Computer Graphics (CS 4731)
Lecture 16: Lighting, Shading and Materials (Part 1)

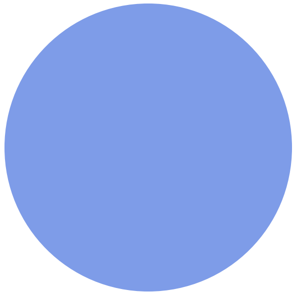
Prof Emmanuel Agu

Computer Science Dept.

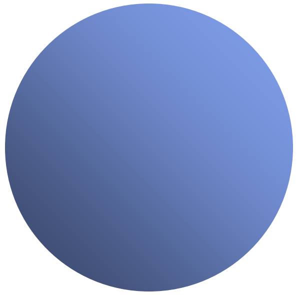
Worcester Polytechnic Institute (WPI)

Why do we need Lighting & shading?

- Sphere without lighting & shading

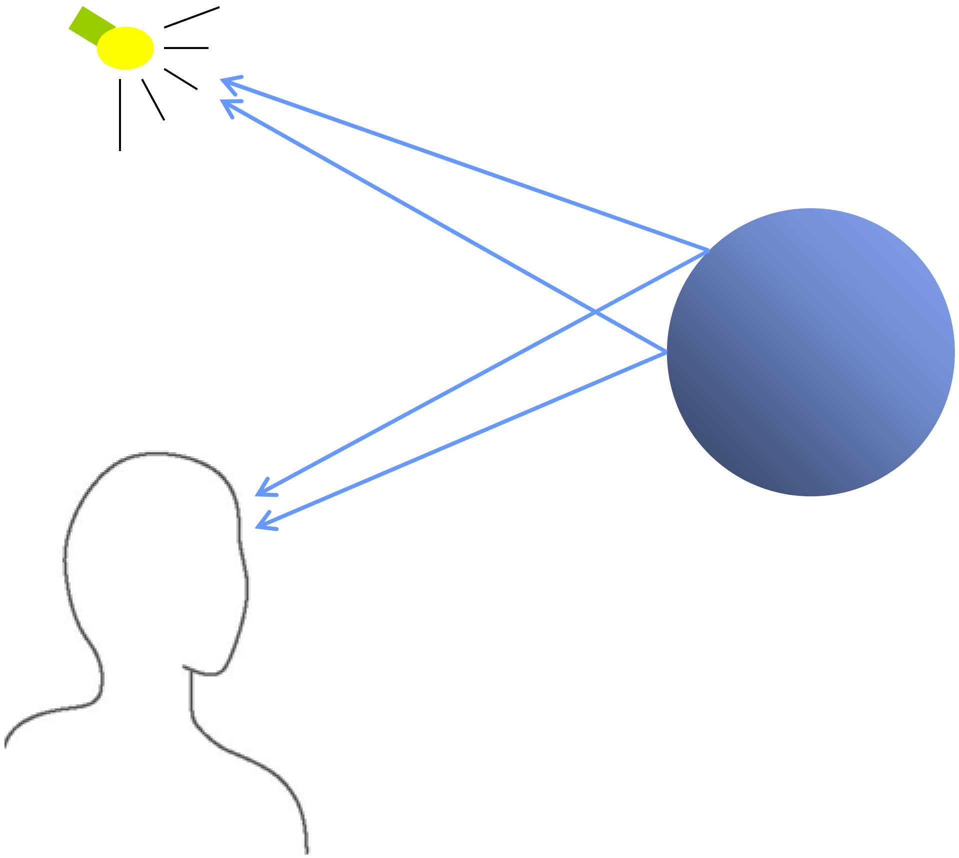


- We want (sphere with shading):
  - Has **visual cues** for humans (shape, light position, viewer position, surface orientation, material properties, etc)



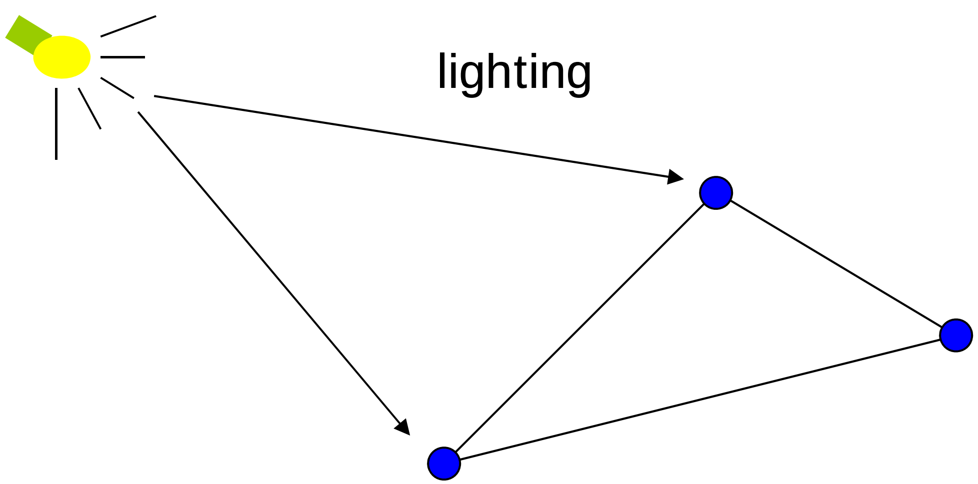
What Causes Shading?

- Shading caused by different angles with light, camera at different points



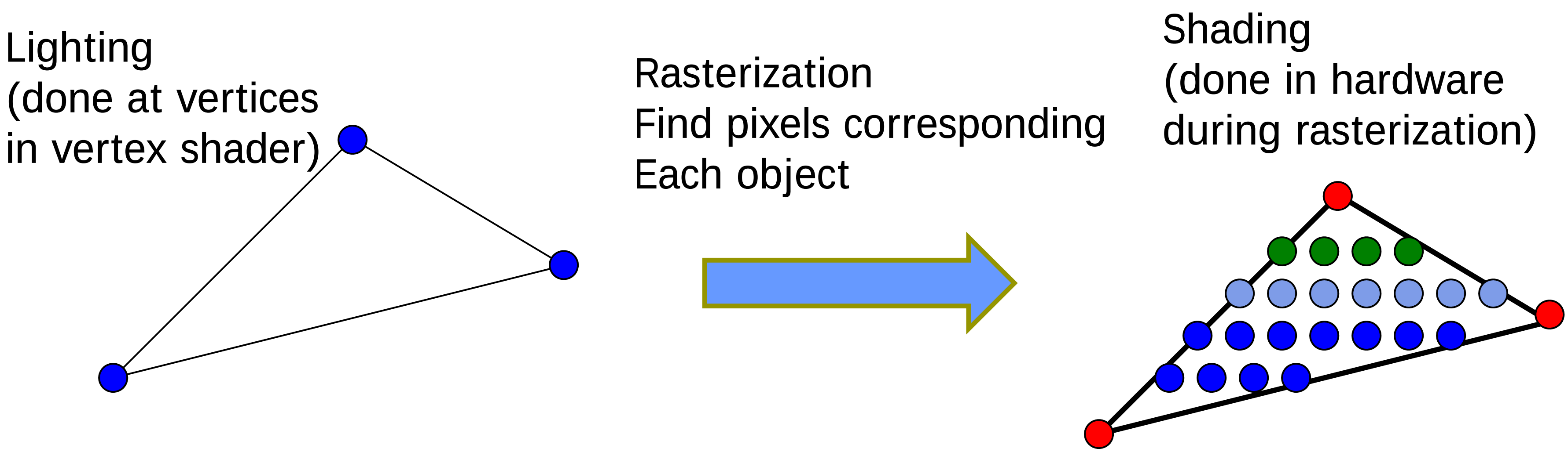
Lighting?

