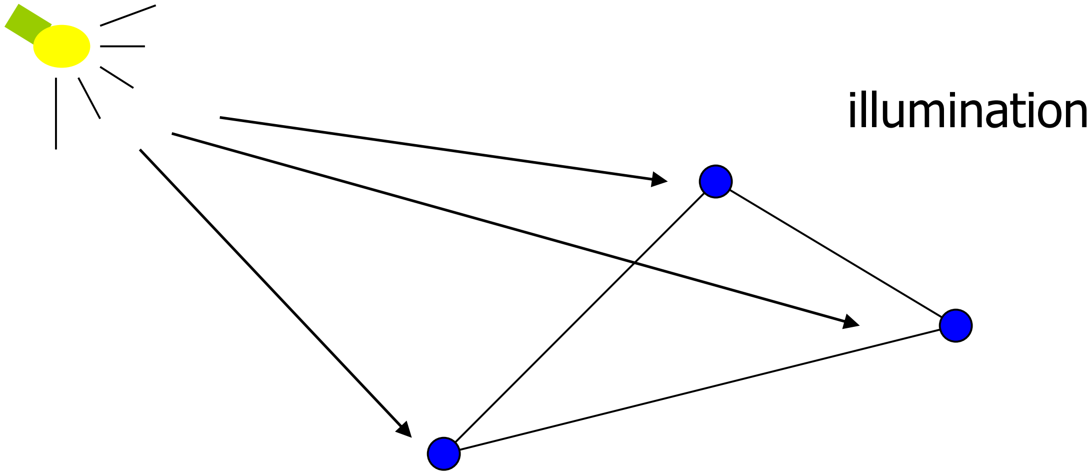
Dr. George Karraz, Ph. D.

Illumination and Shading

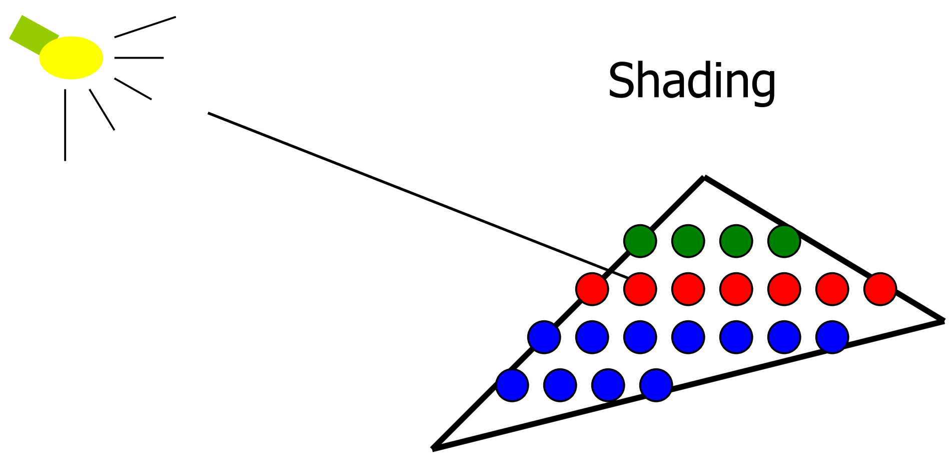
Illumination (Lighting)

- Model the interaction of light with surface points to determine their final color and brightness
- OpenGL computes illumination at vertices



Shading

- Apply the lighting model at a set of points across the entire surface

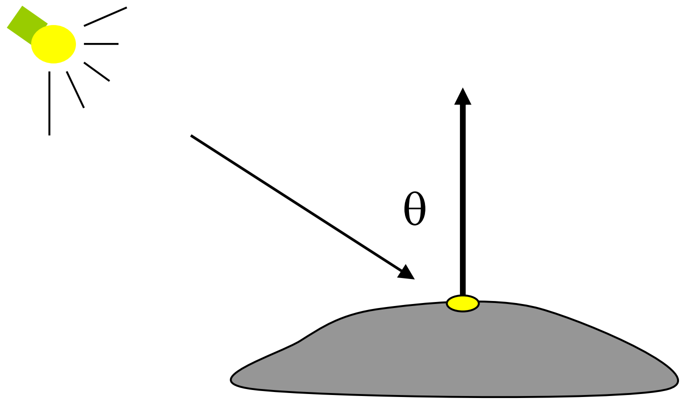


Illumination Model

- A illumination model usually considers:
  - Light attributes (light intensity, color, position, direction, shape)
  - Object surface attributes (color, reflectivity, transparency, etc)
  - Interaction among lights and objects (object orientation)
  - Interaction between objects and eye (viewing dir.)

