5.1 Lighting and Shading

Hao Li

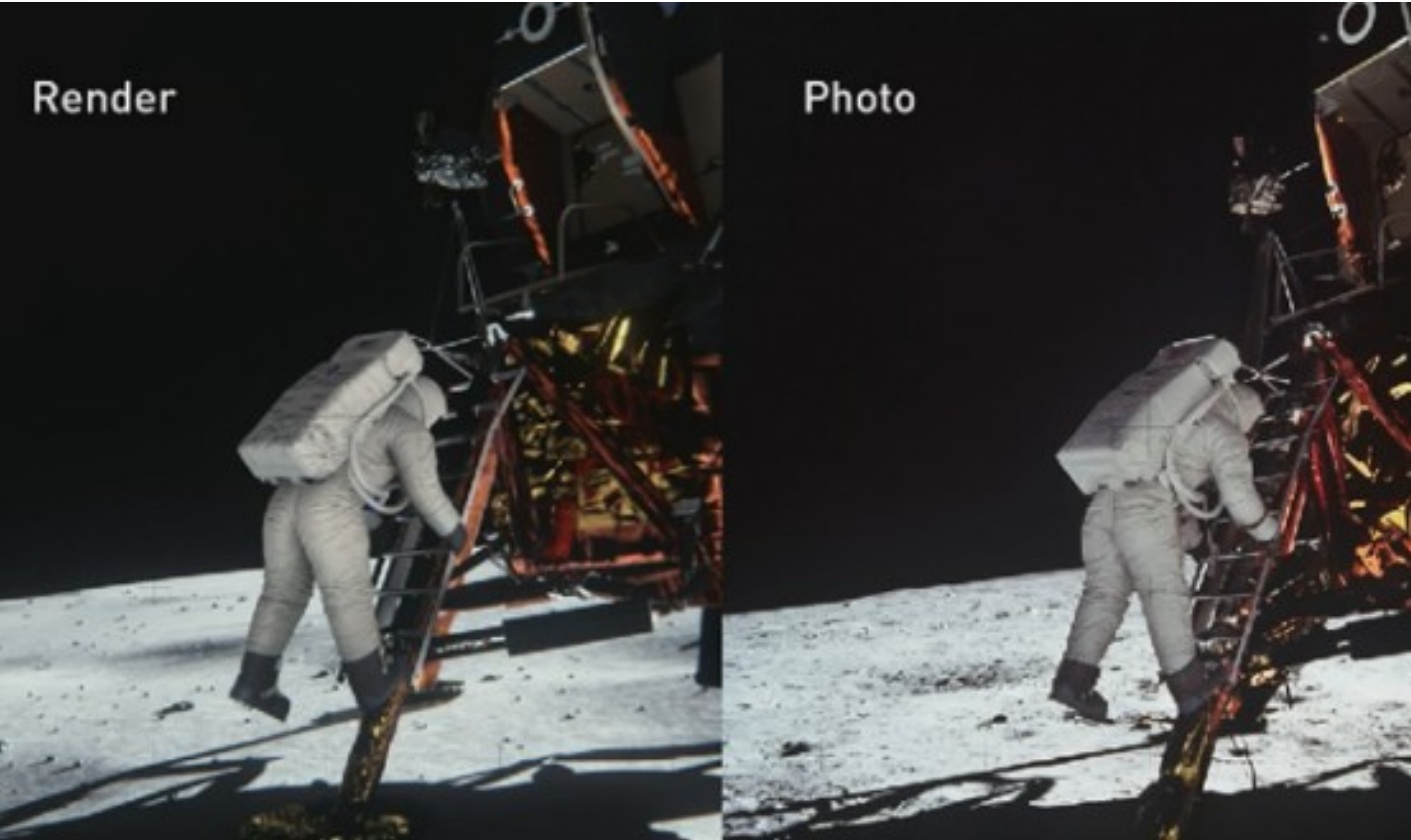
http://cs420.hao-li.com

Debunking Lunar Landing Conspiracies with Global Illumination

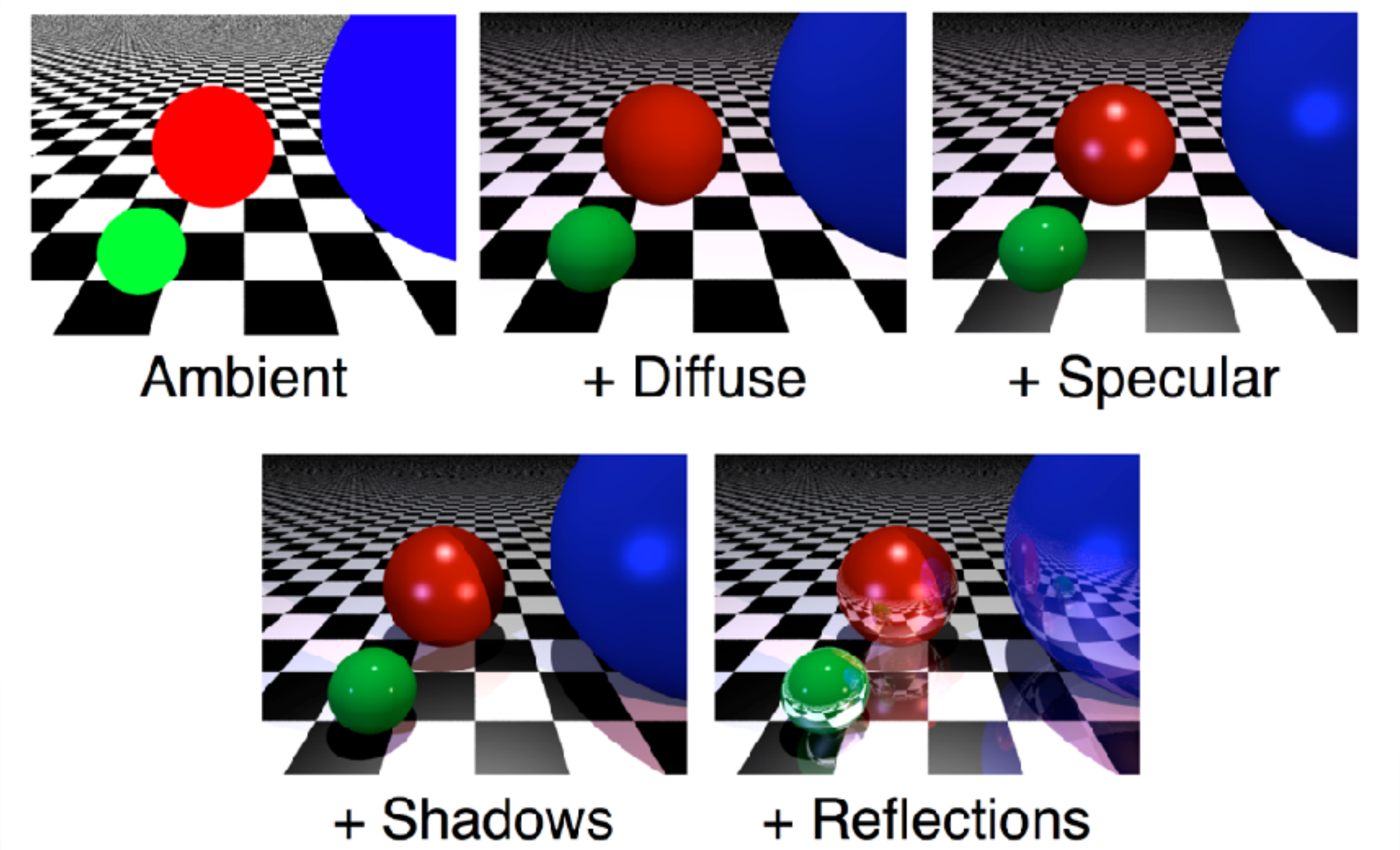
https://www.youtube.com/watch?v=O9y_AVYMEUs

