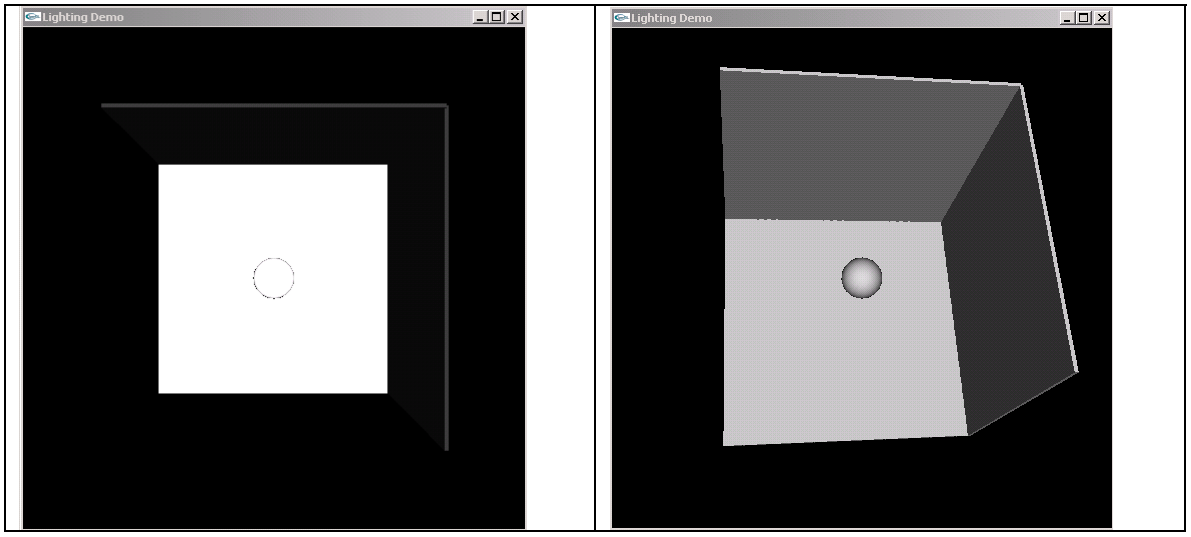
# Lighting in OpenGL – HandsOn Session

In this in-class session, we explore the OpenGL implementation of lighting, and some of the geometry behind illumination. Start by downloading the OpenGL source code **`spotlight.cpp`** from the class website (the Schedule section). This program renders a scene composed of three walls of a room, and allows the user to interactively rotate and translate the scene. We will extend this piece of code to light up the room.

## Activity 1 [Enable Lighting and Light Source]

a) Compile and run the program and observe the output. Then add two lines of code to the EnableLighting function, to enable lighting and the light source GL_LIGHT0. Recompile and run the code. Note that light cancels the effect of glColor3f.

```
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
```




b) The default shading model is GL_FLAT. This means that OpenGL calculates the intensity of the reflected light at the *first vertex* of each polygon, and uses it as the intensity of every point inside the polygon. Specify a smooth shading model using

```
glShadeModel(GL_SMOOTH);
```

This enables OpenGL to calculate the intensity of the reflected light at *each polygon vertex*, and to *interpolate* across the polygon to get the intensity at each point on the polygon.

c) The lighting still doesn't work properly for us. This is because OpenGL uses unit normals at each vertex to compute the vetex color. GLUT shapes already have normals set. We need to tell OpenGL to preserve the unit normals (and not let them get affected by transformations):

```
glEnable(GL_NORMALIZE);
```

Notice the difference in appearance between the two scenes above.

## **Activity 2** [Set Light Properties]

a) OpenGL supports up to eight lights. Light 0 is unique in having a default diffuse and specular setting of fully bright white (1,1,1,1). All other lights have a default settings of totally dark (0,0,0,1). Each of the eight lights has a default position of (0,0,1,0), which would effectively place it as a directional light, facing forward, from just behind the camera.

