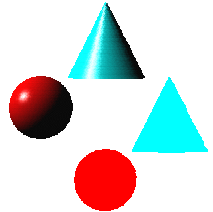


**Lighting Principles**

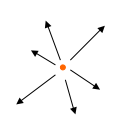
- Lighting simulates how objects reflect light
- Reflected light is based on a number of elements
  - Material composition of object
  - Light's color and position
  - Global lighting parameters
    - Ambient light
    - Two-sided lighting
- Things to consider:
  - Light sources
  - Surface effects
  - Lighting models



5/20/2007 1

**Light Sources**

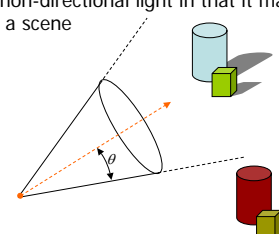
- Simplest form is a *point source*
- Energy is emitted from a single point
  - Typically in a single color, specified with RGB components
- Rays emitted in all directions
- For infinitely distant point sources, rays striking objects are essentially parallel
- Can treat non-point sources as point sources if they are sufficiently distant



5/20/2007 2

**Light Sources**

- Can constrain the output of a light source so that it only travels in one direction
- Called a *directional* or *spotlight* source
  - Constraint determined by the angle  $\theta$
- Differs from non-directional light in that it may not strike all objects in a scene



5/20/2007 3

**Types of Light**

- *Ambient*
  - Light from no direction (due to scattering)
  - Surfaces illuminated by ambient light reflect in all directions
- *Diffuse*
  - Light from a certain direction
  - Reflected equally in all directions from the surface (causing the surface to look equally bright)
- *Specular*
  - Directional light which reflects off a surface in a particular direction
  - Causes the surface to have a shiny highlight
- *Emissive*
  - Light originating within an object
  - Does not affect other objects in the scene
  - We won't cover this in any detail

5/20/2007 4

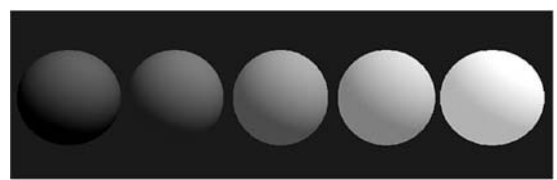
**Ambient Light**

- No apparent source – simply present (this is a real HACK for a complex phenomenon)
- Non-directional, uniform illumination
- Illumination equation:
 
$$A = I_a k_a$$
- $I_a$  is the ambient illumination
- $k_a$  is the *ambient reflection coefficient*
  - The amount of ambient light reflected by an object
  - Property of the object's material
  - Ranges between 0 and 1

5/20/2007 5

**Ambient Light**

- Here is the effect of varying the ambient reflection coefficient from 0.0 to 0.1, 0.3, 0.5, and 0.7:



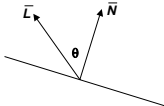
5/20/2007 6

**Diffuse Reflection**

- Reflection from dull, matte surfaces (e.g., chalk)
  - Also called *Lambertian* reflection
- Light comes from a point source
- Light is reflected with equal intensity in all directions
- For a given surface, brightness depends only on the angle between the direction to the light source and the surface normal

$$D = I_p k_d \cos \theta$$

- $k_d$  is the *diffuse reflection coefficient*



5/20/2007 7

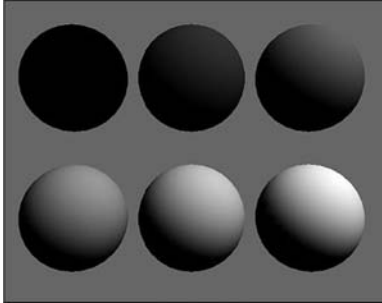
**Diffuse Reflection**

- The angle  $\theta$  must be between  $0^\circ$  and  $90^\circ$  for the light source to have any effect on the object
- This means that a light source behind an object doesn't illuminate it
- We say this object is *self-occluding*
- Technically, we should have a  $\max(\cos \theta, 0)$  term in the equation
- It's simpler to just assume the angle is "legal"

5/20/2007 8

**Diffuse Reflection**

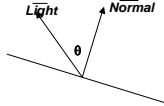
- The effect of varying the diffuse reflection coefficient from 0.0 to 0.2, 0.4, 0.6, 0.8, and 1.0:



5/20/2007 9

**Diffuse Reflection**

- If the vectors have been normalized, we can replace  $\cos \theta$  with their dot product:

$$D = I_p k_d (\text{Normal} \cdot \text{Light})$$


- If the point source is far enough away, it makes essentially the same angle with all surfaces sharing the same surface normal
- In this case, it's called a *directional light source*

5/20/2007 10

**Attenuation – Part I**

- Does distance from a light matter?
- Radiant energy from a light traveling through space is naturally attenuated at a rate of  $1/d^2$  where  $d$  is the distance from the light source
  - In practice, this doesn't look good in our pictures when we use a point source as it tends to produce too much variation of intensities!

