```
/* lightlab.c --- Experimentation with lighting in OpenGL.
 * Tom Kelliher 4/19/2001.
 *
 * 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) Run program as is.  Move around.  Note lack of shadows and
 *       differences between left and right cubes.
 *    2) Comment out the DIRTY define and verify that the cubes look
 *       the same.  Un-comment out the define when finished.
 *    3) In display(), swap the gluLookAt() and glLightfv() calls.
 *       Can see any effect of doing this?  Re-swap the calls when
 *       finished.
 *    4) Try the light source at a few different positions.  Can you
 *       see any difference as a directional vs. positional source?
 *    5) Un-comment out the GET_VALUES define and expeiment with
 *       material and light values.
 *    6) Try to find material and light values that provide
 *       realistic looking results.  (Hint: you may want to
 *       augment the code by changing the constant, linear, and
 *       quadratic distance attenuation coefficients.  See the man
 *       page for glLight.)
 *    7) Add another light source.
 *    8) Augment the code so that each light and each object has
 *       independent properties.
 */

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


// Vertices for right cube.
GLfloat vertices[][3] = {{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},
{1.0,1.0,-1.0}, {-1.0,1.0,-1.0}, {-1.0,-1.0,1.0},
{1.0,-1.0,1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0}};


// Un-comment to be queried for light and material properties.
//#define GET_VALUES

// Un-comment to use "quick and dirty normals" for right cube.
#define DIRTY


#ifdef DIRTY

// These are the "quick and dirty" normals for the right cube,
// 1 per vertex.
GLfloat normals[][3] = {{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},
{1.0,1.0,-1.0}, {-1.0,1.0,-1.0}, {-1.0,-1.0,1.0},
{1.0,-1.0,1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0}};

#else

// These are the real normals for the right cube, 1 per face.
GLfloat normals[][3] = {{0.0,0.0,-1.0},{0.0,1.0,0.0},
{-1.0,0.0,0.0},{1.0,0.0,0.0},{0.0,0.0,1.0},{0.0,-1.0,0.0}};

#endif


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

/* initial viewer location */
static GLdouble viewer[]= {0.0, 0.0, 5.0};


// Guess what this one is.
// Fourth parameter: 0.0 --- directional light source
//                   1.0 --- positional light source.
GLfloat light_position[] = { 0.0, 5.0, 5.0, 0.0 };


#ifdef DIRTY

// Draw a polygon with its normals.
void polygon(int a, int b, int c , int d)
{
   glBegin(GL_POLYGON);
      glNormal3fv(normals[a]);
      glVertex3fv(vertices[a]);
      glNormal3fv(normals[b]);
      glVertex3fv(vertices[b]);
      glNormal3fv(normals[c]);
      glVertex3fv(vertices[c]);
      glNormal3fv(normals[d]);
      glVertex3fv(vertices[d]);
   glEnd();
}

// Draw a cube with lighting normals.
void colorcube()
{
   polygon(0,3,2,1);
   polygon(2,3,7,6);
   polygon(0,4,7,3);
   polygon(1,2,6,5);
   polygon(4,5,6,7);
   polygon(0,1,5,4);
}

#else
// !DIRTY

void polygon(int a, int b, int c , int d, GLfloat normal[])
{
   glBegin(GL_POLYGON);
      glNormal3fv(normal);
      glVertex3fv(vertices[a]);
      glVertex3fv(vertices[b]);
      glVertex3fv(vertices[c]);
      glVertex3fv(vertices[d]);
   glEnd();
}

void colorcube()
```

```
{
        polygon(0,3,2,1,normals[0]);
        polygon(2,3,7,6,normals[1]);
        polygon(0,4,7,3,normals[2]);
        polygon(1,2,6,5,normals[3]);
        polygon(4,5,6,7,normals[4]);
        polygon(0,1,5,4,normals[5]);
}

#endif


// 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, 0.0, 1.0, 0.0);

   // Position the light.
   glLightfv(GL_LIGHT0, GL_POSITION, light_position);

   glPushMatrix();

   // Left cube with "correct normals."
   glTranslatef(-2.0, 0.0, 0.0);
   glutSolidCube(2.0);

   // Sphere centered at origin.
   glTranslatef(2.0, 0.0, 0.0);
   glutSolidSphere(1.0, 32, 32);

   // Right cube with normals determined by DIRTY.
   glTranslatef(2.0, 0.0, 0.0);
   // Rotate cube.
   glRotatef(theta[0], 1.0, 0.0, 0.0);
   glRotatef(theta[1], 0.0, 1.0, 0.0);
   glRotatef(theta[2], 0.0, 0.0, 1.0);
   colorcube();

   /* Now that I'm thoroughly confused, let's re-establish
    * the origin.