Single Light Source for Global Illumination

https://www.youtube.com/watch?v=O9y_AVYMEUs



Lighting



Outline

- Global and Local Illumination
- Normal Vectors
- Light Sources
- Phong Illumination Model

Global Illumination

Tobias R. Metoc

- Ray tracing
- Radiosity
- Photon Mapping
- Follow light rays through a scene
- Accurate, but expensive (off-line)

Raytracing Example

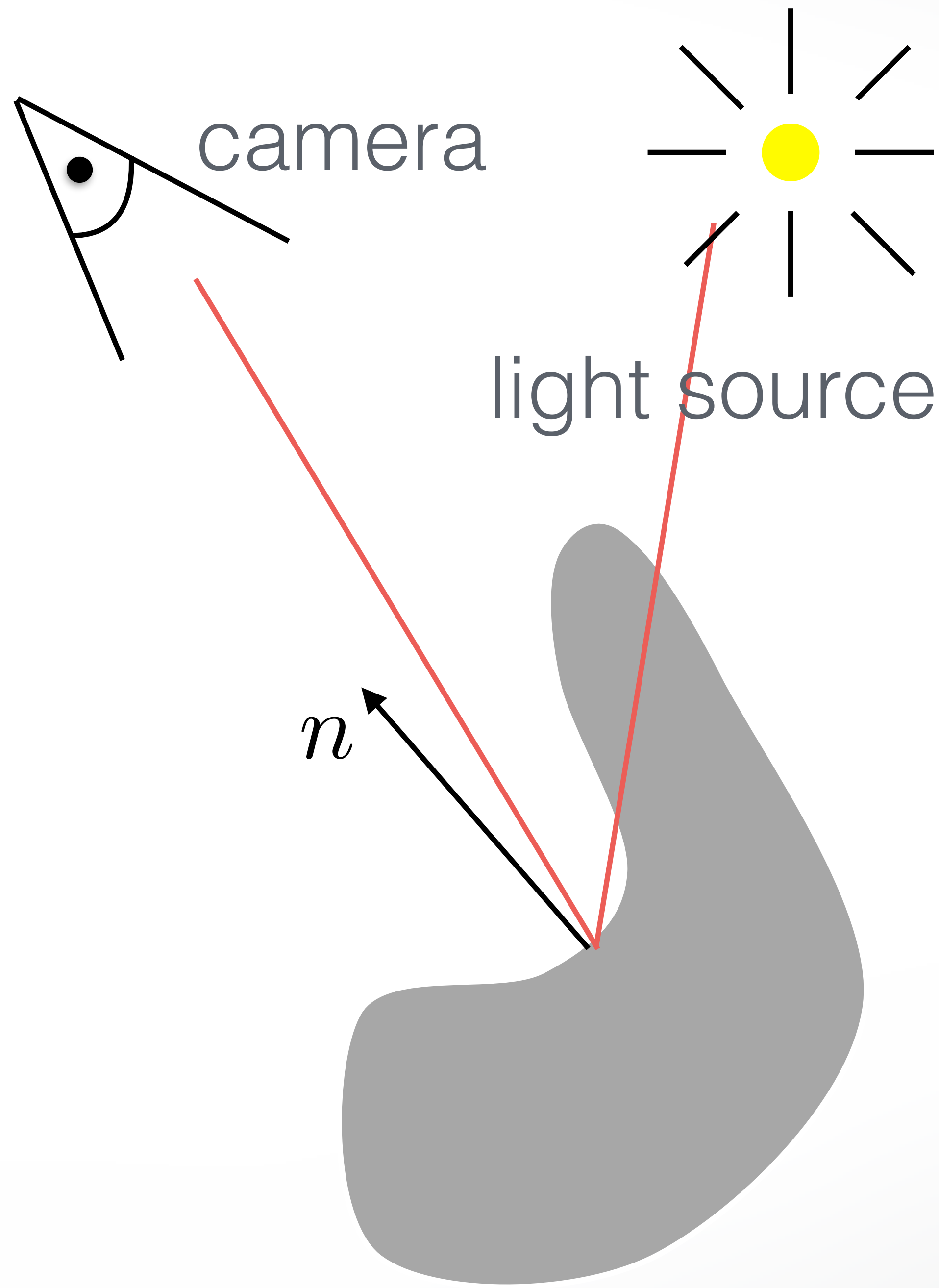
Martin Moeck,
Siemens Lighting

Radiosity Example

Restaurant Interior. Guillermo Leal, Evolucion Visual

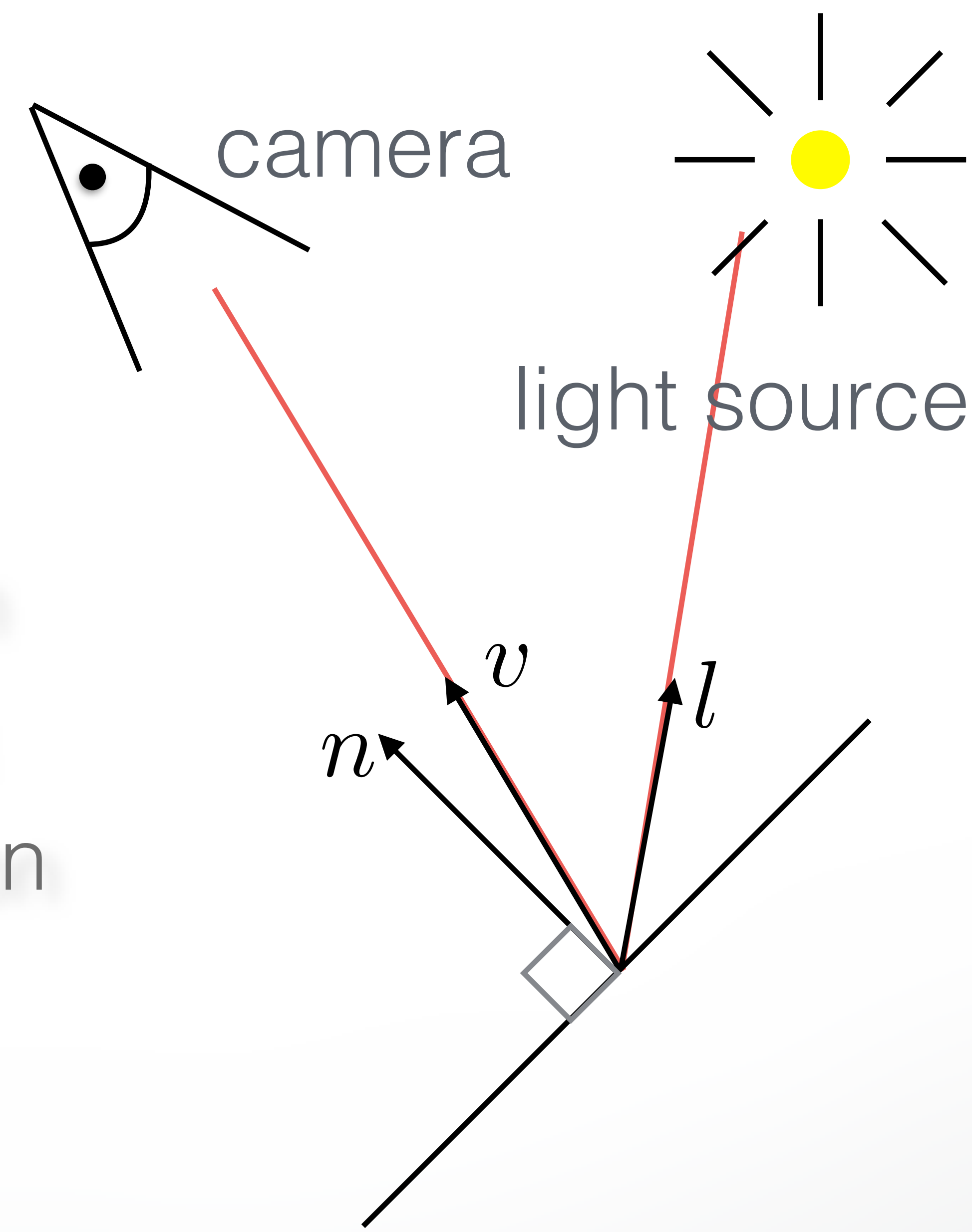
Local Illumination

- Approximate model
- Local interaction between light, surface, viewer
- Phong model (this lecture): fast, supported in OpenGL
- GPU shaders
- Pixar Renderman (offline)



Local Illumination

- Approximate model
- Local interaction between light, surface, viewer
- Color determined only based on surface normal, relative camera position and relative light position
- What effects does this ignore?



