```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\
uniform mat4 utranslate; \n\
uniform mat4 um_view; \n\
void main(void) { \n\
  gl_Position = um_view*urotate*utranslate*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 = -0.5f;

int num_vertices = 4, num_faces = 4;

/*
GLfloat vertices[] = { // partially clipped out
    0.0, 0.5, -0.8, 1.0, // v0
    -0.5, -0.5, -0.5, 1.0, // v1
    0.5, -0.5, -0.5, 1.0, // v2
    0.0, -0.5, -1.3, 1.0, // v3
};
*/

GLfloat vertices[] = { // at center
    0.0, 0.5, 0.0, 1.0, // v0
    -0.5, -0.5, 0.3, 1.0, // v1
    0.5, -0.5, 0.3, 1.0, // v2
    0.0, -0.5, -0.5, 1.0, // v3
};


GLfloat colors[] = {
```

```
57      1.0, 0.0, 0.0, 1.0,  // v0 color
58      0.0, 1.0, 0.0, 1.0,  // v1 color
59      0.0, 0.0, 1.0, 1.0,  // v2 color
60      1.0, 0.0, 1.0, 1.0,  // v3 color
61  };
62
63  GLushort indices[] = {
64      0, 1, 2,  // red
65      1, 0, 3,  // green
66      2, 3, 0,  // blue
67      3, 2, 1,  // purple
68  };
69  void myinit(void) {
70      GLuint status;
71
72      printf("***** Your student number and name *****\n");
73      vs = glCreateShader(GL_VERTEX_SHADER);
74      glShaderSource(vs, 1, &vsSource, NULL);
75      glCompileShader(vs);
76      glGetShaderiv(vs, GL_COMPILE_STATUS, &status);
77      printf("vs compile status = %s\n", (status == GL_TRUE) ? "true" :
          "false");
78      glGetShaderInfoLog(vs, sizeof(buf), NULL, buf);
79      printf("vs log = [%s]\n", buf);
80
81      fs = glCreateShader(GL_FRAGMENT_SHADER);
82      glShaderSource(fs, 1, &fsSource, NULL);
83      glCompileShader(fs);
84      glGetShaderiv(fs, GL_COMPILE_STATUS, &status);
85      printf("fs compile status = %s\n", (status == GL_TRUE) ? "true" :
          "false");
86      glGetShaderInfoLog(fs, sizeof(buf), NULL, buf);
87      printf("fs log = [%s]\n", buf);
88
89      prog = glCreateProgram();
90      glAttachShader(prog, vs);
91      glAttachShader(prog, fs);
92      glLinkProgram(prog);
93      glGetProgramiv(prog, GL_LINK_STATUS, &status);
94      printf("program link status = %s\n", (status == GL_TRUE) ? "true" :
          "false");
95      glGetProgramInfoLog(prog, sizeof(buf), NULL, buf);
96      printf("link log = [%s]\n", buf);
97      glValidateProgram(prog);
98      glGetProgramiv(prog, GL_VALIDATE_STATUS, &status);
99      printf("program validate status = %s\n", (status == GL_TRUE) ? "true" :
          "false");
100     glGetProgramInfoLog(prog, sizeof(buf), NULL, buf);
101     printf("validate log = [%s]\n", buf);
102     glUseProgram(prog);
103
104     GLuint loc;
105     GLuint vbo[1];
106     // using vertex buffer object
107     glGenBuffers(1, vbo);
108     glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
```

```
109     glBufferData(GL_ARRAY_BUFFER, 2 * num_vertices * 4 * sizeof(GLfloat),
          NULL, GL_STATIC_DRAW);
110     glBufferSubData(GL_ARRAY_BUFFER, 0, num_vertices * 4 * sizeof(GLfloat),
          vertices);
111     glBufferSubData(GL_ARRAY_BUFFER, num_vertices * 4 * sizeof(GLfloat),
          num_vertices * 4 * sizeof(GLfloat),
112       colors);
113
114     loc = glGetAttribLocation(prog, "aPosition");
115     glEnableVertexAttribArray(loc);
116     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid *)0);
117
118     loc = glGetAttribLocation(prog, "aColor");
119     glEnableVertexAttribArray(loc);
