Xi'an Jiaotong-Liverpool University

西交利物浦大学

CPT205 Computer Graphics

Lighting and Materials

Lecture 09

2022-23

Yong Yue

Topics for today

- Understanding how real-world lighting works
- Lighting sources
- Lighting effects: ambient, diffuse and specular
- Lighting model and shading
- Rendering illuminated objects by defining the desired light sources
- Material properties of objects being illuminated
- OpenGL functions

Background

- Realistic displays of a scene are obtained by generating perspective projections of objects and applying natural lighting effects to the visible surfaces.
- Photorealism in computer graphics also involves accurate representation of surface properties and good physical description of the lighting effects in the scene (such as light reflection, transparency, surface texture and shadows).

Lighting sources

- A light source can be defined with a number of properties, such as position, colour, direction, shape and reflectivity.
- A single value can be assigned to each of the RGB colour components to describe the *amount* or *intensity* of the colour component.

Lighting sources: Point light

- This is the simplest model for an object, which emits radiant energy with a single colour specified by the three RGB components.
- Large sources can be treated as point emitters if they are not too close to the scene.
- The light rays are generated along radially diverging paths from the light source position.

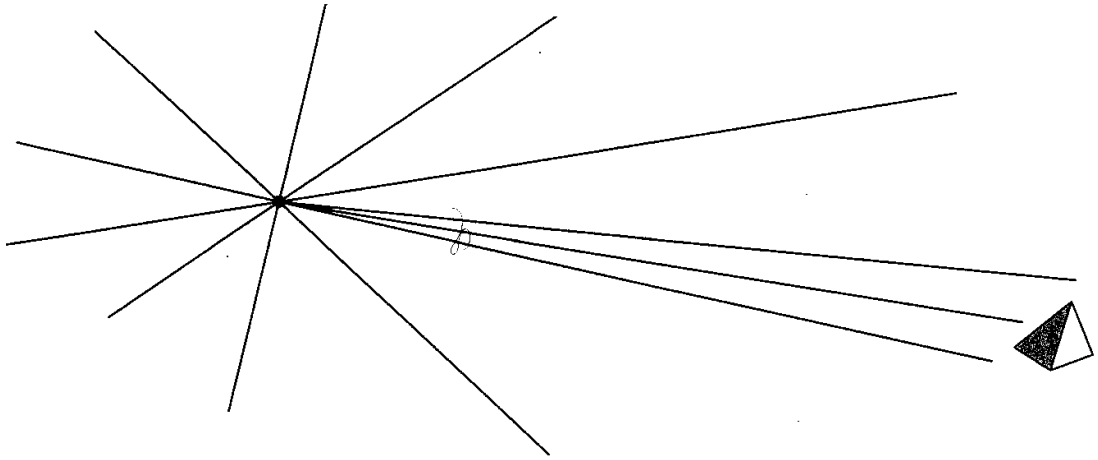
Lighting sources: Directional light

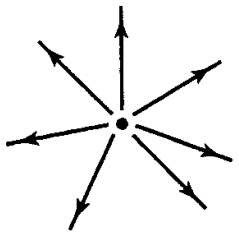
- A large light source (e.g. the sun), which is very far away from the scene, can also be approximated as a point emitter, but there is little variation in its directional effects.
- This is called an infinitely distant / directional light (the light beams are parallel).