Outline

- Global and Local Illumination
- Normal Vectors
- Light Sources
- Phong Illumination Model

Normal Vectors

- Must calculate and specify the normal vector
  - Even in OpenGL!

- Two examples: plane and sphere

Normals of a Plane, Method I

- Method I: given by $ax + by + cz + d = 0$
- Let p_0 be a known point on the plane
- Let p be an arbitrary point on the plane
- Recall: $u \cdot v = 0$ if and only if u orthogonal to v
- $n \cdot (p - p_0) = n \cdot p - n \cdot p_0 = 0$
- Consequently $n_0 = [\ a \quad b \quad c\ ]^T$
- Normalize to $n = \dfrac{n_0}{|n_0|}$

Normals of a Plane, Method II

- Method II: plane given by $p_0,\ p_1,\ p_2$
- Points must not be collinear
- Recall: $u \times v$ orthogonal to u and v
- $n_0 = (p_1 - p_0) \times (p_2 - p_0)$
- Order of cross product determines orientation
- Normalize to $n = \dfrac{n_0}{|n_0|}$

Normals of Sphere

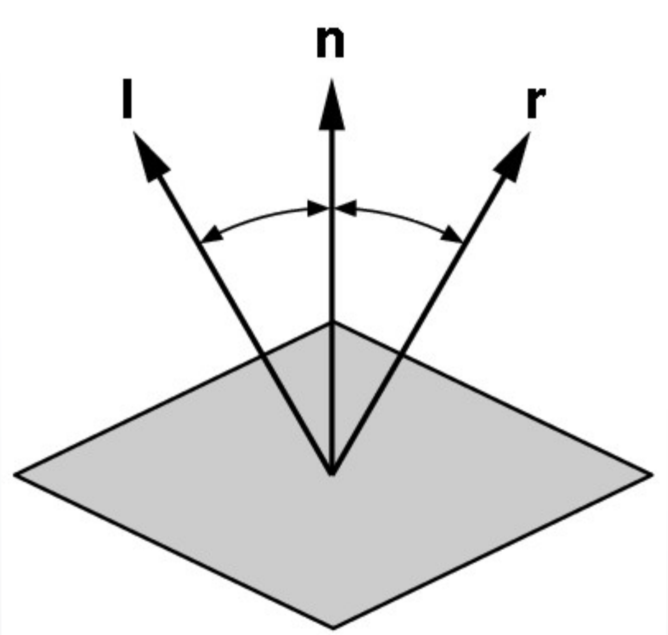
- Implicit Equation $f(x, y, z) = x^2 + y^2 + z^2 - 1 = 0$
- Vector form: $f(p) = p \cdot p - 1 = 0$
- Normal given by gradient vector

$$n_0 = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix} = \begin{bmatrix} 2x \\ 2y \\ 2z \end{bmatrix} = 2p$$

- Normalize $\frac{n_0}{|n_0|} = \frac{2p}{2} = p$

Reflected Vector

- Perfect reflection: angle of incident equals angle of reflection
- Also: l, n, and r lie in the same plane
- Assume $|l| = |n| = 1$, guarantee $|r| = 1$



$$l \cdot n = \cos(\theta) = n \cdot r$$

$$r = \alpha l + \beta n$$

$$Solution : \alpha = -1 \; and \; \beta = 2(l \cdot n)$$

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

Outline

- Global and Local Illumination
- Normal Vectors
- Light Sources
- Phong Illumination Model

Light Sources and Material Properties

- Appearance depends on
  - Light sources, their locations and properties
  - Material (surface) properties:

  - Viewer position

Types of Light Sources

- Ambient light: no identifiable source or direction
- Point source: given only by point
- Distant light: given only by direction
- Spotlight: from source in direction
  - Cut-off angle defines a cone of light
  - Attenuation function (brighter in center)

Point Source

- Given by a point p_0
- Light emitted equally in all directions
- Intensity decreases with square of distance

$$I \propto \frac{1}{|p - p_0|^2}$$

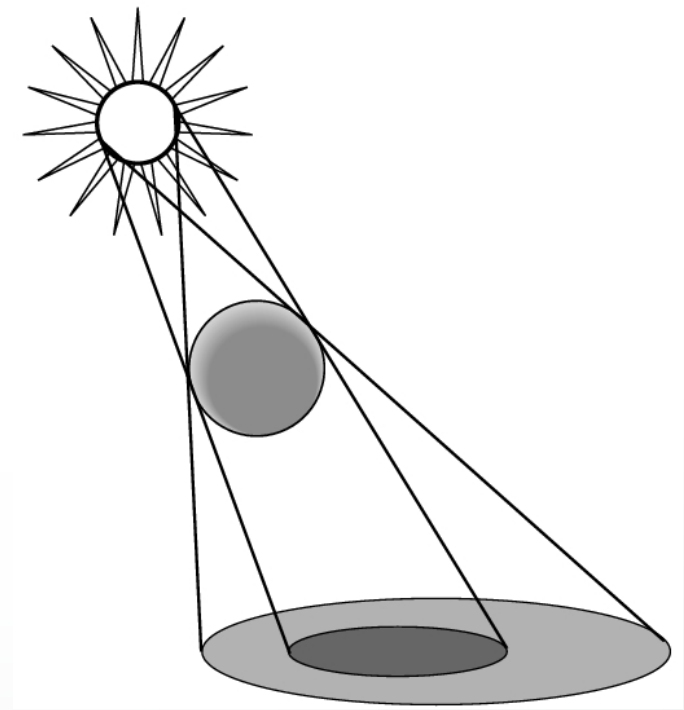
Limitations of Point Sources

- Shading and shadows inaccurate
- Example: penumbra (partial “soft” shadow)
- Similar problems with highlights
- Compensate with attenuation