Illumination Calculation

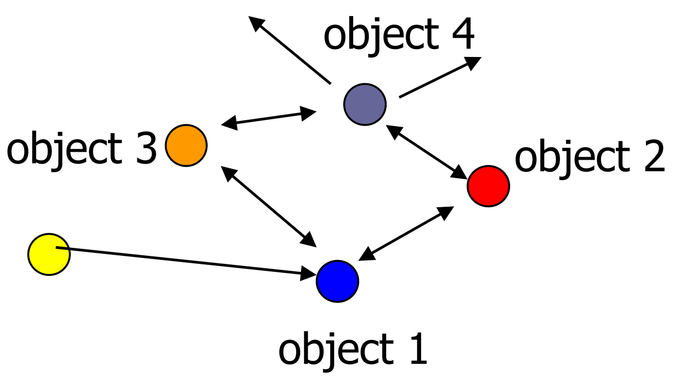
- Local illumination: only consider the light, the observer position, and the object material properties

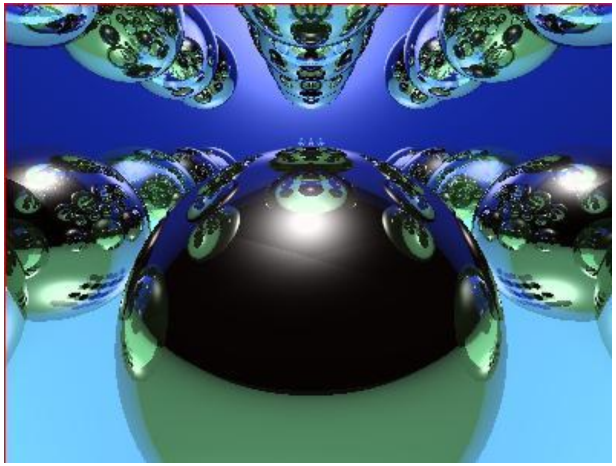


- Example: OpenGL

Illumination Models

- Global illumination: take into account the interaction of light from all the surfaces in the scene

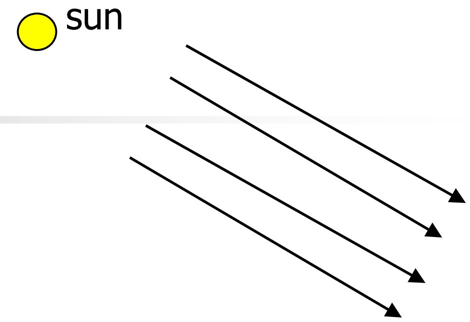




- Example: Ray Tracing

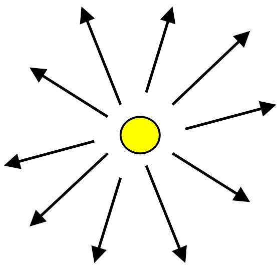
Basic Light Types

- Directional
  - So far away so that light rays are ∥
  - Remember orthogonal projection?



