Lectures
Spotlight Case Study

By
Tom Duff
Pixar Animation Studios
Emeryville, California
and
George Ledin Jr
Sonoma State University
Rohnert Park, California

Directional Light vs Spot Light

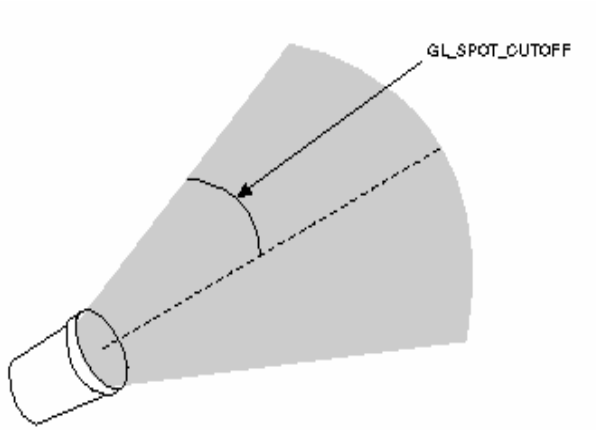
- **Directional Light: The source of light is at infinity. The light is a parallel light from a specified direction.**
  - We use a vector (x,y,z,0) to represent directional light, and we specify this vector by invoking glLight().

    - The “(x,y,z)” is specified at world coordinates.
    - The “(x,y,z)” is to represent the direction of light. This vector always points to the origin.
    - The “0” simply means light is at infinity.
  - Diffuse and specular lighting calculations take into account the light's direction, but not its actual position, and attenuation is disabled.
- **Spot Light: Light has a position, direction and angle.**
  - We use a vector (x,y,z,1) to represent spot light, and we specify the location of spot light at (x,y,z) by invoking glLight().

    - The “(x,y,z)” is specified at camera coordinates.
    - The “(x,y,z)” is to represent the actual position of the light.
    - The “1” simply means that the light is a spotlight.
  - We use another vector (a,b,c) to represent the direction of the spotlight.
    - The (a,b,c) is specified at world coordinates.
    - The vector (a,b,c) always points to the origin.

How to define spot light?

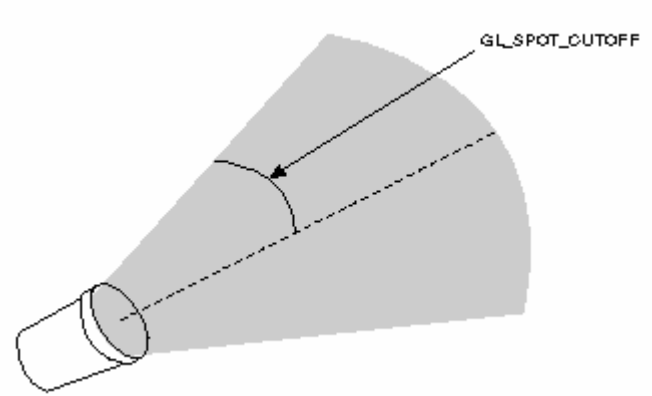
- **Spot Light: Light has a position, direction and angle.**
  - Define Position of light:
    - We use a vector (x,y,z,1) to represent spot light, and we specify the location of spot light at (x,y,z) by invoking glLight().
      - The “(x,y,z)” is specified at the camera view and represents the actual position of the light.
      - The “1” simply means that light is spot light.
    - Example:
      - `GLfloat light_position[] = {0, 0, 2, 1};`
      - glLightfv(GL_LIGHT0, GL_POSITION, light_position);
  - Define Direction of Light:
    - We use a vector (a,b,c) to represent the direction of spot light.
      - The (a,b,c) is specified at world coordinates.
      - The vector (a,b,c) always points to the origin.
    - Example:
      - GLfloat spot_direction[] = {-1.0, -1.0, 0.0};
      - glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spot_direc



  - Define cone of light (The shape of light cone)
    - Example: glLightfv(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0)
  - Define light spot exponent
    - The amount of light within a spotlight can be made to drop off exponentially from its center by giving it a nonzero value for GL_SPOT_EXPONENT.
    - Example: gLightf(GL_LIGHT1, GL_SPOT_EXPONENT, 2.0)

Define the cone of light

- You can have a positional light source act as a spotlight—that is, by restricting the shape of the light it emits to the shape of a cone.
- To create a spotlight, you need to determine the spread of the cone of light you desire, which is called **GL_SPOT_CUTOFF Parameter**.
- Note that no light is emitted beyond the edges of the cone.
- By default, the spot light feature is disabled because GL_SPOT_CUTOFF is 180.0.
- This value means that light is emitted in all directions (the angle at the cone's apex is 360 degrees)
- 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). If you define the cut-off other than [0,90], then the light will be directional light.



Case Study 1
Generic Spot Light Case

Case Study Setup:

- Object setup:
  - Assume in the world coordinates we have one black spheres of radius 2, centered at (0,0,0).

```
glutSolidSphere(2, 100,100);
```

- The camera's setup:

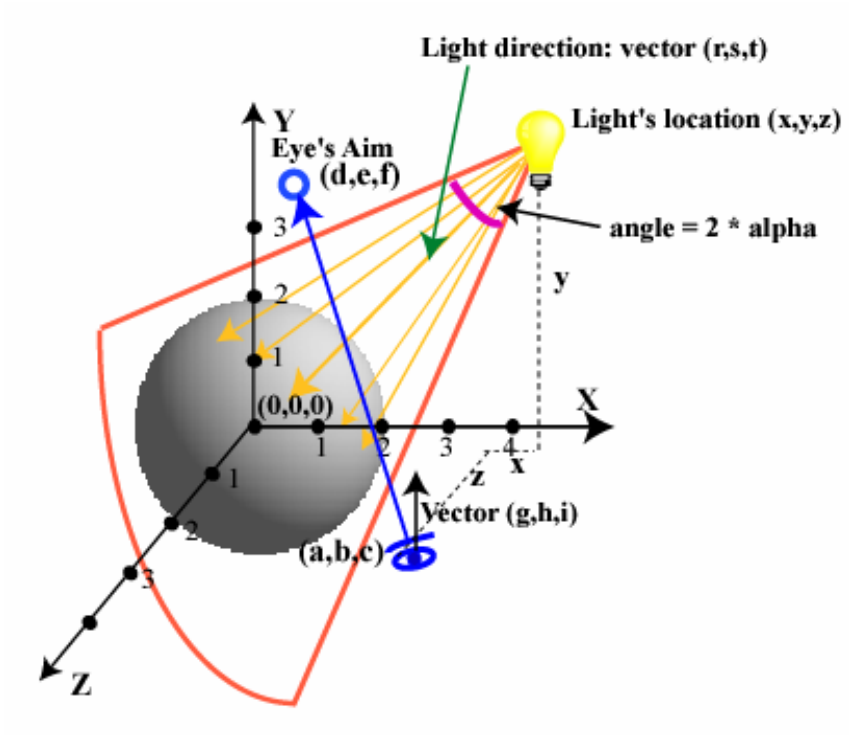
```
gluLookAt(a,b,c, d,e,f, g,h,i);
```

- Light cone setup:

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

- Light direction setup:

```
GLfloat light_position[] = (x,y,z,0);
GLfloat spot_light_position[] = (r,s,t);

glLightfv(GL_LIGHT0,
          GL_POSITION,
          light_position);

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



Question:

How will the object be lit up by the light?

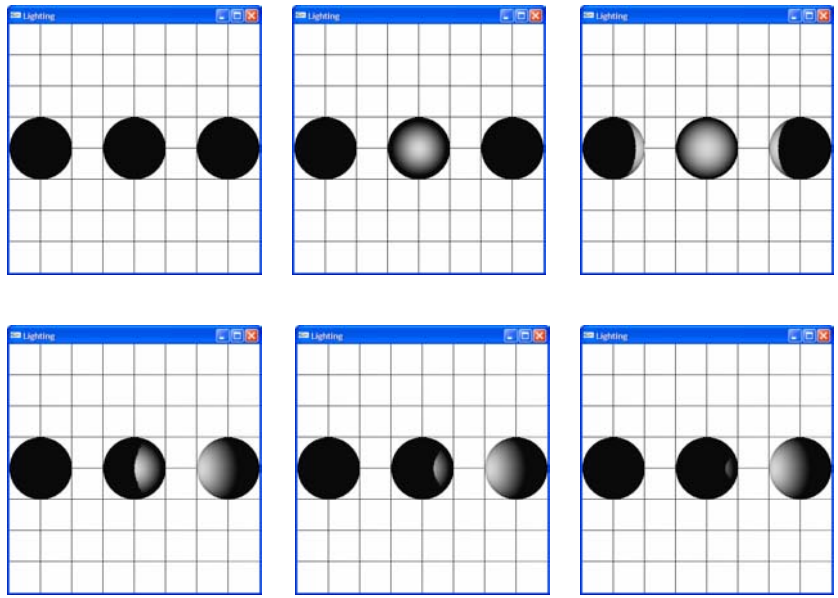
Case Study 2
Spot Lighting

Case Study Setup:

Assume in the world coordinates we have three black spheres of radius 1, centered, respectively, at (0,0,0), (-3,0,0) and (3,0,0).

Code for displaying the three spheres:

```
// center sphere
glutSolidSphere (1, 100,100);

// right sphere
glPushMatrix();
glTranslatef(3,0,0);
glutSolidSphere (1, 100,100);
glPopMatrix();

// left sphere
glPushMatrix();
glTranslatef(-3,0,0);
glutSolidSphere (1, 100,100);
glPopMatrix();
```



Goal:

We will experiment with different light locations and different light directions in world coordinates to see how the three spheres will be lit up.

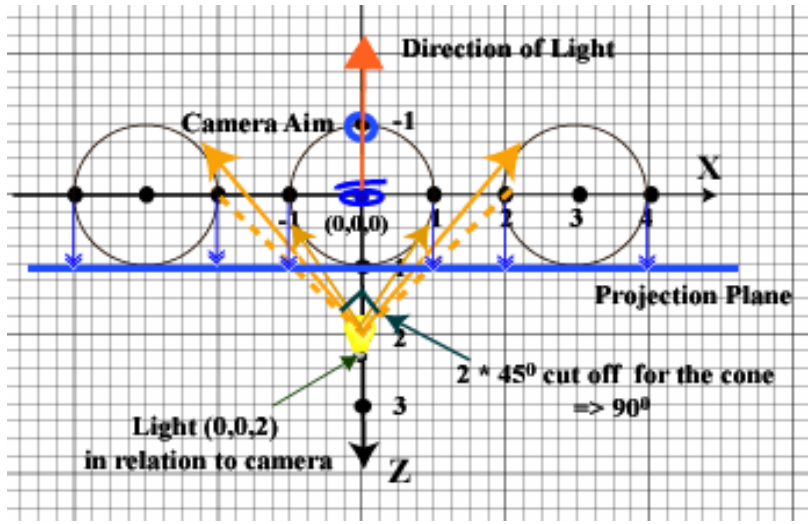
Spot Lighting
1) Set up the Projection View and Model View

```
void reshape (int w, int h)
{
   glViewport (0, 0, (GLsizei) w, (GLsizei) h);

   glMatrixMode (GL_PROJECTION);
   glLoadIdentity();
   if (w <= h)
      glOrtho (-4, 4, -4*(GLfloat)h/(GLfloat)w,
         4*(GLfloat)h/(GLfloat)w, -4.0, 4.0);
   else
      glOrtho (-4*(GLfloat)w/(GLfloat)h,
         4*(GLfloat)w/(GLfloat)h, -4, 4, -4, 4.0);

   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   glEnable(GL_DEPTH_TEST);
   gluLookAt(0,0,0,0,0,-1,0,1,0);
}

//main.c
glutReshapeFunc(reshape);
```

- What is achieved by the reshape function?
  - When w = h, the world coordinate bounding box is 8x8x8. If object is outside the bounding box, it cannot be viewed. If part of an object is outside, that part cannot be viewed.
  - When window is resized, and w>h or w<h, the object will maintain its shape, because reshape function adjusts the object's aspect ratio (width/height) according to window size.
  - Camera is explicitly set as located at (0,0,0), looking at (0,0,-1), which means camera is looking in the –z direction.

Spot Lighting
2) Set up light source and the direction of the light
Example 1 - default spot light direction (0,0,0) → (0,0,-1)