Comment out the line that enables light 0 and set the following light properties for light 1:

```
GLfloat light1_diffuse[]     = {1.0, 1.0, 1.0, 1.0};  /* bright white */
GLfloat light1_specular[]    = {1.0, 1.0, 1.0, 1.0};

glLightfv(GL_LIGHT1, GL_DIFFUSE, light1_diffuse);
glLightfv(GL_LIGHT1, GL_SPECULAR, light1_specular);
```

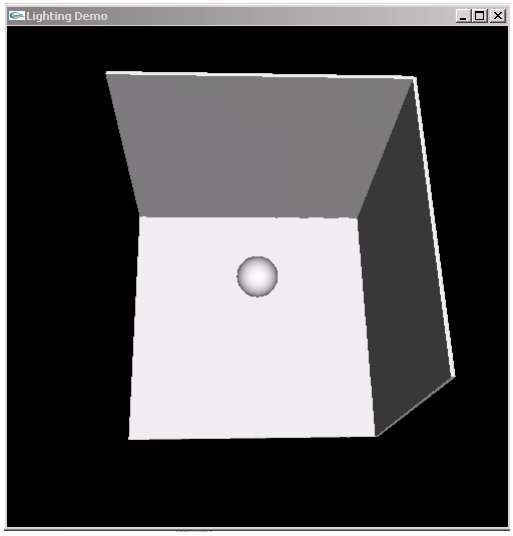
These lines of code should go in the EnableLighting function. Do not forget to enable light 1. The scene should appear as if light 0 were enabled.

b) Add a soft white ambient component to light 1 from the previous activity:

```
GLfloat light1_ambient[]     = {0.8, 0.8, 0.8, 1.0};  /* soft white */

glLightfv(GL_LIGHT1, GL_AMBIENT, light1_ambient);
```

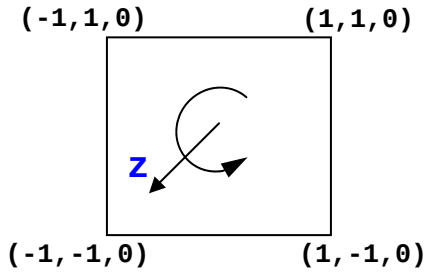
This component adds some level of brightness to the scene.



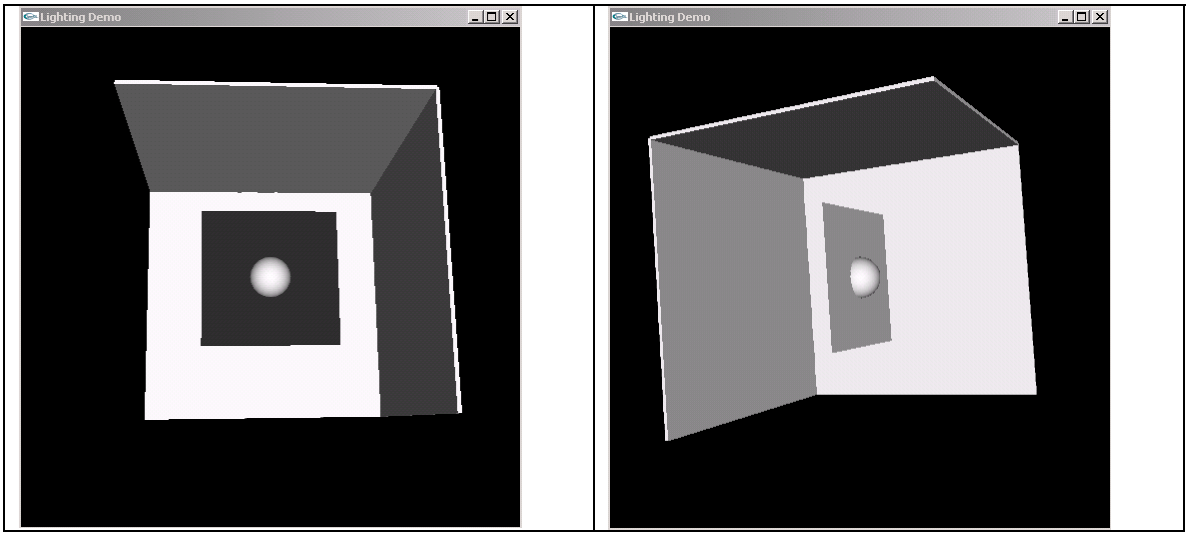
## Activity 3 [Specify Normals]