120     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid *)
          (num_vertices * 4 * sizeof(GLfloat)));
121
122     glProvokingVertex(GL_FIRST_VERTEX_CONVENTION);
123     glEnable(GL_DEPTH_TEST);
124     //  glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
125     glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
126
127 }
128
129 void mykeyboard(unsigned char key, int x, int y) {
130     switch (key) {
131     case 27: // ESCAPE
132         exit(0);
133         break;
134     }
135 }
136
137 void myidle(void) {
138     t += 0.0001f;
139     // redisplay
140     glutPostRedisplay();
141 }
142
143 GLfloat m[16], m_view[16];
144
145 void mydisplay(void) {
146     GLuint loc;
147     glClearColor(0.7f, 0.7f, 0.7f, 1.0f); // gray
148     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
149
150     // rotation about z axis
151     t = 0.0;
152     m[0] = cos(t);    m[4] = -sin(t);   m[8] = 0.0;       m[12] = 0.0;
153     m[1] = sin(t);    m[5] = cos(t);    m[9] = 0.0;       m[13] = 0.0;
154     m[2] = 0.0;       m[6] = 0.0;       m[10] = 1.0;      m[14] = 0.0;
155     m[3] = 0.0;       m[7] = 0.0;       m[11] = 0.0;      m[15] = 1.0;
156     loc = glGetUniformLocation(prog, "urotate");
157     glUniformMatrix4fv(loc, 1, GL_FALSE, m);
158
159     // translation
160     m[0] = 1.0;    m[4] = 0.0;   m[8] = 0.0;    m[12] = 0.0;
```

```
161     m[1] = 0.0;    m[5] = 1.0;   m[9] = 0.0;    m[13] = 0.0;
162     m[2] = 0.0;    m[6] = 0.0;   m[10] = 1.0;   m[14] = 0.0;
163     m[3] = 0.0;    m[7] = 0.0;   m[11] = 0.0;   m[15] = 1.0;
164     loc = glGetUniformLocation(prog, "utranslate");
165     glUniformMatrix4fv(loc, 1, GL_FALSE, m);
166
167     // Example 1
168     m_view[0] = 1.0;    m_view[4] = 0.0;    m_view[8] = 0.0;    m_view[12] =
          0.0;
169     m_view[1] = 0.0;    m_view[5] = 1.0;    m_view[9] = 0.0;    m_view[13] =
          0.0;
170     m_view[2] = 0.0;    m_view[6] = 0.0;    m_view[10] = -1.0;  m_view[14] =
          0.5;
171     m_view[3] = 0.0;    m_view[7] = 0.0;    m_view[11] = 0.0;   m_view[15] =
          1.0;
172
173     /*
174     // Example 2
175     m_view[0] = 0.707107;   m_view[4] = 0.0;         m_view[8] = -0.707107;
          m_view[12] = 0.0;
176     m_view[1] = -0.408248;  m_view[5] = 0.816497;    m_view[9] =-0.408248;
          m_view[13] = 0.0;
177     m_view[2] = -0.577350;  m_view[6] = -0.577350;   m_view[10] = -0.577350;
          m_view[14] = 0.173205;
178     m_view[3] = 0.0;        m_view[7] = 0.0;         m_view[11] = 0.0;
          m_view[15] = 1.0;
179     */
180
181     loc = glGetUniformLocation(prog, "um_view");
182     glUniformMatrix4fv(loc, 1, GL_FALSE, m_view);
183
184     glDrawElements(GL_TRIANGLES, num_faces * 3, GL_UNSIGNED_SHORT, indices);
185     glFlush();
186
187     glutSwapBuffers();
188 }
189
190
191 int main(int argc, char* argv[]) {
192     glutInit(&argc, argv);
193     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
194 //  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
195     glutInitWindowSize(500, 500);
196     glutInitWindowPosition(0, 0);
197     glutCreateWindow("*** Your Student Number and Name ***");
198     glutDisplayFunc(mydisplay);
199     glutIdleFunc(myidle);
200     glutKeyboardFunc(mykeyboard);
201     glewInit();
202     myinit();
203     glutMainLoop();
204     return 0;
205 }
206
```