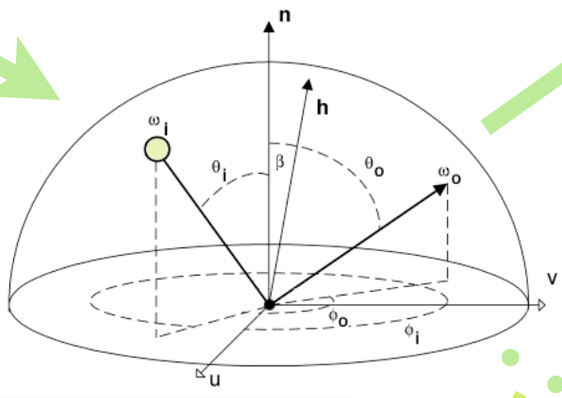
Part V: Shadows

Shading

Part I: What is Light?

Part IV: Lighting

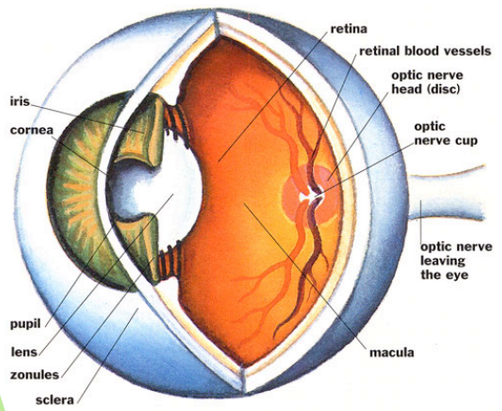
Light



Part III: Reflectance

No Light

Light



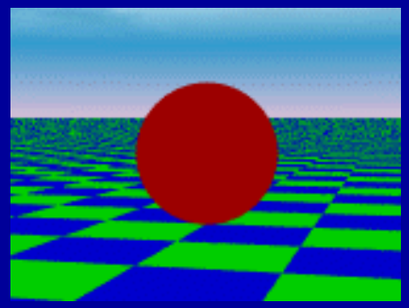
Part II: The Eye

Light

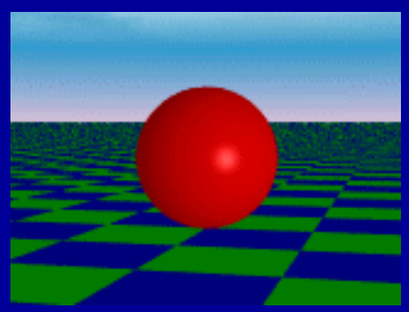
Part V: Shadows

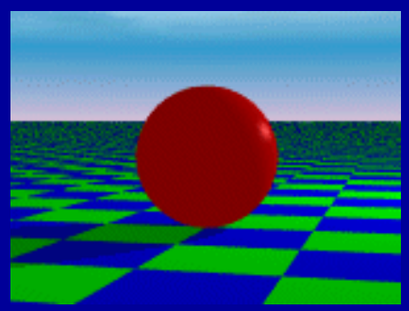
Types of Light Sources

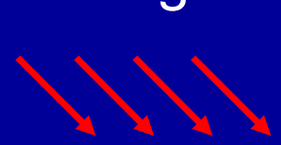
- Ambient: equal light in all directions
  - a hack to model inter-reflections



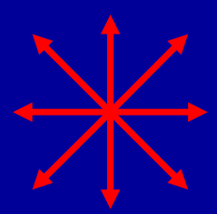
- Directional: light rays oriented in same direction
  - good for distance light sources (sunlight)







- Point: light rays diverge from a single point
  - approximation to a light bulb (but harsher)