a) Add a vertical unit square to the scene, centered at the origin and facing the light:

```
glBegin(GL_POLYGON);
      glVertex3f(1, 1, 0);
      glVertex3f(-1, 1, 0);
      glVertex3f(-1, -1, 0);
      glVertex3f(1, -1, 0);
glEnd();
```




The square shape is not lit up. Explain why.




b) Specify a unit normal for each vertex:

```
glNormal3f(0, 0, 1);
```

This enables OpenGL to calculate the color intensity at vertices of the square, and interpolate across the square. The square has the same color as the back wall (see right image above).

For predefined OpenGL shapes (cube, sphere, etc.), vertex normals are already set.

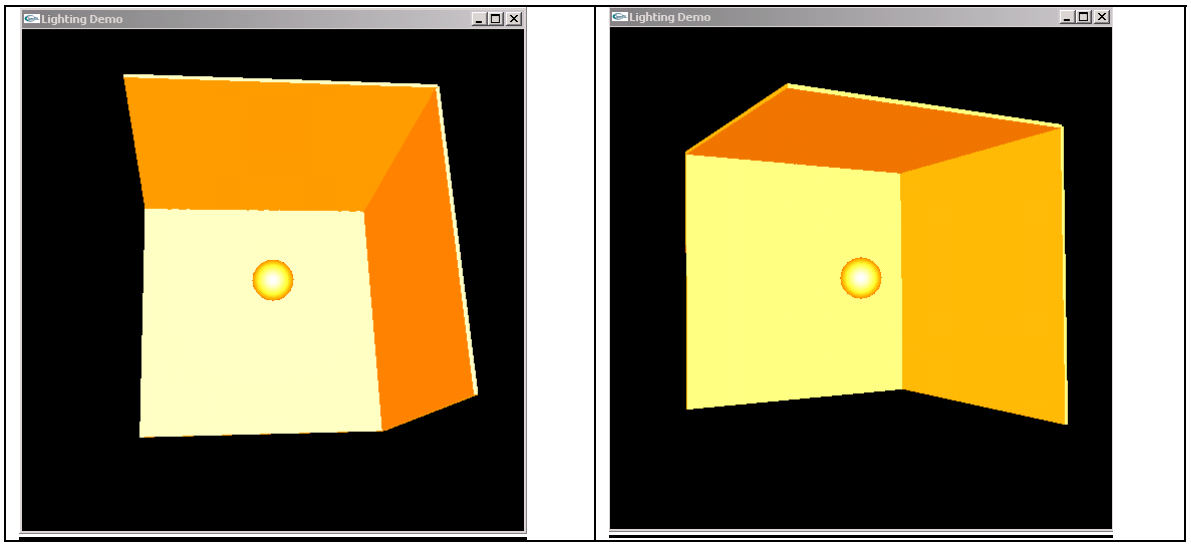
## Activity 4 [Specify Material Color and Shineness]

When light is enabled, glColor3f has no effect. Typically, material properties indirectly define the color of an object. Add the following lines of code to the EnableLighting function:

```
GLfloat mat_specular[]   = {1.0, 1.0, 1.0, 1.0};      /* bright white */
GLfloat mat_diffuse[]    = {1.0, 0.5, 0.0, 1.0};      /* orange color */
GLfloat mat_ambient[]    = {1.0, 0.5, 0.0, 1.0};      /* same as diffuse */
GLfloat mat_shininess    =  5.0;

glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);
```

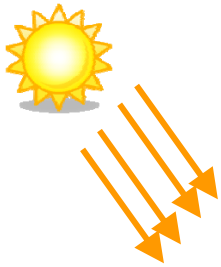
Eliminate the center square tested in the previous activity, recompile and run.