```
void init(void)
{
   // spot light's position
   GLfloat light_position[] = {0, 0, 2, 1};

   glEnable(GL_LIGHTING);  // Enable light
   glEnable(GL_LIGHT0);    // Enable one light, light 0

   // light cut-off is 45, cone angle is 90 degree
   glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);

   // specify spot light position at (0,0,2)
   glLightfv(  GL_LIGHT0, GL_POSITION, light_position);
}
//in main
Init();
// gluLookAt in reshape function
gluLookAt(0,0,0,0,0,-1,0,1,0);
```



Note on light_position: (x, y, z,w) and direction of light:

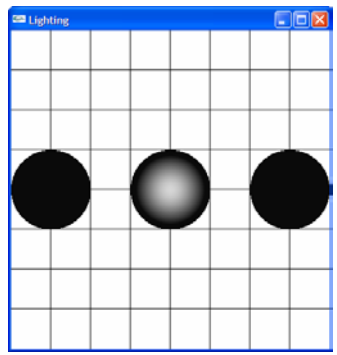
1. **When w = 1, the light is treated as spot light.**

2. **Since camera is located at (0,0,0), light_position (x,y,z) = (0,0,2). Therefore, the light in relation to camera is: (0,0,2).**

3. **The direction of light is a vector specified by (x,y,z) of spot_light_position:**
   `GLfloat spot_light_direction[] = {x, y, z};`
   glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spot_light_direction);

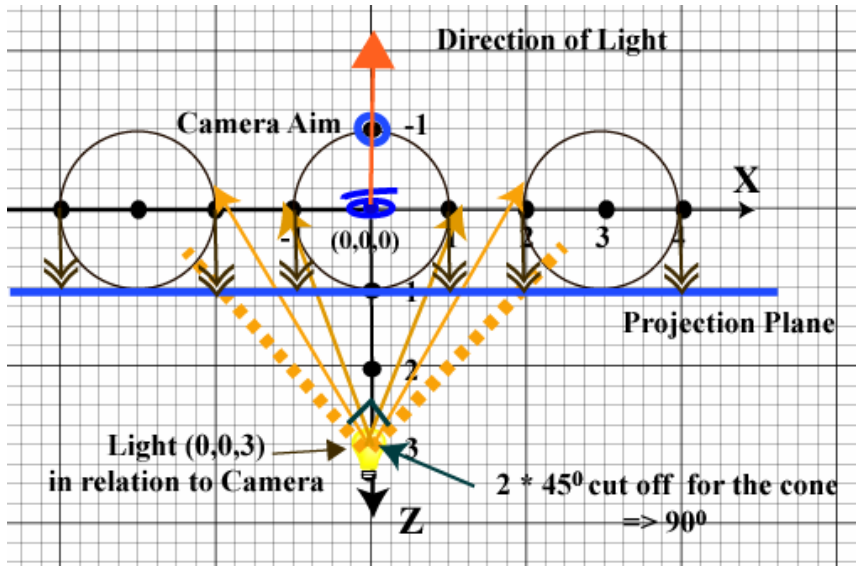
3. If the spot_light_direction is unspecified (as in this case), then the default spot light direction is a vector from origin to (0,0,-1).

Front View



Spot Lighting
Set up light source and the direction of the light
Example 2 - default spot light direction (0,0,0) → (0,0,-1)

```
void init(void)
{
   // spot light's position
   GLfloat light_position[] = {0, 0, 3, 1};

   glEnable(GL_LIGHTING);  // Enable light
   glEnable(GL_LIGHT0);    // Enable one light, light 0

   // light cut-off is 45, cone angle is 90 degree
   glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);

   // specify spot light position at (0,0,3)
   glLightfv(  GL_LIGHT0, GL_POSITION, light_position);
}
//in main
Init();
// gluLookAt in reshape function
gluLookAt(0,0,0,0,0,-1,0,1,0);
```



Note on light_position: (x, y, z,w) and direction of light:

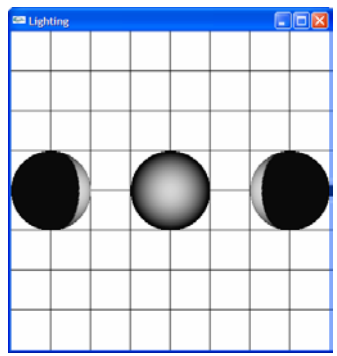
1. **When w = 1, the light is treated as spot light.**

2. **Since camera is located at (0,0,0), light_position (x,y,z) = (0,0,3). Therefore, the light in relation to camera is: (0,0,3).**

3. **The direction of light is a vector specified by (x,y,z) of spot_light_position:**
   `GLfloat spot_light_direction[] = {x, y, z};`
   glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spot_light_direction);

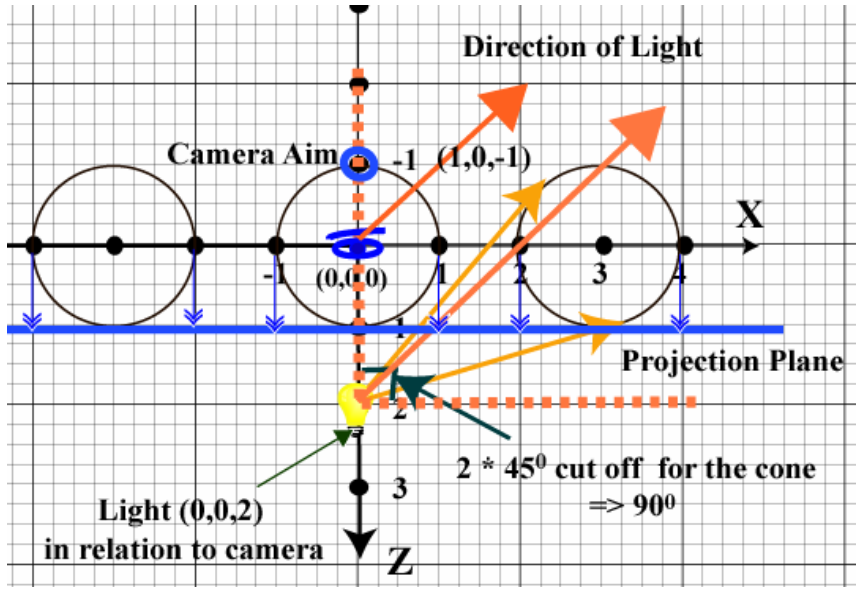
3. If the spot_light_direction is unspecified (as in this case), then the default spot light direction is a vector from the origin to (0,0,-1).

Front View



Spot Lighting
Set up light source and the direction of the light
Example 3 - spot light direction (0,0,0) → (1,0,-1)

```
void init(void)
{
   // spot light's position
   GLfloat light_position[] = {0, 0, 2, 1};
   GLfloat spot_light_direction[] = {1, 0, -1 };

   glEnable(GL_LIGHTING);  // Enable light
   glEnable(GL_LIGHT0);    // Enable one light, light 0

   // light cut-off is 45, cone angle is 90 degree
   glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);

   // specify spot light position at (0,0,3)
   glLightfv(  GL_LIGHT0, GL_POSITION, light_position);
   // specify spot light direction
   glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION,spot_light_direction);
}
//in main
Init();
// gluLookAt in reshape function
gluLookAt(0,0,0,0,0,-1,0,1,0);
```



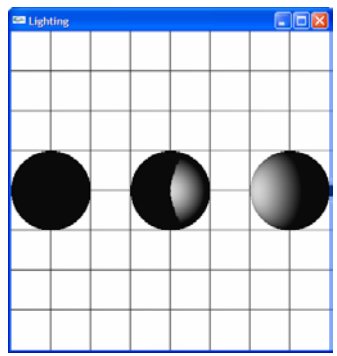
Note on light_position: (x, y, z,w) and direction of light:

1. **When w = 1, the light is treated as spot light.**

2. **Since camera is located at (0,0,0), light_position (x,y,z) = (0,0,2), therefore, the light in relation to camera is located at (0,0,2).**

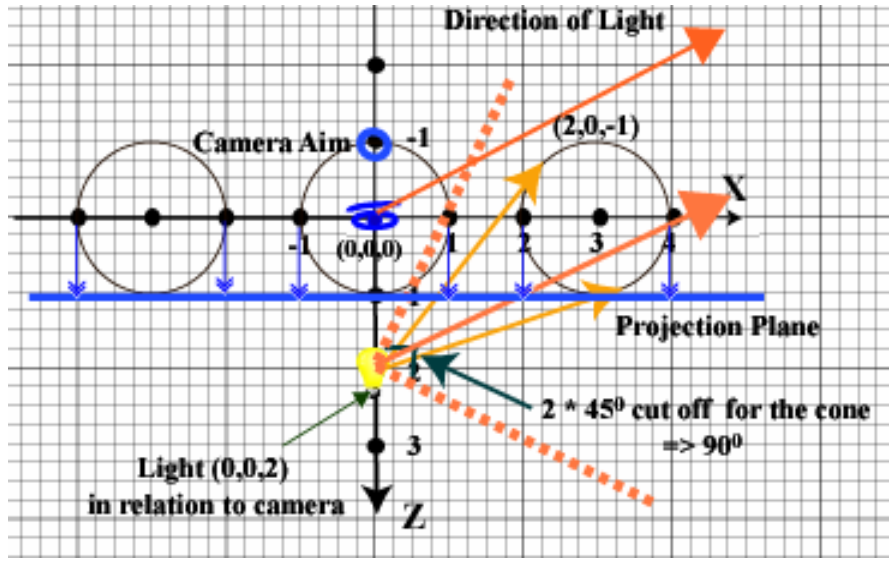
3. **The direction of light is a vector specified by (x,y,z) of spot_light_position (started at the origin). In this case, the direction is:**
   (0,0,0) → (1,0,-1)

Front View



Spot Lighting
Set up light source and the direction of the light
Example 4 - spot light direction (0,0,0) → (2,0,-1)

```
void init(void)
{
   // spot light's position
   GLfloat light_position[] = {0, 0, 3, 1};
   GLfloat spot_light_direction[] = {2, 0, -1 };

   glEnable(GL_LIGHTING);  // Enable light
   glEnable(GL_LIGHT0);    // Enable one light, light 0

   // light cut-off is 45, cone angle is 90 degree
   glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);

   // specify spot light position at (0,0,3)
   glLightfv(  GL_LIGHT0, GL_POSITION, light_position);
   // specify spot light direction
   glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION,spot_light_direction);
}
//in main
Init();
// gluLookAt in reshape function
gluLookAt(0,0,0,0,0,-1,0,1,0);
```



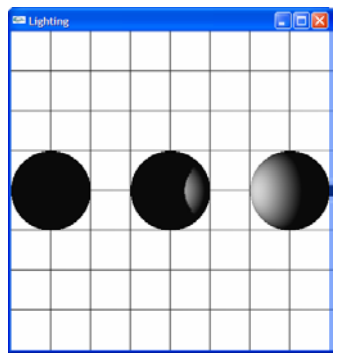
Note on light_position: (x, y, z,w) and direction of light:

1. **When w = 1, the light is treated as spot light.**

2. **Since camera is located at (0,0,0), light_position (x,y,z) = (0,0,3), therefore, the light in relation to camera is located at (0,0,3).**

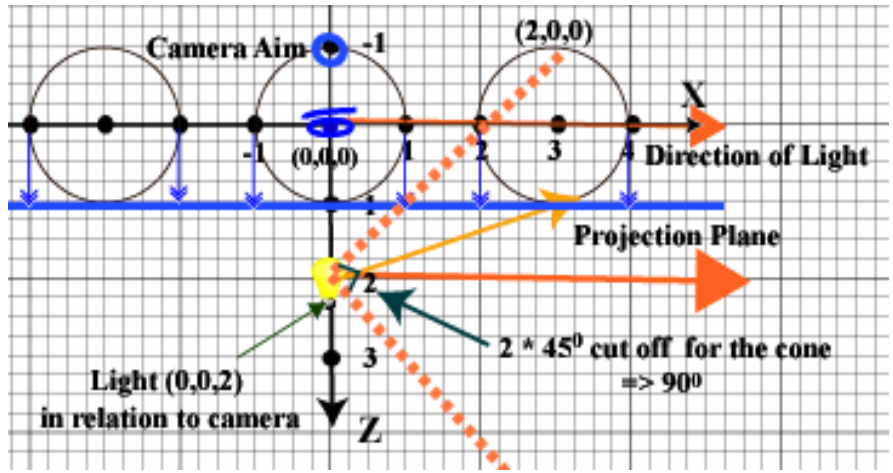
3. **The direction of light is a vector specified by (x,y,z) of spot_light_position (started at the origin). In this case, the direction is:**
   (0,0,0) → (2,0,-1)

Front View



Spot Lighting
Set up light source and the direction of the light
Example 5 (spot light direction (2,0,0))