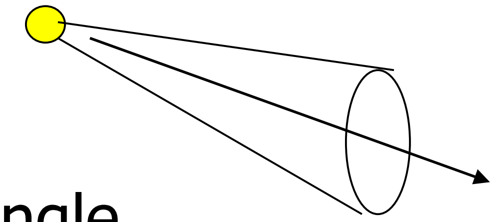
Directional light

- Point
  - Light emanates equally in all directions



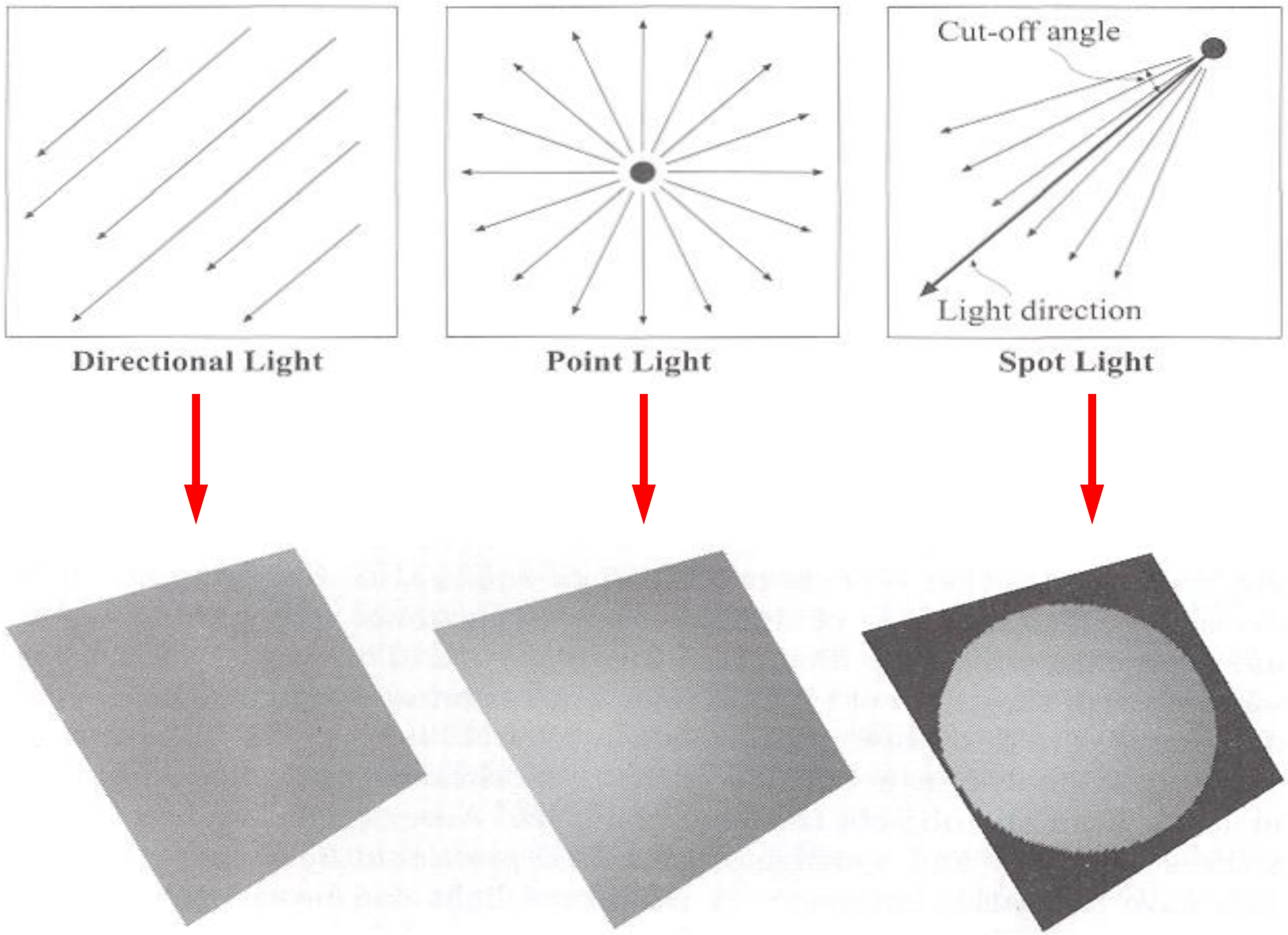
Point light

- Spot
  - Point source limited to an angle



Spot light

Light Source Types



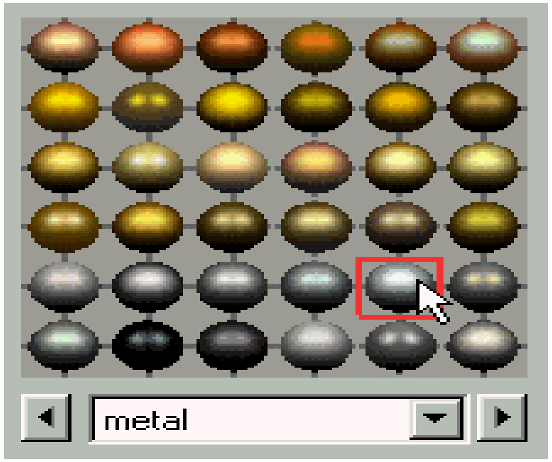
from Akenine-Moller & Haines

Object Properties

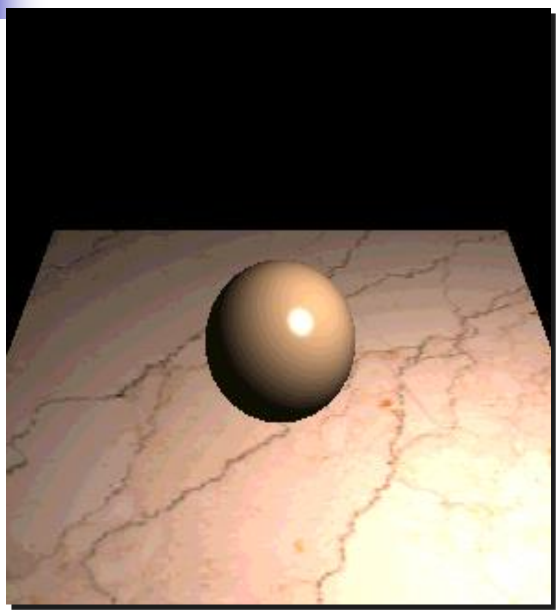
- What happens when light hits an object?
  - Properties of light reflection on an object's surface
    - Reflectance Models
      - Ambient
      - Diffuse
      - Specular
    - Absorption, Emission, Transparency/Translucency
  - Irradiance: All light that arrives at a point on the surface
  - Radiosity: Light leaving a surface in all directions

Object Properties

- Object Material
  - Shiny (Metal), dull (Matte finish), mirror-like, glass, neon, etc.

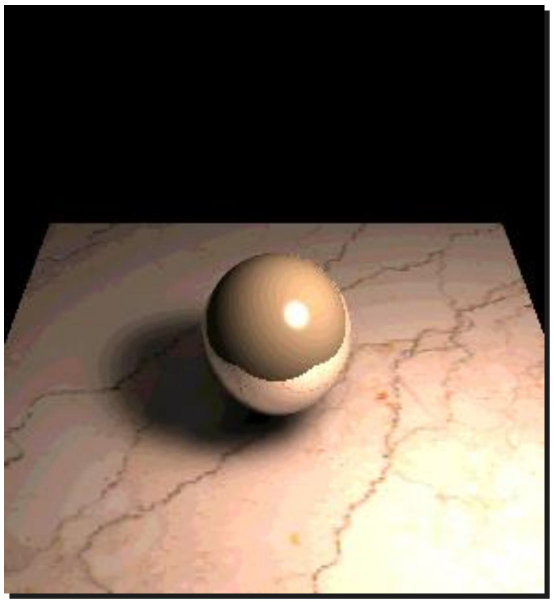


Local vs. Global Illumination



Local

Illumination depends on local object & light sources only



Global

Illumination at a point can depend on any other point in the scene

Simple local illumination

- The model used by OpenGL – considers three types of light contribution to compute the final illumination of an object
  - Ambient
  - Diffuse
  - Specular
- Final illumination of a point (vertex) =