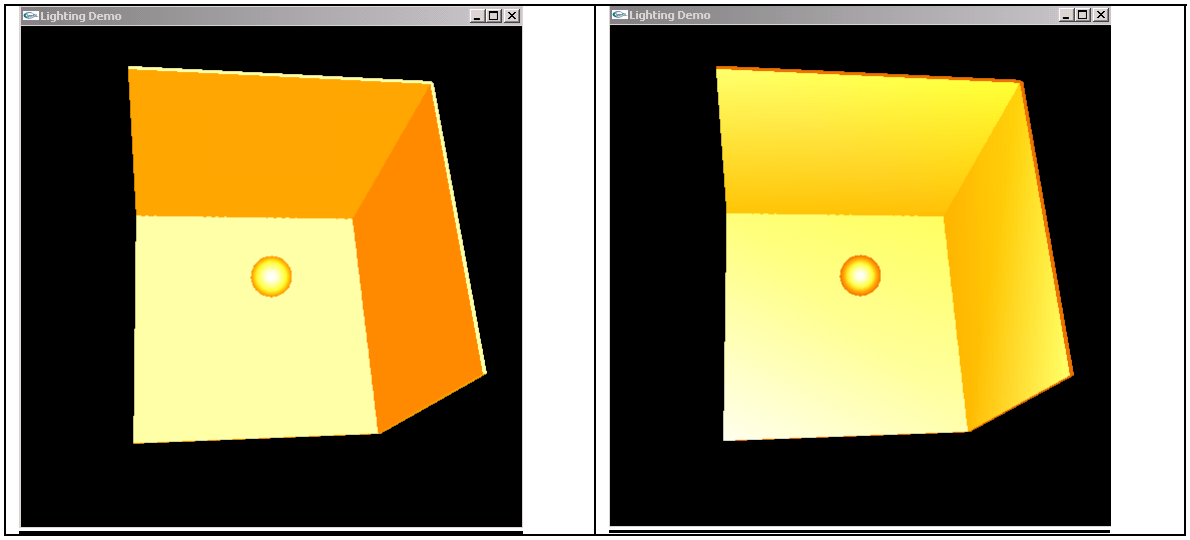

The scene should appear as above (same scene, different orientations).

## Activity 5 [Positional Light Source]

All eight OpenGL lights have a default position of (0,0,1,0). The fourth argument distinguishes between a *directional* light source (0, light at infinity) and *positional* light source (1). So far we have worked with the default directional parameter.



a) Identify the type of light (positional, directional) in the two images below.