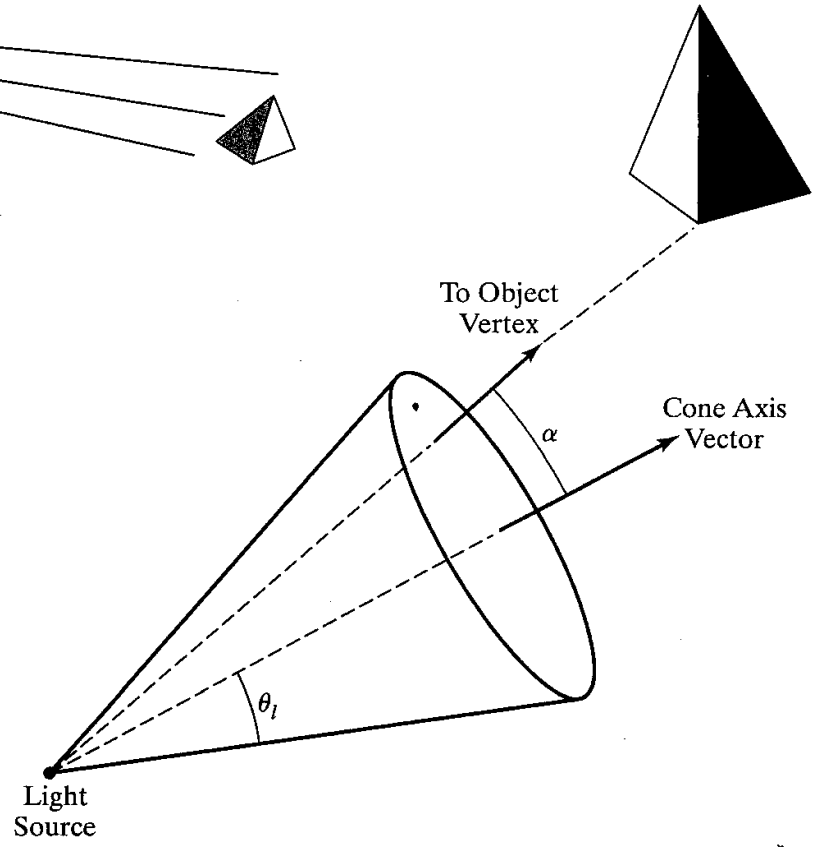
Lighting sources: Spot lights

- A local light source can easily be modified to produce a spotlight beam of light.
- If an object is outside the directional limits of the light source, it is excluded for illumination by that light source.
- A spot light source can be set up by assigning a vector direction and an angular limit θ_l measured from the vector direction along with the position and colour.

Lighting sources: Illustrated







Surface lighting effects

- Although a light source delivers a single distribution of frequencies, the ambient, diffuse and specular components might be different.
- For example, if you have a white light in a room with red walls, the scattered light tends to be red, although the light directly striking objects is white.

Surface lighting effects: Ambient light (1)

- The effect is a general background non-directed illumination. Each object is displayed using an intensity intrinsic to it, i.e. a world of non-reflective, self-luminous object.
- Consequently each object appears as a monochromatic silhouette, unless its constituent parts are given different shades when the object is created.

Surface lighting effects: Ambient light (2)

- The ambient component is the lighting effect produced by the reflected light from various surfaces in the scene.
- When ambient light strikes a surface, it is scattered equally in all directions.
- The light has been scattered so much by the environment that it is impossible to determine its direction - it seems to come from all directions.
- Back lighting in a room has a large ambient component, since most of the light that reaches the eye has been bounced off many surfaces first.
- A spotlight outdoors has a tiny ambient component; most of the light travels in the same direction, and since it is outdoors, very little of the light reaches the eye after bouncing off other objects.

Surface lighting effects: Diffuse reflectance (1)

- Diffuse light comes from one direction, so it is brighter if it comes squarely down on a surface than if it barely glances off the surface.
- Once it hits a surface, however, it is scattered equally in all directions, so it appears equally bright, no matter where the eye is located.
- Any light coming from a particular position or direction probably has a diffuse component.
- Rough or grainy surfaces tend to scatter the reflected light in all directions (called diffuse reflection).

Surface lighting effects: Diffuse reflectance (2)

- Each object is considered to present a dull, matte surface and all objects are illuminated by a point light source whose rays emanate uniformly in all directions.
- The factors which affect the illumination are the point light source intensity, the material's diffuse reflection coefficient, and the angle of incidence of the light.

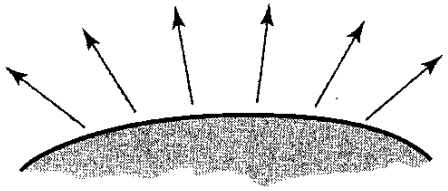
Surface lighting effects: Specular reflectance (1)

- Specular light comes from a particular direction, and it tends to bounce off the surface in a preferred direction.
- A well-collimated laser beam bouncing off a high-quality mirror produces almost 100% specular reflection.
- Shiny metal or plastic has a high specular component, and chalk or carpet has almost none. Specularity can be considered as shininess.