```
void init(void)
{
   // spot light's position
   GLfloat light_position[] = {0, 0, 2, 1};
   GLfloat spot_light_direction[] = {2, 0, 0 };

   glEnable(GL_LIGHTING);  // Enable light
   glEnable(GL_LIGHT0);    // Enable one light, light 0

   // light cut-off is 45, cone angle is 90 degree
   glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);

   // specify spot light position at (0,0,3)
   glLightfv(  GL_LIGHT0, GL_POSITION, light_position);
   // specify spot light direction
   glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION,spot_light_direction);
}
//in main
Init();
// gluLookAt in reshape function
gluLookAt(0,0,0,0,0,-1,0,1,0);
```



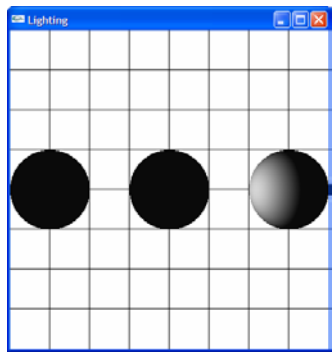
Note on light_position: (x, y, z,w) and direction of light:

1. **When w = 1, the light is treated as spot light.**

2. **Since camera is located at (0,0,0), light_position (x,y,z) = (0,0,2). Therefore, the light in relation to camera is: (0,0,2).**

3. **The direction of light is a vector specified by (x,y,z) of spot_light_position (started at the origin). In this case, the direction is:**
   (0,0,0) → (2,0,0)

Front View

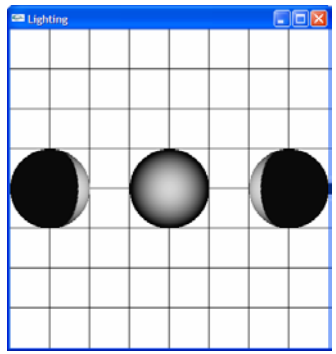


Case Study 3
Spot Light with Camera at location other than (0,0,0)

Case Study Setup:

Assume in the world coordinates we have three black spheres of radius 1, centered, respectively, at (0,0,0), (-3,0,0) and (3,0,0).

Front View



Code for displaying the three spheres:

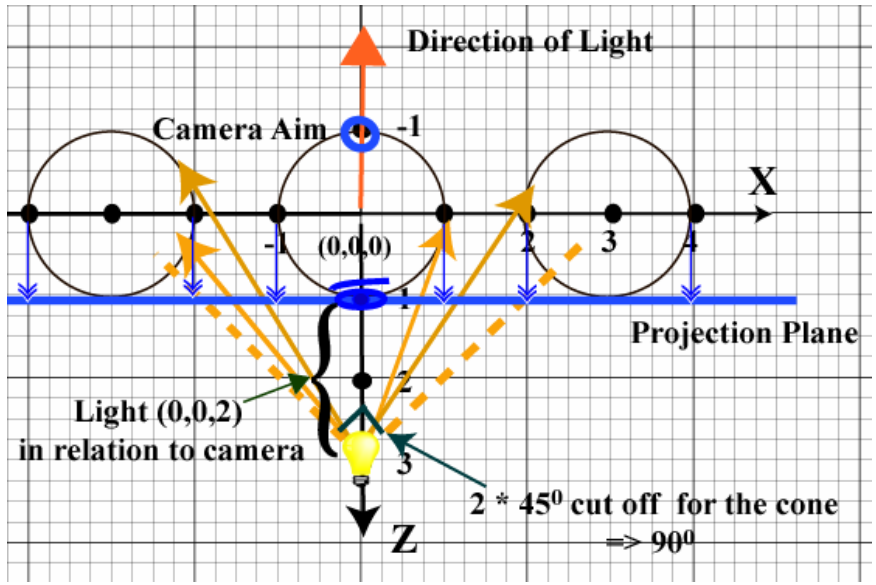
```
glutSolidSphere (1, 100,100);

glPushMatrix();
glTranslatef(3,0,0);
glutSolidSphere (1, 100,100);
glPopMatrix();

glPushMatrix();
glTranslatef(-3,0,0);
glutSolidSphere (1, 100,100);
glPopMatrix();
```

Goal:

We will experiment with spot light's location in relation to camera's location by setting camera's location other than (0,0,0).

Case Study 3
Spot Light with Camera at location (0,0,1), aiming at (0,0,-1)

```
void init(void)
{
   // spot light's position
   GLfloat light_position[] = {0, 0, 2, 1};
   GLfloat spot_light_direction[] = {0, 0, 2 };

   glEnable(GL_LIGHTING);  // Enable light
   glEnable(GL_LIGHT0);    // Enable one light, light 0

   // light cut-off is 45, cone angle is 90 degree
   glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);

   // specify spot light position at (0,0,2)
   glLightfv(  GL_LIGHT0, GL_POSITION, light_position);
   // specify spot light direction
   glLightfv(GL_LIGHT0,
      GL_SPOT_DIRECTION,spot_light_direction);
}
//in main
Init();
// gluLookAt in reshape function
gluLookAt(0,0,1,0,0,-1,0,1,0);
```



Note on light_position: (x, y, z,w) and direction of light:

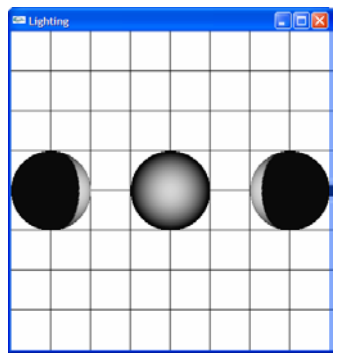
1. **When w = 1, the light is treated as spot light.**

2. **Since camera is located at (0,0,0), light_position (x,y,z) = (0,0,2). Therefore, the light in relation to camera is: (0,0,2).**

3. **The direction of light is a vector specified by (x,y,z) of spot_light_position:**
   ```
   GLfloat spot_light_direction[] = {x, y, z};
   glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spot_light_direction);
   ```

4. **If the spot_light_direction is unspecified (as in this case), then the default spot light direction is a vector from the origin to (0,0,-1).**

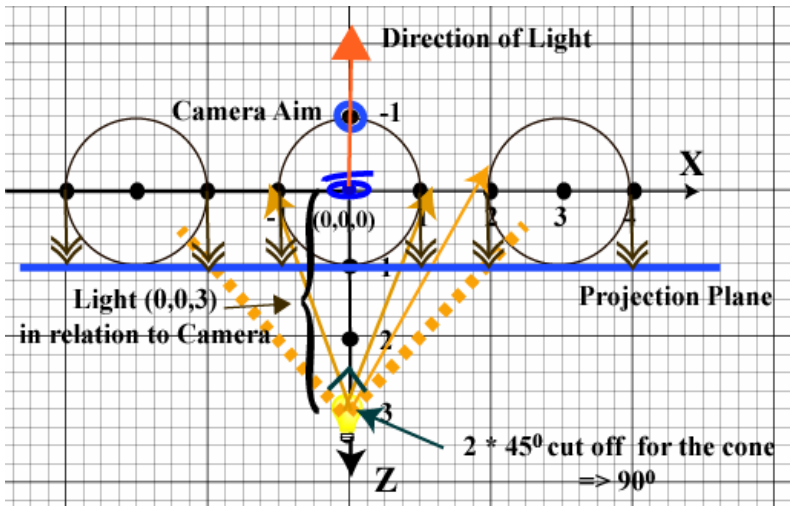
Front View



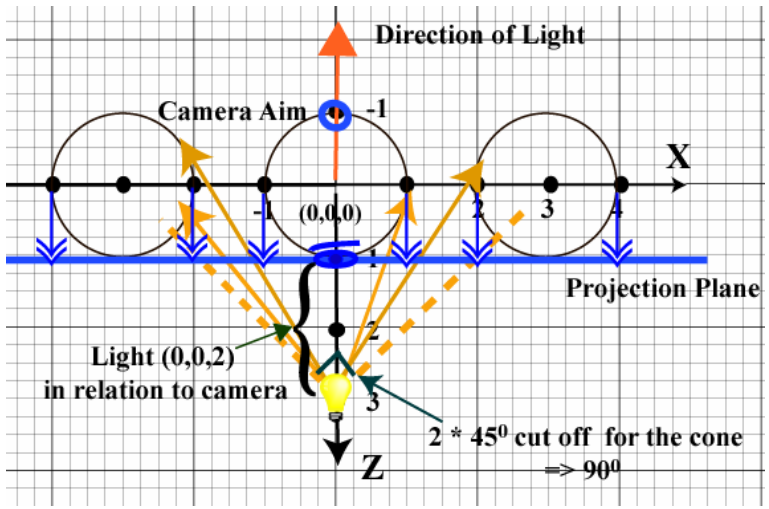
Case Study 3 - Compare Two Cases

1. Camera at (0,0,0), aiming at (0,0,-1). glLight's location (0,0,3)
2. Camera at location (0,0,1), aiming at (0,0,-1). glLight's location (0,0,2)

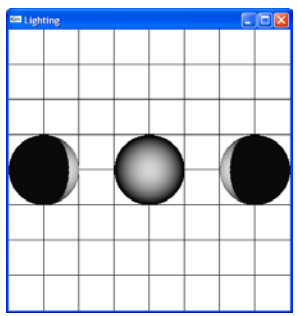
Case 1



Case 2



Both Cases produce the same Front View



Although in the above two cases, camera's locations and the glLight()'s location specifications are both different, since light's actual location is relative to the camera, therefore, the light's actual location of both cases are the same. Hence, the two cases produce the same outputs.

Position and Attenuation

- **There are two kinds of light source.**
  - Directional light source: the effect of an infinite location is that the rays of light can be considered parallel by the time they reach an object.
    - For the directional light, attenuation is disabled.
  - Positional light source: Its exact position within the scene determines the effect it has on a scene and, especially, the direction from which the light rays come.
    - You can attenuate the light from a positional light.
    - diffuse and specular lighting calculations are based on the actual location of the light in eye coordinates, and attenuation is enabled.

Light Attenuation Formula or Spotlight

- You can attenuate the spotlight from a positional light.
- OpenGL attenuates a light source by multiplying the contribution of that source by an attenuation factor:

$$\text{attenuation factor} = \frac{1}{k_c + k_l d + k_q d^2}$$

where

d = distance between the light's position and the vertex

k_c = GL_CONSTANT_ATTENUATION

k_l = GL_LINEAR_ATTENUATION

k_q = GL_QUADRATIC_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:

Example of attenuation constance specification:

Default Attenuation of spot light

```
glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 1.0);
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 0.0);
glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 0.0);
```

Case Study 4 - Spotlighting Attenuation

Case Study Setup:

- Object Setup:
  Assume in the world coordinates we have three black spheres of radius 3, centered at (0,0,0).

