```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 u_rotate;  \n\
uniform float u_scale_factor; \n\
uniform vec2 u_trans_vec; \n\
void main(void) { \n\
  mat4 scalemat = mat4(u_scale_factor); \n\
  scalemat[3][3] = 1.0; \n\
  mat4 transmat = mat4(1.0); \n\
  transmat[3][0] = u_trans_vec[0]; \n\
  transmat[3][1] = u_trans_vec[1]; \n\
  gl_Position = transmat*u_rotate*aPosition; \n\
//  gl_Position = u_rotate*transmat*aPosition; \n\
//  gl_Position = scalemat*transmat*u_rotate*aPosition; \n\
//  gl_Position = u_rotate*scalemat*transmat*aPosition; \n\
//  gl_Position = transmat*u_rotate*scalemat*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, 0.15, 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,
    1, 0, 4,
    2, 3, 1,
    3, 4, 1
};
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 * 5 * 4 * sizeof(GLfloat), NULL, 
        GL_STATIC_DRAW);
    glBufferSubData(GL_ARRAY_BUFFER, 0, 5 * 4 * sizeof(GLfloat), vertices);
    glBufferSubData(GL_ARRAY_BUFFER, 5 * 4 * sizeof(GLfloat), 5 * 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 *)(5 * 4 *
      sizeof(GLfloat)));

    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 += 0.001f;
    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);

    t = 0.0;
//    t = 60.0 * my_PI/180.0;

    // rotation about x-axis
    m[0] = 1.0; m[4] = 0.0;     m[8] = 0.0;      m[12] = 0.0;
    m[1] = 0.0; m[5] = cos(t);  m[9] = -sin(t);  m[13] = 0.0;
    m[2] = 0.0; m[6] = sin(t);  m[10] = cos(t);  m[14] = 0.0;
    m[3] = 0.0; m[7] = 0.0;     m[11] = 0.0;     m[15] = 1.0;

    loc = glGetUniformLocation(prog, "u_rotate");
    glUniformMatrix4fv(loc, 1, GL_FALSE, m);

    float scale_factor = 1.0;
//  float scale_factor = 1.5;
    loc = glGetUniformLocation(prog, "u_scale_factor");
    glUniform1f(loc, scale_factor);

    float trans_vec[] = {0.0, 0.0};
//  float trans_vec[] = { 0.5, 0.5 };
    loc = glGetUniformLocation(prog, "u_trans_vec");
    glUniform2fv(loc, 1, trans_vec);

    glDrawElements(GL_TRIANGLES, 6 * 3, GL_UNSIGNED_SHORT, indices);
    glFlush();
    glutSwapBuffers();
}


int main(int argc, char* argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("*** Your Student Number and Name ***");
    glutDisplayFunc(mydisplay);
    glutIdleFunc(myidle);
    glutKeyboardFunc(mykeyboard);
    glewInit();
    myinit();
    glutMainLoop();
    return 0;
}

```