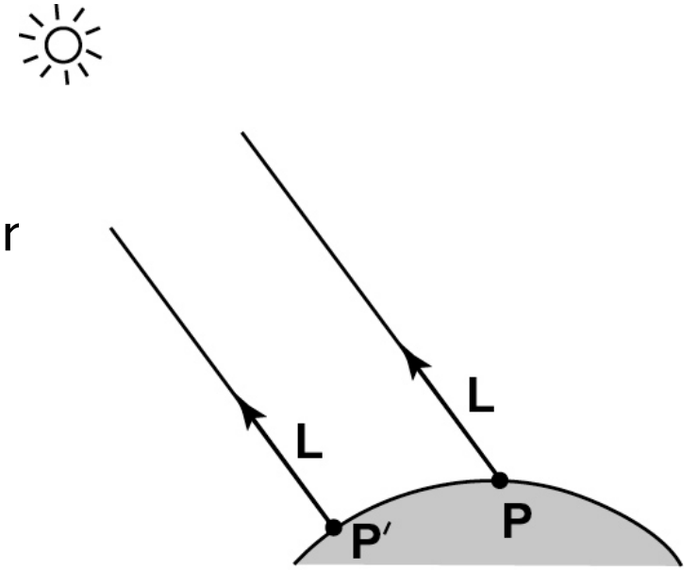


Lights

OpenGL supports three different kinds of lights: ambient, directional, and point. Spot lights are also supported as a special form of point light.

We've seen ambient light sources, which are not really geometric.

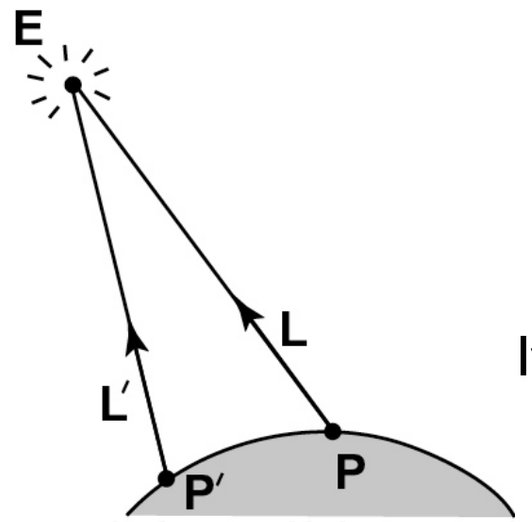
Directional light sources have a single direction and intensity associated with them.



[Note: Lights not discussed ir

Point lights

The direction of a **point light** sources is determined by the vector from the light position to the surface point.



$$\mathbf{L} = \frac{\mathbf{E} - \mathbf{P}}{\|\mathbf{E} - \mathbf{P}\|}$$

Physics tells us the intensity ly with the square of the distance: $d = \|\mathbf{E} - \mathbf{P}\|$

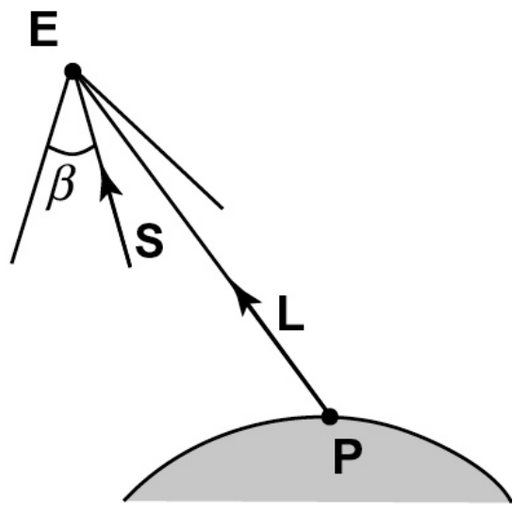
Sometimes, this distance-squared dropoff is considered too "harsh." A common alternative is:

with user-supplied constants for a, b, and c. $f_{\text{atten}} = \frac{1}{d^2}$

$$f_{\text{atten}} = \frac{1}{a + bd + cd^2}$$

Spotlights

OpenGL also allows one to apply a directional attenuation of a point light source, giving a **spotlight** effect.



The spotlight intensity factor is co

where

- **L** is the direction to the point light.
- **S** is the center direction of the spotlight.
- β is the cutoff angle for the spotlight
- e is the angular falloff coefficient $f_{spot} = (\mathbf{L} \times \mathbf{S})_{\beta}^{e}$
-

$$(x)_{\beta}^{e} = \left[\max\left\{\text{acos}(x) - \beta,\ 0\right\}\right]^{e}$$

“Iteration four”

Since light is additive, we can handle multiple lights by taking the sum over every light.