```
glutSolidSphere (3, 100,100);
```

- Light's position

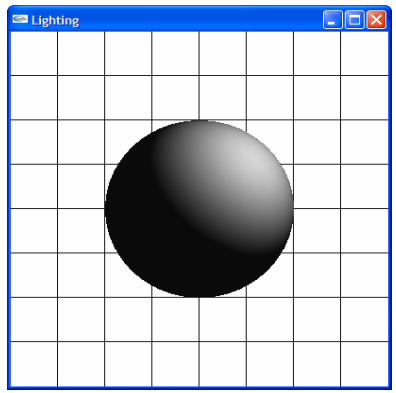
```
GLfloat light_position[] = {3, 3, 3, 1 };
glLightfv(GL_LIGHT0,
          GL_POSITION,
          light_position);
```

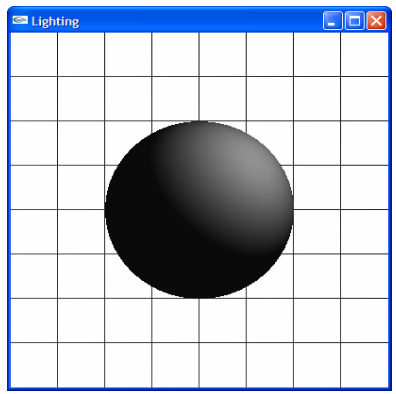
- Light's direction

```
GLfloat spot_light_direction[] = {-1, -1, -1
};
glLightfv(GL_LIGHT0,
           GL_SPOT_DIRECTION,
          spot_light_direction);
```

Goal:

We will experiment with different light attenuation constances to see how the sphere will be lit up.

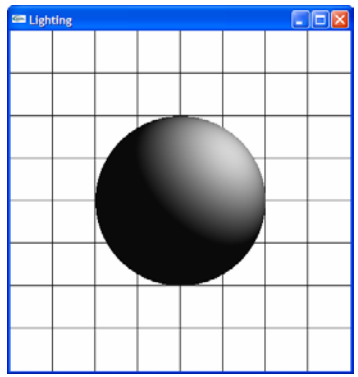




Case Study 4 – Spotlight Attenuation Example 1

```
void init(void)
{
   // spot light's position
   GLfloat light_position[] = {3, 3, 3, 1};
     GLfloat spot_light_direction[] = {-1, -1, -1 };

   glEnable(GL_LIGHTING);  // Enable light
   glEnable(GL_LIGHT0);    // Enable one light, light0

   // light cut-off is 45, cone angle is 90 degree
   glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);

   // specify spotlight position at (3,3,3)
   glLightfv(GL_LIGHT0, GL_POSITION, light_position);

   // specify the spotlight direction
   glLightfv(GL_LIGHT0,
             GL_SPOT_DIRECTION,
             spot_light_direction);

   glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, a);
   glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, b);
   glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, c);
}
//in main
Init();
```

Front View
No Light Attenuation



$$\text{attenuation factor} = \frac{1}{k_c + k_l d + k_q d^2}$$

where

d = distance between the light's position and the vertex

k_c = GL_CONSTANT_ATTENUATION

k_l = GL_LINEAR_ATTENUATION

k_q = GL_QUADRATIC_ATTENUATION

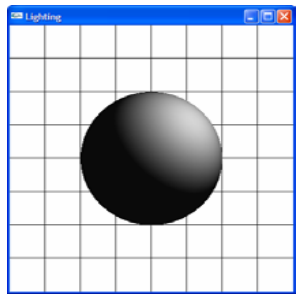
Note:
(Default spotlight attenuation setting: No Attenuation)
GL_CONSTANT_ATTENUATION: a = 1
GL_LINEAR_ATTENUATION: b = 0
GL_QUADRATIC_ATTENUATION: c = 0

Case Study 4 – Spotlight Attenuation Example 2

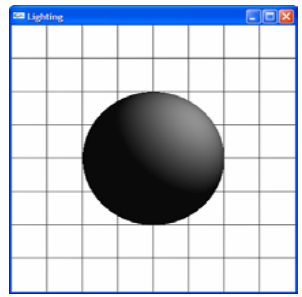
```
void init(void)
{
   // spot light's position
   GLfloat light_position[] = {3, 3, 3, 1};
     GLfloat spot_light_direction[] = {-1, -1, -1 };

   glEnable(GL_LIGHTING);  // Enable light
   glEnable(GL_LIGHT0);    // Enable one light, light0

   // light cut-off is 45, cone angle is 90 degree
   glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);

   // specify spotlight position at (3,3,3)
   glLightfv(GL_LIGHT0, GL_POSITION, light_position);

   // specify the spotlight direction
   glLightfv(GL_LIGHT0,
             GL_SPOT_DIRECTION,
             spot_light_direction);

   glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, a);
   glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, b);
   glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, c);
}
//in main
Init();
```

Front View

No Attenuation



With Attenuation



$$\text{attenuation factor} = \frac{1}{k_c + k_l d + k_q d^2}$$

where

d = distance between the light's position and the vertex

k_c = GL_CONSTANT_ATTENUATION

k_l = GL_LINEAR_ATTENUATION

k_q = GL_QUADRATIC_ATTENUATION

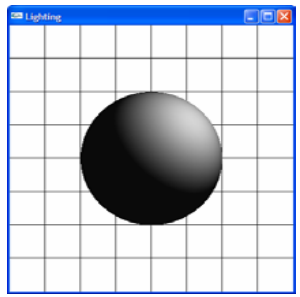
Note:
(default spotlight attenuation)
GL_CONSTANT_ATTENUATION: a = 1.5
GL_LINEAR_ATTENUATION: b = 0
GL_QUADRATIC_ATTENUATION: c = 0

Case Study 4 – Spotlight Attenuation Example 3

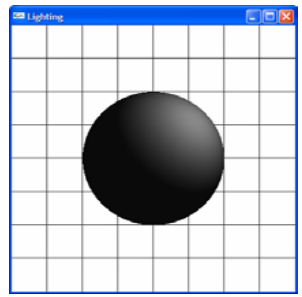
```
void init(void)
{
   // spot light's position
   GLfloat light_position[] = {3, 3, 3, 1};
     GLfloat spot_light_direction[] = {-1, -1, -1 };

   glEnable(GL_LIGHTING);  // Enable light
   glEnable(GL_LIGHT0);    // Enable one light, light0

   // light cut-off is 45, cone angle is 90 degree
   glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);

   // specify spotlight position at (3,3,3)
   glLightfv(GL_LIGHT0, GL_POSITION, light_position);

   // specify the spotlight direction
   glLightfv(GL_LIGHT0,
             GL_SPOT_DIRECTION,
             spot_light_direction);

   glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, a);
   glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, b);
   glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, c);
}
//in main
Init();
```

Front View

No Attenuation



With Attenuation



$$\text{attenuation factor} = \frac{1}{k_c + k_l d + k_q d^2}$$

where

d = distance between the light's position and the vertex

k_c = GL_CONSTANT_ATTENUATION

k_l = GL_LINEAR_ATTENUATION

k_q = GL_QUADRATIC_ATTENUATION

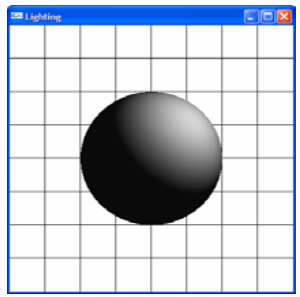
Note:
(default spotlight attenuation)
GL_CONSTANT_ATTENUATION: a = 0
GL_LINEAR_ATTENUATION: b = 0.5
GL_QUADRATIC_ATTENUATION: c = 0

Case Study 4 – Spotlight Attenuation Example 3

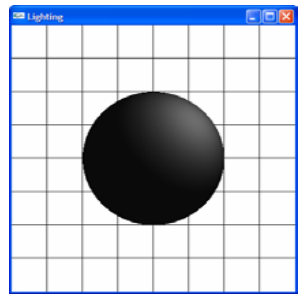
```
void init(void)
{
   // spot light's position
   GLfloat light_position[] = {3, 3, 3, 1};
     GLfloat spot_light_direction[] = {-1, -1, -1 };

   glEnable(GL_LIGHTING);  // Enable light
   glEnable(GL_LIGHT0);    // Enable one light, light0

   // light cut-off is 45, cone angle is 90 degree
   glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);

   // specify spotlight position at (3,3,3)
   glLightfv(GL_LIGHT0, GL_POSITION, light_position);

   // specify the spotlight direction
   glLightfv(GL_LIGHT0,
             GL_SPOT_DIRECTION,
             spot_light_direction);

   glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, a);
   glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, b);
   glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, c);
}
//in main
Init();
```

Front View

No Attenuation



With Attenuation



$$\text{attenuation factor} = \frac{1}{k_c + k_l d + k_q d^2}$$

where

d = distance between the light's position and the vertex

k_c = GL_CONSTANT_ATTENUATION

k_l = GL_LINEAR_ATTENUATION

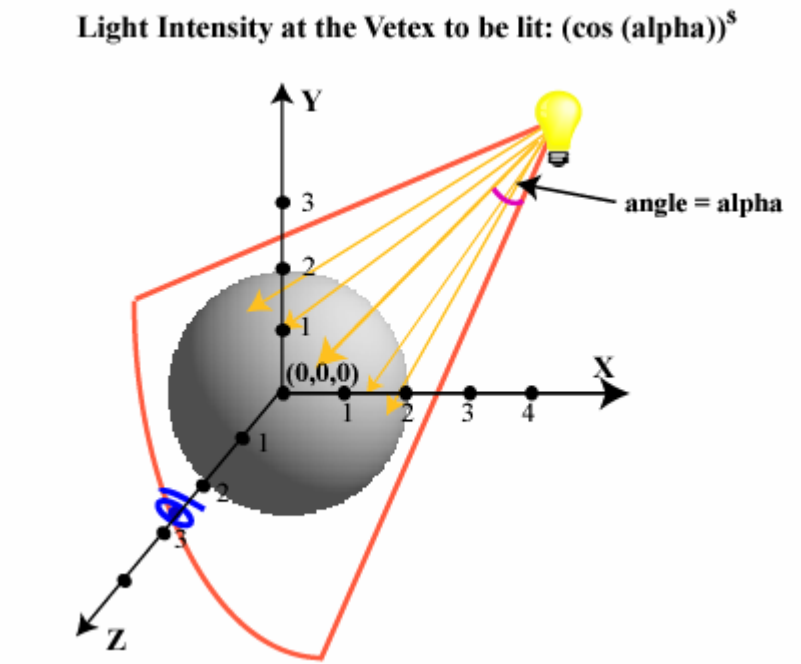
k_q = GL_QUADRATIC_ATTENUATION

Note:
(default spotlight attenuation)
GL_CONSTANT_ATTENUATION: a = 0
GL_LINEAR_ATTENUATION: b = 0
GL_QUADRATIC_ATTENUATION: c = 0.2

Light Intensity - Using Spotlight Exponent

```
void glLightfv(GLenum light, GLenum
    pname, const GLfloat *params)
```

- *light* : Specifies a light.
- *pname:* Specifies a single-valued light source parameter for *light*.
- *param:* Specifies the value that *pname* of light source *light* will be set to by passing a pointer to the value.



- **pname = GL_SPOT_EXPONENT**
  - It specifies the intensity distribution of the light. Only integer and floating point values in the range [0, 128] are accepted.
  - Effective light intensity is attenuated by: $(\cos(\text{alpha}))^s$
    - The “alpha” is the angle between the direction of the light and the direction from the light to the vertex being lit.
    - s: the spot exponent.
  - Thus, higher spot exponents result in a more focused light source, regardless of the spot cutoff angle (see next paragraph). The default spot exponent is 0, resulting in uniform light distribution, which is independent of the angle alpha.

Case Study 5 – Spotlight Intensity

Case Study Setup:

- Object Setup:
  Assume in the world coordinates we have a black spheres of radius 3, centered at (0,0,0).

