```c
#include <GL/glew.h>
#include <GL/glut.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define my_PI 3.141592

static char* vsSource = "#version 130 \n\
in vec4 aPosition; \n\
in vec4 aColor; \n\
flat out vec4 vColor; \n\
// out vec4 vColor; \n\
uniform mat4 urotate;  \n\
void main(void) { \n\
  mat4 scalemat = mat4(5.0); \n\
  scalemat[3][3] = 1.0; \n\
  gl_Position = urotate*scalemat*aPosition; \n\
  vColor = aColor; \n\
}";

static char* fsSource = "#version 130 \n\
flat in vec4 vColor; \n\
// in vec4 vColor; \n\
void main(void) { \n\
  gl_FragColor = vColor; \n\
}";

GLuint vs = 0;
GLuint fs = 0;
GLuint prog = 0;

char buf[1024];
int DRAW_MODE = 0;
float t = 0.0f;

GLfloat vertices[] = {
    0.0, 0.1, 0.0, 1.0, // 0
    -0.1, -0.1, +0.1, 1.0, // 1
    0.1, -0.1, +0.1, 1.0, // 2
    0.1, -0.1, -0.1, 1.0, // 3
    -0.1, -0.1, -0.1, 1.0, // 4
};

GLfloat colors[] = {
    1.0, 0.0, 0.0, 1.0,  //0
    0.0, 1.0, 0.0, 1.0,  //1
    0.0, 0.0, 1.0, 1.0,  //2
    1.0, 0.0, 1.0, 1.0,  //3
    1.0, 1.0, 0.0, 1.0   //4
};

GLushort indices[] = {
    0, 1, 2,
    2, 3, 0,
    4, 0, 3,
```

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

```
107     glEnableVertexAttribArray(loc);
108     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid *)0);
109
110     loc = glGetAttribLocation(prog, "aColor");
111     glEnableVertexAttribArray(loc);
112     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid *)(5 * 4 *
          sizeof(GLfloat)));
113
114     glProvokingVertex(GL_FIRST_VERTEX_CONVENTION);
115     glEnable(GL_DEPTH_TEST);
116     //  glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
117     glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
118 }
119
120 void mykeyboard(unsigned char key, int x, int y) {
121     switch (key) {
122     case 27: // ESCAPE
123         exit(0);
124         break;
125     }
126 }
127
128 void mymenu(int id)
129 {
130     if (id == 0)
131         DRAW_MODE = 0;
132     else if (id == 1)
133         DRAW_MODE = 1;
134     else
135         DRAW_MODE = 2;
136 }
137
138 void myidle(void) {
139     t += 0.001f;
140     // redisplay
141     glutPostRedisplay();
142 }
143
144 GLfloat m[16];
145
146 void mydisplay(void) {
147     GLuint loc;
148     glClearColor(0.7f, 0.7f, 0.7f, 1.0f); // gray
149     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
150     // glClear(GL_COLOR_BUFFER_BIT);
151 //   t = 40.0 * my_PI/180.0;
152
153
154     if (DRAW_MODE == 0) {
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     else if (DRAW_MODE == 1) {
```

```
162         m[0] = cos(t); m[4] = 0.0; m[8] = sin(t);  m[12] = 0.0;
163         m[1] = 0.0; m[5] = 1.0;  m[9] = 0.0;  m[13] = 0.0;
164         m[2] = -sin(t);    m[6] = 0.0;  m[10] = cos(t); m[14] = 0.0;
165         m[3] = 0.0;    m[7] = 0.0;     m[11] = 0.0; m[15] = 1.0;
166     }
167     else if (DRAW_MODE == 2) {
168         // rotation about z-axis
169         m[0] = cos(t); m[4] = -sin(t); m[8] = 0.0;  m[12] = 0.0;
170         m[1] = sin(t); m[5] = cos(t);  m[9] = 0.0;  m[13] = 0.0;
171         m[2] = 0.0;    m[6] = 0.0;  m[10] = 1.0; m[14] = 0.0;
172         m[3] = 0.0;    m[7] = 0.0;     m[11] = 0.0; m[15] = 1.0;
173     }
174     loc = glGetUniformLocation(prog, "urotate");
175     glUniformMatrix4fv(loc, 1, GL_FALSE, m);
176     glDrawElements(GL_TRIANGLES, 6 * 3, GL_UNSIGNED_SHORT, indices);
177     glFlush();
178     glutSwapBuffers();
179 }
180
181
182 int main(int argc, char* argv[]) {
183     glutInit(&argc, argv);
184     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
185 //  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
186     glutInitWindowSize(500, 500);
187     glutInitWindowPosition(0, 0);
188     glutCreateWindow("*** Your Student Number and Name ***");
189     glutDisplayFunc(mydisplay);
190     glutIdleFunc(myidle);
191     glutKeyboardFunc(mykeyboard);
192     glutCreateMenu(mymenu);
193     glutAddMenuEntry("x-axis", 0);
194     glutAddMenuEntry("y-axis", 1);
195     glutAddMenuEntry("z-axis", 2);
196     glutAttachMenu(GLUT_RIGHT_BUTTON);
197     glewInit();
198     myinit();
199     glutMainLoop();
200     return 0;
201 }
202
```