

```
1: /* lightlab.c --- Experimentation with lighting in OpenGL.
2:  * Tom Kelliher 4/19/2001.
3:  *
4:  * Movement is via x/X, y/Y, and z/Z keys. Use space bar to exit.
5:  * Mouse buttons rotate the right cube.
6:  *
7:  * Things to try:
8:  *   1) Run program as is. Move around. Note lack of shadows and
9:  *      differences between left and right cubes.
10: *   2) Comment out the DIRTY define and verify that the cubes look
11: *      the same. Un-comment out the define when finished.
12: *   3) In display(), swap the gluLookAt() and glLightfv() calls.
13: *      Can see any effect of doing this? Re-swap the calls when
14: *      finished.
15: *   4) Try the light source at a few different positions. Can you
16: *      see any difference as a directional vs. positional source?
17: *   5) Un-comment out the GET_VALUES define and experiment with
18: *      material and light values.
19: *   6) Try to find material and light values that provide
20: *      realistic looking results. (Hint: you may want to
21: *      augment the code by changing the constant, linear, and
22: *      quadratic distance attenuation coefficients. See the man
23: *      page for glLight.)
24: *   7) Add another light source.
25: *   8) Augment the code so that each light and each object has
26: *      independent properties.
27: */
28:
29: #include <stdio.h>
30: #include <stdlib.h>
31: #include <GL/glut.h>
32:
33:
34: // Vertices for right cube.
35: GLfloat vertices[][3] = {{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},
36: {1.0,1.0,-1.0}, {-1.0,1.0,-1.0}, {-1.0,-1.0,1.0},
37: {1.0,-1.0,1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0}};
38:
39:
40: // Un-comment to be queried for light and material properties.
41: // #define GET_VALUES
42:
43: // Un-comment to use "quick and dirty normals" for right cube.
44: #define DIRTY
45:
46:
47: #ifdef DIRTY
48:
49: // These are the "quick and dirty" normals for the right cube,
50: // 1 per vertex.
51: GLfloat normals[][3] = {{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},
52: {1.0,1.0,-1.0}, {-1.0,1.0,-1.0}, {-1.0,-1.0,1.0},
53: {1.0,-1.0,1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0}};
54:
55: #else
56:
57: // These are the real normals for the right cube, 1 per face.
58: GLfloat normals[][3] = {{0.0,0.0,-1.0},{0.0,1.0,0.0},
59: {-1.0,0.0,0.0},{1.0,0.0,0.0},{0.0,0.0,1.0},{0.0,-1.0,0.0}};
60:
61: #endif
62:
63:
```

```
64: // The right cube can be rotated. Whoopee.
65: static GLfloat theta[] = {0.0,0.0,0.0};
66: static GLint axis = 2;
67:
68: /* initial viewer location */
69: static GLdouble viewer[] = {0.0, 0.0, 5.0};
70:
71:
72: // Guess what this one is.
73: // Fourth parameter: 0.0 --- directional light source
74: //                      1.0 --- positional light source.
75: GLfloat light_position[] = { 0.0, 5.0, 5.0, 0.0 };
76:
77:
78: #ifdef DIRTY
79:
80: // Draw a polygon with its normals.
81: void polygon(int a, int b, int c , int d)
82: {
83:     glBegin(GL_POLYGON);
84:     glNormal3fv(normals[a]);
85:     glVertex3fv(vertices[a]);
86:     glNormal3fv(normals[b]);
87:     glVertex3fv(vertices[b]);
88:     glNormal3fv(normals[c]);
89:     glVertex3fv(vertices[c]);
90:     glNormal3fv(normals[d]);
91:     glVertex3fv(vertices[d]);
92:     glEnd();
93: }
94:
95: // Draw a cube with lighting normals.
96: void colorcube()
97: {
98:     polygon(0,3,2,1);
99:     polygon(2,3,7,6);
100:    polygon(0,4,7,3);
101:    polygon(1,2,6,5);
102:    polygon(4,5,6,7);
103:    polygon(0,1,5,4);
104: }
105:
106: #else
107: // !DIRTY
108:
109: void polygon(int a, int b, int c , int d, GLfloat normal[])
110: {
111:     glBegin(GL_POLYGON);
112:     glNormal3fv(normal);
113:     glVertex3fv(vertices[a]);
114:     glVertex3fv(vertices[b]);
115:     glVertex3fv(vertices[c]);
116:     glVertex3fv(vertices[d]);
117:     glEnd();
118: }
119:
120: void colorcube()
121: {
122:     polygon(0,3,2,1,normals[0]);
123:     polygon(2,3,7,6,normals[1]);
124:     polygon(0,4,7,3,normals[2]);
125:     polygon(1,2,6,5,normals[3]);
126:     polygon(4,5,6,7,normals[4]);
```

```
127:         polygon(0,1,5,4,normals[5]);
128:     }
129:
130: #endif
131:
132:
133: // Paint the scene.