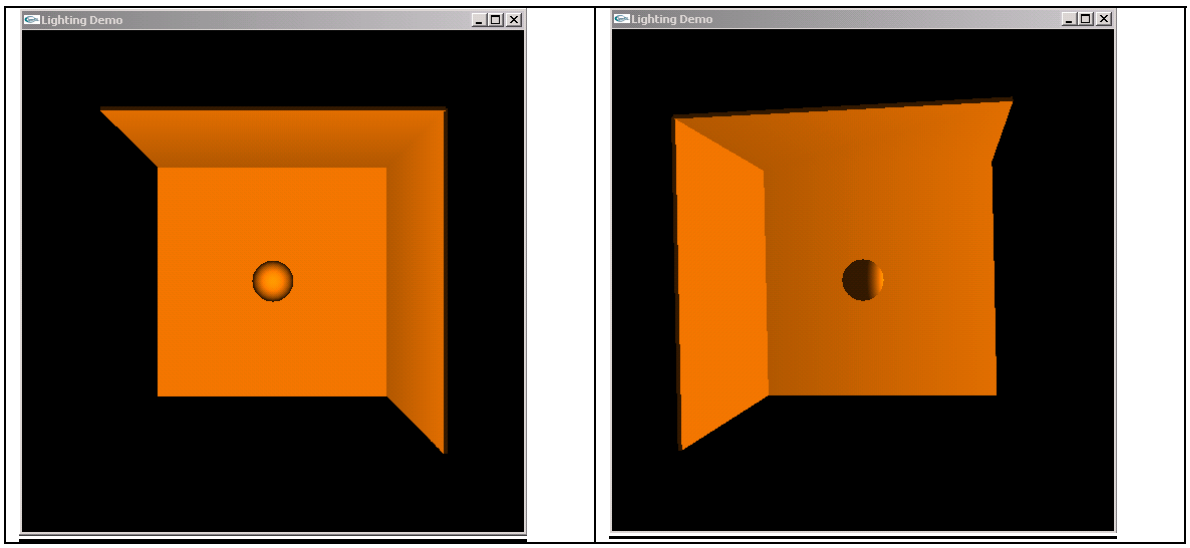

How can you tell the difference? Which lighting method is more expensive?

b) Turn light 1 into a positional light using

```
GLfloat light1_position[] = {0.0, 0.0, 1.0, 1.0};
glLightfv(GL_LIGHT1, GL_POSITION, light1_position);
```

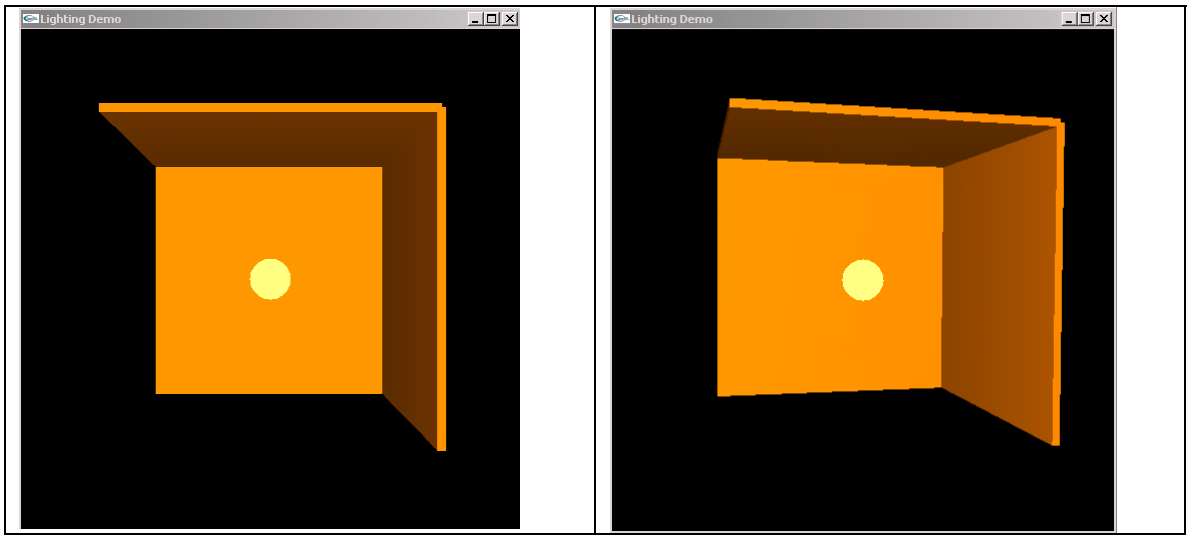
c) The light position gets multiplied by the MODELVIEW matrix to determine the actual position in the scene (think of the light as just another object in the scene).

**Try this out**: Eliminate (comment out) the ambient and specular components of light 1, leaving just the diffuse component in place. Reposition the light source (by calling glLightfv(GL_LIGHT1, GL_POSITION, …) with each rotation/translation of the scene. The light should move with the scene as shown below.