$$\frac{1}{a + bq + cq^2}$$

$$q = distance|p - p_0|$$

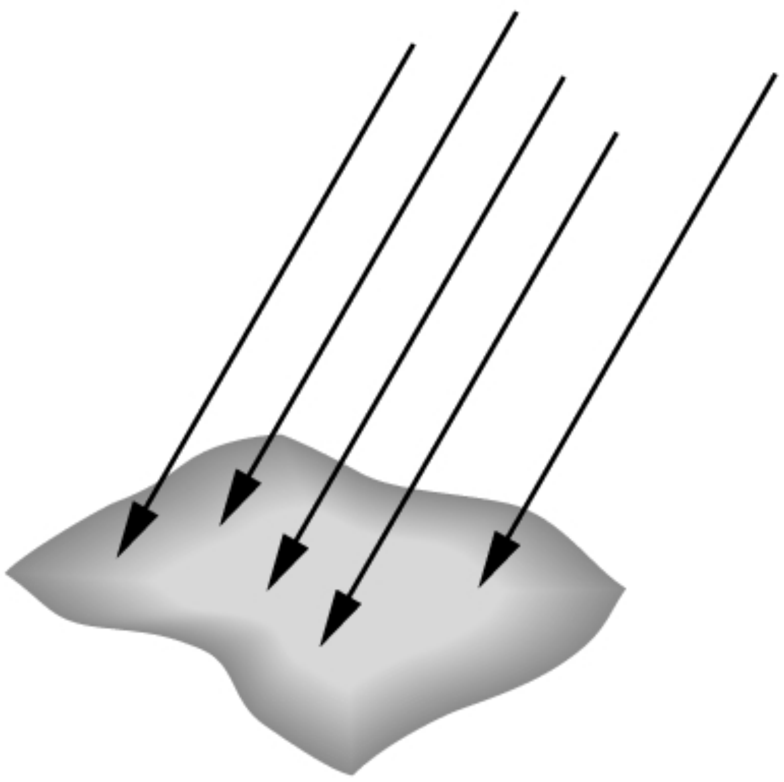
$$a,\ b,\ c\ constants$$



- Softens lighting
- Better with ray tracing
- Better with radiosity

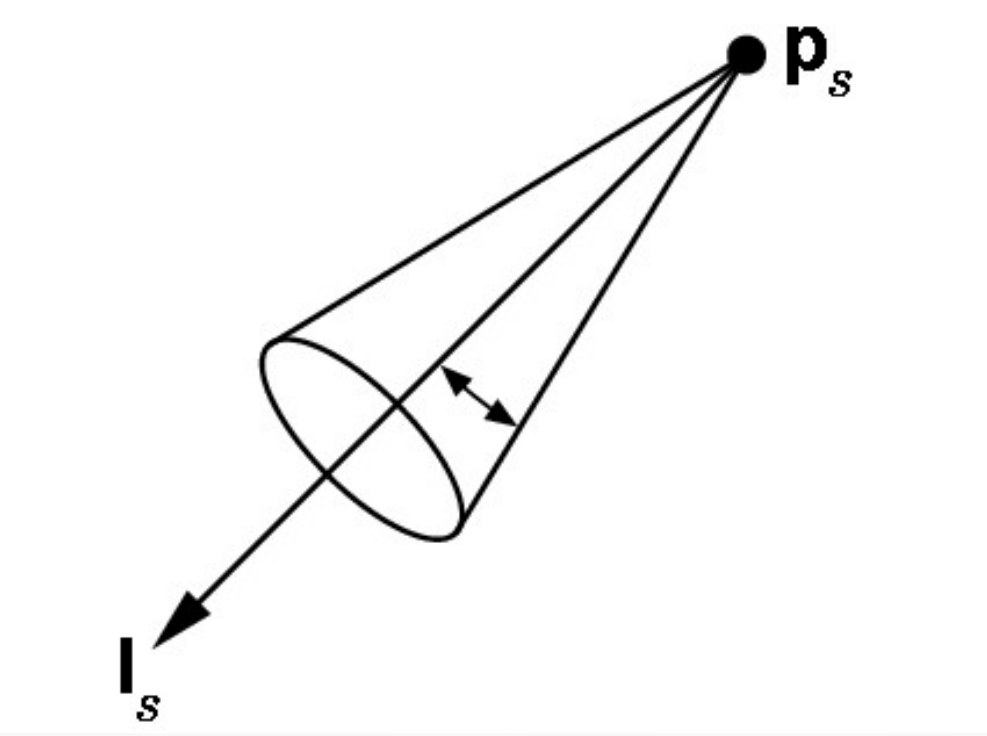
Distant Light Source

- Given by a direction vector
- Simplifies some calculations
- In OpenGL:
  - Point source $\begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$
  - Distant source $\begin{bmatrix} x & y & z & 0 \end{bmatrix}^T$



Spotlight

- Most complex light source in OpenGL
- Light still emanates from point
- Cut-off by cone determined by angle θ



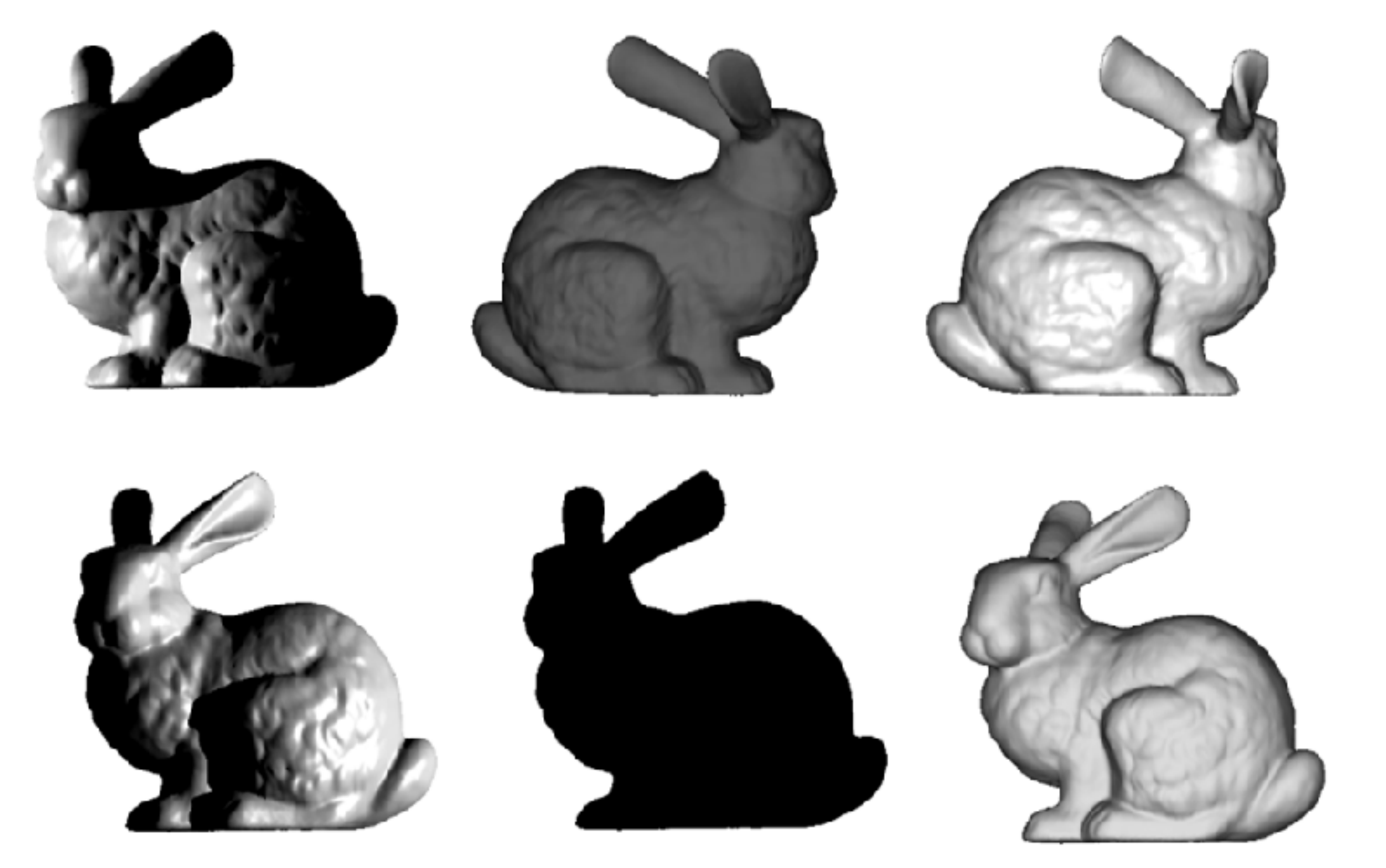
Global Ambient Light

- Independent of light source
- Lights entire scene
- Computationally inexpensive
- Simply add $\begin{bmatrix} G_R & G_G & G_B \end{bmatrix}$ to every pixel on every object
- Not very interesting on its own
  A cheap hack to make the scene brighter