- **Problem:** Model light-surface interaction **at vertices** to determine vertex color and brightness
- Calculate lighting based on angle that surface makes with light, viewer
- Per vertex calculation? Usually done in vertex shader



Shading?

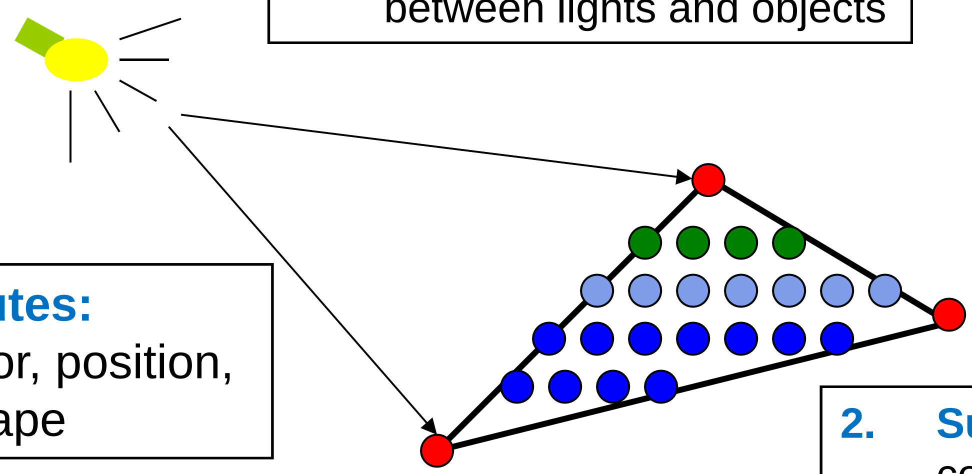
- After triangle is rasterized (drawn in 2D)
  - Triangle converted to pixels
  - Per-vertex lighting calculation means we know color of pixels coinciding with vertices (red dots)
- Shading: figure out color of interior pixels
- How? Assume linear change => interpolate



Lighting (or Illumination) Model?

- Equation for computing illumination
- Usually includes:

3. Interaction
between lights and objects



1. Light attributes:
intensity, color, position, direction, shape

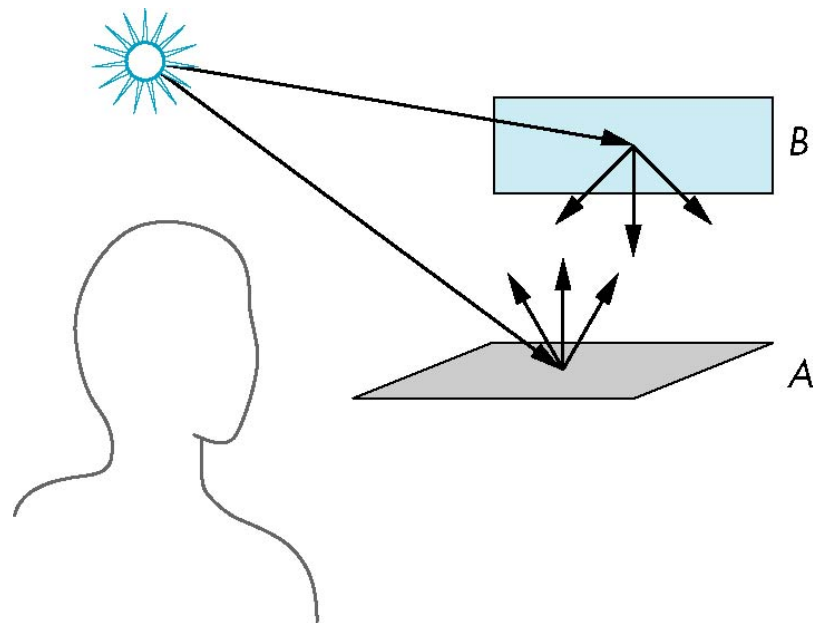
2. Surface attributes
color, reflectivity, transparency, etc

Light Bounces at Surfaces

- Light strikes A
  - Some reflected
  - Some absorbed
- Some reflected light

  from A strikes B
  - Some reflected
  - Some absorbed
- Some of this reflected

light strikes A and so on

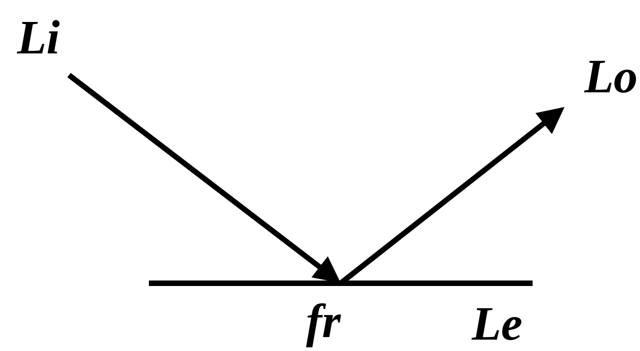


- The infinite reflection, scattering and absorption of light is described by the ***rendering equation***

Rendering Equation

- Introduced by James Kajiya in 86 Siggraph paper.
- Mathematical basis for all global illumination algorithms

$$L_o = L_e(x,\vec{\omega}) + \int_{\Omega} fr(x,\vec{\omega}',\vec{\omega}) Li(x,\vec{\omega}')(\vec{\omega}'\cdot\vec{n})d\vec{\omega}'$$



