```
/* lightlab2.c --- Experimentation with lighting in OpenGL.
 * Tom Kelliher 4/22/2003.
 *
 * This version introduces a moving light and a spotlight anchored
 * to the viewer.  Use l to turn the spot off and L to turn it back
 * on.
 *
 * Movement is via x/X, y/Y, and z/Z keys.  Use space bar to exit.
 * Mouse buttons rotate the right cube.
 *
 * Things to try:
 *
 *    1) Change the lighting so that the background is a dark green
 *       rather than the slate blue.
 *
 *    2) Give each object its own material properties.  Make the
 *       rotating light look more realistic.  (Let it self-emanate.)
 *
 *    3) Add the ability to turn the rotating light on and off.  Its
 *       visible representation should disappear when it's off.
 *
 *    4) As you move in close to the objects, you'll notice the
 *       movable spotlight doesn't react realistically.  Correct
 *       this behavior.
 */


#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>


// Base of the display lists.
GLuint lists;


// The right cube can be rotated.  Whoopee.
static GLfloat theta[] = {45.0,0.0,0.0};
static GLint axis = 2;

/* initial viewer location */
static GLdouble viewer[]= {20.0, 6.0, 6.0};


// Light 0 is the moving light.  Light 1 is the spotlight that moves
// with us.  Note that these are positional lights.
GLfloat light0_position[] = { 0.0, 0.0, 5.0, 1.0 };
GLfloat light1_position[] = { 0.0, 0.0, 0.0, 1.0 };

// Angle of the moving light.
GLint light_angle = 0;


// Paint the scene.
void display(void)
{

   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

   /* Update viewer position in modelview matrix */
   glLoadIdentity();

   gluLookAt(viewer[0],viewer[1],viewer[2], 0.0, 0.0, 0.0,
```

```
   64:              0.0, 1.0, 0.0);
   65:
   66:    // Position the moving light.  It will be positioned in world
   67:    // coordinates.
   68:    glPushMatrix();
   69:    glRotatef(light_angle, 1.0, 0.0, 0.0);
   70:    glLightfv(GL_LIGHT0, GL_POSITION, light0_position);
   71:
   72:    // Render a solid sphere indicating the light's position.  Notice
   73:    // how it's not lit.
   74:    glTranslatef(light0_position[0], light0_position[1],
   75:            light0_position[2]);
   76:    glDisable(GL_LIGHTING);
   77:    glCallList(lists);
   78:    glEnable(GL_LIGHTING);
   79:    glPopMatrix();
   80:
   81:    glPushMatrix();
   82:
   83:    // Left cube.
   84:    glTranslatef(-2.0, 0.0, 0.0);
   85:    glCallList(lists + 1);
   86:
   87:    // Sphere centered at origin.
   88:    glTranslatef(2.0, 0.0, 0.0);
   89:    glCallList(lists + 2);
   90:
   91:    // Right cube.
   92:    glTranslatef(2.0, 0.0, 0.0);
   93:    // Rotate cube.
   94:    glRotatef(theta[0], 1.0, 0.0, 0.0);
   95:    glRotatef(theta[1], 0.0, 1.0, 0.0);
   96:    glRotatef(theta[2], 0.0, 0.0, 1.0);
   97:    glCallList(lists + 1);
   98:
   99:    // Now that I'm thoroughly confused, let's re-establish the
  100:    // origin.
  101:    glPopMatrix();
  102:
  103:    // The cone.
  104:    // Up y-axis by 1.
  105:    glTranslatef(0.0, 1.0, 0.0);
  106:    // Rotate z-axis up to y-axis.
  107:    glRotatef(-90.0, 1.0, 0.0, 0.0);
  108:    glCallList(lists + 3);
  109:
  110:    // The torus.
  111:    // Translate up the cone.
  112:    glTranslatef(0.0, 0.0, 1.0);
  113:    glCallList(lists + 4);
  114:
  115:    glutSwapBuffers();
  116: }
  117:
  118:
  119: // Rotate the right cube.  Looks kinda neat when it cuts into
  120: // the sphere.
  121: void mouse(int btn, int state, int x, int y)
  122: {
  123:         if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN) axis = 0;
  124:         if(btn==GLUT_MIDDLE_BUTTON && state == GLUT_DOWN) axis = 1;
  125:         if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN) axis = 2;
  126:         theta[axis] += 2.0;
```

```
  127:         if( theta[axis] > 360.0 ) theta[axis] -= 360.0;
  128:         glutPostRedisplay();
  129: }
  130:
  131:
  132: void keys(unsigned char key, int x, int y)
  133: {
  134:
  135: /* Use x, X, y, Y, z, and Z keys to move viewer */
  136:
  137:    if(key == 'x') viewer[0]-= 1.0;
  138:    if(key == 'X') viewer[0]+= 1.0;
  139:    if(key == 'y') viewer[1]-= 1.0;
  140:    if(key == 'Y') viewer[1]+= 1.0;
  141:    if(key == 'z') viewer[2]-= 1.0;
  142:    if(key == 'Z') viewer[2]+= 1.0;
  143:    if(key == 'l') glDisable(GL_LIGHT1);
  144:    if(key == 'L') glEnable(GL_LIGHT1);
  145:    if(key == ' ') exit(0);
  146:
  147:    printf("v[x]: %f, v[y]: %f, v[z]: %f.\n",
  148:           viewer[0], viewer[1], viewer[2]);
  149:
  150:    glutPostRedisplay();
  151: }
  152:
  153:
  154: void myReshape(int w, int h)
  155: {
  156:    glViewport(0, 0, w, h);
  157:
  158:    /* Use a perspective view */
  159:
  160:    glMatrixMode(GL_PROJECTION);
  161:    glLoadIdentity();
  162:    gluPerspective(35.0, (GLfloat) w/h, 1.0, 100.0);
  163:    glMatrixMode(GL_MODELVIEW);
  164: }
  165:
  166:
  167: // Rotate the moving light.
