```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\
void main(void) { \n\
  gl_Position = 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[] = {
    1.0, 0.0, 0.0, 1.0,  // v0 color
    0.0, 1.0, 0.0, 1.0,  // v1 color
    0.0, 0.0, 1.0, 1.0,  // v2 color
    1.0, 0.0, 1.0, 1.0,  // v3 color
};

GLushort indices[] = {
    0, 1, 2,  // red
    1, 0, 3,  // green
    2, 3, 0,  // blue
    3, 2, 1,  // purple
};
void myinit(void) {
    GLuint status;

    printf("***** Your student number and name *****\n");
    vs = glCreateShader(GL_VERTEX_SHADER);
    glShaderSource(vs, 1, &vsSource, NULL);
    glCompileShader(vs);
    glGetShaderiv(vs, GL_COMPILE_STATUS, &status);
    printf("vs compile status = %s\n", (status == GL_TRUE) ? "true" : "false");
    glGetShaderInfoLog(vs, sizeof(buf), NULL, buf);
    printf("vs log = [%s]\n", buf);

    fs = glCreateShader(GL_FRAGMENT_SHADER);
    glShaderSource(fs, 1, &fsSource, NULL);
    glCompileShader(fs);
    glGetShaderiv(fs, GL_COMPILE_STATUS, &status);
    printf("fs compile status = %s\n", (status == GL_TRUE) ? "true" : "false");
    glGetShaderInfoLog(fs, sizeof(buf), NULL, buf);
    printf("fs log = [%s]\n", buf);

    prog = glCreateProgram();
    glAttachShader(prog, vs);
    glAttachShader(prog, fs);
    glLinkProgram(prog);
    glGetProgramiv(prog, GL_LINK_STATUS, &status);
    printf("program link status = %s\n", (status == GL_TRUE) ? "true" : "false");
    glGetProgramInfoLog(prog, sizeof(buf), NULL, buf);
    printf("link log = [%s]\n", buf);
    glValidateProgram(prog);
    glGetProgramiv(prog, GL_VALIDATE_STATUS, &status);
    printf("program validate status = %s\n", (status == GL_TRUE) ? "true" : "false");
    glGetProgramInfoLog(prog, sizeof(buf), NULL, buf);
    printf("validate log = [%s]\n", buf);
    glUseProgram(prog);

    GLuint loc;
    GLuint vbo[1];
    // using vertex buffer object
    glGenBuffers(1, vbo);
    glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
    glBufferData(GL_ARRAY_BUFFER, 2 * num_vertices * 4 * sizeof(GLfloat), NULL, GL_STATIC_DRAW);
    glBufferSubData(GL_ARRAY_BUFFER, 0, num_vertices * 4 * sizeof(GLfloat),
        vertices);
    glBufferSubData(GL_ARRAY_BUFFER, num_vertices * 4 * sizeof(GLfloat),
        num_vertices * 4 * sizeof(GLfloat),
        colors);

    loc = glGetAttribLocation(prog, "aPosition");
    glEnableVertexAttribArray(loc);
    glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid *)0);

    loc = glGetAttribLocation(prog, "aColor");
    glEnableVertexAttribArray(loc);
    glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid *)
        (num_vertices * 4 * sizeof(GLfloat)));

    glProvokingVertex(GL_FIRST_VERTEX_CONVENTION);
    glEnable(GL_DEPTH_TEST);
    //  glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);

}

void mykeyboard(unsigned char key, int x, int y) {
    switch (key) {
    case 27: // ESCAPE
        exit(0);
        break;
    }
}

void myidle(void) {
    t += 0.0001f;


    // redisplay
    glutPostRedisplay();
}

GLfloat m[16];

void mydisplay(void) {
    GLuint loc;
    glClearColor(0.7f, 0.7f, 0.7f, 1.0f); // gray
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glDrawElements(GL_TRIANGLES, num_faces * 3, GL_UNSIGNED_SHORT, indices);
    glFlush();

    glutSwapBuffers();
}


int main(int argc, char* argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
//  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
```

```c
    glutInitWindowPosition(0, 0);
    glutCreateWindow("*** Your Student Number and Name ***");
    glutDisplayFunc(mydisplay);
    glutIdleFunc(myidle);
    glutKeyboardFunc(mykeyboard);
    glewInit();
    myinit();
    glutMainLoop();
    return 0;
}

```