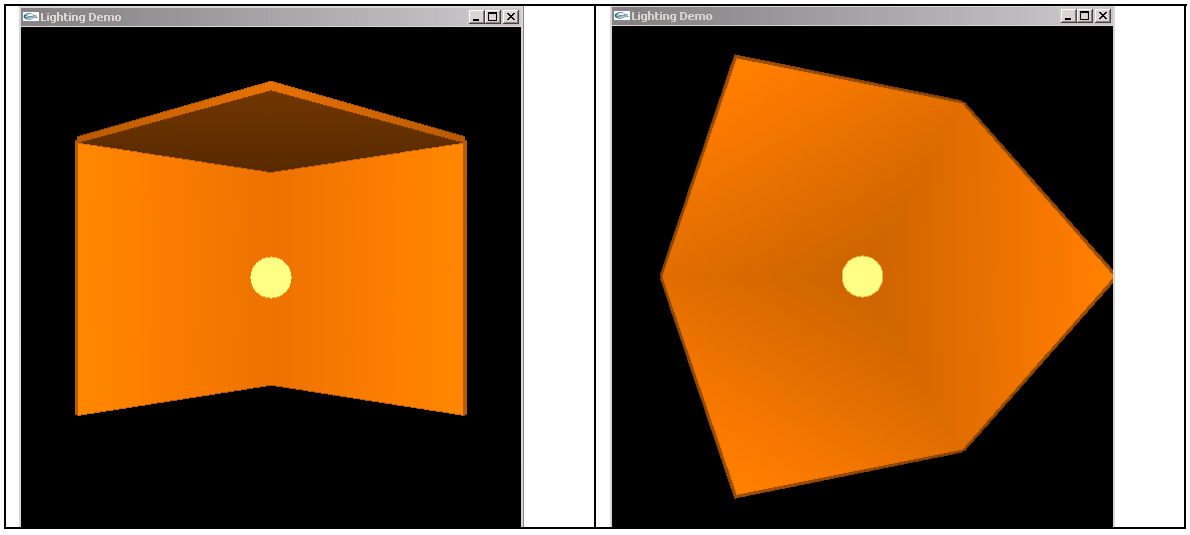
## Exercise 1 [The Geometry behind Diffuse Lighting]

The light in the scene below has a diffuse component only, and is oriented along the z-axis. The room is centered at the origin of the coordinate system, and its three walls are initially orthogonal to the three coordinate axis, as in the left of the image below. The scene uses smooth polygonal shading.




a) The room has been rotated clockwise about the x-axis by angle θ1, and counterclockwise about the y-axis by angle θ2, as in the right of the image above. Determine the relationship between θ1 and θ2, based on the intensity of the color on the three walls.

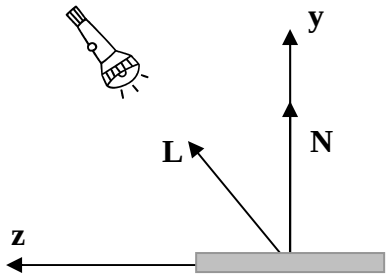
b) In the left image below, the corner between the back wall and the right wall is indistinguishable. What rotation has been applied to the scene relative to the original position? Specify the rotation angle and the rotation axis.




What about the image on the right?

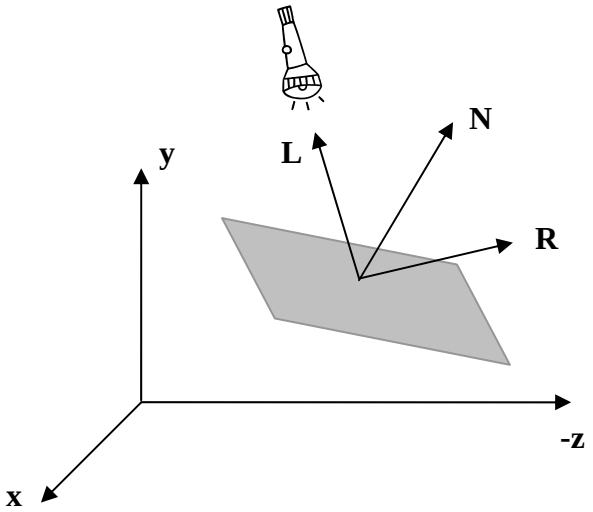
## Exercise 2 [The Geometry behind Specular Lighting]

a) [**Warmup**] Imagine you have a specular surface positioned so that its center passes through the origin and lies in the X-Z plane. There is a flashlight positioned at (0.0, 4.0, 3.0) and is oriented so that it points at the origin. A cross-sectional diagram is shown below.