```
glutSolidSphere (3, 100,100);
```

- Light's position

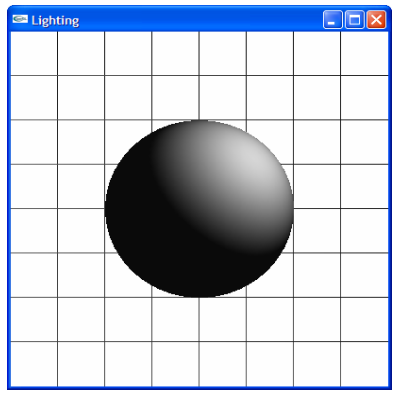
```
GLfloat light_position[] = {3, 3, 3, 1 };
glLightfv(GL_LIGHT0,
          GL_POSITION,
          light_position);
```

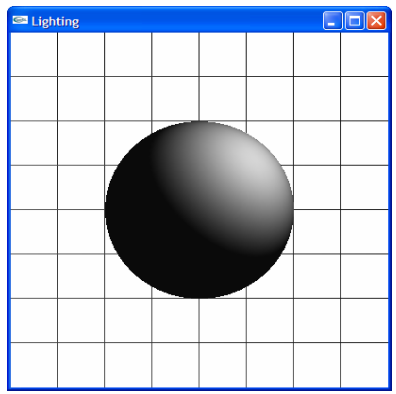
- Light's direction

```
GLfloat spot_light_direction[] = {-1, -1, -1};
glLightfv(GL_LIGHT0,
           GL_SPOT_DIRECTION,
          spot_light_direction);
```

Goal:

We will experiment with different light attenuation constants to see how the spheres will be lit up.

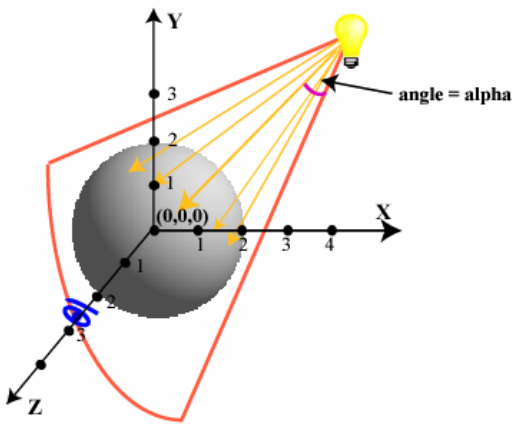




Case Study 5 – Spotlight Intensity Example 1

```
void init(void)
{
   // spotlight position
   GLfloat light_position[] = {3, 3, 3, 1};
     GLfloat spot_light_direction[] = {-1, -1, -1 };

   glEnable(GL_LIGHTING);  // Enable light
   glEnable(GL_LIGHT0);    // Enable one light, light0

   // light cut-off is 45, cone angle is 90 degree
   glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);

   // specify spotlight position at (3,3,3)
   glLightfv(GL_LIGHT0, GL_POSITION, light_position);

   // specify the spotlight direction
   glLightfv(GL_LIGHT0,
             GL_SPOT_DIRECTION,
             spot_light_direction);

   // spotlight intensity
   glLightf(GL_LIGHT0, GL_SPOT_EXPONENT,s);
}
```

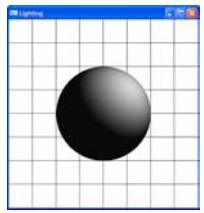
Light Intensity at the Vetex to be lit: $(\cos(\text{alpha}))^s$



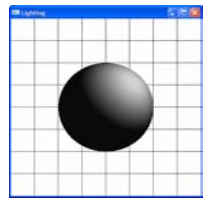
Front View

Uniform Intensity

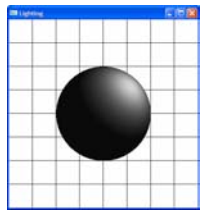
s = 0



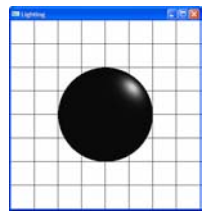
s = 0.5



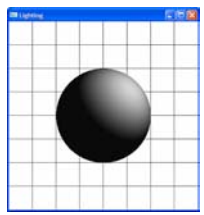
s = 10



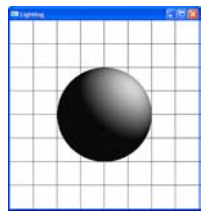
s = 128



s = -1



s = 130



Note:
(default spotlight intensity - uniform) GL_SPOT_EXPONENT: s = 0

When s > 128 or s < 0, the output is the same as s = 0.

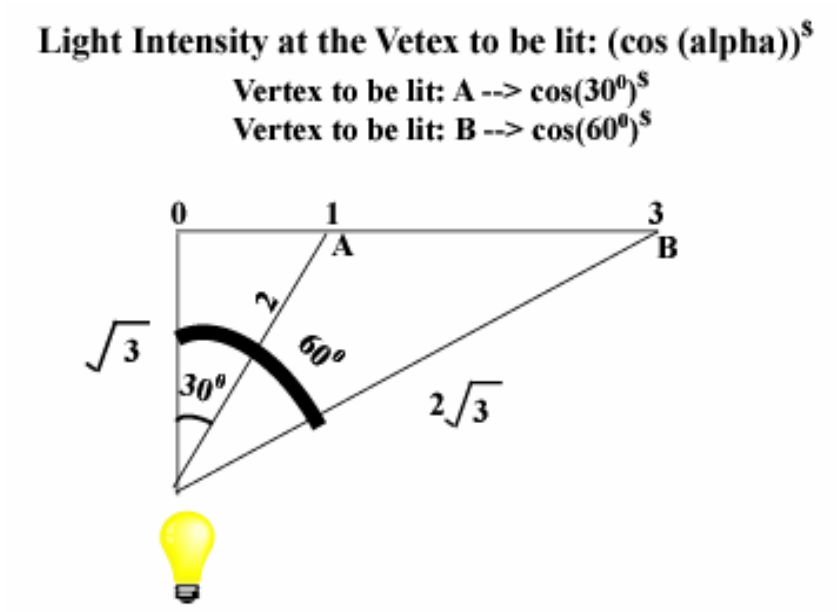
Case Study 5 - Spotlighting Intensity

Case Study Setup:

- Assume in the world coordinates the light is √3 away from the object to be lit.
- Vertex A and vertex B are both two points on the object to be lit.
  - For vertex A: cos(alpha) = cos (30^0) = 0.866
  - For vertex B: cos(alpha) = cos(60^0) = 0.5

Goal:

We will experiment with different spot light exponent, s, to find the light intensity ratio between Vertex A and B.



Case Study 5 – Spotlight Intensity - Example 2

Light Intensity Code: **`glLightf(GL_LIGHT0, GL_SPOT_EXPONENT,s);`**

Case A: s = 0
Uniform Intensity

Vertex A: $(\cos 30^0)^0 = 1$
Vertex B: $(\cos 60^0)^0 = 1$

A : B = 1 : 1

Case B: s = 0.5

Vertex A: $(\cos 30^0)^{0.5} = 0.93$
Vertex B: $(\cos 60^0)^{0.5} = 0.707$

A : B = 0.93 : 0.707 = 1.3 : 1

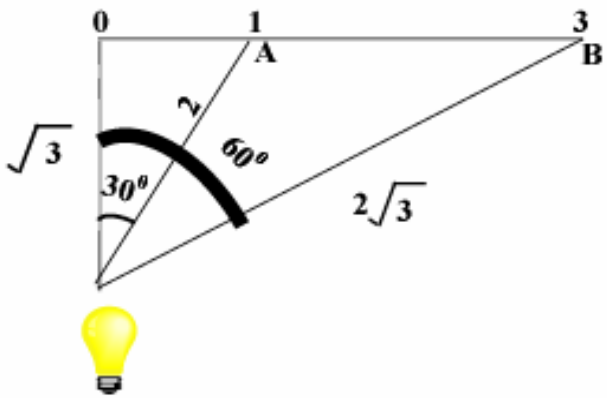
Case C: s = 10

Vertex A: $(\cos 30^0)^{10} = 0.23$
Vertex B: $(\cos 60^0)^{10} = 0.000976$

A : B = 235 : 1

Light Intensity at the Vetex to be lit: $(\cos(\text{alpha}))^s$

Vertex to be lit: A --> $\cos(30^0)$
Vertex to be lit: B --> $\cos(60^0)$



Note: This example shows that be higher the spotlight exponent, s, the more focused the spotlight.

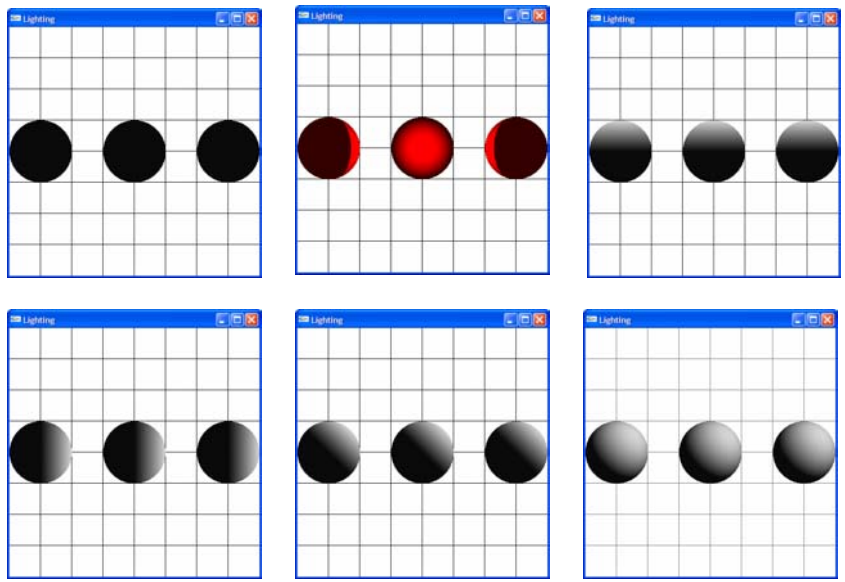
Case Study 6
Spot Lighting with Color Material

Case Study Setup:

Assume in the world coordinates we have three black spheres of radius 1, centered, respectively, at (0,0,0), (-3,0,0) and (3,0,0).

Code for displaying the three spheres:

```
glColor4f(1.0,0.0,0.0,0.0);
glutSolidSphere (1, 100,100);

glPushMatrix();
glTranslatef(3,0,0);
glutSolidSphere (1, 100,100);
glPopMatrix();


glPushMatrix();
glTranslatef(-3,0,0);
glutSolidSphere (1, 100,100);
glPopMatrix();
```

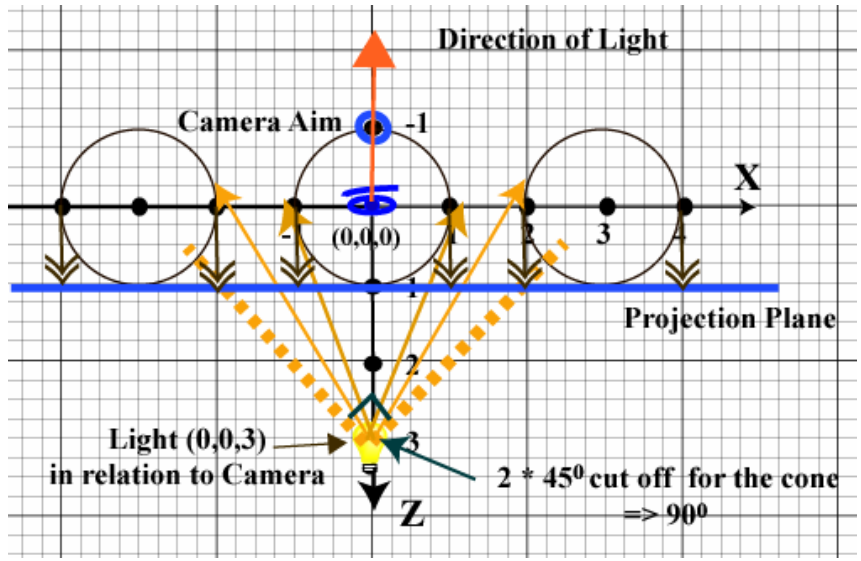


Goal:

We will experiment with different light directions in the world coordinates to see how the three spheres will be lit up.

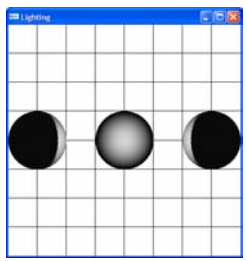
Case Study 6 - Lighting with color

```
void init(void)
{
   // spot light's position
   GLfloat light_position[] = {0, 0, 3, 1};

   glEnable(GL_LIGHTING);  // Enable light
   glEnable(GL_LIGHT0);    // Enable one light, light0

   // light cut-off is 45, cone angle is 90 degree
   glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);

   // specify spot light position at (0,0,3)
   glLightfv(GL_LIGHT0, GL_POSITION, light_position);

    glEnable(GL_COLOR_MATERIAL);
}
//in main
Init();
// in display function
glColor4f(1.0,0.0,0.0,0.0);  // red
```

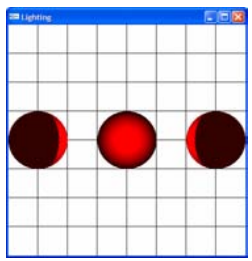


Front View

Without GL_COLOR_MATERIAL enabled



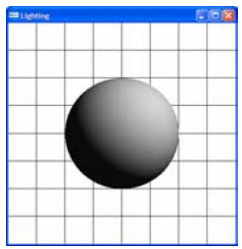
With GL_COLOR_MATERIAL enabled

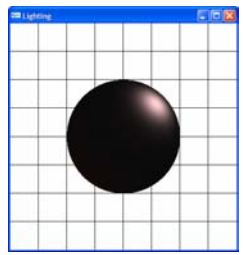


Case 7 - Direction light vs spotlight
Spotlight can specify GL_SPOT_EXPONENT, but directional light can't