Outline

- Global and Local Illumination
- Normal Vectors
- Light Sources
- Phong Illumination Model

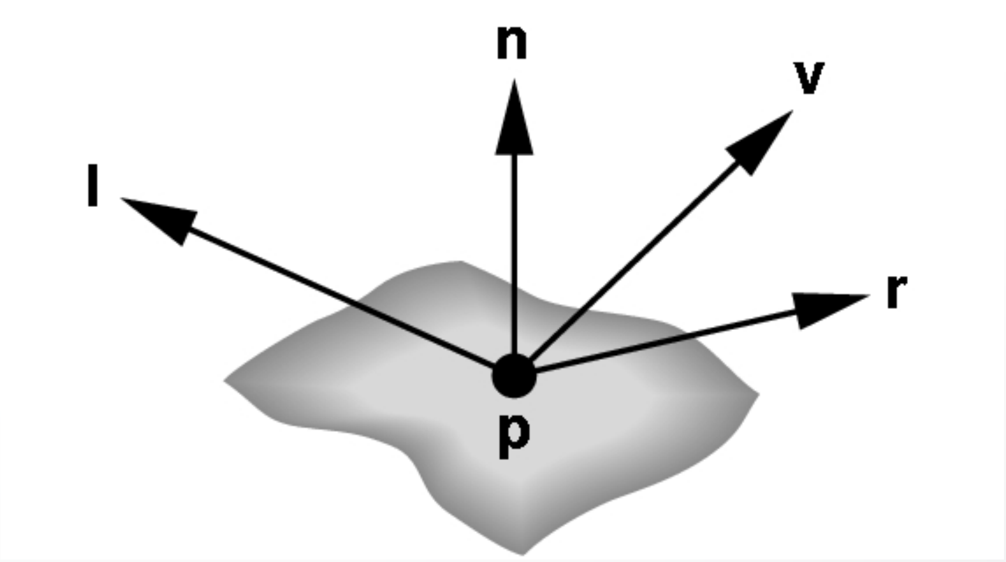
Phong Lighting



Phong Illumination Overview

- Calculate color for arbitrary point on surface
- Compromise between realism and efficiency
- Local computation (no visibility calculations)
- Basic inputs are material properties and $l,\ n,\ v$:

l = unit vector to light source
n = surface normal
v = unit vector to viewer
r = reflection of l at p
(determined by l and n)



Phong Illumination Overview

1. Start with global ambient light $\begin{bmatrix} G_R & G_G & G_B \end{bmatrix}$

2. Add contributions from each light source

3. Clamp the final result to [0, 1]

- Calculate each color channel (R,G,B) **separately**

- Light source contributions decomposed into
  - Ambient reflection
  - Diffuse reflection
  - Specular reflection

- Based on ambient, diffuse, and specular lighting and material properties

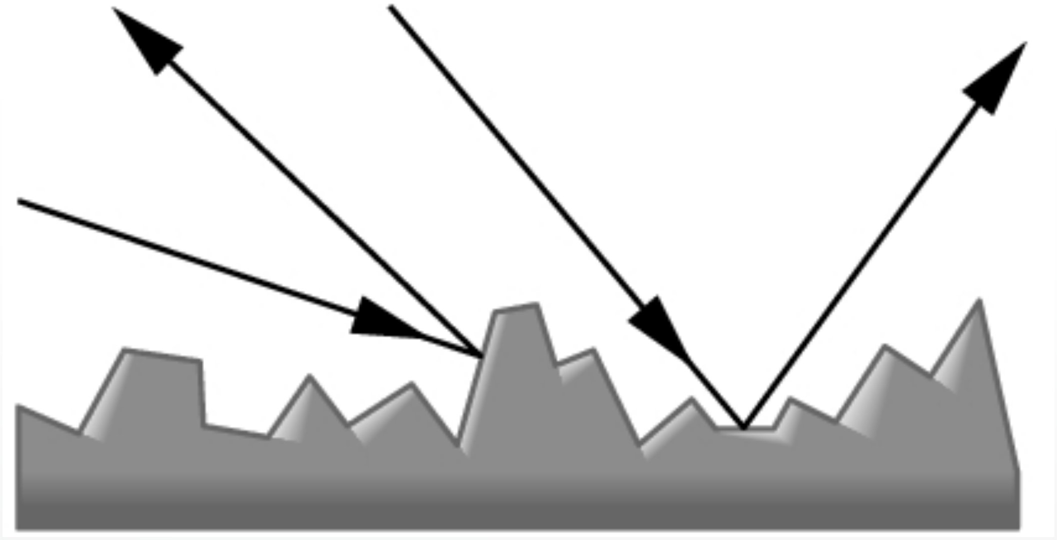
Ambient Reflection

$$\boxed{I_a = k_a L_a}$$

- Intensity of ambient light is uniform at every point
- Ambient reflection coefficient k_a (material), $0 \leq k_a \leq 1$
- May be different for every surface and r,g,b
- Determines reflected fraction of ambient light
- L_a = ambient component of light source
  (can be set to different value for each light source)
- Note: L_a is not a physically meaningful quantity