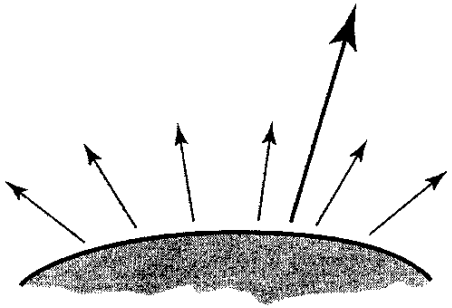
Surface lighting effects: Specular reflectance (2)

- This effect can be seen on any shiny surface and the light reflected off the surface tends to have the same colour as the light source (a white spot from a white light reflected off a red apple).
- The reflectance tends to fall off rapidly as the viewpoint direction veers away from the direction of reflection (which is equal to the direction to the light source mirrored about the surface normal); for perfect reflectors (i.e. perfect mirrors), specular light appears only in the direction of reflection.
- The factors that affect the amount of light reflected are the material's specular-reflection coefficient, the angle of viewing, relative to the direction of reflection, and the light source intensity.

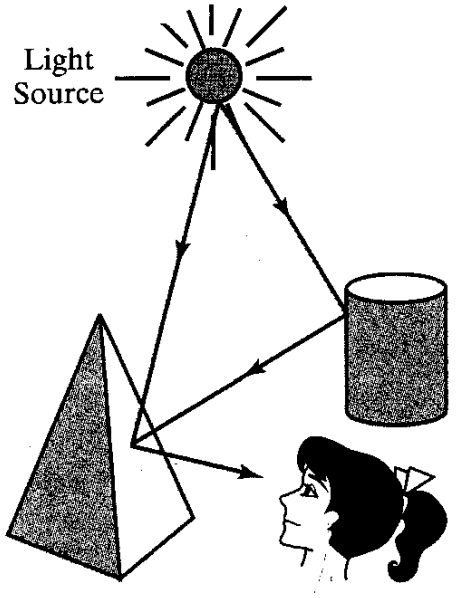
Surface lighting effects: Illustrated



Diffuse reflection
from a surface



Specular reflection
superimposed on
diffuse reflection vectors



Combined illumination
from light sources
and reflection from
other surface

Attenuation

- For positional/point light sources, we consider the attenuation of light received due to the distance from the source.
- Attenuation is disabled for directional lights as they are infinitely far away, and it does not make sense to attenuate their intensity over distance.
- Although for an ideal source the attenuation is inversely proportional to the square of the distance d, we can gain more flexibility by using the following distance-attenuation model,

$$f(d) = \frac{1}{k_c + k_l d + k_q d^2}$$

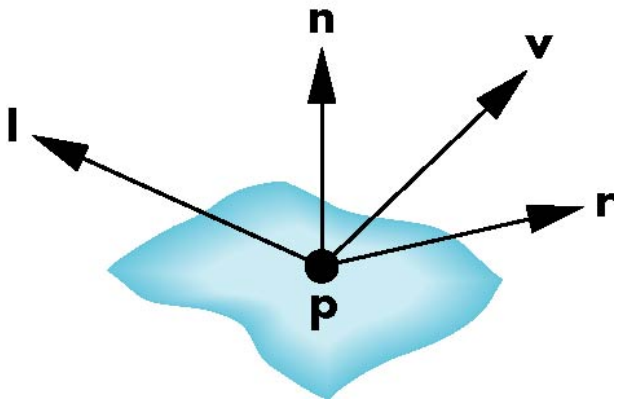
which contains constant, linear and quadratic terms. Three floats for these values, k_c, k_l and k_q can be used for these terms.

Lighting model

- A *lighting model* (also called *illumination* or *shading* model) is used to calculate the colour of an illuminated position on the surface of an object.
- The lighting model computes the lighting effects for a surface using various optical properties, which have been assigned to the surface, such as the degree of transparency, colour reflectance coefficients and texture parameters.
- The lighting model can be applied to every projection position, or the surface rendering can be accomplished by interpolating colours on the surface using a small set of lighting-model calculations.
- A *surface rendering method* uses the colour calculation from a lighting model to determine the pixel colours for all projected positions in a scene.

