```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\
out vec4 vColor; \n\
uniform mat4 umx;  \n\
uniform mat4 umy;  \n\
uniform mat4 umz;  \n\
void main(void) { \n\
  gl_Position = umz*umy*umx*aPosition; \n\
  vColor = aColor; \n\
}";

static char* fsSource = "#version 130 \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, 0.3, 0.0, 1.0, // 0
    -0.2, -0.2, +0.2, 1.0, // 1
    0.2, -0.2, +0.2, 1.0, // 2
    0.2, -0.2, -0.2, 1.0, // 3
    -0.2, -0.2, -0.2, 1.0, // 4
};

/*
GLfloat vertices[] = {
    0.5, 0.8, 0.0, 1.0, // 0
    0.3, 0.3, +0.2, 1.0, // 1
    0.7, 0.3, +0.2, 1.0, // 2
    0.7, 0.3, -0.2, 1.0, // 3
    0.3, 0.3, -0.2, 1.0, // 4
};
*/

GLfloat colors[] = {
    1.0, 0.0, 0.0, 1.0,  //0
    0.0, 1.0, 0.0, 1.0,  //1
```

```
57      0.0, 0.0, 1.0, 1.0,  //2
58      1.0, 0.0, 1.0, 1.0,  //3
59      1.0, 1.0, 0.0, 1.0   //4
60  };
61
62  GLushort indices[] = {
63      0, 1, 2,
64      2, 3, 0,
65      4, 0, 3,
66      1, 0, 4,
67      2, 3, 1,
68      3, 4, 1
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" :
          "false");
79      glGetShaderInfoLog(vs, sizeof(buf), NULL, buf);
80      printf("vs log = [%s]\n", buf);
81
82      fs = glCreateShader(GL_FRAGMENT_SHADER);
83      glShaderSource(fs, 1, &fsSource, NULL);
84      glCompileShader(fs);
85      glGetShaderiv(fs, GL_COMPILE_STATUS, &status);
86      printf("fs compile status = %s\n", (status == GL_TRUE) ? "true" :
          "false");
87      glGetShaderInfoLog(fs, sizeof(buf), NULL, buf);
88      printf("fs log = [%s]\n", buf);
89
90      prog = glCreateProgram();
91      glAttachShader(prog, vs);
92      glAttachShader(prog, fs);
93      glLinkProgram(prog);
94      glGetProgramiv(prog, GL_LINK_STATUS, &status);
95      printf("program link status = %s\n", (status == GL_TRUE) ? "true" :
          "false");
96      glGetProgramInfoLog(prog, sizeof(buf), NULL, buf);
97      printf("link log = [%s]\n", buf);
98      glValidateProgram(prog);
99      glGetProgramiv(prog, GL_VALIDATE_STATUS, &status);
100     printf("program validate status = %s\n", (status == GL_TRUE) ? "true" :
          "false");
101     glGetProgramInfoLog(prog, sizeof(buf), NULL, buf);
102     printf("validate log = [%s]\n", buf);
103     glUseProgram(prog);
104
105     GLuint loc;
106     GLuint vbo[1];
107     // using vertex buffer object
108     glGenBuffers(1, vbo);
```

```
109     glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
110     glBufferData(GL_ARRAY_BUFFER, 2 * 5 * 4 * sizeof(GLfloat), NULL,
          GL_STATIC_DRAW);
111     glBufferSubData(GL_ARRAY_BUFFER, 0, 5 * 4 * sizeof(GLfloat), vertices);
112     glBufferSubData(GL_ARRAY_BUFFER, 5 * 4 * sizeof(GLfloat), 5 * 4 * sizeof
          (GLfloat),
113       colors);
114
115     loc = glGetAttribLocation(prog, "aPosition");
116     glEnableVertexAttribArray(loc);
117     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid *)0);
118
119     loc = glGetAttribLocation(prog, "aColor");
120     glEnableVertexAttribArray(loc);
121     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid *)(5 * 4 *
          sizeof(GLfloat)));
122
123     glEnable(GL_DEPTH_TEST);
124 //  glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
125     glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
126 }
127
128 void mykeyboard(unsigned char key, int x, int y) {
129     switch (key) {
130     case 27: // ESCAPE
131         exit(0);
132         break;
133     }
134 }
135
136 void myidle(void) {
137     t += 0.001f;
138     // redisplay
139     glutPostRedisplay();
140 }
141
142 GLfloat mx[16], my[16], mz[16];
143 GLfloat tx, ty, tz;
144 void mydisplay(void) {
145     GLuint loc;
146     glClearColor(0.7f, 0.7f, 0.7f, 1.0f); // gray
147     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
148
149     tx = ty = tz = 0.0;
150
151     tx = 60.0 * my_PI / 180.0;
152     ty = 10.0 * my_PI / 180.0;
153     tz = -30.0 * my_PI / 180.0;
154
155     // rotation about x-axis
156     mx[0] = 1.0; mx[4] = 0.0;     mx[8] = 0.0;      mx[12] = 0.0;
157     mx[1] = 0.0; mx[5] = cos(tx);  mx[9] = -sin(tx);  mx[13] = 0.0;
158     mx[2] = 0.0; mx[6] = sin(tx);  mx[10] = cos(tx);  mx[14] = 0.0;
159     mx[3] = 0.0; mx[7] = 0.0;     mx[11] = 0.0;     mx[15] = 1.0;
160
161     // rotation about y-axis
```

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