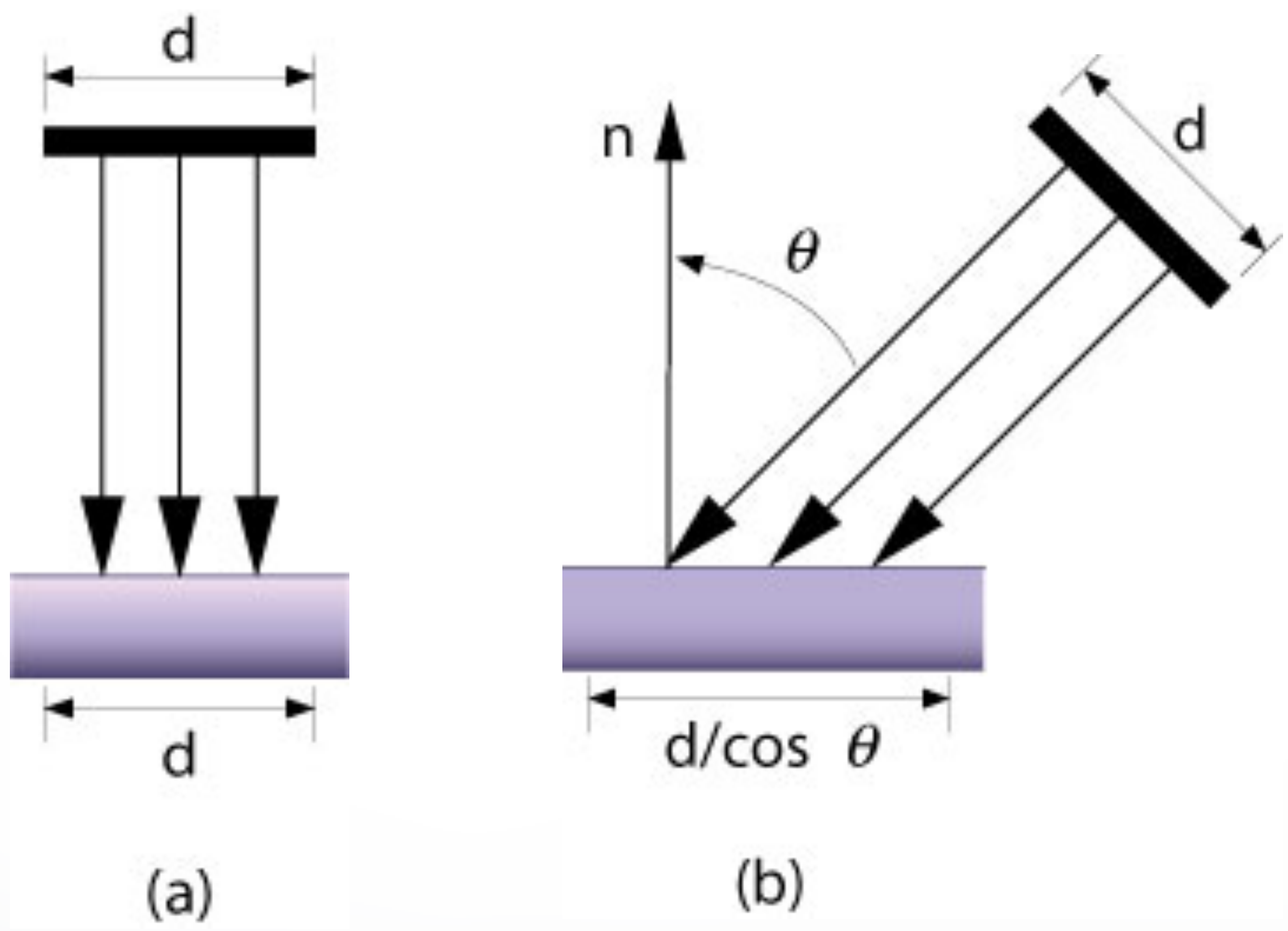
Diffuse Reflection

- Diffuse reflector scatters light
- Assume equally all direction
- Called Lambertian surface
- Diffuse reflection coefficient k_d (material), $0 \leq k_d \leq 1$
- Angle of incoming light is important



Lambert's Law

- Intensity depends on angle of incoming light.



Diffuse Light Intensity Depends On Angle Of Incoming Light

- Recall

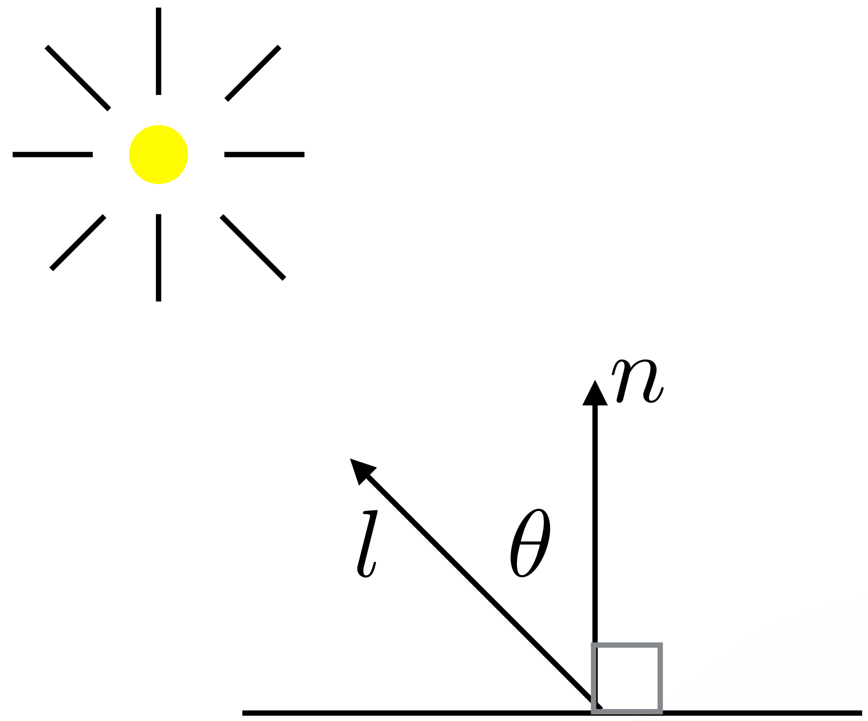
  $l =$ unit vector to light

  $n =$ unit surface normal

  $\theta =$ angle to normal

- $\cos(\theta) = l \cdot n$

- $$I_d = k_d L_d (l \cdot n)$$



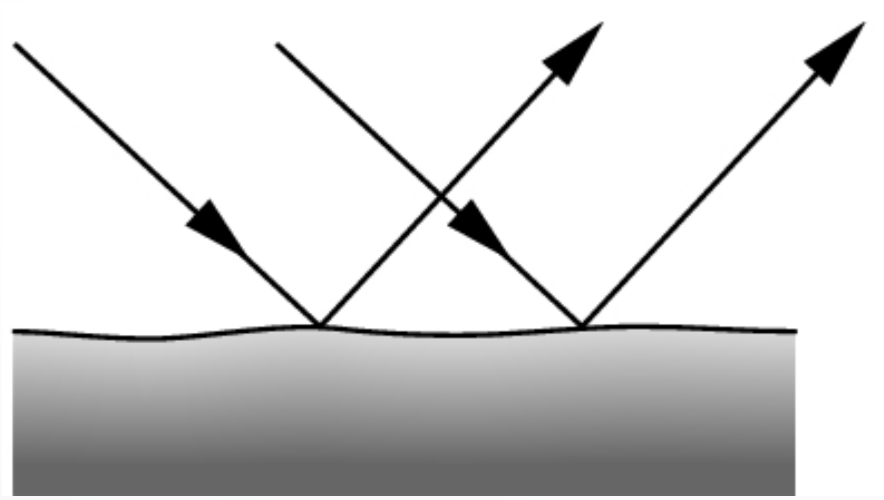
- With attenuation:

$$I_d = \frac{k_d L_d}{a + bq + cq^2}(l \cdot n)$$

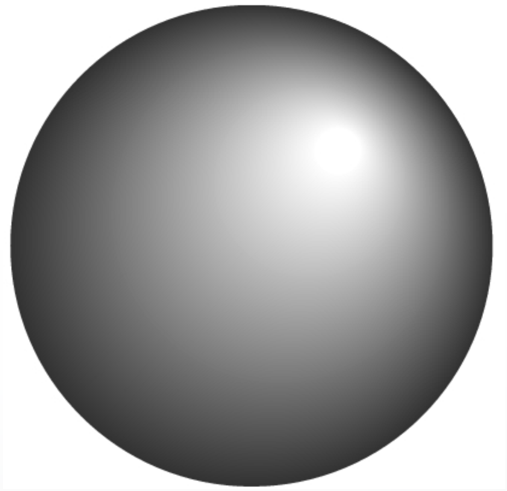
q = distance to light source,
L_d = diffuse component of light

Specular Reflection

- Specular reflection coefficient k_s (material), $0 \leq k_s \leq 1$
- Shiny surfaces have high specular coefficient
- Used to model specular highlights
- Does not give mirror effect
  (need other techniques)