- *Lo* is outgoing radiance
- *Li* incident radiance
- *Le* emitted radiance,
- ***fr*** is bidirectional reflectance distribution function (BRDF)
  - Describes how a surface reflects light energy
  - Fraction of incident light reflected

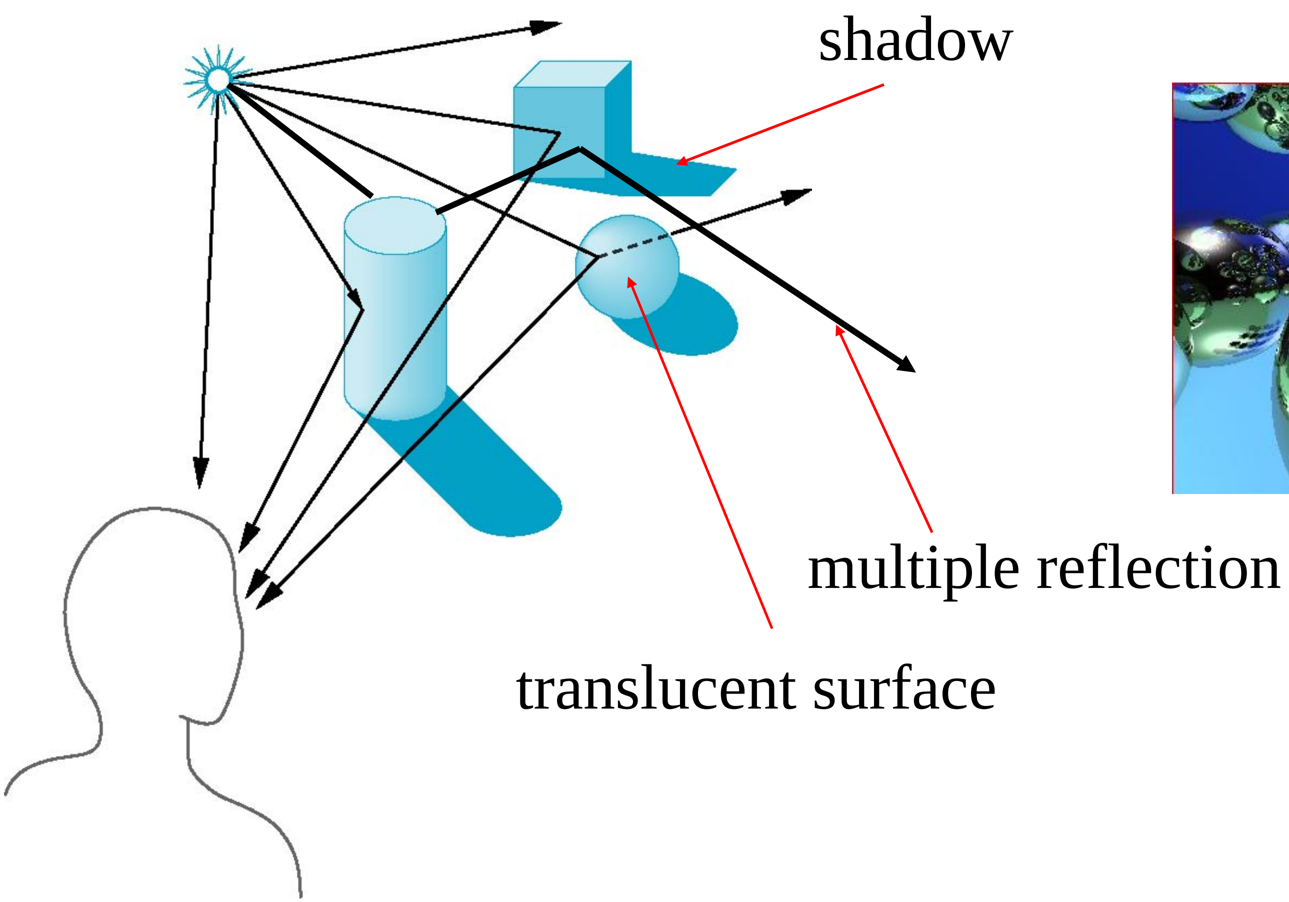
Rendering Equation

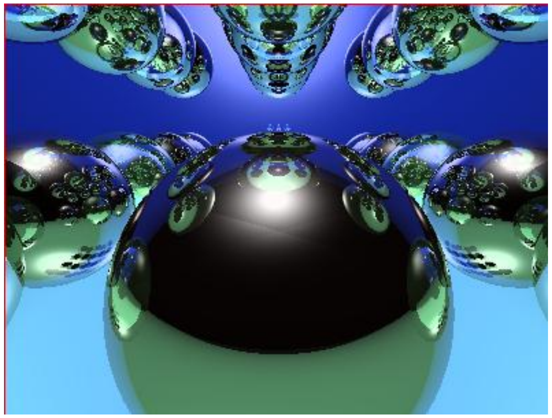
$$L_o = L_e(x,\vec{\omega}) + \int_{\Omega} fr(x,\vec{\omega}',\vec{\omega}) Li(x,\vec{\omega}')(\vec{\omega}' \cdot \vec{n}) d\vec{\omega}'$$

- Rendering equation includes many effects
  - Reflection
  - Shadows
  - Multiple scattering from object to object
- ***Rendering equation*** cannot be solved in general
- Rendering algorithms solve approximately. E.g. by sampling discretely

Global Illumination (Lighting) Model

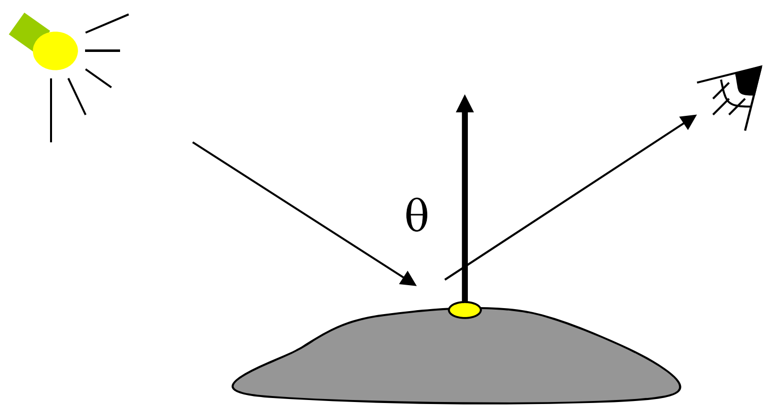
- Global illumination: model interaction of light from all surfaces in scene (track multiple bounces)