5/20/2007 11

**Attenuation – Part II**

- Distance from a light source affects the amount of illumination seen on an object
- We use an *attenuation factor*,  $f_{att}$ , to account for this
  - Result is limited (clamped) to a maximum of 1

$$f_{att} = \min\left(\frac{1}{k_c + k_l d_L + k_q d_L^2}, 1\right)$$

- $d_L$  is the distance light travels from the source to the surface
- The clamping is for a source at infinity
- $k_c$ ,  $k_l$ , and  $k_q$  are *attenuation constants*
  - Constant, linear, and quadratic
  - $k_c$  is there to prevent division by zero

5/20/2007 12

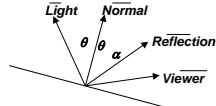
**Colored Lights and Surfaces**

- Commonly, we deal with colored lights and surfaces by writing separate equations for each component of the color model
- We represent an object's *diffuse color* with  $O_d$ 
  - One term for each component:  $(O_{dR}, O_{dG}, O_{dB})$
- The light's components  $I_{pR}$ ,  $I_{pG}$ , and  $I_{pB}$  are reflected in proportion to  $k_p O_{dR}$ ,  $k_p O_{dG}$ , and  $k_p O_{dB}$
- Thus, for the red component,
 
$$I_R = I_{dR} k_a O_{dR} + f_{att} I_{pR} k_d O_{dR} (N \cdot L)$$
- To account for other color models, we replace the color indicator with the symbol  $\lambda$ :
 
$$I_\lambda = I_{d\lambda} k_a O_{d\lambda} + f_{att} I_{p\lambda} k_d O_{d\lambda} (N \cdot L)$$

5/20/2007 13

**Specular Reflection**

- Observable on any shiny surface
  - "Highlights" are caused by specular reflection
- On a perfectly reflective surface, light is reflected only in one direction
  - Angle of reflection = angle of incidence
- The angle between the reflection and the viewpoint,  $\alpha$ , determines the amount of reflection seen
  - Also affected by the *specular reflection exponent* of the material
  - For a perfect mirror, can only see reflection when the angle between the Reflection and Viewer ( $\alpha$ ) is zero



5/20/2007 14

**Specular Reflection**

- Maximum reflection occurs when  $\alpha$  is zero
- Falloff is approximated by  $\cos^n \alpha$ 
  - $n$  is the specular reflection exponent
  - Low values of  $n$  give gentle falloff; high values give sharp, focused highlights




Figure 9.3: specular highlights with specular coefficients 20, 50, and 80 (left, center, and right), respectively

5/20/2007 15

**Final Light**

- Final light is the sum of these contributing types:
 
$$I_f = \sum Ambient + \sum Diffuse + \sum Specular$$
- Compute separately for R, G, and B
- Values are clamped to 1
- We calculate light at vertices of polygons
- Need to know *surface normals* to finish computation

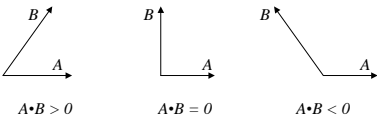
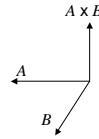
5/20/2007 16

**Math Reminder (Again)**

- For a vector  $w$ :
  - Magnitude:  $|w| = \sqrt{w_1^2 + w_2^2 + \dots + w_n^2}$
  - Normalized unit vector:  $\hat{w} = \frac{w}{|w|}$   $|\hat{w}| = 1$
  - Dot product of vectors  $A$  and  $B$ :
 
$$A \cdot B = a_x b_x + a_y b_y + a_z b_z = |A| |B| \cos \theta$$
  - Cross product of vectors  $A$  and  $B$ :
 
$$A \times B = \begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ A_x & A_y & A_z \\ B_x & B_y & B_z \end{vmatrix} \quad |A \times B| = |A| |B| \sin \theta$$

5/20/2007 17

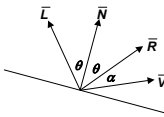
**Math Reminder**

- The sign of the dot product:
 
- The sign of the cross product uses the right hand rule
  - Fingers align with  $A$ , palm faces direction of  $B$
  - Positive non-zero values point in direction of thumb.

