

```
1  #include <GL/glew.h>
2  #include <GL/glut.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <math.h>
6
7  #define my_PI 3.141592
8
9  static char* vsSource = "#version 120 WnW
10 in vec4 aPosition; WnW
11 in vec4 aColor; WnW
12 out vec4 vColor; WnW
13 uniform mat4 urotate; WnW
14 void main(void) { WnW
15     gl_Position = urotate*aPosition; WnW
16     vColor = aColor; WnW
17 }";
18
19 static char* fsSource = "#version 120 WnW
20 in vec4 vColor; WnW
21 void main(void) { WnW
22     gl_FragColor = vColor; WnW
23 }";
24
25 GLuint vs = 0;
26 GLuint fs = 0;
27 GLuint prog = 0;
28
29 char buf[1024];
30 float t = 0.0f;
31
32 GLfloat vertices[] = {
33     -0.2, -0.2, -0.2, 1.0, // 0
34     -0.2, -0.2, +0.2, 1.0, // 1
35     -0.2, +0.2, -0.2, 1.0, // 2
36     -0.2, +0.2, +0.2, 1.0, // 3
37     +0.2, -0.2, -0.2, 1.0, // 4
38     +0.2, -0.2, +0.2, 1.0, // 5
39     +0.2, +0.2, -0.2, 1.0, // 6
40     +0.2, +0.2, +0.2, 1.0, // 7
41 };
42
43
44 GLfloat colors[] = {
45     1.0, 0.0, 0.0, 1.0,
46     0.0, 1.0, 0.0, 1.0,
47     0.0, 0.0, 1.0, 1.0,
48     1.0, 1.0, 0.0, 1.0,
49     0.0, 1.0, 1.0, 1.0,
50     1.0, 0.0, 1.0, 1.0,
51     1.0, 0.5, 0.2, 1.0,
52     0.2, 1.0, 1.0, 1.0,
53 };
54
55 GLushort indices[] = { // 36 points, 12 triangles
56     0, 4, 6,
```

```
57     6, 2, 0,
58     4, 5, 7,
59     7, 6, 4,
60     1, 3, 7,
61     7, 5, 1,
62     0, 2, 3,
63     3, 1, 0,
64     2, 6, 7,
65     7, 3, 2,
66     0, 1, 5,
67     5, 4, 0,
68 };
69
70 void myinit(void) {
71     GLuint status;
72
73     printf("***** Your student number and name *****\n");
74     vs = glCreateShader(GL_VERTEX_SHADER);
75     glShaderSource(vs, 1, &vsSource, NULL);
76     glCompileShader(vs);
77     glGetShaderiv(vs, GL_COMPILE_STATUS, &status);
78     printf("vs compile status = %s\n", (status == GL_TRUE) ? "true" :
79           "false");
80     glGetShaderInfoLog(vs, sizeof(buf), NULL, buf);
81     printf("vs log = [%s]\n", buf);
82
83     fs = glCreateShader(GL_FRAGMENT_SHADER);
84     glShaderSource(fs, 1, &fsSource, NULL);
85     glCompileShader(fs);
86     glGetShaderiv(fs, GL_COMPILE_STATUS, &status);
87     printf("fs compile status = %s\n", (status == GL_TRUE) ? "true" :
88           "false");
89     glGetShaderInfoLog(fs, sizeof(buf), NULL, buf);
90     printf("fs log = [%s]\n", buf);
91
92     prog = glCreateProgram();
93     glAttachShader(prog, vs);
94     glAttachShader(prog, fs);
95     glLinkProgram(prog);
96     glGetProgramiv(prog, GL_LINK_STATUS, &status);
97     printf("program link status = %s\n", (status == GL_TRUE) ? "true" :
98           "false");
99     glGetProgramInfoLog(prog, sizeof(buf), NULL, buf);
100    printf("link log = [%s]\n", buf);
101    glValidateProgram(prog);
102    glGetProgramiv(prog, GL_VALIDATE_STATUS, &status);
103    printf("program validate status = %s\n", (status == GL_TRUE) ? "true" :
104           "false");
105    glGetProgramInfoLog(prog, sizeof(buf), NULL, buf);
106    printf("validate log = [%s]\n", buf);
107    glUseProgram(prog);
108
109    GLuint loc;
110    GLuint vbo[1];
111    // using vertex buffer object
112    glGenBuffers(1, vbo);
```

```

109     glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
110     glBufferData(GL_ARRAY_BUFFER, 2 * 8 * 4 * sizeof(GLfloat), NULL,
111                 GL_STATIC_DRAW);
112     glBufferSubData(GL_ARRAY_BUFFER, 0, 8 * 4 * sizeof(GLfloat), vertices);
113     glBufferSubData(GL_ARRAY_BUFFER, 8 * 4 * sizeof(GLfloat), 8 * 4 * sizeof
114                     (GLfloat),
115                     colors);
116
117     loc = glGetAttribLocation(prog, "aPosition");
118     glEnableVertexAttribArray(loc);
119     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid *)0);
120
121     loc = glGetAttribLocation(prog, "aColor");
122     glEnableVertexAttribArray(loc);
123     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid *) (3 * 4 *
124                             sizeof(GLfloat)));
125 }
126
127 void mykeyboard(unsigned char key, int x, int y) {
128     switch (key) {
129         case 27: // ESCAPE
130             exit(0);
131             break;
132     }
133 }
134
135 void myidle(void) {
136     // redisplay
137     glutPostRedisplay();
138 }
139
140 GLfloat m[16];
141
142 void mydisplay(void) {
143     GLuint loc;
144     glClearColor(0.7f, 0.7f, 0.7f, 1.0f); // gray
145     glClear(GL_COLOR_BUFFER_BIT);
146
147     t = 30.0 * my_PI/180.0;
148
149     /* rotation about z-axis
150     m[0] = cos(t); m[4] = -sin(t); m[8] = 0.0; m[12] = 0.0;
151     m[1] = sin(t); m[5] = cos(t); m[9] = 0.0; m[13] = 0.0;
152     m[2] = 0.0; m[6] = 0.0; m[10] = 1.0; m[14] = 0.0;
153     m[3] = 0.0; m[7] = 0.0; m[11] = 0.0; m[15] = 1.0;
154     */
155     // rotation about x-axis
156     m[0] = 1.0; m[4] = 0.0; m[8] = 0.0; m[12] = 0.0;
157     m[1] = 0.0; m[5] = cos(t); m[9] = -sin(t); m[13] = 0.0;
158     m[2] = 0.0; m[6] = sin(t); m[10] = cos(t); m[14] = 0.0;
159     m[3] = 0.0; m[7] = 0.0; m[11] = 0.0; m[15] = 1.0;
160
161     loc = glGetUniformLocation(prog, "urotate");

```

```
162     glUniformMatrix4fv(loc, 1, GL_FALSE, m);
163     glDrawElements(GL_TRIANGLES, 12 * 3, GL_UNSIGNED_SHORT, indices);
164     glFlush();
165     glutSwapBuffers();
166 }
167
168
169 int main(int argc, char* argv[]) {
170     glutInit(&argc, argv);
171     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
172     glutInitWindowSize(500, 500);
173     glutInitWindowPosition(0, 0);
174     glutCreateWindow("*** Your Student Number and Name ***");
175     glutDisplayFunc(mydisplay);
176     glutIdleFunc(myidle);
177     glutKeyboardFunc(mykeyboard);
178     glewInit();
179     myinit();
180     glutMainLoop();
181     return 0;
182 }
183
```