Our equation is now:

$$I = k_e + k_a L_a + \sum_j \frac{\left(\mathbf{L}_j \cdot \mathbf{S}_j\right)^{e_j}_{\beta_j}}{a_j + b_j d_j + c_j {d_j}^2} L_j \left[k_d \left(\mathbf{N} \cdot \mathbf{L}_j\right)_+ + k_s \left(\mathbf{V} \cdot \mathbf{R}_j\right)^{n_s}_+ \right]$$

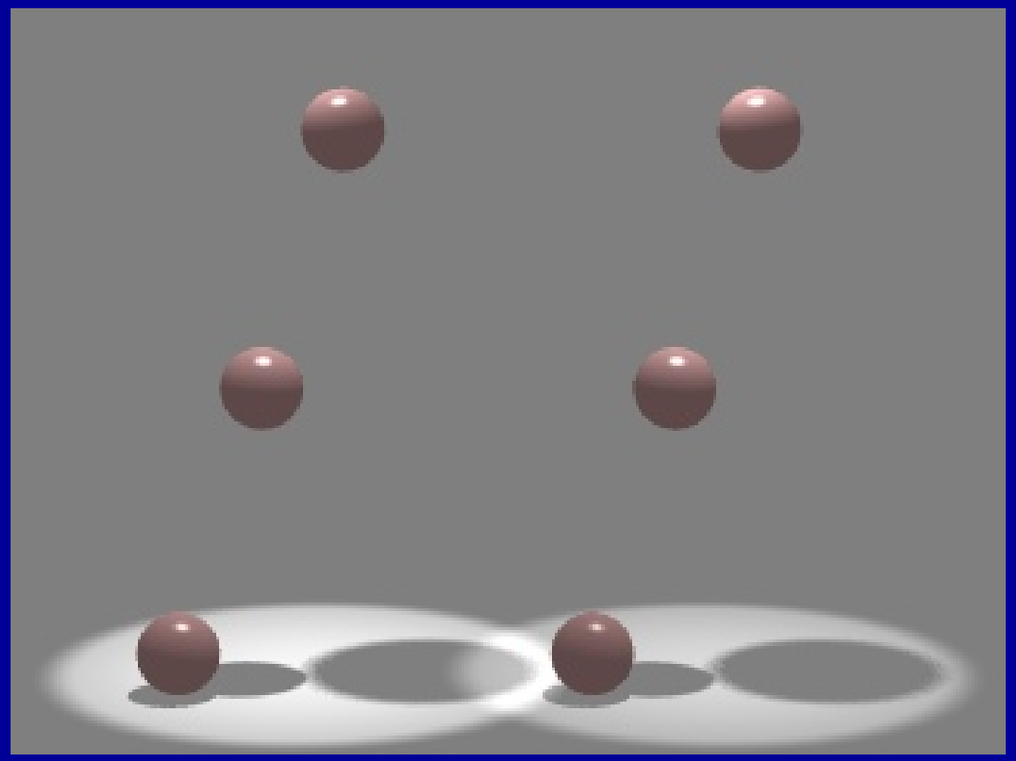
This is the Phong illumination model.

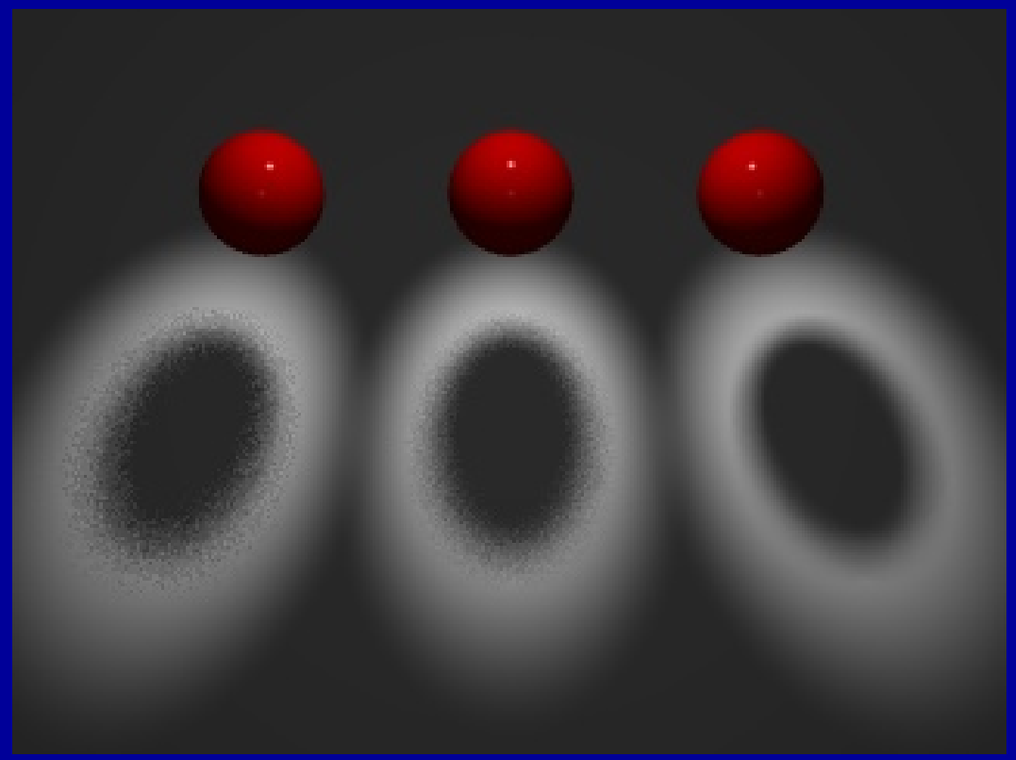
Which quantities are spatial vectors?