5/20/2007 18

**Phong Illumination Model**

- Developed in 1975 by Phong Bui-Tuong
- Illumination model for non-perfect reflectors
- Phong model was the first to account for viewers and lights at arbitrary positions
- If  $W(\theta)$  is the fraction of specularly reflected light, Phong's model is

$$I_\lambda = I_a k_a O_{d\lambda} + f_{att} I_{p\lambda} [k_d O_{d\lambda} \cos \theta + W(\theta) \cos^n \alpha]$$


5/20/2007 19

**Phong Illumination Model**

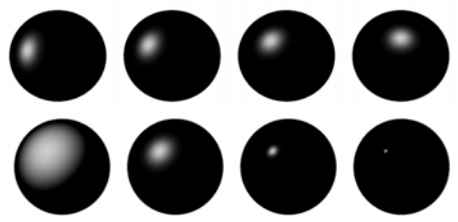
- If we assume that all vectors are normalized,  $\cos \alpha$  is the dot-product  $R \cdot V$
- $W(\theta)$  is typically set to  $k_s$ , the *specular reflection coefficient*
- Resulting model:

$$I_\lambda = I_a k_a O_{d\lambda} + f_{att} I_{p\lambda} [k_d O_{d\lambda} (N \cdot L) + k_s (R \cdot V)^n]$$

5/20/2007 20

**Phong Lighting**

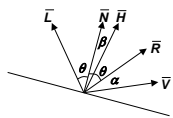
Lighting with different Values for L and n



5/20/2007 21

**Halfway Vector**

- An alternative to Phong's model
- $H$  is the halfway vector
  - Also known as the direction of maximum highlights
- This term can be expressed as  $(N \cdot H)$ , where

$$\vec{H} = \frac{(\vec{L} + \vec{V})}{|\vec{L} + \vec{V}|}$$


- When the viewer and the light source are both at infinity, this offers computational simplicity

5/20/2007 22

**How OpenGL Simulates Lights**

- Phong lighting model
  - Computed at vertices
- Lighting contributors
  - Surface material properties
  - Light properties
  - Lighting model properties

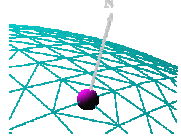
5/20/2007 23

**Surface Normals**

- Normals define how a surface reflects light
- OpenGL maintains a *current normal*, which is used to compute color at vertices
- Setting the current normal:
 

```
glNormal3f(x, y, z);
glNormal3fv(v);
```
- Use unit normals for proper lighting
  - Scaling affects a normal's length
 

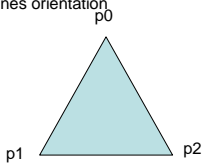
```
glEnable(GL_NORMALIZE)
or
glEnable(GL_RESCALE_NORMAL)
```



5/20/2007 24

**Determining the Normal**


- Given a plane with points p0, p1, p2
- Points may not be collinear
- Recall: U x V orthogonal to U and V
- $n0 = (p1 - p0) \times (p2 - p0)$
- Remember that U x V is NOT equal to V x U!
  - Order of cross produce determines orientation
- Normalize to  $n = n0/|n0|$



5/20/2007 25

**Normal of a 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:
  - Take partial derivative with respect to x, y, and z and what do you get?
- Normalize  $n0/|n0| = 2p/2 = p$



5/20/2007 26

**To Get the Normal of a Shared Vertex**

- Average the normal vectors of all polygons in the surface mesh that share that vertex
  - $n0 = \text{sum}(n_k) / |\text{sum}(n_k)|$
- For shared vertices: 
$$N = \frac{\sum a_i N_i}{\sum a_i}$$

where  $a_i$  is the inverse *cos* of the dot product of two edges that meet at the vertex

5/20/2007 27

**Lights in OpenGL**

- Two possible light sources:
  - Local (point) light sources
  - Infinite (directional) light sources
- Type of light controlled by w coordinate in (x, y, z, w):
  - w = 0: infinite light, directed along (x, y, z)
  - w ≠ 0: local light, positioned at (x/w, y/w, z/w)
- A light positioned at the origin of the world:
 

```
float positionalLight[] = { 0.0, 0.0, 0.0, 1.0 };
```
- A directional light moving towards the +z axis:
 

```
float directionalLight[] = { 0.0, 0.0, 1.0, 0.0 };
```