Phong model

- A simple model that can be computed rapidly
- Three components considered
  - Diffuse
  - Specular
  - Ambient
- Four vectors used
  - To light source **l**
  - To viewer **v**
  - Face normal **n**
  - Perfect reflector **r**



Phong model

- For <u>each</u> point light source, there are 9 coefficients
  - $I_{dr}, I_{dg}, I_{db}, I_{sr}, I_{sg}, I_{sb}, I_{ar}, I_{ag}, I_{ab}$
- Material properties match light source properties
  - 9 absorption coefficients

    $k_{dr}, k_{dg}, k_{db}, k_{sr}, k_{sg}, k_{sb}, k_{ar}, k_{ag}, k_{ab}$
  - Shininess coefficient α
- For <u>each</u> light source and <u>each</u> colour component, the Phong model can be written (without the distance terms) as
  - $I = k_d I_d \, \mathbf{l} \cdot \mathbf{n} + k_s I_s (\mathbf{v} \cdot \mathbf{r})^{\alpha} + k_a I_a$
- For <u>each</u> colour component, we add contributions from <u>all</u> sources

Polygonal Shading

- In vertex shading, shading calculations are done for each vertex
  - Vertex colours become vertex shades and can be sent to the vertex shader as a vertex attribute
  - Alternately, we can send the parameters to the vertex shader and have it compute the shade
- By default, vertex shades are interpolated across an object if passed to the fragment shader as a varying variable (smooth shading)
- We can also use uniform variables to shade with a single shade (flat shading)

Flat (constant) shading

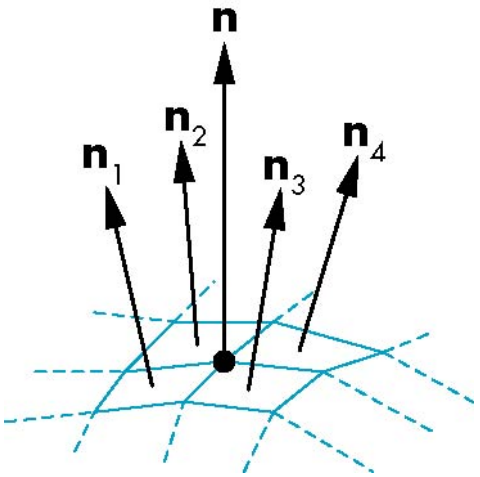
- If the three vectors ($\mathbf{l}$, $\mathbf{v}$, $\mathbf{n}$) are constant, then the shading calculation needs to be carried out only once for each polygon, and each point on the polygon is assigned the same shade.
- The polygonal mesh is usually designed to model a smooth surface, and flat shading will almost always be disappointing because we can see even small differences in shading between adjacent polygons.

Smooth (interpolative) shading

- The rasteriser interpolates colours assigned to vertices across a polygon.
- Suppose that the lighting calculation is made at each vertex using the material properties and vectors $\mathbf{n}$, $\mathbf{v}$, and $\mathbf{l}$ computed for each vertex. Thus, each vertex will have its own colour that the rasteriser can use to interpolate a shade for each fragment.
- Note that if the light source is distant, and either the viewer is distant or there are no specular reflections, then smooth (or interpolative) shading shades a polygon in a constant colour.

Gouraud shading

- The vertex normals are determined (average of the normals around a mesh vertex: $\mathbf{n} = (\mathbf{n}_1+\mathbf{n}_2+\mathbf{n}_3+\mathbf{n}_4)/|\mathbf{n}_1+\mathbf{n}_2+\mathbf{n}_3+\mathbf{n}_4|$) for a polygon and used to calculate the pixel intensities at each vertex, using whatever lighting model.



- Then the intensities of all points on the edges of the polygon are calculated by a process of weighted averaging (linear interpolation) of the vertex values.

- The vertex intensities are linearly interpolated over the surface of the polygon.

Material colours

- A lighting model may approximate a material colour depending on the percentages of the incoming red, green, and blue light reflected.
- For example, a perfectly red ball reflects all the incoming red light and absorbs all the green and blue light striking it.
- If the ball is lit in white light (composed of equal amounts of red, green and blue light), all the red is reflected, and a red ball is seen. If the ball is viewed in pure red light, it also appears to be red.
- If, however, the ball is viewed in pure green light, it appears black (all the green light is absorbed, and there is no incoming red, so no light is reflected).