Which are RGB triples?

Which are scalars?

More Light Sources





- Spotlight: point source with directional fall-off
  - intensity is maximal along some direction D, falls off away from D
  - specified by color, point, direction, fall-off parameters
- Area Source: Luminous 2D surface
  - radiates light from all points on its surface
  - generates soft shadows

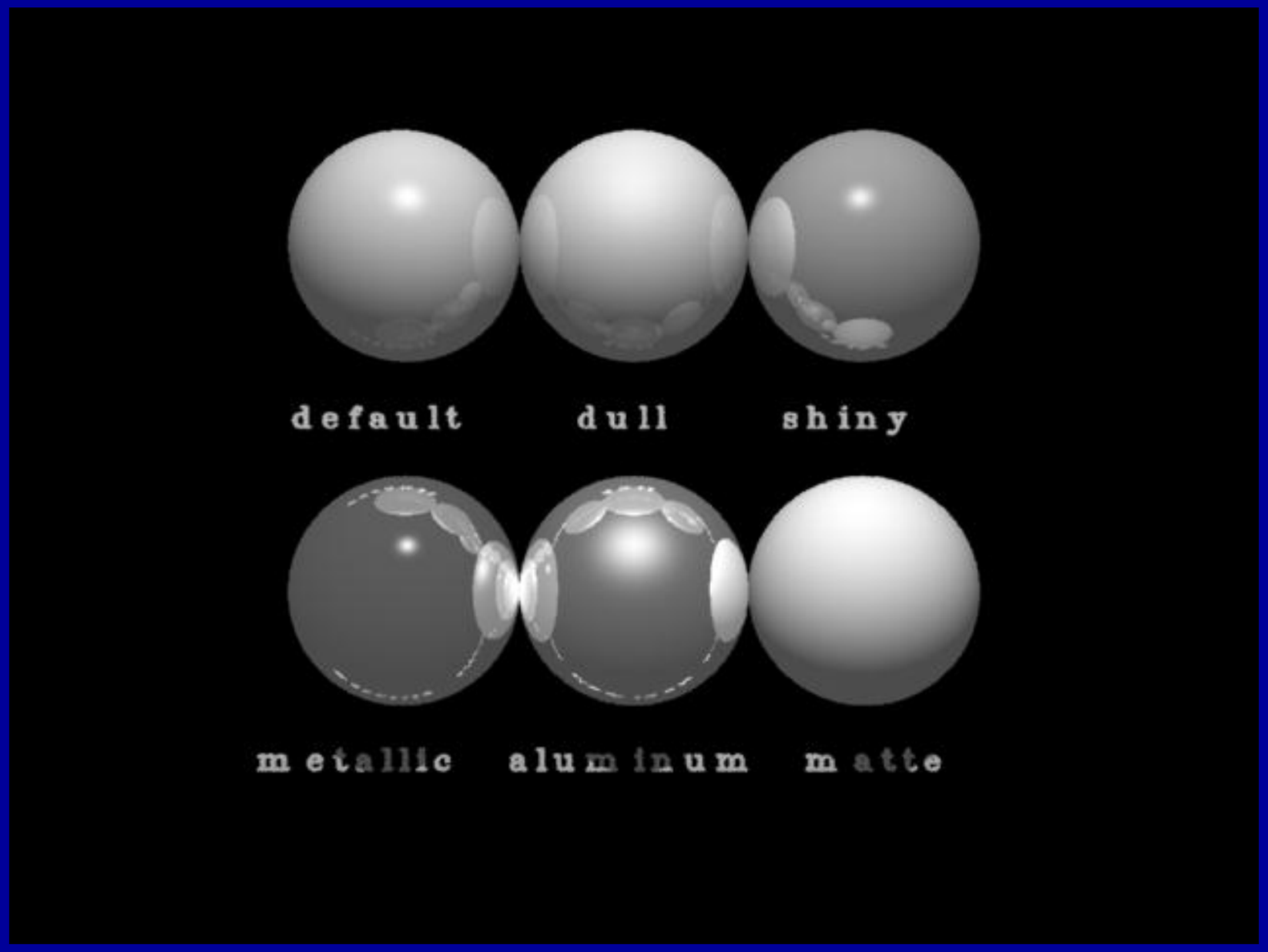
Putting It All Together

- Combining ambient, diffuse, and specular illumination

$$I = k_a I_a + f_{att} I_{light} \left[k_d \cos\theta + k_s (\cos\phi)^p \right]$$

- For multiple light sources
  - Repeat the diffuse and specular calculations for each light source