  ambient + diffuse + specular

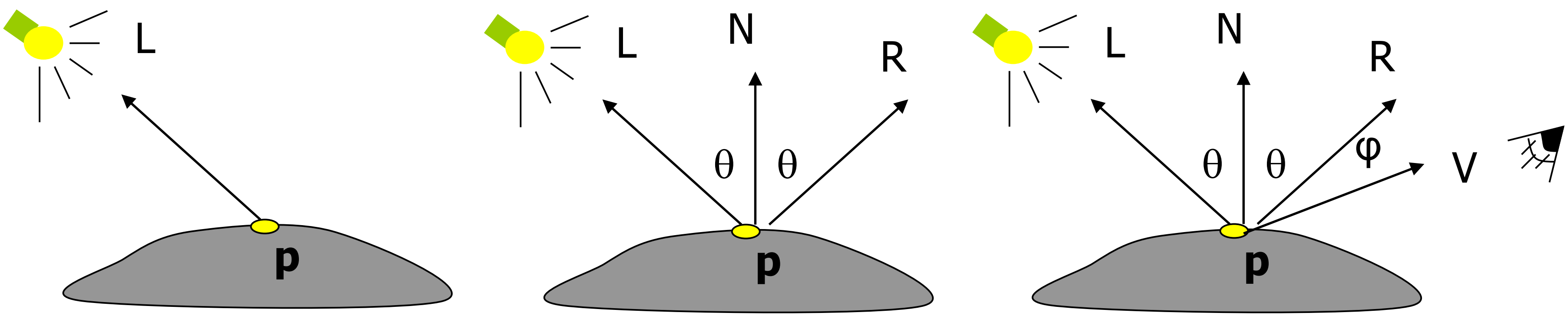
Ambient lighting example

Diffuse lighting example

Specular light example

Light Reflectance Components

Take a point P on the object surface:



L: Light Vector **R: Reflection Vector** **V: View Vector**

Reflects about
the **Normal (N)** to
the surface

Ambient Reflection

- Background light scattered by the environment
  - Light bounces off of many objects
  - Simple Global Illumination
- Simple reflectance model
  - Independent of …
    - Light position
    - Object orientation
    - Viewer's position
- $\mathbf{k_a}$: Ambient reflection coefficient
  - Ambient light an object reflects
  - $0 \leq \mathbf{k_a} \leq 1$

Ambient = $\mathbf{I_a} * \mathbf{k_a}$

Diffuse Reflection

- Lambert's Law:
  - Radiant energy **D** that a small surface patch receives from a light source:

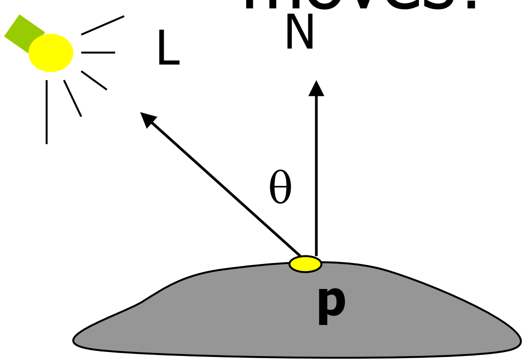
$$\mathbf{D = I_d * \cos(\theta)}$$

  - $\mathbf{I_d}$ = light intensity, θ = Angle between **L** and **N**

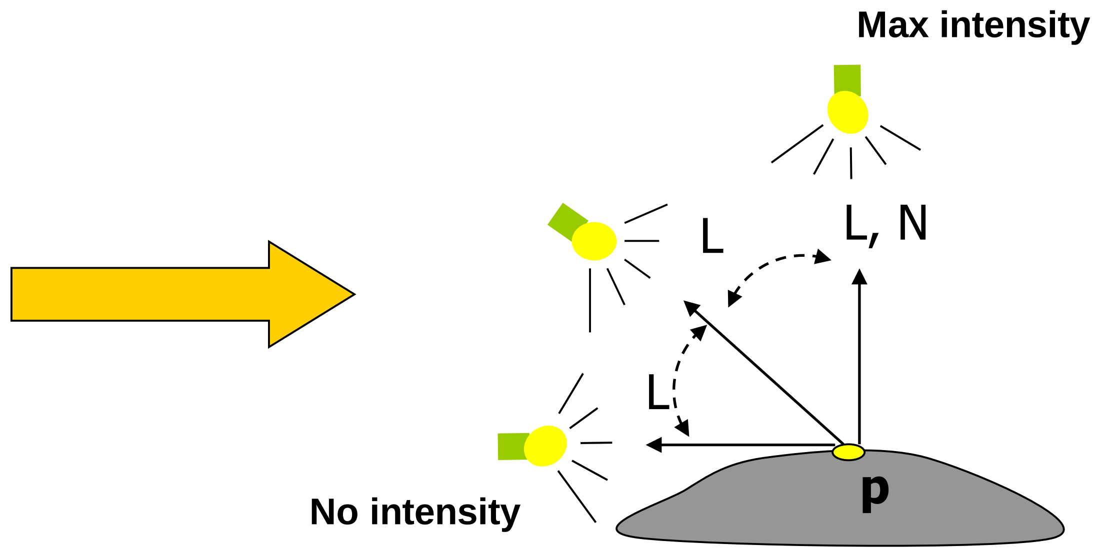
- Also called Lambertian or Matte surfaces