Material colours: Ambient, diffuse and specular

- Like lights, materials have different ambient, diffuse and specular colours, which determine the ambient, diffuse and specular reflectances of the material.
- The ambient reflectance of a material is combined with the ambient component of each incoming light source, the diffuse reflectance with the light's diffuse component, and similarly for the specular reflectance component.
- Ambient and diffuse reflectances define the material colour and are typically similar if not identical.
- Specular reflectance is usually white or grey, so that specular highlights become the colour of the specular intensity of the light source. If a white light shines on a shiny red plastic sphere, most of the sphere appears red, but the shiny highlight is white.

RGB values for lights

- The colour components specified for lights mean something different from those for materials.
- For a light, the numbers correspond to a percentage of full intensity for each colour. If the R, G and B values for a light colour are all 1.0, the light is the brightest white.
- If the values are 0.5, the colour is still white, but only at half intensity, so it appears grey. If R=G=1 and B=0 (full red and green with no blue), the light appears yellow.

RGB values for lights and materials

- For materials, the numbers correspond to the reflected proportions of those colours. So if R=1, G=0.5 and B=0 for a material, the material reflects all the incoming red light, half the incoming green and none of the incoming blue light.
- In other words, if a light has components (R_L, G_L, B_L), and a material has corresponding components (R_M, G_M, B_M); then, ignoring all other reflectivity effects, the light that arrives at the eye is given by (R_L*R_M, G_L*G_M, B_L*B_M).

RGB values for two lights

- Similarly, if two lights, (R_1, G_1, B_1) and (R_2, G_2, B_2) are sent to the eye, these components are added up, giving (R_1+R_2, G_1+G_2, B_1+B_2).
- If any of the sums are greater than 1 (corresponding to a colour brighter than the equipment can display), the component is clamped to 1.

OpenGL: Light sources and colour (1)

- Light sources have a number of properties, such as colour, position and direction.
- The command used to specify all properties of lights is **`glLight*()`**.
- It takes three arguments: the identity of the light, the property and the desired value for that property.

OpenGL: Light sources and colour (2)

- **`void glLight{if}[v](GLenum` *`light`*`, GLenum` *`pname`*`, TYPEparam`**`)`** creates the light specified by *light*, which can be GL_LIGHT0, GL_LIGHT1, ... , GL_LIGHT7.
- The characteristic of the light being set is defined by *pname*, which specifies a named parameter (see table on next slide).
- The *TYPEparam* argument indicates the values to which the *pname* characteristic is set; it is a pointer to a group of values if the vector version is used, or the value itself if the non-vector version is used. The non-vector version can be used to set only single-valued light characteristics.

OpenGL: Light sources and colour (3)

Parameter Name	Default Value	Meaning
GL_AMBIENT	(0.0, 0.0, 0.0, 1.0)	ambient RGBA intensity of light
GL_DIFFUSE	(1.0, 1.0, 1.0, 1.0)	diffuse RGBA intensity of light
GL_SPECULAR	(1.0, 1.0, 1.0, 1.0)	specular RGBA intensity of light
GL_POSITION	(0.0, 0.0, 1.0, 0.0)	(*x, y, z, w*) position of light
GL_SPOT_DIRECTION	(0.0, 0.0, -1.0)	(*x, y, z*) direction of spotlight
GL_SPOT_EXPONENT	0.0	spotlight exponent
GL_SPOT_CUTOFF	180.0	spotlight cut-off angle
GL_CONSTANT_ATTENUATION	1.0	constant attenuation factor
GL_LINEAR_ATTENUATION	0.0	linear attenuation factor
GL_QUADRATIC_ATTENUATION	0.0	quadratic attenuation factor

OpenGL: Light sources and colour (4)

- The default values listed for GL_DIFFUSE and GL_SPECULAR in the table apply only to GL_LIGHT0. For other lights, the default value is (0.0, 0.0, 0.0, 1.0) for both GL_DIFFUSE and GL_SPECULAR. Below is an example of using **`glLight*()`**:

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

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);
```

- If the last value of LIGHT_POSITION, w, is 0.0, the corresponding light source is directional, and the (x, y, z) values describe its direction.

OpenGL: Light sources and colour (5)

- Arrays are defined for the parameter values, and **`glLightfv()`** is called repeatedly to set various parameters. In this example, the first three calls to **`glLightfv()`** are superfluous, since they are used to specify the default values for the GL_AMBIENT, GL_DIFFUSE, and GL_SPECULAR parameters.
- Remember to turn on each light with **`glEnable()`**;
- The parameters for **`glLight*()`** and their possible values are further explained shortly. These parameters interact with those that define the overall lighting model for a particular scene and the object's material properties.

OpenGL: Light sources and colour (6)

- OpenGL allows you to associate three different colour-related parameters (GL_AMBIENT, GL_DIFFUSE and GL_SPECULAR) with any particular light.
- The GL_AMBIENT parameter refers to the RGBA intensity of the ambient light that a particular light source adds to the scene.
- The GL_DIFFUSE parameter probably most closely correlates with what you naturally think of as “the colour of a light”. It defines the RGBA colour of the diffuse light that a particular light source adds to a scene.

OpenGL: Light sources and colour (7)

- The GL_SPECULAR parameter affects the colour of the specular highlight on an object.
- Typically, a real-world object such as a glass bottle has a specular highlight that is the colour of the light shining on it (often white).
- Therefore, if you want to create a realistic effect, set the GL_SPECULAR parameter to the same value as the GL_DIFFUSE parameter.

OpenGL: Light position and direction (1)

- A light source can be treated as though it is located infinitely far away from the scene or close to the scene.
- The first type is referred to as a *directional* light source; the effect of an infinite location is that the rays of light can be considered parallel.
- The second type is called a *positional* light source, since its exact position within the scene determines its effect on the scene and, specifically, the direction from which the light rays come. A desk lamp is an example of a positional light source. The light specified below is directional (where $w = 0$).

```
GLfloat light_position[] = {1.0, 1.0, 1.0, 0.0};
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

OpenGL: Light position and direction (2)

- As shown, a vector of four values (*x, y, z, w*) is specified for GL_POSITION. If the last value, *w*, is 0.0, the corresponding light source is a directional one, and the (*x, y, z*) values describe its direction. This direction is transformed by the model-view matrix. By default, GL_POSITION is (0, 0, 1, 0), which defines a directional light that points along the negative *z*-axis.
- If the *w* value is nonzero, the light is positional, and the (*x, y, z*) values specify the location of the light in homogeneous object co-ordinates.
- This location is transformed by the model-view matrix and stored in eye co-ordinates.

OpenGL: Light position and direction (3)

- OpenGL treats the position and direction of a light source just as it treats the position of a geometric primitive. In other words, a light source is subject to the same matrix transformations as a primitive.
- More specifically, when **`glLight*()`** is called to specify the position or the direction of a light source, the position or direction is transformed by the current model-view matrix. This means you can manipulate the position or direction of a light source by changing the contents of the model-view matrix stack.
- Note that the projection matrix has no effect on light position or direction.

OpenGL: Light position and direction (4)

The following three effects can be implemented by changing the point in the program at which the light position is set, relative to modelling or viewing transformations:

- A light position that remains fixed
- A light that moves around a stationary object
- A light that moves along with the viewpoint

OpenGL: Attenuation

- By default, k_c is 1.0 and both k_l and k_q are zero, but you can give these parameters different values

```
glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 2.0);
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 1.0);
glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 0.5);
```

- Note that the ambient, diffuse and specular contributions are all attenuated. Only the emission and global ambient values are not attenuated.

OpenGL: Spotlight (1)

- Note that no light is emitted beyond the edges of the cone.
- By default, the spotlight feature is disabled because the GL_SPOT_CUTOFF parameter is 180.0. This value means that light is emitted in all directions (the angle at the cone's apex is 360 degrees, so it is not a cone at all).
- The value for GL_SPOT_CUTOFF is restricted to being within the range [0.0,90.0] (unless it has the special value 180.0).
- The following line sets the cut-off parameter to 45 degrees:

```
glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);
```

OpenGL: Spotlight (2)

- You also need to specify a spotlight direction, which determines the axis of the cone:

```
GLfloat spot_direction[] = {-1.0, -1.0, 0.0};

glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spot_direction);
```

- The direction is specified in homogeneous object co-ordinates, the default direction is (0.0, 0.0, -1.0), i.e. the light points down the negative *z*-axis.