```
// direction light direction
  GLfloat light_position[] = {3, 3, 3, 0};

  glEnable(GL_LIGHTING);  // Enable light

  glEnable(GL_LIGHT0);    // Enable one light,
    light0

  // specify directional light position at (3,3,3)
  glLightfv(GL_LIGHT0,GL_POSITION, ight_position);

  // spotlight intensity
  glLightf(GL_LIGHT0, GL_SPOT_EXPONENT,50);
```

```
// spotlight position
  GLfloat light_position[] = {3, 3, 3, 1};
  GLfloat spot_light_direction[] = {-1, -1, -1 };

  glEnable(GL_LIGHTING);  // Enable light
  glEnable(GL_LIGHT0);    // Enable one light, light0

  // light cut-off is 45, cone angle is 90 degrees
  glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);
  glLightfv(GL_LIGHT0, GL_POSITION, light_position);

  // specify the spotlight direction
  glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spot_light_direction);

  // spotlight intensity
  glLightf(GL_LIGHT0, GL_SPOT_EXPONENT,50);
```





glLight uses default values:
- **GL_AMBIENT={0,0,0,1} (black)**
- **GL_DIFFUSE={1,1,1,1) (white)**
- **GL_SPECULAR={1,1,1,1} (white)**

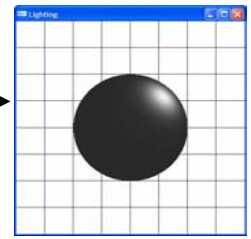
glMaterial uses default values:
- **GL_AMBIENT={0.2,0.2,0.2,1} (grey)**
- **GL_DIFFUSE={0.8,0.8,0.8,1) (lighter grey)**
- **GL_SPECULAR={0,0,0,1} (black)**

Case 8 - Adds color to spotlight

```
// spotlight position
GLfloat lightPosition[] = {3, 3, 3, 1};
GLfloat spotLightDirection[] = {-1, -1, -1 };
GLfloat fLtAmbient1[4] = { a, b, c, 1 };
GLfloat fLtDiffuse1[4] = { d, e, f, 1 };
GLfloat fLtSpecular1[4] = {x,y,z,1}; //white

 // material values and code
 GLfloat fMatAmbient[4] = {0.8,0.8,0.8,1}; //grey
GLfloat fMatDiffuse[4] = {0.8,0.8,0.8,1};//default-grey
GLfloat fMatSpecular[4] = {0.8,0.8,0.8,1}; //grey
GLfloat fShine = 50; // somewhat medium

glEnable(GL_LIGHTING);  // Enable light
glEnable(GL_LIGHT0);    // Enable one light, light0

// light cut-off is 45, cone angle is 90 degree
glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotLightDirectio
n);
glLightf(GL_LIGHT0, GL_SPOT_EXPONENT,50); //spotlight
intensity

glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
glLightfv(GL_LIGHT0, GL_AMBIENT, fLtAmbient1);
glLightfv(GL_LIGHT0, GL_DIFFUSE, fLtDiffuse1);
glLightfv(GL_LIGHT0, GL_SPECULAR, fLtSpecular1);

glMaterialfv(GL_FRONT, GL_AMBIENT, fMatAmbient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, fMatDiffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, fMatSpecular);
glMaterialf(GL_FRONT, GL_SHININESS, fShine);
```

Ambient: a=1,b=1,c=1 (white)
Diffuse:d=0,e=0,f=0 (black)
Specular:x=0,y=0,z=0 (black)

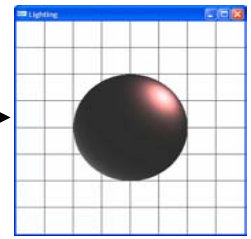
Ambient: a=0,b=0,c=0 (black)
Diffuse:d=1,e=1,f=1 (white)
Specular:x=0,y=0,z=0(black)



White spot

Ambient: a=1,b=1,c=1 (white)
Diffuse:d=1,e=0,f=0) (red)
Specular:x=0,y=0,z=0(white)

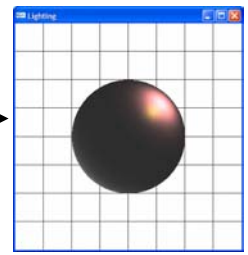
Ambient: a=1,b=0,c=0 (red)
Diffuse:d=1,e=1,f=1 (white)
Specular:x=0,y=0,z=0(white)



Red+ white spot

Ambient: a=1,b=1,c=1 (white)
Diffuse:d=1,e=0,f=0 (red)
Specular:x=1,y=1,z=0(yellow)

Ambient: a=1,b=0,c=0 (red)
Diffuse:d=1,e=1,f=1) (white)
Specular:x=1,y=1,z=0(yellow)



Red+ white+ yellow spot

glLight uses default values:
GL_AMBIENT={0,0,0,1} (black)
GL_DIFFUSE={1,1,1,1) (white)
GL_SPECULAR={1,1,1,1} (white)
glMaterial uses default values:

GL_AMBIENT={0.2,0.2,0.2,1} (grey)
GL_DIFFUSE={0.8,0.8,0.8,1) (lighter grey)
GL_SPECULAR={0,0,0,1} (black)

Therefore, we study them together.

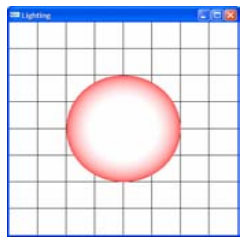
Case 9 - Another way to define material color
Let material track color of light

```
// sphere's material color
glColor4f(r,s,t,1); //red=r,green=s,blue=t
// enable material color
glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT,GL_AMBIENT_AND_DIFFUSE);//default
```

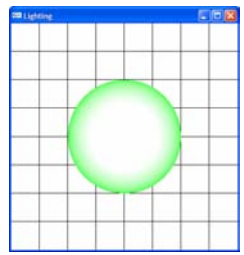
```
GLfloat lightPosition[] = {0, 0, 1, 1};
GLfloat spotLightDirection[] = {-1, -1, -1 };
GLfloat fLtAmbient1[4] = { a, b, c, 1 };
GLfloat fLtDiffuse1[4] = { 1, 1, 1, 1 };//white
GLfloat fLtSpecular1[4] = {0,0,0};

glEnable(GL_LIGHTING);  // Enable light
glEnable(GL_LIGHT0);    // Enable one light, light0

glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);
glLightfv(GL_LIGHT0,
          GL_SPOT_DIRECTION,
          spotLightDirection);
glLightf(GL_LIGHT0, GL_SPOT_EXPONENT,50);

glLightfv(GL_LIGHT0, GL_AMBIENT, fLtAmbient1);
glLightfv(GL_LIGHT0, GL_DIFFUSE, fLtDiffuse1);
glLightfv(GL_LIGHT0, GL_SPECULAR, fLtSpecular1);

glEnable(GL_LIGHTING);
glEnable(GL_COLOR_MATERIAL);
```

r=1,s=1,t=1 (white material)

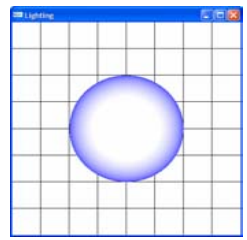
a=1, b=0, c=0
red



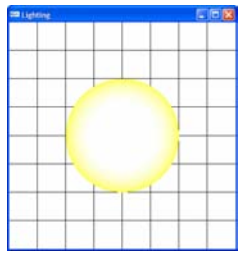
a=0, b=1, c=0
green



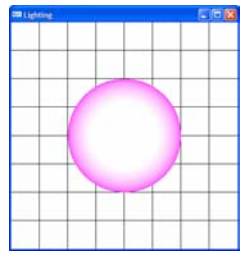
a=0, b=0, c=1
blue



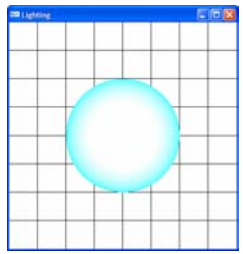
a=1, b=1, c=0
yellow



a=0, b=1, c=1
purple



a=0, b=1, c=1
cyan



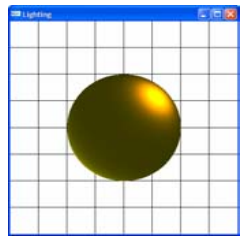
Case 9 - Turn on material-tracking-light color, continued

```
// spotlight position
GLfloat lightPosition[] = {3, 3, 3, 1};
GLfloat spotLightDirection[] = {-1, -1, -1 };
GLfloat fLtAmbient1[4] = { a, b, c, 1 };
GLfloat fLtDiffuse1[4] = { d, e, f, 1 };
GLfloat fLtSpecular1[4] = {0,0,0,1}; //black

glEnable(GL_LIGHTING);  // Enable light
glEnable(GL_LIGHT0);    // Enable one light, light0

// light cut-off is 45, cone angle is 90 degree
glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotLightDirection);
glLightf(GL_LIGHT0, GL_SPOT_EXPONENT,50); //light intensity

glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
glLightfv(GL_LIGHT0, GL_AMBIENT, fLtAmbient1);
glLightfv(GL_LIGHT0, GL_DIFFUSE, fLtDiffuse1);
glLightfv(GL_LIGHT0, GL_SPECULAR, fLtSpecular1);
```

Material: r=1,s=1,t=0 (yellow)

Ambient: a=1,b=0,c=0 (red)
Diffuse:d=1,e=1,f=1 (white)

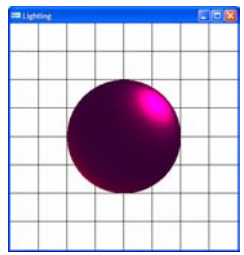
Ambient: a=1,b=1,c=1 (white)
Diffuse:d=1,e=0,f=0 (red)



Material: r=1,s=0,t=1 (purple)

Ambient: a=1,b=0,c=0 (red)
Diffuse:d=1,e=1,f=1 (white)

Ambient: a=1,b=1,c=1 (white)
Diffuse:d=1,e=0,f=0 (red)



```
// sphere's material color
glColor4f(r,s,t,1); //red=r,green=s,blue=t
// enable material color
glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT,GL_AMBIENT_AND_DIFFUSE);//default
```

When enable material-tracking-color, material's color (yellow or purple) is added to the original reddish spot.

What does glColorMaterial do? When is it equivalent to glMaterial?

```
GLfloat lightPosition[] = {3, 3, 3, 1};
GLfloat spotLightDirection[] = {-1, -1, -1 };
GLfloat fLtAmbient1[4] = { 1, 0, 0, 1 };
GLfloat fLtDiffuse1[4] = { 0.8, 0.8, 0.8, 1 };
GLfloat fLtSpecular1[4] = {0,0,0,1};

// material values and code
GLfloat fMatAmbient[4] = {a,b,c,1};
GLfloat fMatDiffuse[4] = {d,e,f,1};
GLfloat fMatSpecular[4] = {g,h,i,1};
GLfloat fShine = 0; // median

glEnable(GL_LIGHTING);  // Enable light
glEnable(GL_LIGHT0);    // Enable one light, light0
glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);
glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotLightDirection);
glLightf(GL_LIGHT0, GL_SPOT_EXPONENT,50); // spotlight intensity

glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
glLightfv(GL_LIGHT0, GL_AMBIENT, fLtAmbient1);
glLightfv(GL_LIGHT0, GL_DIFFUSE, fLtDiffuse1);
glLightfv(GL_LIGHT0, GL_SPECULAR, fLtSpecular1);
```

```
//set material color
glMaterialfv(GL_FRONT, GL_AMBIENT, fMatAmbient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, fMatDiffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, fMatSpecular);
glMaterialf(GL_FRONT, GL_SHININESS, fShine);
```

? =