  - Add the components from all light sources
  - The ambient term contributes only once
- The different reflectance coefficients can differ.
  - Simple “metal”: k_a and k_d share material color, k_s is white
  - Simple plastic: k_s also includes material color
  - More on these kinds of parameters when we talk about reflectance models in a few weeks

Some Examples

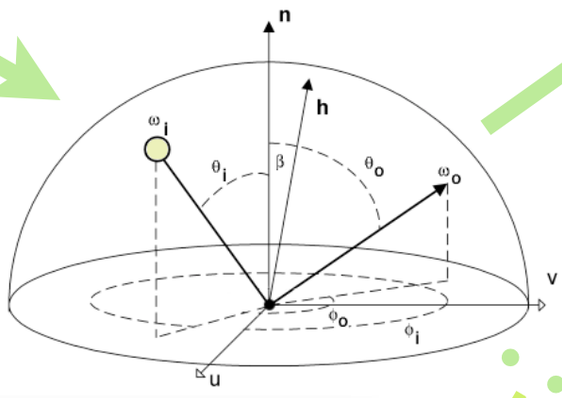


Shading

Part I: What is Light?

Part IV: Lighting

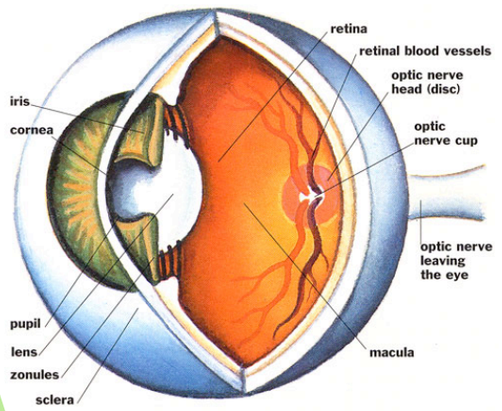
Light



Part III: Reflectance

No Light

Light



Part II: The Eye

Light

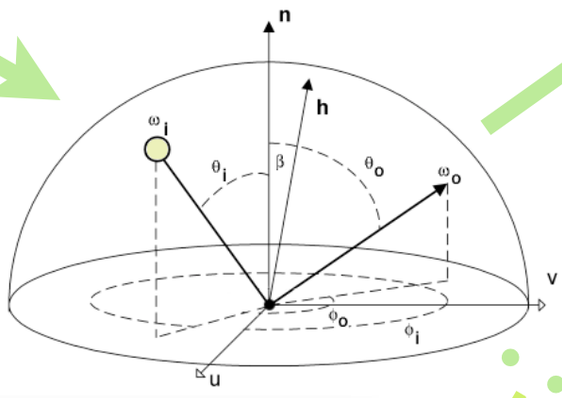
Part V: Shadows

Shading

Part I: What is Light?