- Also, keep in mind that a spotlight direction is transformed by the model-view matrix just as though it were a normal vector.

OpenGL: Spotlight (3)

- In addition to the spotlight cut-off angle and direction, you can control the intensity distribution of the light within the cone, in two ways.
- First, you can set the distance attenuation factors described earlier.
- You can also set the GL_SPOT_EXPONENT parameter, to control how concentrated the light is.
- The light intensity is the highest in the centre of the cone. It is attenuated towards the edges of the cone by the cosine of the angle between the direction of the light and the direction from the light to the vertex lit exponentially. Thus, a higher spot exponent results in a more focused light source.

OpenGL: Multiple lights (1)

- As aforementioned, you can have up to 8 lights in your scene (possibly more, depending on your OpenGL implementation).
- Since OpenGL needs to perform calculations to determine how much light each vertex receives from each light source, increasing the number of lights adversely affects performance.
- The following lines of code define a point light and a spotlight:

OpenGL: Multiple lights (2)

```
GLfloat light1_ambient[] = {0.2, 0.2, 0.2, 1.0};
GLfloat light1_diffuse[] = {1.0, 1.0, 1.0, 1.0};
GLfloat light1_specular[] = {1.0, 1.0, 1.0, 1.0};
GLfloat light1_position[] = {-2.0, 2.0, 1.0, 1.0};
GLfloat spot_direction[] = {-1.0, -1.0, 0.0};

glLightfv(GL_LIGHT1, GL_AMBIENT, light1_ambient);
glLightfv(GL_LIGHT1, GL_DIFFUSE, light1_diffuse);
glLightfv(GL_LIGHT1, GL_SPECULAR, light1_specular);
glLightfv(GL_LIGHT1, GL_POSITION, light1_position);
glLightf(GL_LIGHT1, GL_CONSTANT_ATTENUATION, 1.5);
glLightf(GL_LIGHT1, GL_LINEAR_ATTENUATION, 0.5);
glLightf(GL_LIGHT1, GL_QUADRATIC_ATTENUATION, 0.2);

glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);
glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spot_direction);
glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, 2.0);

glEnable(GL_LIGHT1);
glEnable(GL_LIGHT0)
```

OpenGL: Selecting a lighting model (1)

- ***Global Ambient Light***. Each light source can contribute ambient light to a scene. In addition, there can be other ambient light that is not from any particular source. The GL_LIGHT_MODEL_AMBIENT parameter is used to specify the RGBA intensity of such global ambient light

```
GLfloat lmodel_ambient[] = {0.2, 0.2, 0.2, 1.0};
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient);
```

- In the example, the values used for *lmodel_ambient* are the default values for GL_LIGHT_MODEL_AMBIENT. Since these numbers yield a small amount of white ambient light, even if you do not add a specific light source to your scene, you can still see the objects in the scene.

OpenGL: Selecting a lighting model (2)

- ***Local or Infinite Viewpoint***. The location of the viewpoint affects the calculations for highlights produced by specular reflectance.
- More specifically, the intensity of the highlight at a particular vertex depends on the normal at that vertex, the direction from the vertex to the light source, and the direction from the vertex to the viewpoint.
- Keep in mind that the viewpoint is not actually moved by calls to lighting commands (you need to change the projection transformation); instead, different assumptions are made for the lighting calculations as if the viewpoint were moved.

OpenGL: Selecting a lighting model (3)

- With an infinite viewpoint, the direction between it and any vertex in the scene remains constant. A local viewpoint tends to yield more realistic results, but since the direction has to be calculated for each vertex, overall performance is decreased.
- By default, an infinite viewpoint is assumed. Here is how to change to a local viewpoint

```
glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);
```

- This call places the viewpoint at (0, 0, 0). To switch back to an infinite viewpoint, pass in GL_FALSE as the argument.