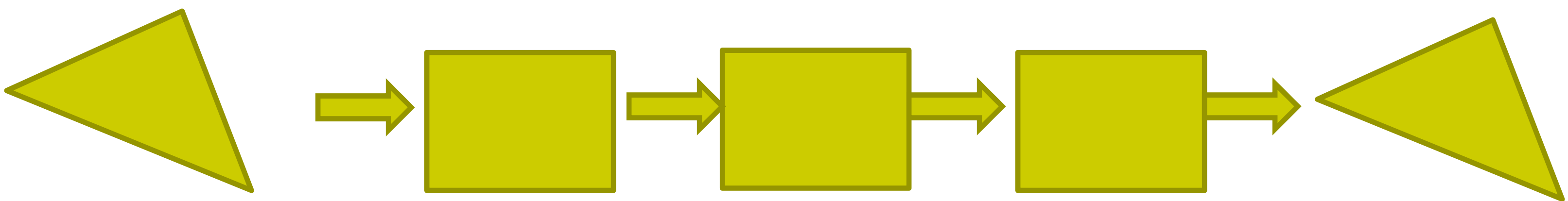
Local Illumination (Lighting) Model

- One bounce!
  - Doesn't track inter-reflections, transmissions
- Simple! Only considers
  - Light
  - Viewer position
  - Surface Material properties



Local vs Global Rendering

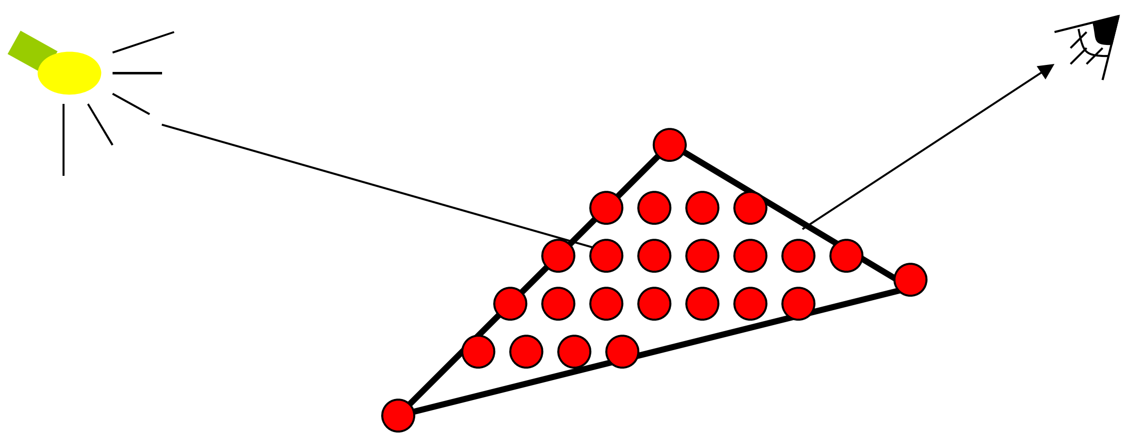
- Global Illumination is accurate, looks real
  - But raster graphics pipeline (like OpenGL) renders each polygon independently (local rendering)



- OpenGL cannot render full global illumination
- However, we can use techniques exist for approximating (faking) global effects

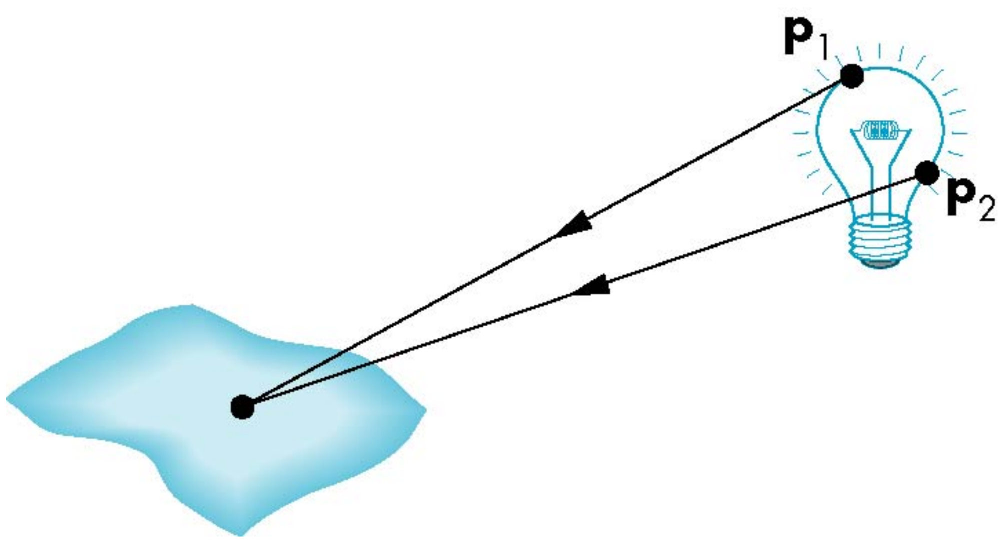
Light-Material Interaction

- Light strikes object, some absorbed, some reflected
- Fraction reflected determines object color and brightness
  - **Example:** A surface looks red under white light because red component of light is reflected, other wavelengths absorbed
- Reflected light depends on surface **smoothness** and **orientation**



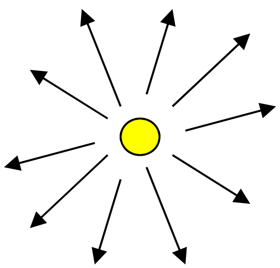
Light Sources

- General light sources are difficult to model because we must compute effect of light coming from all points on light source

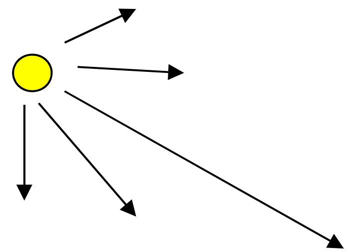


Basic Light Sources

- We generally use simpler light sources
- Abstractions that are easier to model

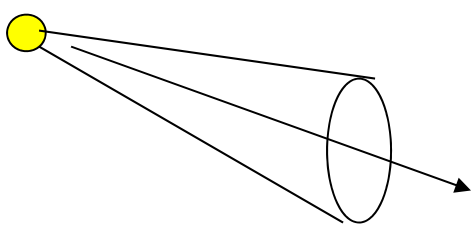


Point light

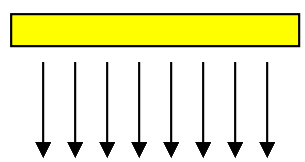


Directional light

Light intensity can be **independent** or **dependent** of the distance between object and the light source