Lambert's Law (1)

- How does **D** change if the light source moves?

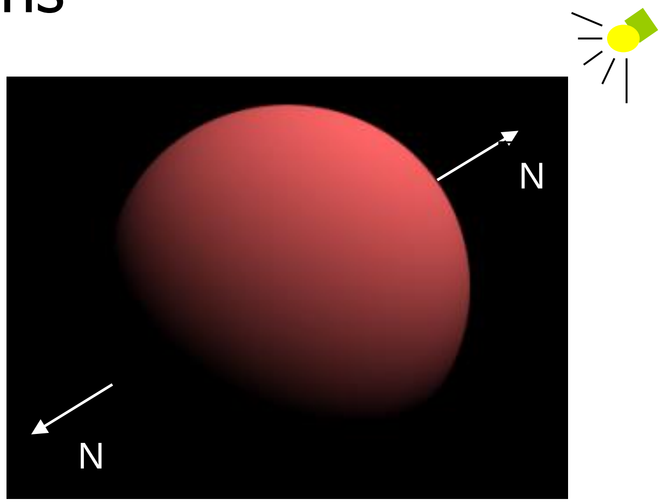


$$\mathbf{D = I_d * \cos(\theta)}$$



Lambert's Law (2)

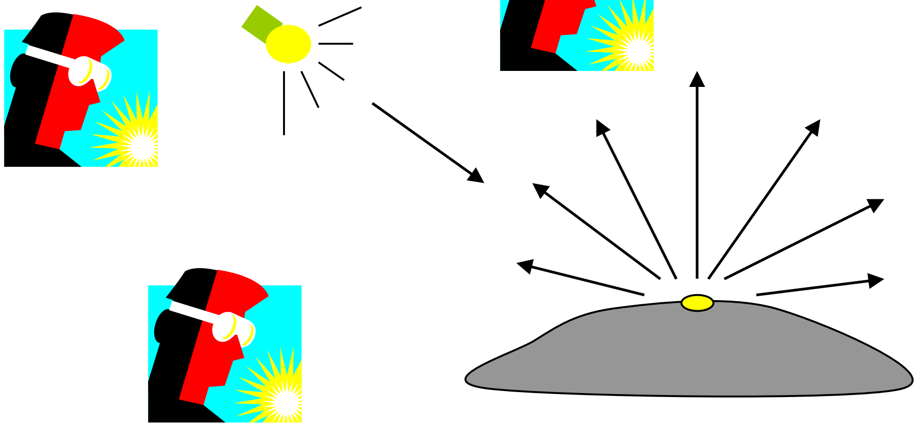
- How does **D** change on an object's surface?
  - A sphere's surface has all possible normal directions



Diffuse Reflection

- Energy **D** is reflected <u>equally</u> in all directions on the surface
  - Independent of …
    - Viewer's position

- $\mathbf{k_d}$: Diffuse reflection coefficient
  - Diffuse light an object reflects
  - $0 \leq \mathbf{k_d} \leq 1$



$$\text{Diffuse} = \mathbf{I_d} * \mathbf{k_d} * \mathbf{cos}(\theta)$$
$$= \mathbf{I_d} * \mathbf{k_d} * (\mathbf{N} \bullet \mathbf{L})$$