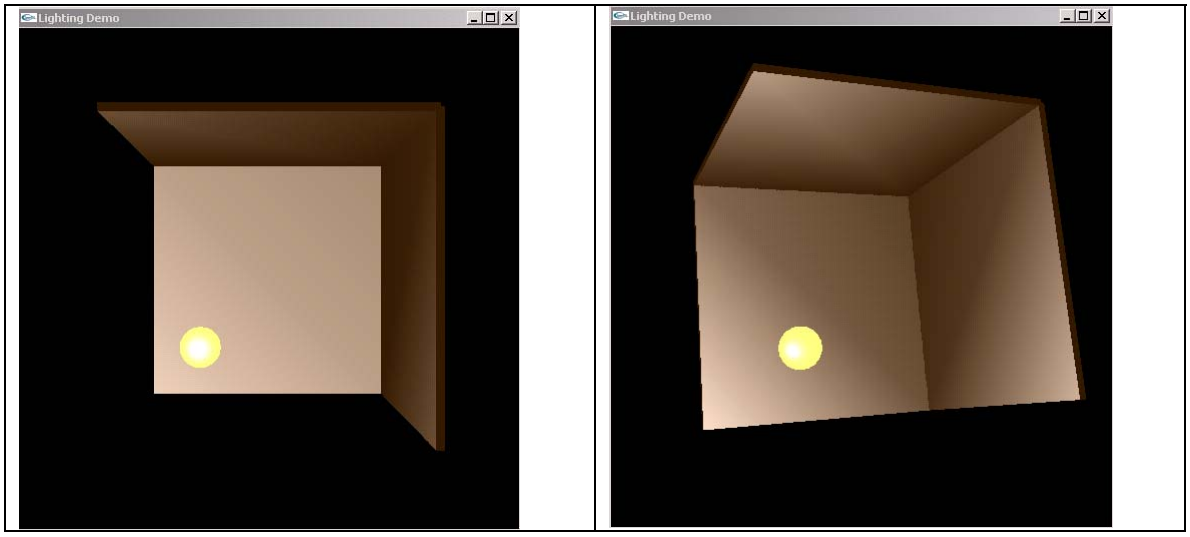

Suppose that you want to position a camera in the scene so that it is looking directly at the origin from a distance of 10 and is receiving maximum reflection? Where would you place the camera? Draw your answer first on the diagram and then give the (x, y, x) coordinates for the position of the camera.

b) [**Challenge**] When a ray of light projects off of a surface, the angle between the surface normal N and the direction of light L is equal to the angle between N and the direction of the reflection R.




Notice that there are an infinite number of rays whose angle with N is the same as the angle between N and L. Of this infinite number of rays, the reflected light travels in the same plane as L and N. Derive a formula for R in terms of L and N. You can assume that L and N are unit vectors.

c) The light in the scene below has a white specular component only. The light is positioned at (-1, -1, 1) in the scene. The images below show two snapshots of the scene from different viewpoints. Give a vector indicating the direction of the viewer that sees the brightest spot in the center (0, 0, -2) of the back wall.