    */
   glPopMatrix();

   // Up y-axis by 1.
   glTranslatef(0.0, 1.0, 0.0);
   // Rotate z-axis up to y-axis.
   glRotatef(-90.0, 1.0, 0.0, 0.0);
   glutSolidCone(1.0, 2.0, 32, 32);
   // Translate up the cone.
   glTranslatef(0.0, 0.0, 1.0);
   glutSolidTorus(0.25, 1.0, 32, 32);

   glFlush();
   glutSwapBuffers();
}


/* Rotate the right cube.  Looks kinda neat when it cuts
 * into the sphere.
 */

void mouse(int btn, int state, int x, int y)
{
        if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN) axis = 0;
        if(btn==GLUT_MIDDLE_BUTTON && state == GLUT_DOWN) axis = 1;
        if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN) axis = 2;
        theta[axis] += 2.0;
        if( theta[axis] > 360.0 ) theta[axis] -= 360.0;
        glutPostRedisplay();
}


void keys(unsigned char key, int x, int y)
{

/* Use x, X, y, Y, z, and Z keys to move viewer */

   if(key == 'x') viewer[0]-= 1.0;
   if(key == 'X') viewer[0]+= 1.0;
   if(key == 'y') viewer[1]-= 1.0;
   if(key == 'Y') viewer[1]+= 1.0;
   if(key == 'z') viewer[2]-= 1.0;
   if(key == 'Z') viewer[2]+= 1.0;
   if(key == ' ') exit(0);

   printf("v[x]: %f, v[y]: %f, v[z]: %f.\n",
          viewer[0], viewer[1], viewer[2]);

   glutPostRedisplay();
}


void myReshape(int w, int h)
{
   glViewport(0, 0, w, h);

   /* Use a perspective view */

   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(35.0, (GLfloat) w/h, 1.0, 100.0);
   glMatrixMode(GL_MODELVIEW);
}


// Set material and light values.
void init(void)
{
   // Default values for material and light properties.
   GLfloat mat_ambient[] = { 0.5, 0.0, 0.0, 1.0 };
   GLfloat mat_diffuse[] = { 1.0, 0.5, 0.5, 1.0 };
   GLfloat mat_specular[] = { 1.0, 0.5, 0.5, 1.0 };
   GLfloat mat_shininess[] = { 50.0 };

```

```
241:    GLfloat light_ambient[] = { 1.0, 1.0, 1.0, 1.0 };
242:    GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
243:    GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };
244:
245: #ifdef GET_VALUES
246:    printf("Enter RGB float values for:\n");
247:    printf("   Material ambient properties: ");
248:    scanf("%f %f %f", mat_ambient, mat_ambient+1, mat_ambient+2);
249:    printf("   Material diffuse properties: ");
250:    scanf("%f %f %f", mat_diffuse, mat_diffuse+1, mat_diffuse+2);
251:    printf("   Material specular properties: ");
252:    scanf("%f %f %f", mat_specular, mat_specular+1, mat_specular+2);
253:    printf("\nEnter shininess float value: ");
254:    scanf("%f", mat_shininess);
255:
256:    printf("\nEnter RGB float values for:\n");
257:    printf("   Light ambient properties: ");
258:    scanf("%f %f %f", light_ambient, light_ambient+1,
259:          light_ambient+2);
260:    printf("   Light diffuse properties: ");
261:    scanf("%f %f %f", light_diffuse, light_diffuse+1,
262:          light_diffuse+2);
263:    printf("   Light specular properties: ");
264:    scanf("%f %f %f", light_specular, light_specular+1,
265:          light_specular+2);
266: #endif
267:
268:    glClearColor (0.2f, 0.3f, 0.4f, 1.0);
269:    glShadeModel (GL_SMOOTH);
270:
271:    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
272:    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
273:    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
274:    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
275:
276:    glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
277:    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
278:    glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
279:
280:    glEnable(GL_LIGHTING);
281:    glEnable(GL_LIGHT0);
282:    glEnable(GL_DEPTH_TEST);
283:
284:    /* The "quick and dirty" normals haven't been normalized. */
285:    glEnable(GL_NORMALIZE);
286: }
287:
288:
289: void
290: main(int argc, char **argv)
291: {
292:    glutInit(&argc, argv);
293:    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
294:    glutInitWindowSize(500, 500);
295:    glutCreateWindow("Light Lab");
296:    glutReshapeFunc(myReshape);
297:    glutDisplayFunc(display);
298:    glutMouseFunc(mouse);
299:    glutKeyboardFunc(keys);
300:    init();
301:    glutMainLoop();
302: }
```