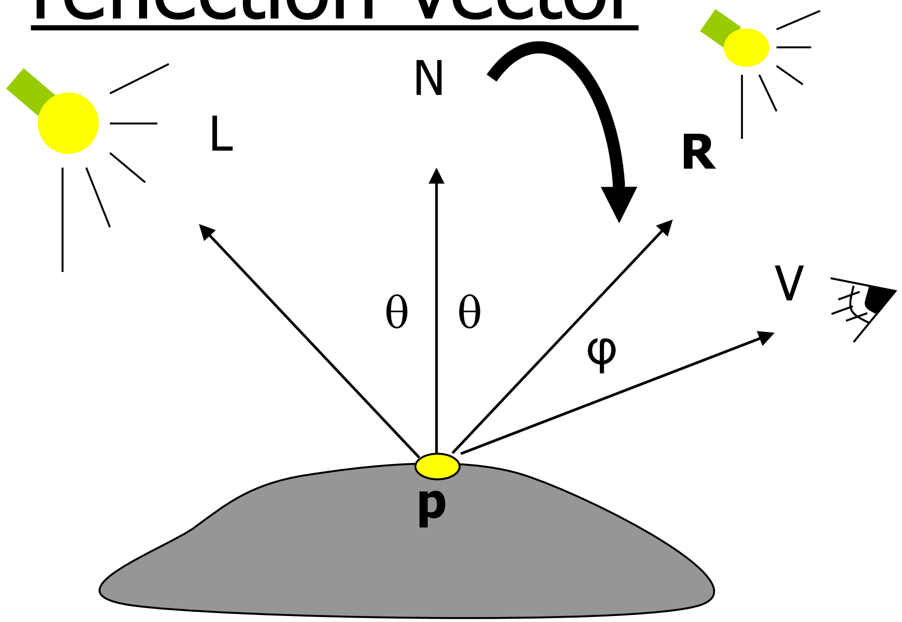
N and **L** must be normalized

Specular Reflection (1)

- The reflection of the light source on the object
- Shiny/Glossy surfaces
  - Not a perfect mirror

Specular Reflection (2)

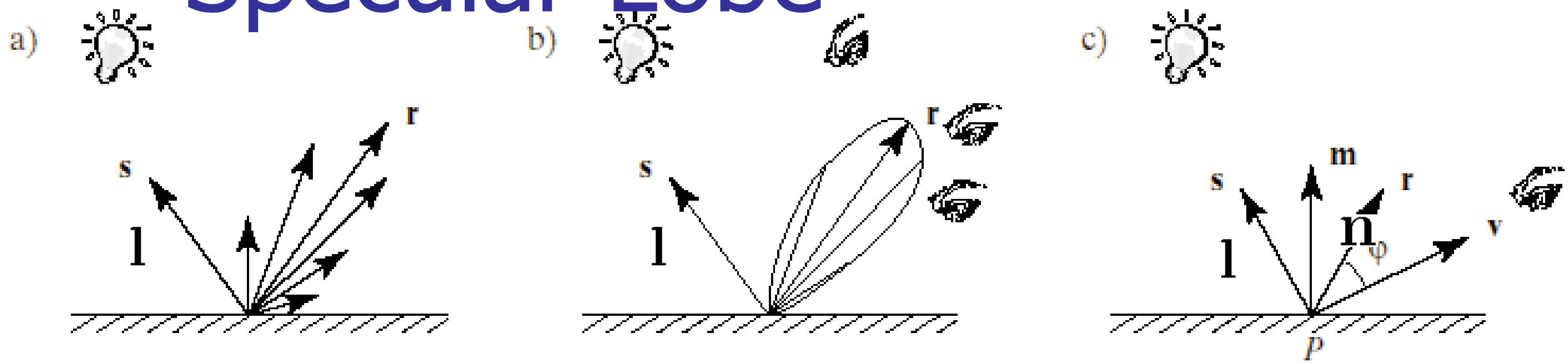
- The object reflects maximum light intensity in the direction of the <u>reflection vector</u>



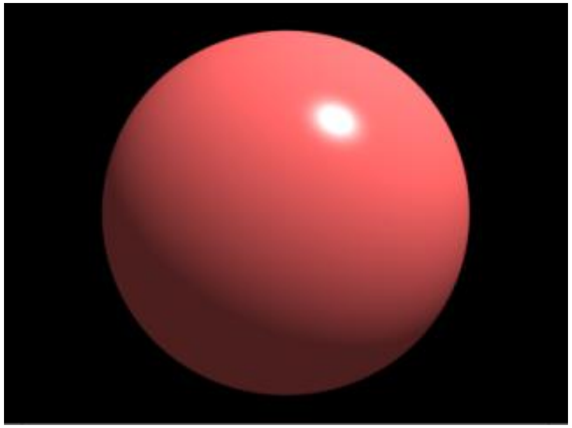
Light intensity increases as **V** gets closer to **R**

V • R = cos(φ)

Specular Lobe



- The reflection of the light source is maximum at the reflection direction
- Falls off quickly as the viewer moves away
- The size of the lobe determines the shininess of the object
- The shinier the object ⇒ the smaller the lobe

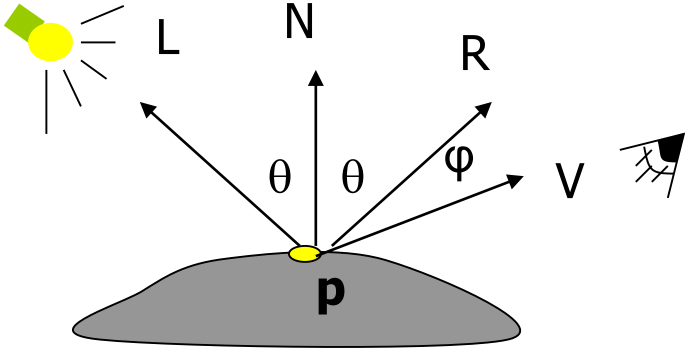


$(\cos(\varphi))^{shine}$

Specular Reflection

- k_s: Specular reflection coefficient
  - Specular light an object reflects
  - $0 \leq k_s \leq 1$