```
//the following code will
//set material properties to follow glColor values
glColorMaterial(GL_COLOR_MATERIAL);
(1)glColorMaterial(GL_FRONT,GL_AMBIENT);
(2)glColorMaterial(GL_FRONT,GL_DIFFUSE);
(3)glColorMaterial(GL_FRONT,GL_AMBIENT_AND_DIFFUSE);
(4)glColorMaterial(GL_FRONT,GL_SPECULAR);
   glMaterialfv(GL_FRONT,GL_SPECULAR,specref);
```

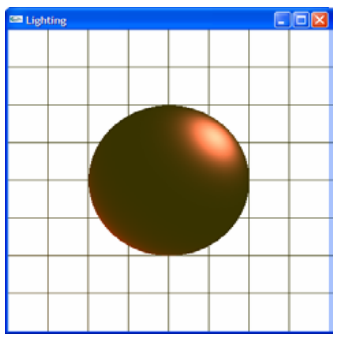
Study of glColorMaterial(…,GL_AMBIENT)

glColor3f(1,1,0) // yellow

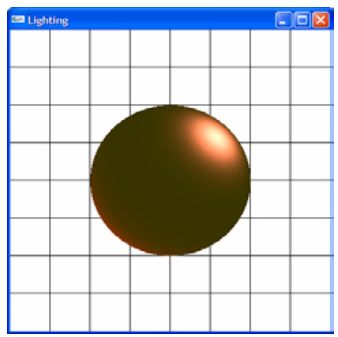
```
GLfloat lightPosition[] = {3, 3, 3, 1};
GLfloat spotLightDirection[] = {-1, -1, -1 };
GLfloat fLtAmbient1[4] = { 1, 0, 0, 1 };
GLfloat fLtDiffuse1[4] = { 0.8, 0.8, 0.8, 1 };
GLfloat fLtSpecular1[4] = {0,0,0,1};
```

```
X1a

GLfloat fMatAmbient[4] = {1,1,0,1};
GLfloat fMatDiffuse[4] = {0.8,0.8,0.8,1};
GLfloat fMatSpecular[4] = {0,0,0,1};

glMaterialfv(GL_FRONT, GL_AMBIENT, fMatAmbient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, fMatDiffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, fMatSpecular);
```

```
X2 –glColor changes ambient reflection

glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT,GL_AMBIENT);
```



≠



Study of glColorMaterial(…,GL_AMBIENT)

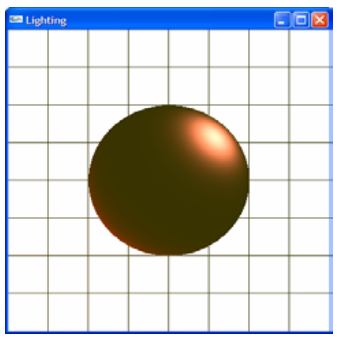
glColor3f(1,1,0) // yellow

```
GLfloat lightPosition[] = {3, 3, 3, 1};
GLfloat spotLightDirection[] = {-1, -1, -1 };
GLfloat fLtAmbient1[4] = { 1, 0, 0, 1 };
GLfloat fLtDiffuse1[4] = { 0.8, 0.8, 0.8, 1 };
GLfloat fLtSpecular1[4] = {0,0,0,1};
```

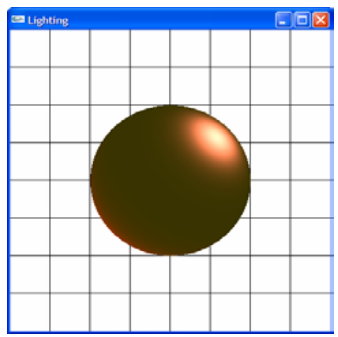
```
X1b

GLfloat fMatAmbient[4] = {1,1,0,1};
GLfloat fMatDiffuse[4] = {1,1,1,1};
GLfloat fMatSpecular[4] = {1,1,1,1};

glMaterialfv(GL_FRONT, GL_AMBIENT, fMatAmbient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, fMatDiffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, fMatSpecular);
```

X2 –glColor changes ambient reflection

```
glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT,GL_AMBIENT);
```



=



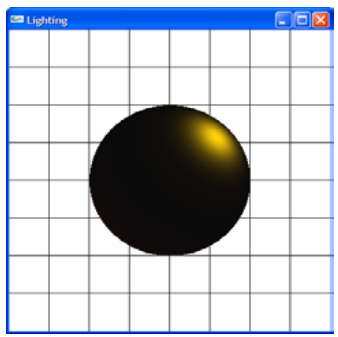
Study of glColorMaterial(…,GL_DIFFUSE)

glColor3f(1,1,0) // yellow

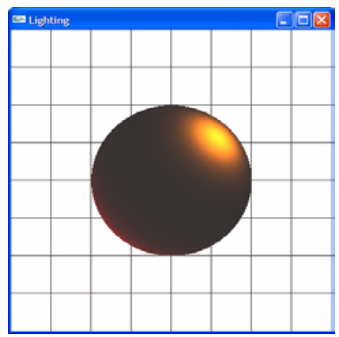
```
GLfloat lightPosition[] = {3, 3, 3, 1};
GLfloat spotLightDirection[] = {-1, -1, -1 };
GLfloat fLtAmbient1[4] = { 1, 0, 0, 1 };
GLfloat fLtDiffuse1[4] = { 0.8, 0.8, 0.8, 1 };
GLfloat fLtSpecular1[4] = {0,0,0,1};
```

```
Y1a

GLfloat fMatAmbient[4] = {0.2,0.2,0.2,1};
GLfloat fMatDiffuse[4] = {1,1,0,1};
GLfloat fMatSpecular[4] = {0,0,0,1};

glMaterialfv(GL_FRONT, GL_AMBIENT, fMatAmbient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, fMatDiffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR,fMatSpecular);
```

```
Y2 –glColor changes diffuse reflection

glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT,GL_DIFFUSE);
```



≠



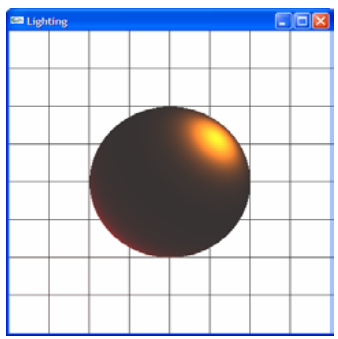
Study of glColorMaterial(…,GL_DIFFUSE)

glColor3f(1,1,0) // yellow

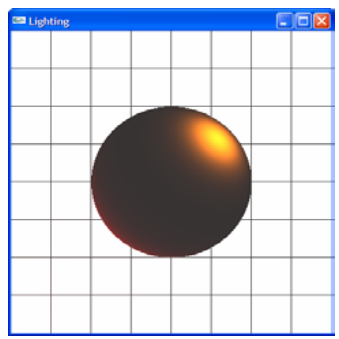
```
GLfloat lightPosition[] = {3, 3, 3, 1};
GLfloat spotLightDirection[] = {-1, -1, -1 };
GLfloat fLtAmbient1[4] = { 1, 0, 0, 1 };
GLfloat fLtDiffuse1[4] = { 0.8, 0.8, 0.8, 1 };
GLfloat fLtSpecular1[4] = {0,0,0,1};
```

```
Y1b

GLfloat fMatAmbient[4] = {1,1,1,1};
GLfloat fMatDiffuse[4] = {1,1,0,1};
GLfloat fMatSpecular[4] = {1,1,1,1};


glMaterialfv(GL_FRONT, GL_AMBIENT, fMatAmbient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, fMatDiffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, fMatSpecular);
```

```
Y2 -glColor changes diffuse reflection

glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT,GL_DIFFUSE);
```



=



Study of glColorMaterial(…,GL_AMBIENT_AND_DIFFUSE)

glColor3f(1,1,0) // yellow

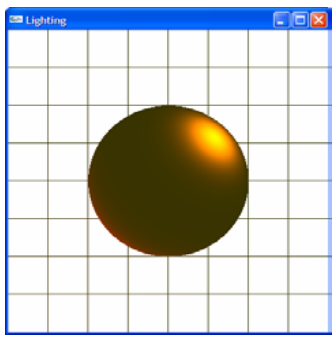
```
GLfloat lightPosition[] = {3, 3, 3, 1};
GLfloat spotLightDirection[] = {-1, -1, -1 };
GLfloat fLtAmbient1[4] = { 1, 0, 0, 1 };
GLfloat fLtDiffuse1[4] = { 0.8, 0.8, 0.8, 1 };
GLfloat fLtSpecular1[4] = {0,0,0,1};
```

Z1

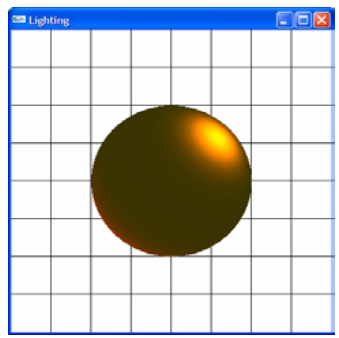
```
GLfloat fMatAmbient[4] = {1,1,0,1};
GLfloat fMatDiffuse[4] = {1,1,0,1};
GLfloat fMatSpecular[4] = {1,1,1,1};

glMaterialfv(GL_FRONT, GL_AMBIENT, fMatAmbient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, fMatDiffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, fMatSpecular);
```

Z2 –glColor changes ambient and diffuse reflection

```
glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT,GL_AMBIENT_AND_DIFFUSE);
```



=

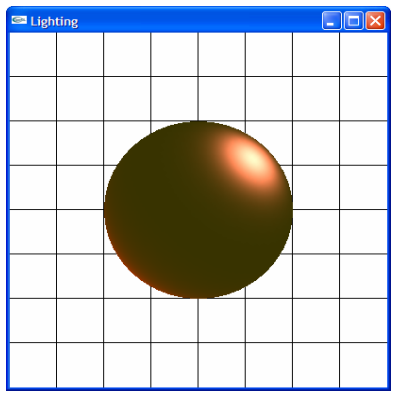


Compare 3 glColorMaterial(GL_FRONT, x)

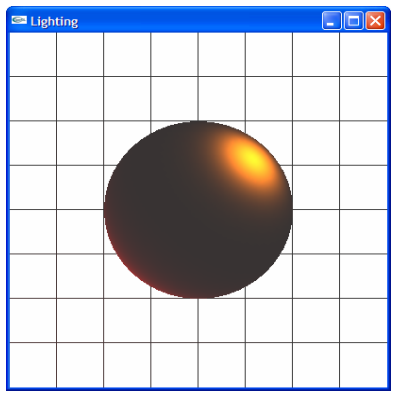
glColor3f(1,1,0) // yellow

```
GLfloat lightPosition[] = {3, 3, 3, 1};
GLfloat spotLightDirection[] = {-1, -1, -1 };
GLfloat fLtAmbient1[4] = { 1, 0, 0, 1 };
GLfloat fLtDiffuse1[4] = { 0.8, 0.8, 0.8, 1 };
GLfloat fLtSpecular1[4] = {0,0,0,1};
…
glEnable(GL_COLOR_MATERIAL);
```

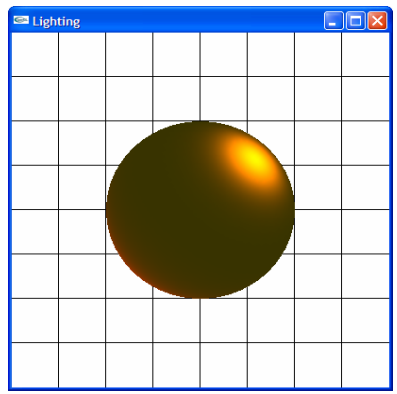
X = GL_AMBIENT



X = GL_DIFFUSE



X = GL_AMBIENT_AND_DIFFUSE



Study of glColorMaterial(…,GL_SPECULAR)

glColor3f(1,1,0) // yellow

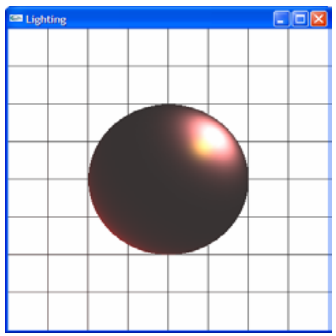
```
GLfloat lightPosition[] = {3, 3, 3, 1};
GLfloat spotLightDirection[] = {-1, -1, -1 };
GLfloat fLtAmbient1[4] = { 1, 0, 0, 1 };//red
GLfloat fLtDiffuse1[4] = { 1, 1, 1, 1 };//white
GLfloat fLtSpecular1[4] = {1,1,0,1};//yellow
GLfloat fMatShine = 50; // median
glMaterialf(GL_FRONT, GL_SHININESS, fMatShine);
```

```
P1
GLfloat fMatAmbient[4] = {1,1,1,1};
GLfloat fMatDiffuse[4] = {1,1,1,1};
GLfloat fMatSpecular[4] = {1,1,0,1};

glMaterialfv(GL_FRONT, GL_AMBIENT, fMatAmbient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, fMatDiffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, fMatSpecular);
```

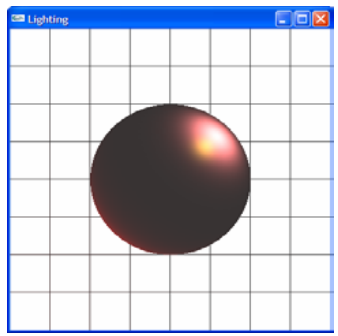
Z2 –glColor changes specular reflection

glEnable(GL_COLOR_MATERIAL);

glColorMaterial(GL_FRONT,GL_SPECULAR);



=



Conclusion: glMaterial vs glColorMaterial