Spot light



Area light

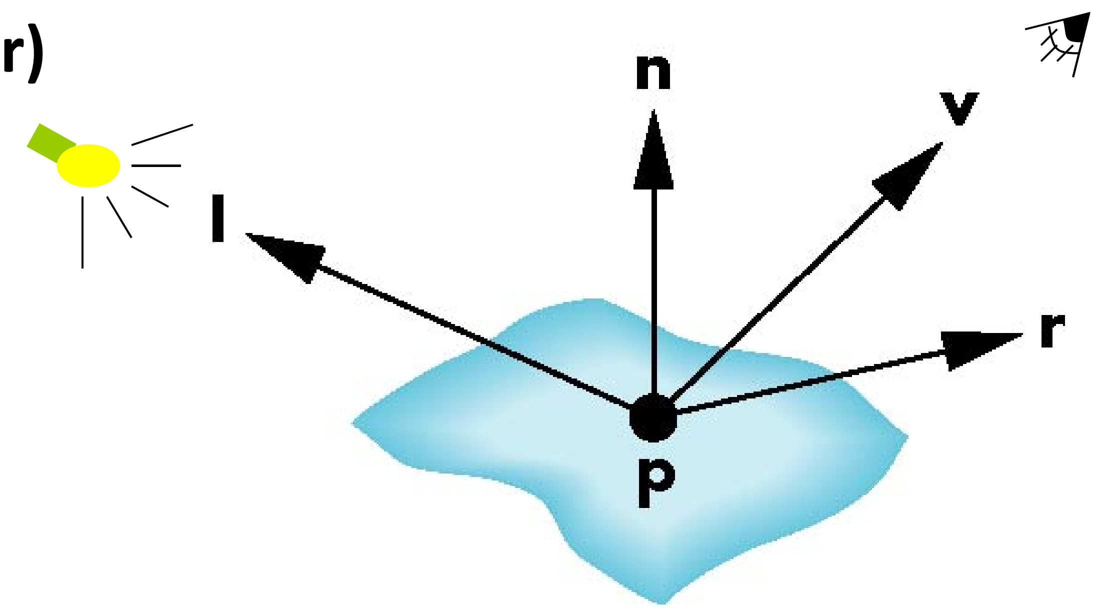
Phong Model

- Simple lighting model that can be computed quickly
- 3 components
  - Diffuse
  - Specular
  - Ambient
- Compute each component separately
- Vertex Illumination =

  ambient + diffuse + specular
- Materials reflect each component differently
  - Material reflection coefficients control reflection

Phong Model

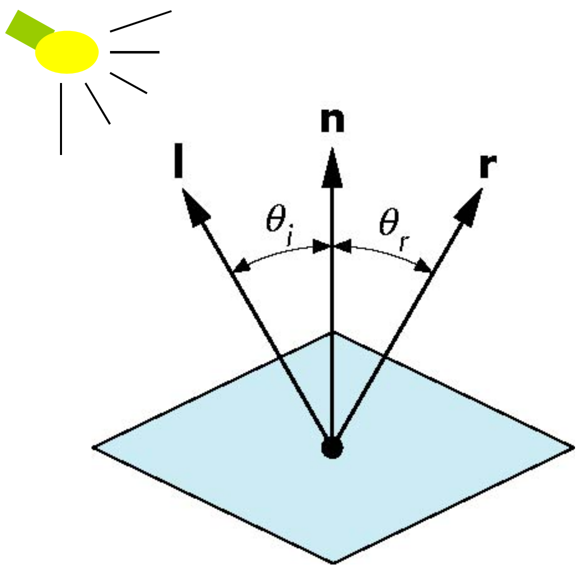
- Compute lighting (components) at each vertex (P)
- Uses 4 vectors, from vertex
  - To light source **(l)**
  - To viewer **(v)**
  - Normal **(n)**
  - Mirror direction **(r)**



Mirror Direction?

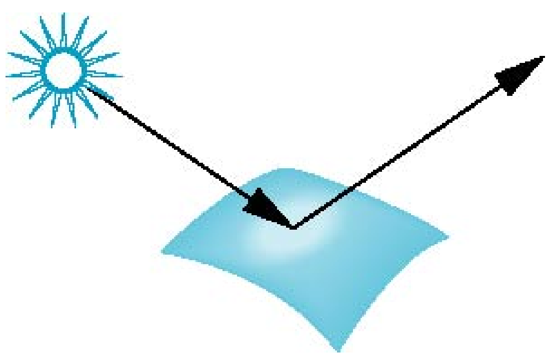
- Angle of reflection = angle of incidence
- Normal is determined by surface orientation
- The three vectors must be coplanar

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

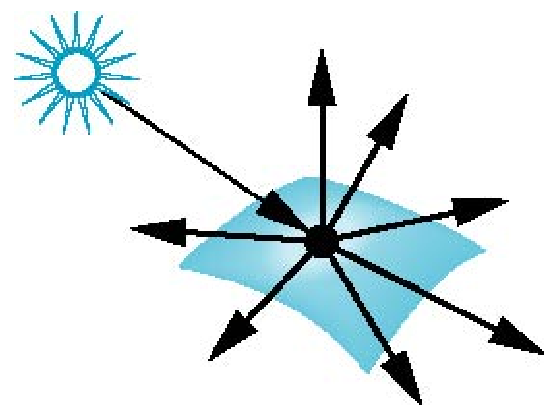


Surface Roughness

- **Smooth surfaces:** more reflected light concentrated in mirror direction
- **Rough surfaces:** reflects light in all directions