N	: surface normal at P
I_s	: light intensity
φ	: angle between **V** and **R**
n	: shininess factor

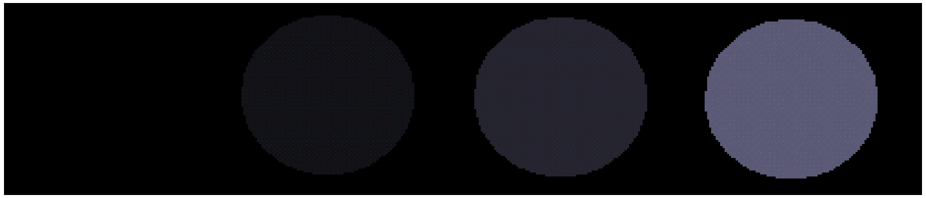


$$\text{Spec} = I_s * k_s * \cos^n(\varphi)$$
$$= I_s * k_s * (V \bullet R)^n$$

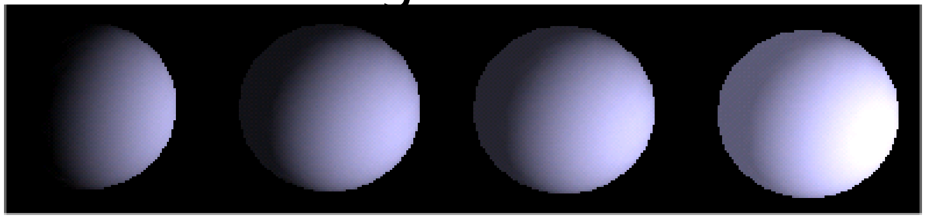
V and **R** must be unit vectors

Ambient/Diffuse/Specular

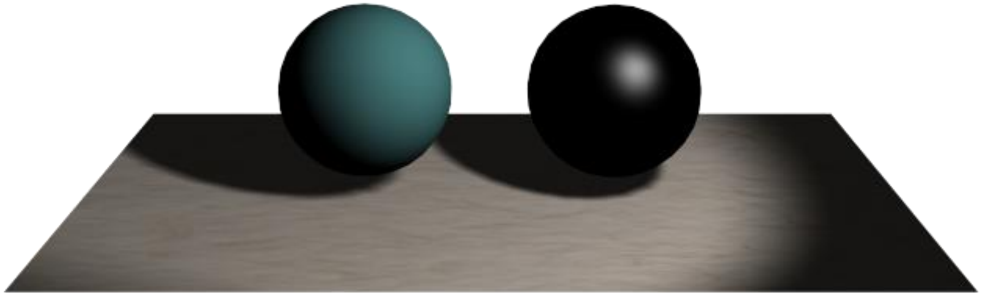
- Just ambient light:



- Diffuse and change Ambient

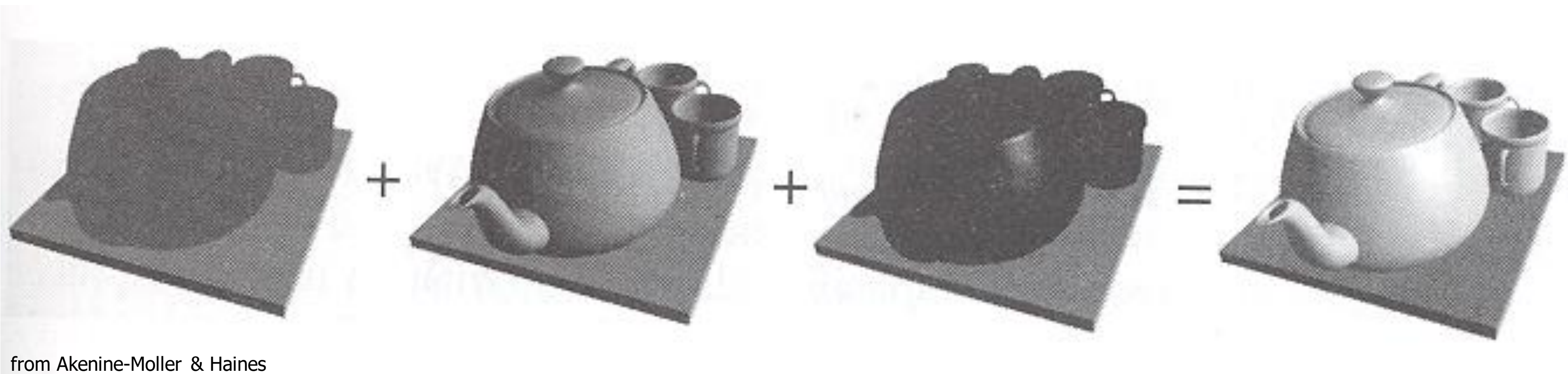


- Left: Sphere with just diffuse reflection
- Right:Sphere with just specular reflection



Basic Reflectance Equation

- Reflectance =



Ambient **Diffuse** **Specular** **Final**

$$= \mathbf{I_a} * \mathbf{k_a} + I_d * \mathbf{k_d} * (\mathbf{N} \bullet \mathbf{L}) + \mathbf{I_s} * \mathbf{k_s} * (\mathbf{R} \bullet \mathbf{V})^n$$

Put it all together

- Illumination from a single light source:
  - Illum = ambient + diffuse + specular

$$= Ka \times I$$
$$+ Kd \times I \times \max(0, N \cdot L)$$
$$+ Ks \times I \times \max(0, R \cdot V)^{n}$$

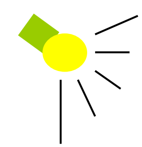
- Note that the K's and the I's are vectors (RGB).