OpenGL: Selecting a lighting model (4)

- ***Two-sided Lighting***. Lighting calculations are performed for all polygons, whether they are front-facing or back-facing.
- You usually set up lighting conditions with the front-facing polygons in mind, but the back-facing polygons typically are not correctly illuminated.
- If the object is a sphere, only the front faces are ever seen, since they are the ones on the outside of the sphere. So, in this case, it does not matter what the back-facing polygons look like. If the sphere is cut away so that its inside surface would be visible, however, you may want to have the inside surface fully lit according to the lighting conditions defined; you may also want to supply a different material description for the back faces.

OpenGL: Defining material properties (1)

- Most of the material properties are conceptually similar to those used to create light sources. The mechanism for setting them is similar, except that the command used is called **`glMaterial*()`**.

```
void glMaterial{if}[v](GLenum face, GLenum pname,
                       TYPEparam)
```

- The code above specifies a current material property for lighting calculations.
- The face parameter can be GL_FRONT, GL_BACK, or GL_FRONT_AND_BACK to indicate which face of the object the material should be applied to.

OpenGL: Defining material properties (2)

- The particular material property being set is identified by *pname* and the desired values for that property are given by *TYPEparam*, which is either a pointer to a group of values (if the vector version is used) or the actual value (if the non-vector version is used).
- The non-vector version works only for setting GL_SHININESS.
- The possible values for *pname* are shown in the table on the next slide. Note that GL_AMBIENT_AND_DIFFUSE allows you to set both the ambient and diffuse material colours simultaneously to the same RGBA values.

OpenGL: Defining material properties (3)

Parameter Name	Default Value	Meaning
GL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	ambient colour of material
GL_DIFFUSE	(0.8, 0.8, 0.8, 1.0)	diffuse colour of material
GL_AMBIENT_AND_DIFFUSE		ambient and diffuse colour of material
GL_SPECULAR	(0.0, 0.0, 0.0, 1.0)	specular colour of material
GL_SHININESS	0.0	specular exponent
GL_EMISSION	(0.0, 0.0, 0.0, 1.0)	emissive colour of material
GL_COLOR_INDEXES	(0,1,1)	ambient, diffuse and specular colour indices

OpenGL: Defining material properties (4)

- You can choose to have lighting calculations performed differently for the front- and back-facing polygons of objects. If the back faces may be seen, you can supply different material properties for the front and the back surfaces by using the *face* parameter of **`glMaterial*()`**.
- Note that most of the material properties set with **`glMaterial*()`** are RGBA colours. Regardless of what alpha values are supplied for other parameters, the alpha value at any particular vertex is the diffuse-material alpha value, that is, the alpha value given to GL_DIFFUSE with the **`glMaterial*()`**.
- Also, none of the RGBA material properties apply in colour-index mode.

OpenGL: Defining material properties (5)

- Diffuse reflectance plays the most important role in determining what you perceive the colour of an object.
- Ambient reflectance affects the overall colour of the object. Because ambient reflectance is the brightest where an object is directly illuminated, it is the most noticeable where an object receives no direct illumination.
- The total ambient reflectance for an object is affected by the global ambient light and ambient light from individual light sources.
- Like diffuse reflectance, ambient reflectance is not affected by the position of the viewpoint.

OpenGL: Defining material properties (6)

- Specular reflection from an object produces highlights. Unlike ambient and diffuse reflections, the amount of specular reflection seen by a viewer depends on the location of the viewpoint – it is brightest along the direct angle of reflection.
- By specifying an RGBA colour for GL_EMISSION, you can make an object appear to be giving off light of that colour. Since most real-world objects (except lights) do not emit light, this feature is probably used to simulate lamps and other light sources in a scene.

Summary

- Lighting and materials together have important effects on the graphics rendering.
- There is a range of factors to consider: lighting sources (positional, spot and directional) and effects (ambient, diffuse and specular), lighting models (e.g. Phong model) and shading methods.
- Global ambient light and multiple light sources are combined with material properties.
- Approximations (e.g. ignoring angles between light rays for a light source at a relatively large distance, or angles between the viewer and any vertex in a scene for an infinite viewpoint) are applied in order to reduce the expensive computational work.