smooth surface

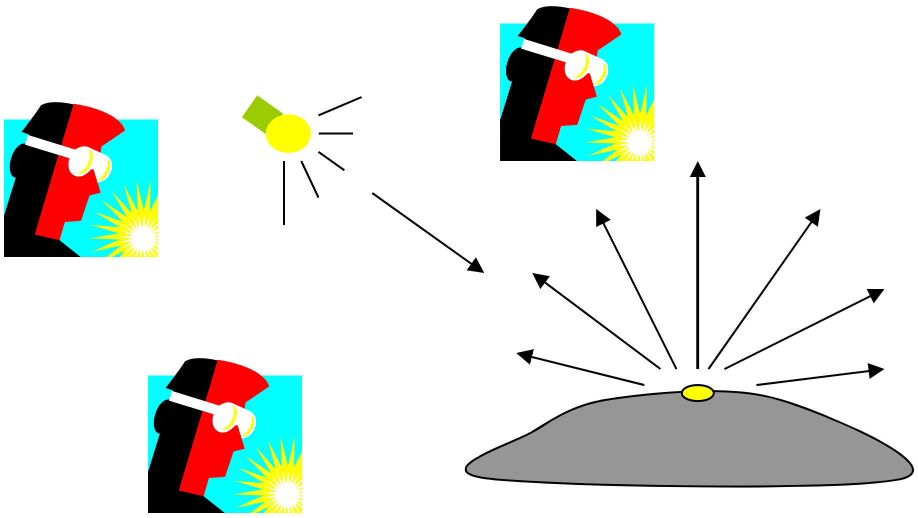


rough surface

Diffuse Lighting Example

Diffuse Light Reflected

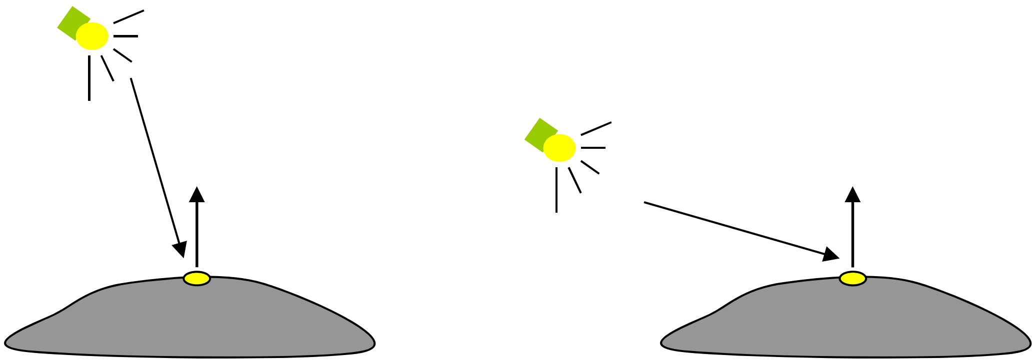
- Illumination surface receives from a light source and reflects equally in all directions



Eye position does not matter

Diffuse Light Calculation

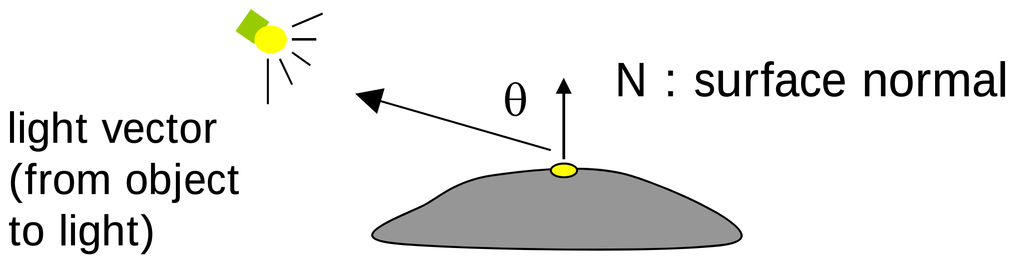
- How much light received from light source?
- Based on Lambert's Law



Receive more light

Receive less light

Diffuse Light Calculation



- Lambert's law: radiant energy D a small surface patch receives from a light source is:

$$D = I \times k_D \cos(\theta)$$

  - I: light intensity
  - θ: angle between light vector and surface normal
  - k_D: Diffuse reflection coefficient.
    Controls how much diffuse light surface reflects

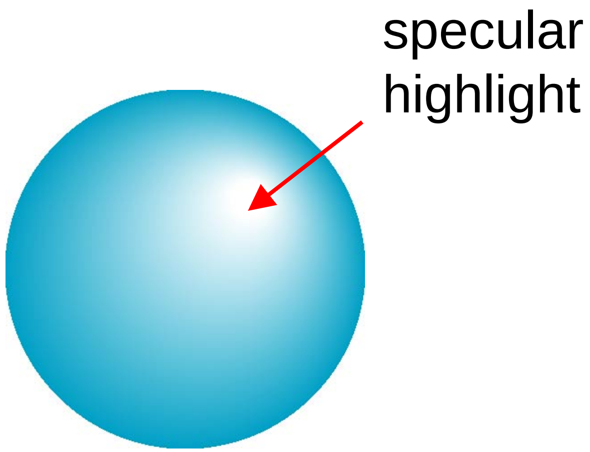
Specular light example

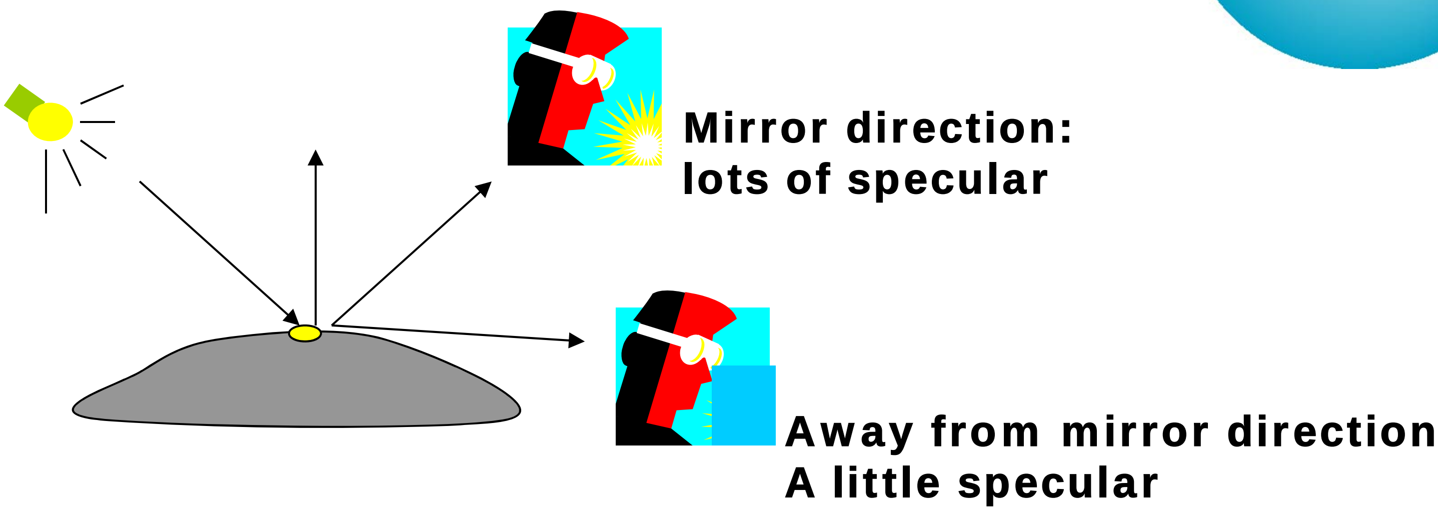
Specular?
Bright spot
on object

Specular light contribution

- Incoming light reflected out in small surface area
- Specular bright in **mirror direction**
- Drops off away from mirror direction
- Depends on viewer position relative