Put it all together

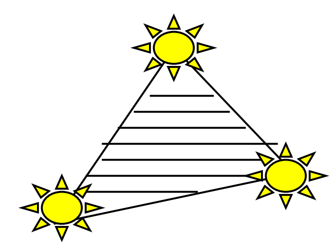
- If there are N lights
  - Total illumination for a point P = Σ (Illum)
- Some more terms to be added (in OpenGL):
  - Self emission
  - Global ambient
  - Light distance attenuation and spot light effect

Lighting in OpenGL

- Adopt Phong lighting model (specular) plus diffuse and ambient lights
  - Lighting is computed at vertices
    - Interpolate across surface (Gouraud/smooth shading)   OR
    - Use a constant illumination (get it from one of the vertices)



- Setting up OpenGL Lighting:
  - Light Properties
  - Enable/Disable lighting
  - Surface material properties
  - Provide correct surface normals
  - Light model properties



Light Properties

- Properties:
  - Colors / Position and type / attenuation

glLightfv(light, property, value)

① ② ③

(1) constant: specify which light you want to set the property
example: GL_LIGHT0, GL_LIGHT1, GL_LIGHT2 … you can create multiple lights (OpenGL allows at least 8 lights)

(2) constant: specify which light property you want to set the value
example: GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_POSITION
(check the red book for more)

(3) The value you want to set to the property

Property Example

- Define colors and position a light

```
GLfloat light_ambient[] = {0.0, 0.0, 0.0, 1.0};
GLfloat  light_diffuse[] = {1.0, 1.0, 1.0, 1.0};
GLfloat light_specular[] = {1.0, 1.0, 1.0, 1.0};
GLfloat light_position[] = {0.0, 0.0, 1.0, 1.0};

glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

colors (the first three lines)

Position (light_position)

What if I set the Position to (0,0,1,0)?

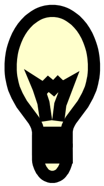
Types of lights

- OpenGL supports two types of lights
  - Local light (point light)
  - Infinite light (directional light)
- Determined by the light positions you provide
  - w = 0: infinite light source (faster)
  - w != 0: point light – position = (x/w, y/w, z/w)

```
GLfloat light_position[] = {x,y,z,w};
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

Turning on the lights

- Turn on the power (for all the lights)
  - glEnable(GL_LIGHTING);
  - glDisable(GL_LIGHTING);



- Flip each light's switch
  - glEnable(GL_LIGHT*n*) (n = 0,1,2,…)

Material Properties

- The color and surface properties of a material (dull, shiny, etc.)
- How much the surface reflects the incident lights (ambient/diffuse/specular reflection coefficients)

glMaterialfv(face, property, value)

Face: material property for which face (e.g. GL_FRONT, GL_BACK, GL_FRONT_AND_BACK)
Property: what material property you want to set (e.g. GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_SHININESS, GL_EMISSION, etc)
Value: the value you can to assign to the property

Material Example

- Define ambient/diffuse/specular reflection and shininess

```
GLfloat mat_amb_diff[] = {1.0, 0.5, 0.8, 1.0};
GLfloat mat_specular[] = {1.0, 1.0, 1.0, 1.0};      <- refl. coefficient
GLfloat shininess[] = {5.0};      <- (range: dull 0 – very shiny128)

glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE,
             mat_amb_diff);
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, shininess);
```

Global light properties

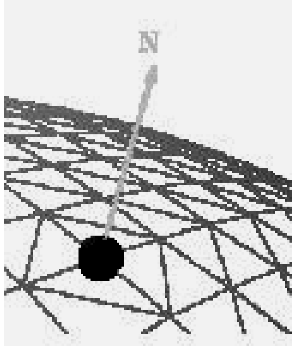
glLightModelfv(property, value)

- Enable two-sided lighting
  - property = GL_LIGHT_MODEL_TWO_SIDE
  - value = GL_TRUE (GL_FALSE if you don't want two sided lighting)
- Global ambient color
  - Property = GL_LIGHT_MODEL_AMBIENT
  - Value = (red, green, blue, 1.0);
- Check the red book for others

Surface Normals

- Correct normals are essential for correct lighting
- Associate a normal to each vertex

```
glBegin(…)
  glNormal3f(x,y,z)
  glVertex3f(x,y,z)
  …
glEnd()
```



- The normals you provide need to have a unit length
  - You can use glEnable(GL_NORMALIZE) to have OpenGL normalize all the normals.
  - Why not always have OpenGL do this?