specular reflection



specular highlights

Specular Reflection

- Recall

  $v =$ unit vector to camera

  $r =$ unit reflected vector

  $\phi =$ angle between v and r

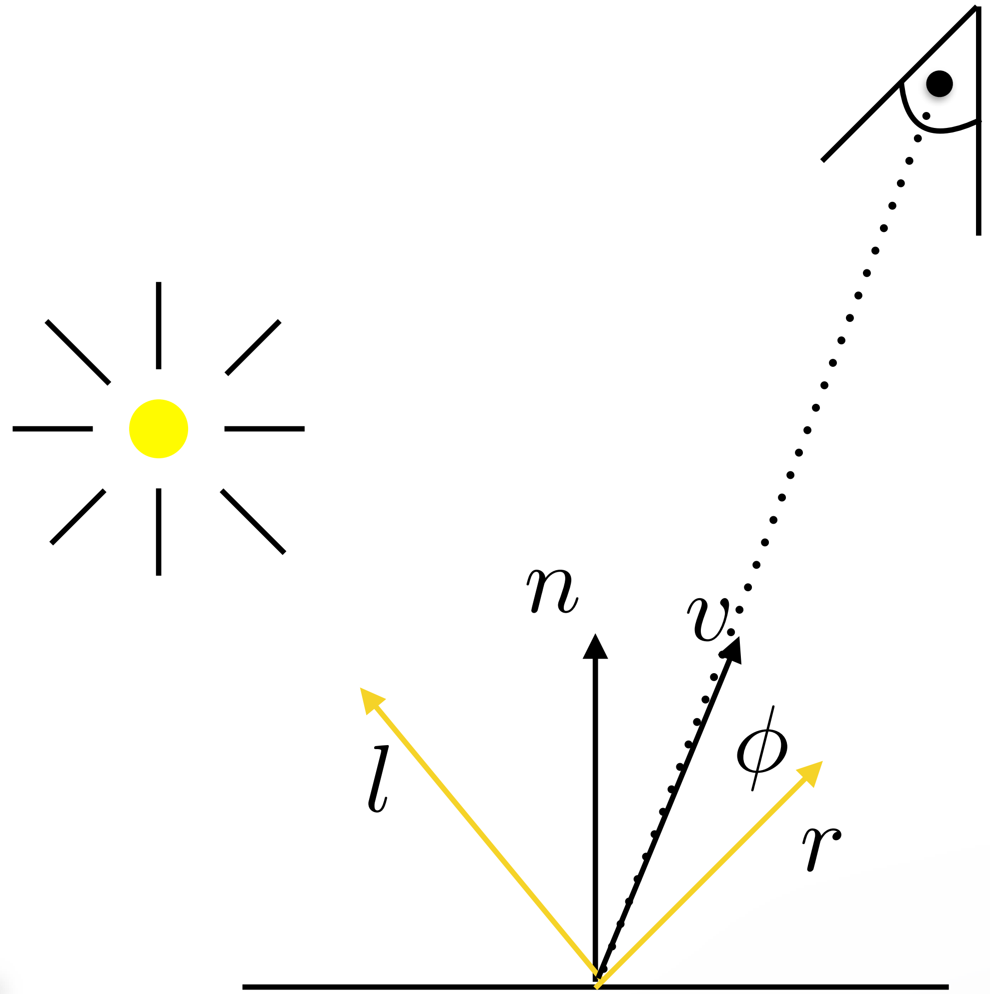
- $\cos(\phi) = v \cdot r$

- $$I_s = k_s L_s (\cos\phi)^\alpha$$

- L_s is specular component of light

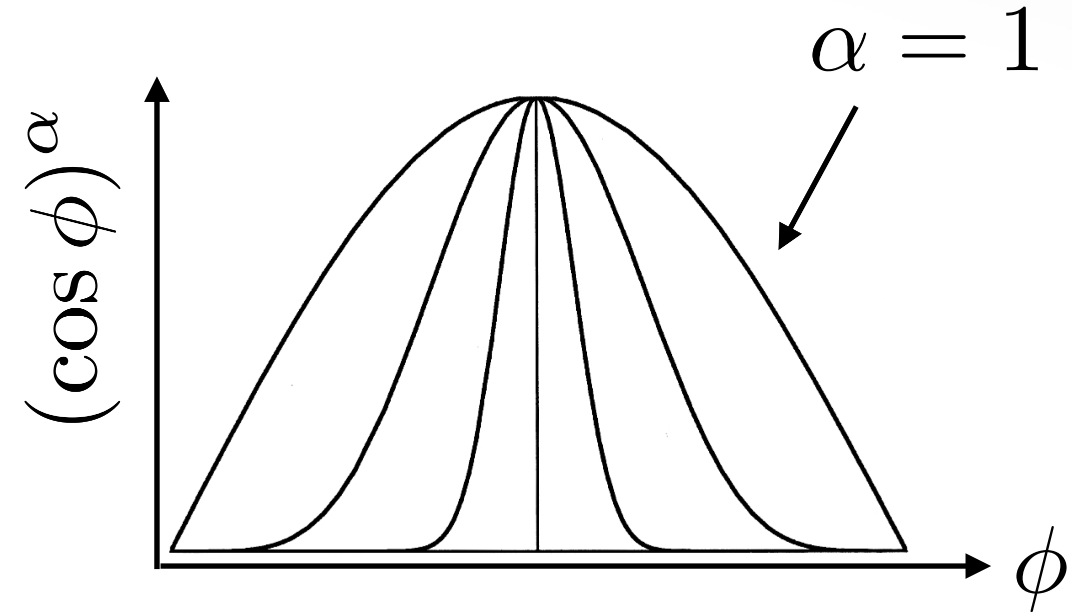
- α is shininess coefficient

- Can add distance term as well

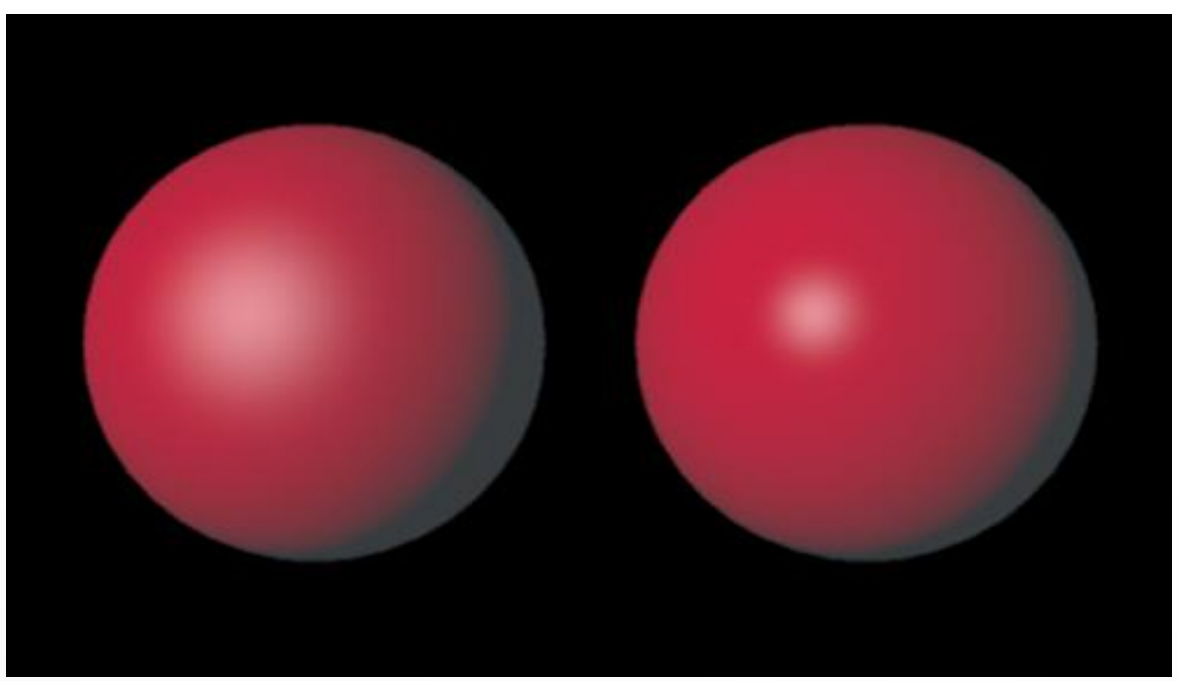


Shininess Coefficient

- $I_s = k_s L_s (\cos\phi)^\alpha$
- α is the shininess coefficient



Higher α gives narrower curves



low α high α

Source:
Univ. of Calgary

Summary of Phong Model

- Light components for each color:
  - Ambient (L_a), diffuse (L_d), specular (L_s)
- Material coefficients for each color:
  - Ambient (k_a), diffuse (k_d), specular (k_s)
- Distance q for surface point from light source