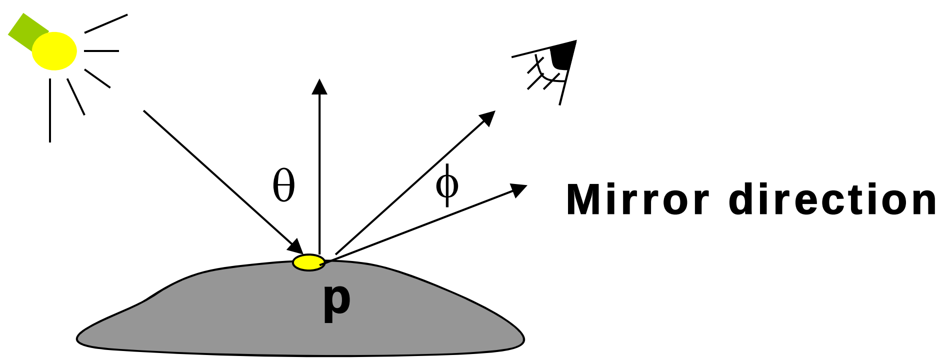
  to mirror direction



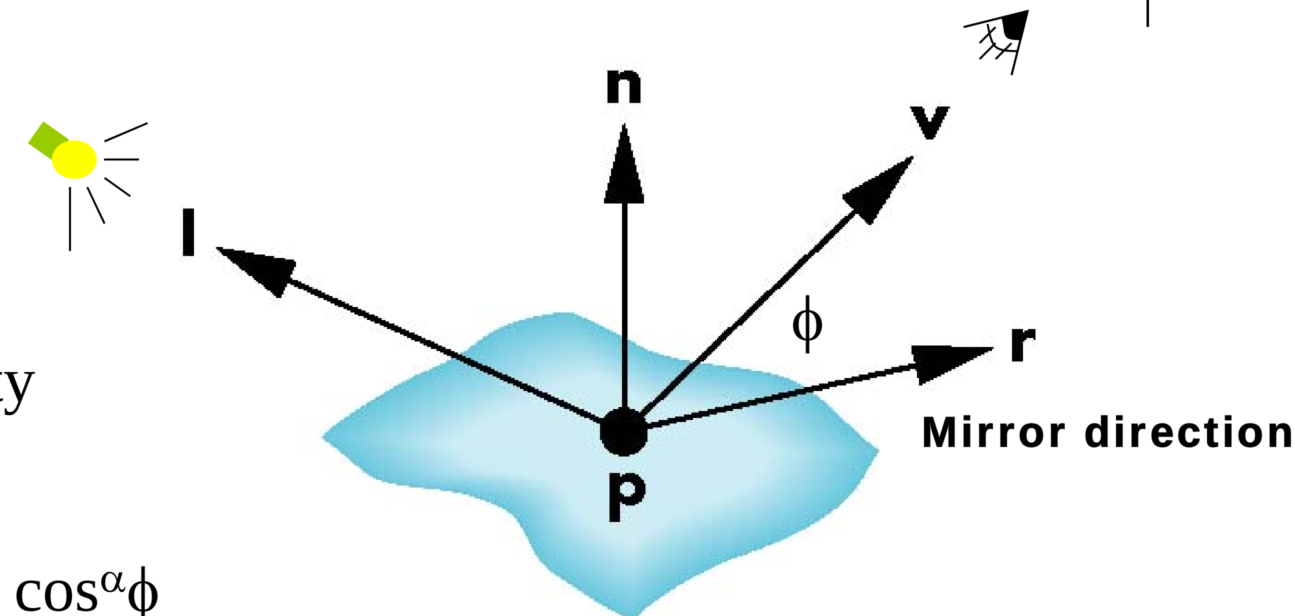


Specular light calculation

- Perfect reflection surface: all specular seen in mirror direction
- Non-perfect (real) surface: some specular still seen away from mirror direction
- ϕ is deviation of view angle from mirror direction
- Small ϕ = more specular



Modeling Specular Relections



incoming intensity

$$\mathbf{I}_s = k_s\, \mathbf{I} \cos^{\alpha}\phi$$

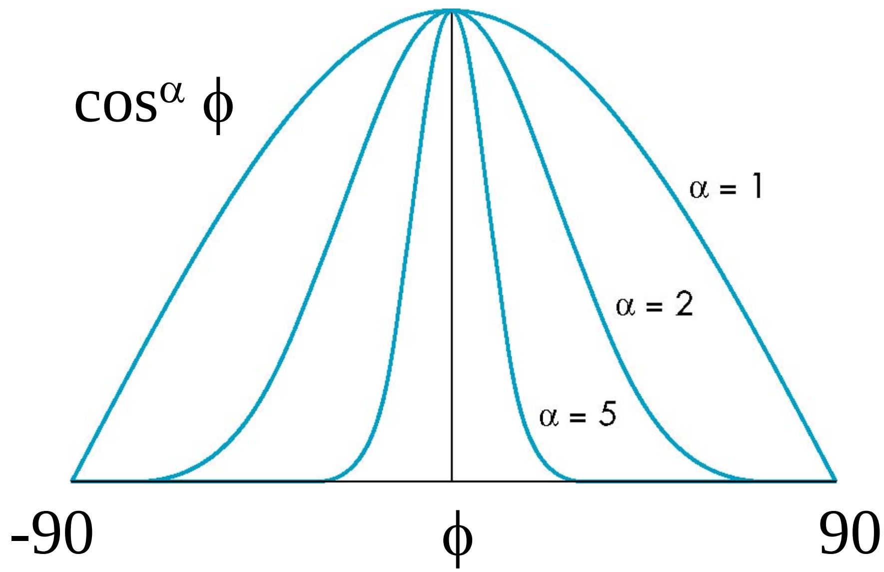
reflected intensity

shininess coef

Absorption coef

The Shininess Coefficient, α

- α controls falloff sharpness
- High α = sharper falloff = small, bright highlight
- Low α = slow falloff = large, dull highlight
  - α between 100 and 200 = metals
  - α between 5 and 10 = plastic look



Specular light: Effect of ‘α’