5/20/2007 28

**Lighting Variables**

- Lights values are specified in terms of R, G, B and A components
  - Floating point values
  - Range is 0.0 (no intensity) to 1.0 (full intensity)
- Global light reflecting off everything will be blue-ish:
 

```
float ambientLight[] = { 0.3, 0.5, 0.8, 1.0 };
```
- The surface that diffuse light hits will be lighter than areas which it barely hits (which will be dark)
 

```
float diffuseLight[] = { 0.25, 0.25, 0.25, 1.0 };
```

5/20/2007 29

**Lights in OpenGL**

```
glEnable( GL_LIGHTn )
```

- If enabled, include light n in the evaluation of the lighting equation
- Equivalent to flipping the light's switch
- At least eight in implementation (GL\_LIGHT0 through GL\_LIGHT7)
- Up to eight can be used in a scene
- `glGetIntegerv( GL_MAX_LIGHTS, &n );`

```
glEnable( GL_LIGHTING )
```

- If enabled, compute the vertex color or index using the current lighting parameters
- Otherwise, simply associate the current color or index with each vertex
- Equivalent to turning on the power to all enabled lights

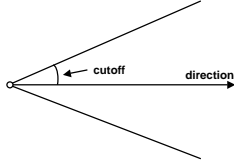
```
glEnable( GL_NORMALIZE )
```

- If enabled, normal vectors specified with `glNormal` are scaled to unit length after transformation
- Before any geometry is specified, will automatically normalize vectors

5/20/2007 30

**Light Properties in OpenGL**

- Position or direction
- Color
- How it is attenuated (diminished) over distance
- Omni-directional (default) or spotlight
  - Direction (3D vector)
  - Cutoff (0 to 90)
  - Dropoff exponent



5/20/2007 31

**Setting Light Properties**

```
glLight[fi]( light, property, value );
glLight[fi]v( light, property, *value );
```

- **light** specifies which light
  - GL\_LIGHT0, etc.
- **property** is a light source parameter
  - Color
  - Position and type
  - Attenuation
- **value** is the desired setting
  - May be a single value (e.g., attenuation factor) or a vector (e.g., intensities for RGBA)

5/20/2007 32

**Parameter Settings for Lights**

- Some parameter names and defaults:

Parameter	Description	Default(s)
GL_AMBIENT	ambient intensity, RGBA	(0,0,0,1)
GL_DIFFUSE	diffuse intensity, RGBA	(1,1,1,1)
GL_SPECULAR	specular intensity, RGBA	(1,1,1,1)
GL_POSITION	light position, world coordinates	(0,0,1,0)
GL_SPOT_DIRECTION	eye coordinates	(0,0,-1)
GL_SPOT_EXPONENT	intensity distribution [0,128]	0
GL_SPOT_CUTOFF	maximum spread angle [0,90]	180
GL_CONSTANT_ATTENUATION		1
GL_LINEAR_ATTENUATION		0
GL_QUADRATIC_ATTENUATION		0

5/20/2007 33

**Controlling a Light's Position**

- Modelview matrix affects a light's position
- Different effects based on *when* position is specified
  - Eye coordinates
  - World coordinates
  - Model coordinates
- Push and pop matrices to uniquely control a light's position

5/20/2007 34

**Lighting Models**

- OpenGL's lighting model lets you define 4 components
  - The ambient light intensity of the scene
  - The location of the viewport (local or infinite)
    - Affects specular reflection angle calculation
  - One-sided or two-sided lighting
  - Whether specular color is separate from ambient and diffuse

5/20/2007 35

**Light Model Properties**

```
glLightModel[fi]( property, value );
glLightModel[fi]v( property, *value );
```

- **GL\_LIGHT\_MODEL\_TWO\_SIDE**
  - Specifies whether one- (0, front only, default) or two-sided lighting (non-zero) calculations are done for polygons
- **GL\_LIGHT\_MODEL\_AMBIENT**
  - Ambient RGBA intensity of the entire scene
- **GL\_LIGHT\_MODEL\_LOCAL\_VIEWER**
  - How specular reflection angles are computed
  - 0 (default): view direction parallel to & in the direction of the -z axis
  - Other: from the origin of the eye coordinate system
- **GL\_LIGHT\_MODEL\_COLOR\_CONTROL**
  - Specular color calculated separately from ambient and diffuse

5/20/2007 36

**Lighting Models**

- To specify the amount of global ambient light:
 

```
// medium light
float ambientLightModel[] = { 0.5, 0.5, 0.5, 1.0 };
glLightModelfv( GL_LIGHT_MODEL_AMBIENT,
                ambientLightModel );
```
- To specify the location of the viewpoint:
 

```
// local viewport
glLightModeli( GL_LIGHT_LOCAL_VIEWER, GL_TRUE );
```

