

```
1: /* lightlab2.c --- Experimentation with lighting in OpenGL.
2: * Tom Kelliher 4/22/2003.
3: *
4: * This version introduces a moving light and a spotlight anchored
5: * to the viewer. Use l to turn the spot off and L to turn it back
6: * on.
7: *
8: * Movement is via x/X, y/Y, and z/Z keys. Use space bar to exit.
9: * Mouse buttons rotate the right cube.
10: *
11: * Things to try:
12: *
13: * 1) Change the lighting so that the background is a dark green
14: * rather than the slate blue.
15: *
16: * 2) Give each object its own material properties. Make the
17: * rotating light look more realistic. (Let it self-emanate.)
18: *
19: * 3) Add the ability to turn the rotating light on and off. Its
20: * visible representation should disappear when it's off.
21: *
22: * 4) As you move in close to the objects, you'll notice the
23: * movable spotlight doesn't react realistically. Correct
24: * this behavior.
25: */
26:
27:
28: #include <stdio.h>
29: #include <stdlib.h>
30: #include <GL/glut.h>
31:
32:
33: // Base of the display lists.
34: GLuint lists;
35:
36:
37: // The right cube can be rotated. Whoopee.
38: static GLfloat theta[] = {45.0,0.0,0.0};
39: static GLint axis = 2;
40:
41: /* initial viewer location */
42: static GLdouble viewer[] = {20.0, 6.0, 6.0};
43:
44:
45: // Light 0 is the moving light. Light 1 is the spotlight that moves
46: // with us. Note that these are positional lights.
47: GLfloat light0_position[] = { 0.0, 0.0, 5.0, 1.0 };
48: GLfloat light1_position[] = { 0.0, 0.0, 0.0, 1.0 };
49:
50: // Angle of the moving light.
51: GLint light_angle = 0;
52:
53:
54: // Paint the scene.
55: void display(void)
56: {
57:
58:     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
59:
60:     /* Update viewer position in modelview matrix */
61:
62:     glLoadIdentity();
63:
64:     gluLookAt(viewer[0],viewer[1],viewer[2], 0.0, 0.0, 0.0,
65:              0.0, 1.0, 0.0);
66:
67:     // Position the moving light. It will be positioned in world
68:     // coordinates.
69:     glPushMatrix();
70:     glRotatef(light_angle, 1.0, 0.0, 0.0);
71:     glLightfv(GL_LIGHT0, GL_POSITION, light0_position);
72:
73:     // Render a solid sphere indicating the light's position. Notice
74:     // how it's not lit.
75:     glTranslatef(light0_position[0], light0_position[1],
76:                light0_position[2]);
77:     glDisable(GL_LIGHTING);
78:     glCallList(lists);
79:     glEnable(GL_LIGHTING);
80:     glPopMatrix();
81:
82:     glPushMatrix();
83:
84:     // Left cube.
85:     glTranslatef(-2.0, 0.0, 0.0);
86:     glCallList(lists + 1);
87:
88:     // Sphere centered at origin.
89:     glTranslatef(2.0, 0.0, 0.0);
90:     glCallList(lists + 2);
91:
92:     // Right cube.
93:     glTranslatef(2.0, 0.0, 0.0);
94:     // Rotate cube.
95:     glRotatef(theta[0], 1.0, 0.0, 0.0);
96:     glRotatef(theta[1], 0.0, 1.0, 0.0);
97:     glRotatef(theta[2], 0.0, 0.0, 1.0);
98:     glCallList(lists + 1);
99:
100:    // Now that I'm thoroughly confused, let's re-establish the
101:    // origin.
102:    glPopMatrix();
103:
104:    // The cone.
105:    // Up y-axis by 1.
106:    glTranslatef(0.0, 1.0, 0.0);
107:    // Rotate z-axis up to y-axis.
108:    glRotatef(-90.0, 1.0, 0.0, 0.0);
109:    glCallList(lists + 3);
110:
111:    // The torus.
112:    // Translate up the cone.
113:    glTranslatef(0.0, 0.0, 1.0);
114:    glCallList(lists + 4);
115:
116:    glutSwapBuffers();
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:     glutInit(&argc, argv);
239:     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
240:     glutInitWindowSize(500, 500);

```

```
241:  glutCreateWindow("Light Lab Two");
242:  glutReshapeFunc(myReshape);
243:  glutDisplayFunc(display);
244:  glutMouseFunc(mouse);
245:  glutKeyboardFunc(keys);
246:  glutIdleFunc(idle);
247:  init();
248:
249:  // Get five list handles.
250:  lists = glGenLists(5);
251:
252:  // For the moving light.
253:  glNewList(lists, GL_COMPILE);
254:      glutSolidCube(0.1);
255:  glEndList();
256:
257:  // For the two cubes.
258:  glNewList(lists + 1, GL_COMPILE);
259:      glutSolidCube(2.0);
260:  glEndList();
261:
262:  // The sphere at the origin.
263:  glNewList(lists + 2, GL_COMPILE);
264:      glutSolidSphere(1.0, 32, 32);
265:  glEndList();
266:
267:  // The cone.
268:  glNewList(lists + 3, GL_COMPILE);
269:      glutSolidCone(1.0, 2.0, 32, 32);
270:  glEndList();
271:
272:  // The torus.
273:  glNewList(lists + 4, GL_COMPILE);
274:      glutSolidTorus(0.25, 1.0, 32, 32);
275:  glEndList();
276:
277:  glutMainLoop();
278: }
```