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

α = 10

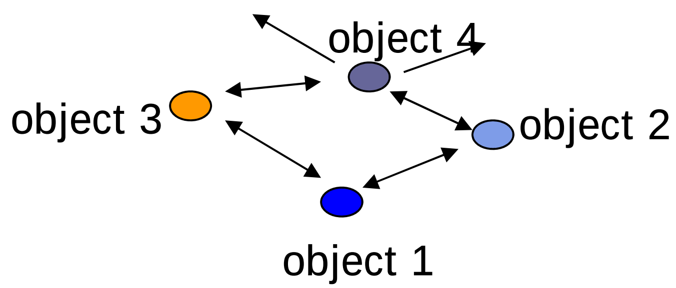
α = 90

α = 30

α = 270

Ambient Light Contribution

- Very simple approximation of global illumination
  (Lump 2nd, 3rd, 4th, …. etc bounce into single term)
- **Assume to be a constant**
- No direction!
  - Independent of light position, object orientation, observer's position or orientation



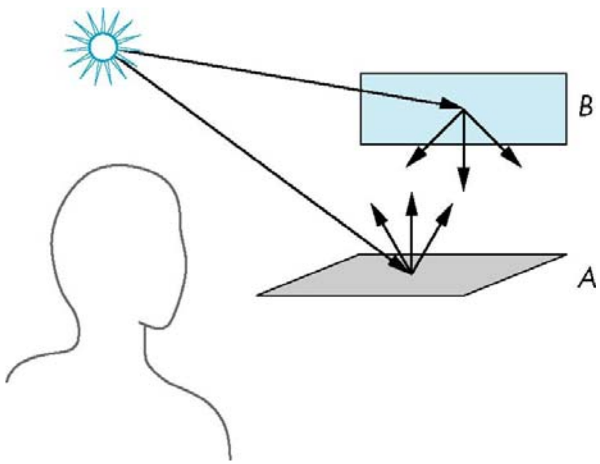
Ambient = Ia x Ka ← constant

Ambient Light Example

Ambient: background light, scattered by environment

Light Attentuation with Distance

- Light reaching a surface **inversely proportional** to square of distance
- We can multiply by factor of form $1/(ad + bd + cd^2)$ to **diffuse** and **specular** terms



Adding up the Components

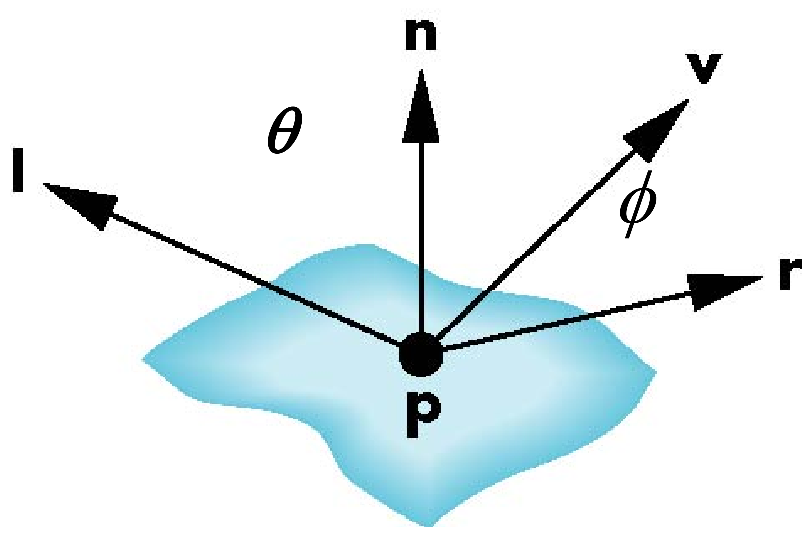
- Adding all components (no attentuation term) , phong model for each light source can be written as

diffuse + specular + ambient

$$I = k_d I_d \cos\theta + k_s I_s \cos\phi^{\alpha} + k_a I_a$$

$$= k_d I_d (l \cdot n) + k_s I_s (\mathbf{v} \cdot \mathbf{r})^{\alpha} + k_a I_a$$

- **Note:**
  - $\cos\theta = \mathbf{l} \cdot \mathbf{n}$
  - $\cos\phi = \mathbf{v} \cdot \mathbf{r}$



Separate RGB Components

- We can separate red, green and blue components
- Instead of 3 light components I_d, I_s, I_a,
  - E.g. $I_d = I_{dr}, I_{dg}, I_{db}$
  - 9 coefficients for each point source
  - $I_{dr}, I_{dg}, I_{db}, I_{sr}, I_{sg}, I_{sb}, I_{ar}, I_{ag}, I_{ab}$
- Instead of 3 material components k_d, k_s, k_a,
  - E.g. $k_d = k_{dr}, k_{dg}, k_{db}$
  - 9 material absorption coefficients
  - $k_{dr}, k_{dg}, k_{db}, k_{sr}, k_{sg}, k_{sb}, k_{ar}, k_{ag}, k_{ab}$

Put it all together

- Can separate red, green and blue components. Instead of:

$$I = k_d\, I_d\, (l \cdot n) \;+\; k_s\, I_s\, (\mathbf{v} \cdot \mathbf{r})^{\alpha} \;+\; k_a\, I_a$$

- We computing lighting for RGB colors separately

$$I_r = k_{dr}\, I_{dr}\; \mathbf{l} \cdot \mathbf{n} \;+ k_{sr}\, I_{sr}\, (\mathbf{v} \cdot \mathbf{r})^{\alpha} + k_{ar}\, I_{ar}$$ **Red**

$$I_g = k_{dg}\, I_{dg}\; \mathbf{l} \cdot \mathbf{n} \;+ k_{sg}\, I_{sg}\, (\mathbf{v} \cdot \mathbf{r})^{\alpha} + k_{ag}\, I_{ag}$$ **Green**

$$I_b = k_{db}\, I_{db}\; \mathbf{l} \cdot \mathbf{n} \;+ k_{sb}\, I_{sb}\, (\mathbf{v} \cdot \mathbf{r})^{\alpha} + k_{ab}\, I_{ab}$$ **Blue**

- Above equation is just for one light source!!
- For N lights, repeat calculation for each light

Total illumination for a point P = Σ (Lighting for all lights)

Coefficients for Real Materials

Material	Ambient Kar, Kag,kab	Diffuse Kdr, Kdg,kdb	Specular Ksr, Ksg,ksb	Exponent, α
Black plastic	0.0 0.0 0.0	0.01 0.01 0.01	0.5 0.5 0.5	32
Brass	0.329412 0.223529 0.027451	0.780392 0.568627 0.113725	0.992157 0.941176 0.807843	27.8974
Polished Silver	0.23125 0.23125 0.23125	0.2775 0.2775 0.2775	0.773911 0.773911 0.773911	89.6

Figure 8.17, Hill, courtesy of McReynolds and Blythe

References

- Interactive Computer Graphics (6th edition), Angel and Shreiner
- Computer Graphics using OpenGL (3rd edition), Hill and Kelley