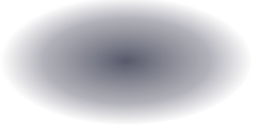
Part IV: Lighting

Light

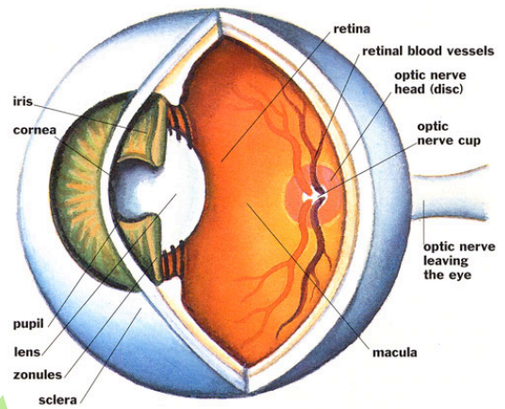


Part III: Reflectance

No Light



Light



Part II: The Eye

Light

Part V: Shadows

Shadows

Shadows occur where objects are hidden from a light source

omit any intensity contribution from hidden light sources

How does the z-buffer work?

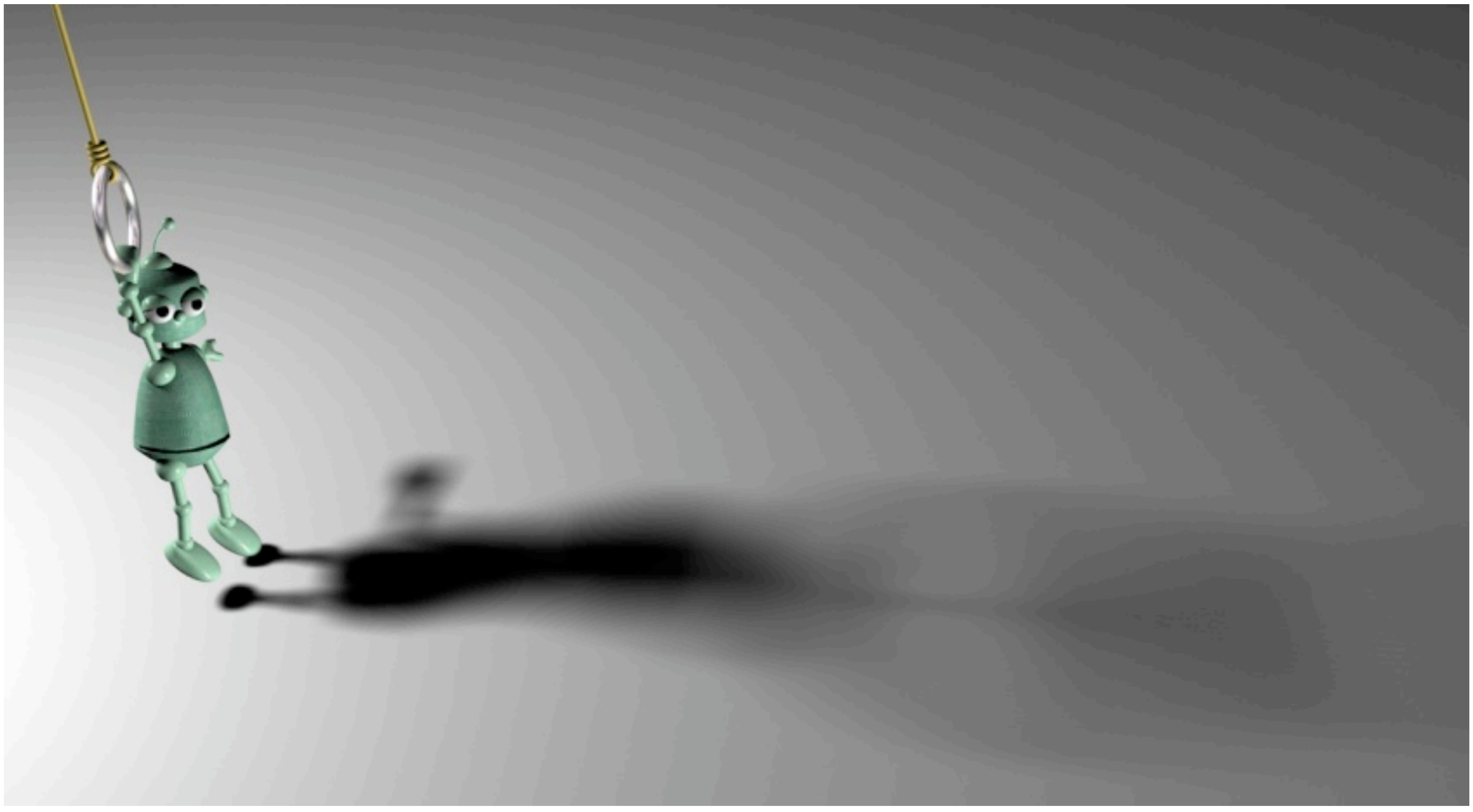
How can we use the z-buffer to compute shadows?