```
GLfloat fMatAmbient[4] = {a,b,c,1};
GLfloat fMatDiffuse[4] = {1,1,1,1};
GLfloat fMatSpecular[4] = {1,1,1,1};

glMaterialfv(GL_FRONT, GL_AMBIENT, fMatAmbient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, fMatDiffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, fMatSpecular);
```

=

```
//glColor changes ambient reflection
glColor3f(a,b,c)
glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT,GL_AMBIENT);
```

```
GLfloat fMatAmbient[4] = {1,1,1,1};
GLfloat fMatDiffuse[4] = {a,b,c,1};
GLfloat fMatSpecular[4] = {1,1,1,1};

glMaterialfv(GL_FRONT, GL_AMBIENT, fMatAmbient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, fMatDiffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, fMatSpecular);
```

=

```
//glColor changes diffuse reflection
glColor3f(a,b,c)
glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT,GL_DIFFUSE);
```

```
GLfloat fMatAmbient[4] = {a,b,c,1};
GLfloat fMatDiffuse[4] = {a,b,c,1};
GLfloat fMatSpecular[4] = {1,1,1,1};

glMaterialfv(GL_FRONT, GL_AMBIENT, fMatAmbient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, fMatDiffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, fMatSpecular);
```

=

```
//glColor changes ambient/diffuse reflection
glColor3f(a,b,c)
glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT,GL_AMBIENT_AND_DIFFUSE);
```

```
GLfloat fMatAmbient[4] = {1,1,1,1};
GLfloat fMatDiffuse[4] = {1,1,1,1};
GLfloat fMatSpecular[4] = {a,b,c,1};

glMaterialfv(GL_FRONT, GL_AMBIENT, fMatAmbient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, fMatDiffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, fMatSpecular);
```

=

```
//glColor changes specular reflection
glColor3f(a,b,c)
glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT,GL_SPECULAR);
```

Conclusion: glMaterial vs glColorMaterial, continued …

- With glMaterial, we can be more specific as to exactly whatColor components we need for the scene.

  - If you need to change more than one material parameter, use glMaterial*()

- With glColorMaterial, where material tracks glColor, it is easier to use, with fewer lines of code to write.

  - After calling glColorMaterial(), you need to call glEnable() with GL_COLOR_MATERIAL as the parameter.

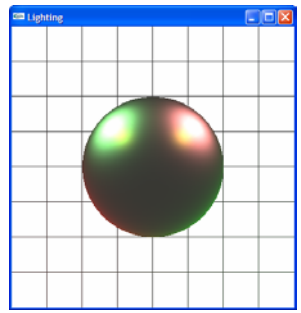
  - You should use glColorMaterial() whenever you need to change a single material parameter for most vertices in your scene.

Case 10 - Multiple spotlights

```
void Light1()
        {
        GLfloat lightPosition1[] = {3, 3, 3, 1};
        GLfloat spotLightDirection1[] = {-1, -1, -1 };
        GLfloat fLtAmbient1[4] = { 1, 0, 0, 1 };
        GLfloat fLtDiffuse1[4] = { 1, 1, 1, 1 };
        GLfloat fLtSpecular1[4] = {1,1,0,1};

        GLfloat fMatAmbient1[4] = {1,1,1,1};
        GLfloat fMatDiffuse1[4] = {1,1,1,1};
        GLfloat fMatSpecular1[4] = {1,1,0,1};
        GLfloat fMatShine1 = 50; // median

        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);

        glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);
        glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,spotLightDirection1);
        glLightf(GL_LIGHT0, GL_SPOT_EXPONENT,50);

        glLightfv(GL_LIGHT0, GL_POSITION, lightPosition1);
        glLightfv(GL_LIGHT0, GL_AMBIENT, fLtAmbient1);
        glLightfv(GL_LIGHT0, GL_DIFFUSE, fLtDiffuse1);
        glLightfv(GL_LIGHT0, GL_SPECULAR, fLtSpecular1);

        glMaterialfv(GL_FRONT, GL_AMBIENT, fMatAmbient1);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, fMatDiffuse1);
        glMaterialfv(GL_FRONT, GL_SPECULAR, fMatSpecular1);
        glMaterialf(GL_FRONT, GL_SHININESS, fMatShine1);
        }
```

```
void Light2()
        {
        GLfloat lightPosition2[] = {-3, 3, 3, 1};
        GLfloat spotLightDirection2[] = {1, -1, -1 };
        GLfloat fLtAmbient2[4] = { 0, 1, 0, 1 };
        GLfloat fLtDiffuse2[4] = { 1, 1, 1, 1 };
        GLfloat fLtSpecular2[4] = {1,1,0,1};

        GLfloat fMatAmbient2[4] = {1,1,1,1};
        GLfloat fMatDiffuse2[4] = {1,1,1,1};
        GLfloat fMatSpecular2[4] = {1,1,0,1};
        GLfloat fMatShine2 = 50; // median

        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT1);

        glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, 45.0);
        glLightfv(GL_LIGHT1,GL_SPOT_DIRECTION,spotLightDirection2);
        glLightf(GL_LIGHT1, GL_SPOT_EXPONENT,50);

        glLightfv(GL_LIGHT1, GL_POSITION, lightPosition2);
        glLightfv(GL_LIGHT1, GL_AMBIENT, fLtAmbient2);
        glLightfv(GL_LIGHT1, GL_DIFFUSE, fLtDiffuse2);
        glLightfv(GL_LIGHT1, GL_SPECULAR, fLtSpecular2);

        glMaterialfv(GL_FRONT, GL_AMBIENT, fMatAmbient2);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, fMatDiffuse2);
        glMaterialfv(GL_FRONT, GL_SPECULAR, fMatSpecular2);
        glMaterialf(GL_FRONT, GL_SHININESS, fMatShine2);
        }
```

Light1();
Light2();



Overview of how to create lighting in OpenGL

- Before Start:

  0. Determine the world coordinates

- Main Steps:

  1. Define Normal Vectors for Each Vertex of Every Object
  2. An object's normals determine its orientation relative to the light sources.
  3. Create, Position, and Enable One or More Light Sources
  4. Select a Lighting Model
  5. Define Material Properties for the Objects in the Scene

How to keep the light Stationary?

- **Keeping the Light Stationary**

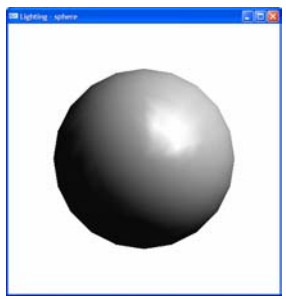
```
glViewport (0, 0, (GLsizei) w, (GLsizei) h);
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
if (w <= h)
    glOrtho (-1.5, 1.5, -1.5*h/w, 1.5*h/w, -
    10.0, 10.0);
else
    glOrtho (-1.5*w/h, 1.5*w/h, -1.5, 1.5, -
    10.0, 10.0);

glMatrixMode (GL_MODELVIEW);
glLoadIdentity();
…
/* later in init() */
GLfloat light_position[] = { 1.0, 1.0, 1.0,
    1.0 };
glLightfv(GL_LIGHT0, GL_POSITION, position);
/* NO other MODELVIEW transformation is set…*/
```

- Answer:
  - After setting the light position

(3) Select a Lighting Model

```
void init(void)
{
   GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
   GLfloat mat_shininess[] = { 50.0 };
   GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };

   SetBackground(1,1,1,0);
   glShadeModel (GL_SMOOTH);

   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
   glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
   glLightfv(GL_LIGHT0, // default color is white
     GL_POSITION,       // use light position for
     lighting
     light_position);      // light's position

   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glEnable(GL_DEPTH_TEST);
}
```

- This example uses the default settings for these two aspects of the light model.
  - an infinite viewer
    - Using a local viewer adds significantly to the complexity of the calculations that must be performed, because OpenGL must calculate the angle between the viewpoint and each object.
  - One sided lighting
    - Lighting calculations on the front facing only.
- If you need to specify Light Model, use glLightModel*() function.



Selecting a Lighting Model

- The OpenGL notion of a lighting model has three components:
  - The global ambient light intensity
  - Whether the viewpoint position is local to the scene or considered to be an infinite distance away
  - Whether lighting calculations should be performed differently for both the front and back faces of objects
- void glLightModel{if}(GLenum *pname*, *TYPE param*);
- void glLightModel{if}v(GLenum *pname*, *TYPE *param*);

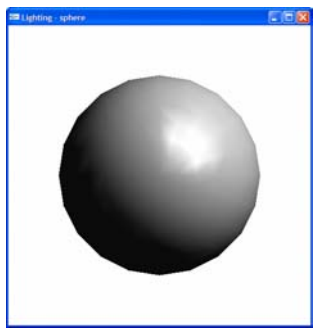
Default values for Lighting Model

Parameter Name	Default Value	Meaning
GL_LIGHT_MODEL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	ambient RGBA intensity of the entire scene
GL_LIGHT_MODEL_LOCAL_VIEWER	0.0 or GL_FALSE	how specular reflection angles are computed
GL_LIGHT_MODEL_TWO_SIDE	0.0 or GL_FALSE	choose between one-sided or two-sided lighting

Define Material Properties for the objects in the scene

```
void init(void)
{
   GLfloat mat_specular[] = { 1.0, 1.0, 1.0,
    1.0 };
   GLfloat mat_shininess[] = { 50.0 };
   GLfloat light_position[] = { 1.0, 1.0, 1.0,
    0.0 };

   SetBackground(1,1,1,0);
   glShadeModel (GL_SMOOTH);

   glMaterialfv(GL_FRONT, GL_SPECULAR,
    mat_specular);
   glMaterialfv(GL_FRONT, GL_SHININESS,
    mat_shininess);

   glLightfv(GL_LIGHT0, // default color is
    white
    GL_POSITION,       // use light position
    for lighting
    light_position);      // light's position

   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glEnable(GL_DEPTH_TEST);
}
```

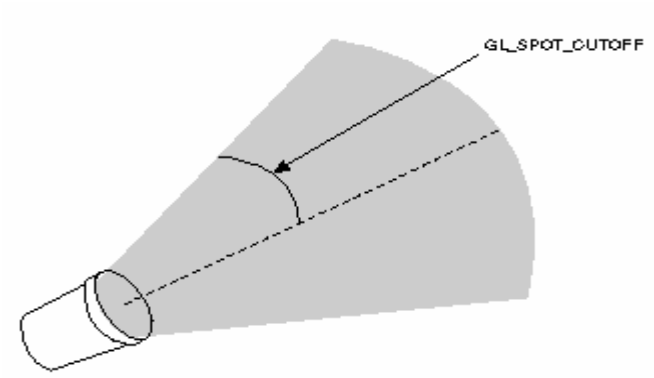
- You can specify a material's ambient, diffuse, and specular colors and how shiny it is.
- In this example, only these last two material properties—the specular material color and shininess—are explicitly specified (with the glMaterialfv() calls).



Spotlights

- You can have a positional light source act as a spotlight—that is, by restricting the shape of the light it emits to a cone.
- To create a spotlight, you need to determine the spread of the cone of light you desire, which is called **GL_SPOT_CUTOFF Parameter**.
- 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)
- 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).



Creating Light Sources

- **Default Values for pname Parameter of glLight*()**
- GL_AMBIENT (0.0, 0.0, 0.0, 1.0)
  - ambient RGBA intensity of lightGL_
- DIFFUSE (1.0, 1.0, 1.0, 1.0)
  - diffuse RGBA intensity of lightGL_
- 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 cutoff 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

- *pname*
- Specifies a light source parameter for *light*.
  - **GL_AMBIENT**, **GL_DIFFUSE**, **GL_SPECULAR**, **GL_POSITION**, **GL_SPOT_DIRECTION**, **GL_SPOT_EXPONENT**, **GL_SPOT_CUTOFF**, **GL_CONSTANT_ATTENUATION**, **GL_LINEAR_ATTENUATION**, **GL_QUADRATIC_ATTENUATION**

Creating Light Sources
Default Values for pname Parameter of glLight*()

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 cutoff 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