  168: void idle(void)
  169: {
  170:         light_angle = (light_angle + 1) % 360;
  171:         glutPostRedisplay();
  172: }
  173:
  174:
  175: // Set material and light values.
  176: void init(void)
  177: {
  178:    // Default values for material and light properties.
  179:    GLfloat mat_ambient[] = { 0.5, 0.0, 0.0, 1.0 };
  180:    GLfloat mat_diffuse[] = { 1.0, 0.5, 0.5, 1.0 };
  181:    GLfloat mat_specular[] = { 1.0, 0.5, 0.5, 1.0 };
  182:    GLfloat mat_shininess[] = { 50.0 };
  183:
  184:    // No ambient light whatsoever.
  185:    GLfloat light_ambient[] = { 0.0, 0.0, 0.0, 1.0 };
  186:    GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
  187:    GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };
  188:
  189:    GLfloat light1_ambient[] = { 0.0, 0.0, 0.0, 1.0 };
```

```
  190:    GLfloat light1_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
  191:    GLfloat light1_specular[] = { 1.0, 1.0, 1.0, 1.0 };
  192:
  193:    glClearColor (0.2f, 0.3f, 0.4f, 1.0);
  194:    glShadeModel (GL_SMOOTH);
  195:
  196:    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
  197:    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
  198:    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
  199:    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
  200:
  201:    // Note use of linear attenuation values, the spot cutoff
  202:    // half-angle, and the spot exponent (intensity dropoff
  203:    // within the cone).
  204:
  205:    glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
  206:    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
  207:    glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
  208:    glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 0.1f);
  209:
  210:    // Specifying the light position here causes the position to be
  211:    // interpreted in eye coordinates.  The default direction of the
  212:    // spotlight is (0, 0, -1), which is also interpreted in eye
  213:    // coordinates in this case.
  214:
  215:    glLoadIdentity();
  216:    glLightfv(GL_LIGHT1, GL_POSITION, light1_position);
  217:
  218:    glLightfv(GL_LIGHT1, GL_AMBIENT, light1_ambient);
  219:    glLightfv(GL_LIGHT1, GL_DIFFUSE, light1_diffuse);
  220:    glLightfv(GL_LIGHT1, GL_SPECULAR, light1_specular);
  221:    glLightf(GL_LIGHT1, GL_LINEAR_ATTENUATION, 0.2f);
  222:    glLighti(GL_LIGHT1, GL_SPOT_CUTOFF, 5);
  223:    glLightf(GL_LIGHT1, GL_SPOT_EXPONENT, 0.2f);
  224:
  225:    // Declare the viewer to be local.
  226:
  227:    glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);
  228:    glEnable(GL_LIGHTING);
  229:    glEnable(GL_LIGHT0);
  230:    glEnable(GL_LIGHT1);
  231:    glEnable(GL_DEPTH_TEST);
  232: }
  233:
  234:
  235: void
  236: main(int argc, char **argv)
  237: {
  238:    setvbuf(stdout, (char *)NULL, _IONBF, 0);
  239:    setvbuf(stderr, (char *)NULL, _IONBF, 0);
  240:
  241:    glutInit(&argc, argv);
  242:    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
  243:    glutInitWindowSize(500, 500);
  244:    glutCreateWindow("Light Lab Two");
  245:    glutReshapeFunc(myReshape);
  246:    glutDisplayFunc(display);
  247:    glutMouseFunc(mouse);
  248:    glutKeyboardFunc(keys);
  249:    glutIdleFunc(idle);
  250:    init();
  251:
  252:    // Get five list handles.
```

```
  253:    lists = glGenLists(5);
  254:
  255:    // For the moving light.
  256:    glNewList(lists, GL_COMPILE);
  257:       glutSolidCube(0.1);
  258:    glEndList();
  259:
  260:    // For the two cubes.
  261:    glNewList(lists + 1, GL_COMPILE);
  262:       glutSolidCube(2.0);
  263:    glEndList();
  264:
  265:    // The sphere at the origin.
  266:    glNewList(lists + 2, GL_COMPILE);
  267:       glutSolidSphere(1.0, 32, 32);
  268:    glEndList();
  269:
  270:    // The cone.
  271:    glNewList(lists + 3, GL_COMPILE);
  272:    glutSolidCone(1.0, 2.0, 32, 32);
  273:    glEndList();
  274:
  275:    // The torus.
  276:    glNewList(lists + 4, GL_COMPILE);
  277:    glutSolidTorus(0.25, 1.0, 32, 32);
  278:    glEndList();
  279:
  280:    glutMainLoop();
  281: }
```