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

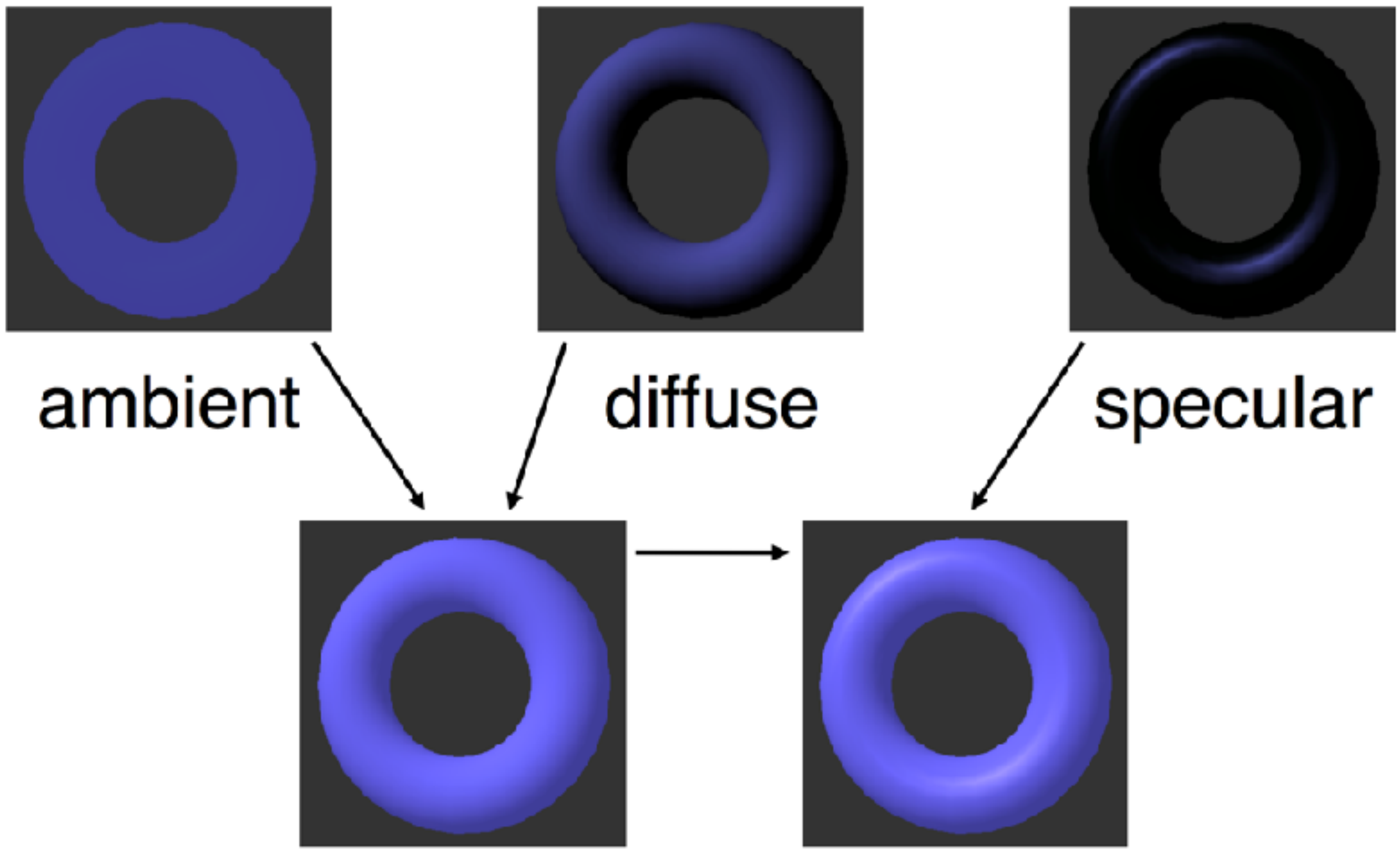
l = unit vector to light

$r = l$ reflected about n

n = surface normal

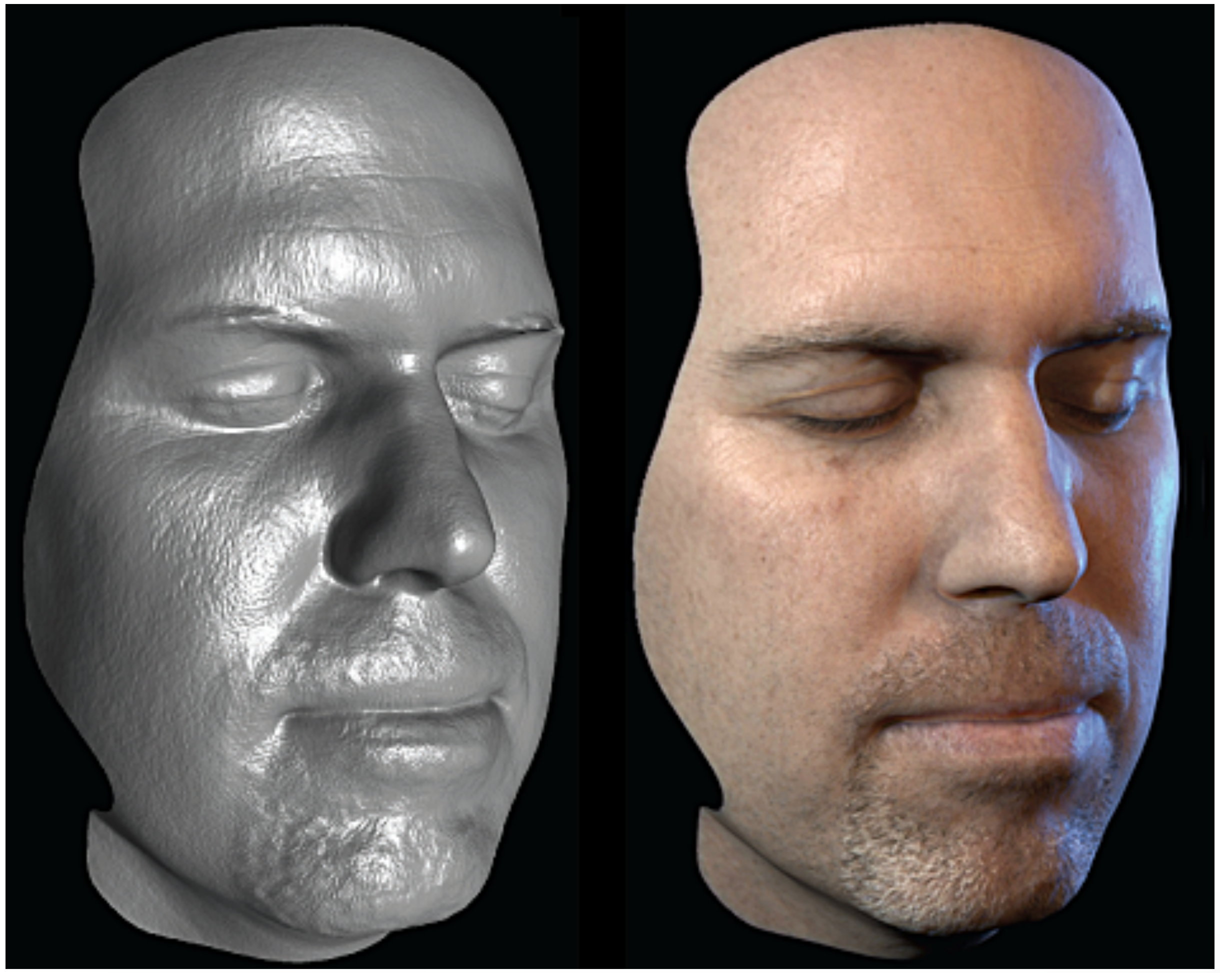
v = vector to viewer

Summary of Phong Model



BRDF

- Bidirectional Reflection Distribution Function
- Must measure for real materials
- Isotropic vs. anisotropic
- Mathematically complex
- Programmable pixel shading



Lighting properties of a human face were captured and face re-rendered; Institute for Creative Technologies

Summary

- Global and Local Illumination
- Normal Vectors
- Light Sources
- Phong Illumination Model

http://cs420.hao-li.com

Thanks!