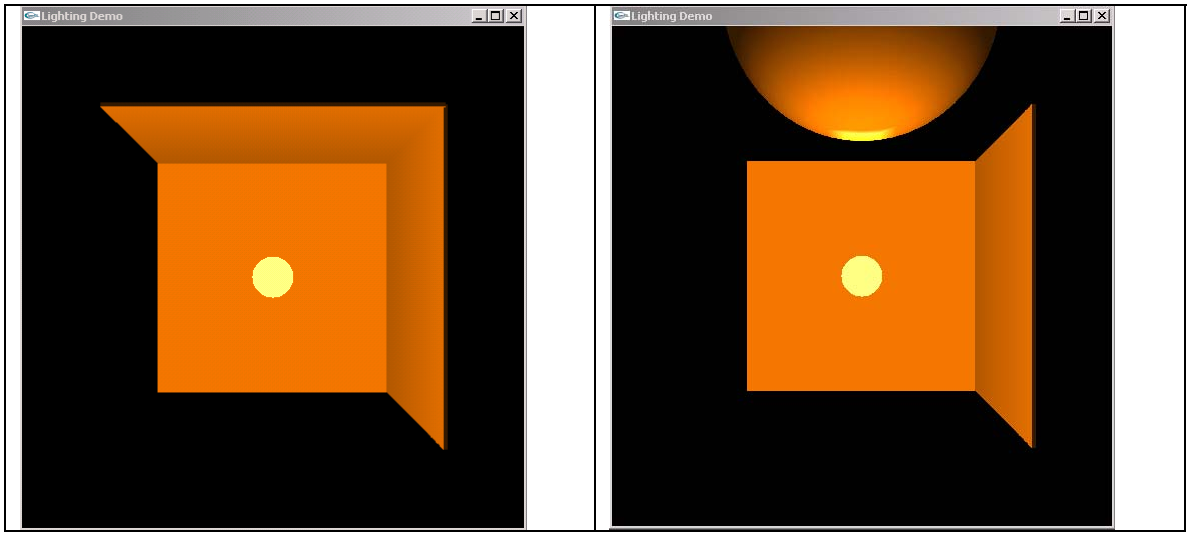
## **Activity 6** [Spotlights]

OpenGL supports spotlights, which are *positional* light sources that limit the light to a cone-shaped region of space. To create a spotlight, you need to specify the angle between the axis of the cone and a ray along the edge of the cone, using the GL_SPOT_CUTOFF parameter. You also need to specify the spotlight's direction, which determines the axis of the cone of light. You can also set the GL_SPOT_EXPONENT parameter, which by default is zero, to control how concentrated the light is. The light's intensity is highest in the center of the cone.

a) Comment out the ambient and specular components of light 1, leaving just a diffuse component for the light. Add a spotlight to the scene, pointing to the ceiling:

```
GLfloat dirVector [] = {0.0, 1.0, 0.0};
glLightfv(GL_LIGHT2, GL_SPOT_DIRECTION, dirVector);
glLightf(GL_LIGHT2, GL_SPOT_CUTOFF, 10.0);
glLightf(GL_LIGHT2, GL_SPOT_EXPONENT, 5);
/* set white diffuse and specular components for light 2 */
/* position light 2 at the origin */
```

Is the spotlight visible? Explain what happens.




b) Replace the room ceiling with a sphere of about the same size.

Is the spotlight visible? Explain what happens.

c) Add to your code the function **`tiledRoom`** below, which constructs the three room walls out of very thin cylinders (this is so that we don't have to write the code for drawing tiled rectangular walls ourselves ☺). This code should enable your spotlight to work properly on all walls. Define and initialize variables used by this code, as appropriate.

```
void tiledRoom()
{
   /* ceiling */
   glPushMatrix();
   glTranslatef(0,ROOM_SIZE/2,-ROOM_SIZE/2);
   glScalef(ROOM_SIZE/2, WALL_THICKNESS, ROOM_SIZE);
   gluCylinder(topRoomObj, 1, 1, 1, 20, 20);
   glPopMatrix();

   /* right wall */
   glPushMatrix();
   glTranslatef(ROOM_SIZE/2,0,-ROOM_SIZE/2);
   glScalef(WALL_THICKNESS, ROOM_SIZE/2, ROOM_SIZE);
   gluCylinder(rightRoomObj, 1, 1, 1, 20, 20);
   glPopMatrix();

   /* back wall */
   glPushMatrix();
   glTranslatef(0,-ROOM_SIZE/2,-ROOM_SIZE/2);
   glRotatef(-90, 1, 0, 0);
   glScalef(ROOM_SIZE/2, WALL_THICKNESS, ROOM_SIZE);
   gluCylinder(backRoomObj, 1, 1, 1, 20, 20);
   glPopMatrix();
}
```

d) Change the light direction to point in turn to each of the three walls.