5/20/2007 37

**Putting it All Together**

```
glEnable(GL_LIGHTING);

// set up materials
glLightfv( GL_LIGHT0, GL_AMBIENT, ambientLight );
glLightfv( GL_LIGHT0, GL_DIFFUSE, diffuseLight );
glLightfv( GL_LIGHT0, GL_POSITION, lightPosition );

glEnable(GL_LIGHT0);
// draw scene (specify normals in step using
// glNormal3f() and its corresponding vertices)
```

- By default there is no ambient light
- Default RGBA for **GL\_DIFFUSE** for **GL\_LIGHT0**:
 

```
( 1.0, 1.0, 1.0, 1.0 )
```
- Default RGBA for **GL\_DIFFUSE** for all other lights:
 

```
( 0.0, 0.0, 0.0, 0.0 )
```

5/20/2007 38

**Specular Highlight**

- The bright reflection seen on an object under a specially-directed light
- GL\_SPECULAR** determines the color of this highlight
  - Often this is specified with the same values as **GL\_DIFFUSE**
- Defaults for **GL\_SPECULAR**
  - GL\_LIGHT0**: 

```
( 1.0, 1.0, 1.0, 1.0 )
```
  - Others: 

```
( 0.0, 0.0, 0.0, 0.0 )
```

```
float specularLight[] = { 1.0, 1.0, 1.0, 1.0 };
float lightPosition[] = { 0.0, 0.0, 0.0, 0.0 };

glLightfv( GL_LIGHT0, GL_SPECULAR, specularLight );
glLightfv( GL_LIGHT0, GL_POSITION, lightPosition );
// define specular material properties
```

5/20/2007 39

**Attenuation**

- Intensity of light decreases as you get further away from the origin of the light
  - e.g., a street lamp at night in the fog
- Only affects positional light sources
- OpenGL supports three attenuation factors:
 

Parameter	Default
<b>GL_CONSTANT_ATTENUATION</b>	1.0
<b>GL_LINEAR_ATTENUATION</b>	0.0
<b>GL_QUADRATIC_ATTENUATION</b>	0.0

5/20/2007 40

**Attenuation**

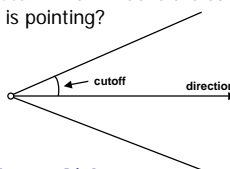
- Specified with **glLightf()** function (*not* **glLightfv()**):
 

```
glLightf( GL_LIGHT0, GL_CONSTANT_ATTENUATION, 4.0 );
```
- Does not affect emission and global ambient light values
- Affects all other ambient, diffuse and specular light properties
- Graphics engine may run slower due to distance calculations

5/20/2007 41


**Spotlights**

- Reduce the radiance of the positional light from all directions to a specific direction
- Must specify 3 additional parameters to a normal positional light:
  - Spotlight cutoff** – how wide is the cone of light in the direction it is pointing?



```
// a 30 degree light cone
glLightf( GL_LIGHT0, GL_SPOT_CUTOFF, 15.0 )
```

5/20/2007 42



**Spotlights**


---

- **Spotlight direction** – direction the spotlight is facing, as a vector (x, y, z)
  - The default is (0.0, 0.0, -1.0), down the -z axis

```
// point light down the -y axis
float spotLightDirection[] = { 0.0, -1.0, 0.0 };
glLightfv( GL_LIGHT0, GL_SPOT_DIRECTION,
           spotLightDirection );
```
- **Spotlight focus** – concentration of the spotlight in the center of the light cone
  - As you move away from the center of the cone, the light is attenuated at a certain ratio
  - **GL\_SPOT\_EXPONENT** sets the amount of concentration
    - Larger values → more focused light source

```
glLightfv( GL_LIGHT0, GL_SPOT_EXPONENT, 10.0 );
```

5/20/2007
43




**Controlling a Light's Position**

---

- The light's position specified by `glLight*()` is manipulated by the current Modelview matrix
- Static lights are positioned before any transformation occurs
  - In OpenGL, this would be after the camera is specified
- Can apply a series of transformations on the light source position (vertices) to change the lights dynamically

5/20/2007
44



**Tips for Better Lighting**

---

- Recall that lighting/illumination is computed only at vertices due to Gouraud shading model
  - Model tessellation heavily affects lighting results
  - Better results but more geometry to process
- Use a single infinite light for fastest lighting
  - Minimal computation per vertex

5/20/2007
45