Light Sources

- What is a good model for the sun?
- What is a good model for the lights in this room?

Hard Shadows

Soft Shadows



- Why? How?

Transmission with Refraction

- Refraction:
  - the bending of light due to its different velocities through different materials
- Refractive index:
  - light travels at speed *c/n* in a material of refractive index *n*
  - *c* is the speed of light in a vacuum
  - varies with wavelength hence rainbows and prisms

MATERIAL	INDEX OF REFRACTION
Air/Vacuum	1
Water	1.33
Glass	about 1.5
Diamond	2.4

Snell's Law

Light bends by the physics *principle of least time*

light travels from point A to point B by the fastest path

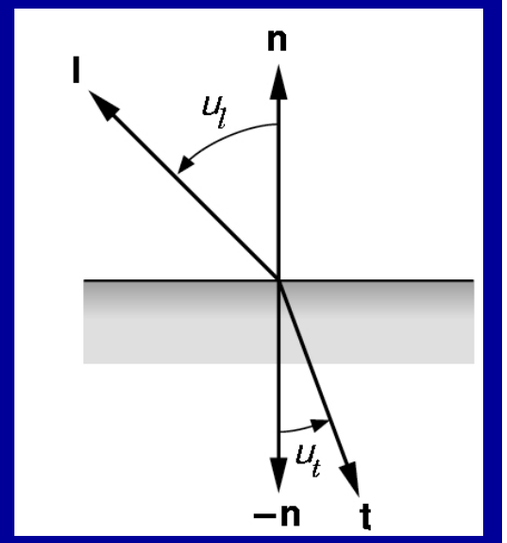
when passing from a material of index n_1 to one of index n_2 *Snell's law* gives the angle of refraction:
$n_1 \sin \theta_1 = n_2 \sin \theta_2$
where θ_1 and θ_2 are the angles from perpendicular

When traveling into a denser material (larger n), light bends to be more perpendicular (eg air to water) and vice versa

light travels further in the faster material

if the indices are the same the light doesn't bend

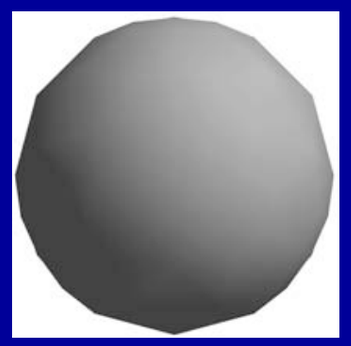
When traveling into a less dense material total internal reflection occurs if $\theta_1 > \sin^{-1}(n_2/n_1)$



Shading

Given an equation to calculate surface radiance, we still must apply it to the real model

- Usually performed during scan conversion
- There are efficient methods for doing this quickly (which we will discuss in more detail later in the semester



blackboard

Flat shaded

Gouraud: Normal at vertex is average of normals for adjacent faces

Phong: interpolate normals instead of intensities

Uniformly shaded surfaces are still unrealistic

Real objects have surface features, or texture

One option: use a huge number of polygons with appropriate surface coloring and reflectance characteristics

Texture mapping gets you more detail at less cost

- Assign radiance based on an image

Or use *Procedural shaders* to specify any function you want to define radiance

- Generate radiance on the fly, during shading

OpenGL Materials

```
GLfloat white8[] = {.8, .8, .8, 1.}, white2 = {.2,.2,.2,1.},black={0.,0.,0.};
GLfloat mat_shininess[] = {50.};          /* Phong exponent */

glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, black);
glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, white8);
glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, white2);
glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS, mat_shininess);
```

OpenGL Lighting

```
GLfloat white[] = {1., 1., 1., 1.};
GLfloat light0_position[] = {1., 1., 5., 0.}; /* directional light (w=0) */

glLightfv(GL_LIGHT0, GL_POSITION, light0_position);
glLightfv(GL_LIGHT0, GL_DIFFUSE, white);
glLightfv(GL_LIGHT0, GL_SPECULAR, white);
glEnable(GL_LIGHT0);

glEnable(GL_NORMALIZE);     /* normalize normal vectors */
glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);    /* two-sided lighting*/

glEnable(GL_LIGHTING);
```