134: void display(void)
135: {
136:
137:     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
138:
139:     /* Update viewer position in modelview matrix */
140:     glLoadIdentity();
141:     gluLookAt(viewer[0],viewer[1],viewer[2],
142:              0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
143:
144:     // Position the light.
145:     glLightfv(GL_LIGHT0, GL_POSITION, light_position);
146:
147:     glPushMatrix();
148:
149:     // Left cube with "correct normals."
150:     glTranslatef(-2.0, 0.0, 0.0);
151:     glutSolidCube(2.0);
152:
153:     // Sphere centered at origin.
154:     glTranslatef(2.0, 0.0, 0.0);
155:     glutSolidSphere(1.0, 32, 32);
156:
157:     // Right cube with normals determined by DIRTY.
158:     glTranslatef(2.0, 0.0, 0.0);
159:     // Rotate cube.
160:     glRotatef(theta[0], 1.0, 0.0, 0.0);
161:     glRotatef(theta[1], 0.0, 1.0, 0.0);
162:     glRotatef(theta[2], 0.0, 0.0, 1.0);
163:     colorcube();
164:
165:     /* Now that I'm thoroughly confused, let's re-establish
166:      * the origin.
167:      */
168:     glPopMatrix();
169:
170:     // Up y-axis by 1.
171:     glTranslatef(0.0, 1.0, 0.0);
172:     // Rotate z-axis up to y-axis.
173:     glRotatef(-90.0, 1.0, 0.0, 0.0);
174:     glutSolidCone(1.0, 2.0, 32, 32);
175:     // Translate up the cone.
176:     glTranslatef(0.0, 0.0, 1.0);
177:     glutSolidTorus(0.25, 1.0, 32, 32);
178:
179:     glFlush();
180:     glutSwapBuffers();
181: }
182:
183:
184: /* Rotate the right cube.  Looks kinda neat when it cuts
185:  * into the sphere.
186:  */
187:
188: void mouse(int btn, int state, int x, int y)
189: {
```

```
190:         if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN) axis = 0;
191:         if(btn==GLUT_MIDDLE_BUTTON && state == GLUT_DOWN) axis = 1;
192:         if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN) axis = 2;
193:         theta[axis] += 2.0;
194:         if( theta[axis] > 360.0 ) theta[axis] -= 360.0;
195:         glutPostRedisplay();
196:     }
197:
198:
199: void keys(unsigned char key, int x, int y)
200: {
201:
202: /* Use x, X, y, Y, z, and Z keys to move viewer */
203:
204:     if(key == 'x') viewer[0]-= 1.0;
205:     if(key == 'X') viewer[0]+= 1.0;
206:     if(key == 'y') viewer[1]-= 1.0;
207:     if(key == 'Y') viewer[1]+= 1.0;
208:     if(key == 'z') viewer[2]-= 1.0;
209:     if(key == 'Z') viewer[2]+= 1.0;
210:     if(key == ' ') exit(0);
211:
212:     printf("v[x]: %f, v[y]: %f, v[z]: %f.\n",
213:           viewer[0], viewer[1], viewer[2]);
214:
215:     glutPostRedisplay();
216: }
217:
218:
219: void myReshape(int w, int h)
220: {
221:     glViewport(0, 0, w, h);
222:
223:     /* Use a perspective view */
224:
225:     glMatrixMode(GL_PROJECTION);
226:     glLoadIdentity();
227:     gluPerspective(35.0, (GLfloat) w/h, 1.0, 100.0);
228:     glMatrixMode(GL_MODELVIEW);
229: }
230:
231:
232: // Set material and light values.
233: void init(void)
234: {
235:     // Default values for material and light properties.
236:     GLfloat mat_ambient[] = { 0.5, 0.0, 0.0, 1.0 };
237:     GLfloat mat_diffuse[] = { 1.0, 0.5, 0.5, 1.0 };
238:     GLfloat mat_specular[] = { 1.0, 0.5, 0.5, 1.0 };
239:     GLfloat mat_shininess[] = { 50.0 };
240:
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: }
```