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

static char* vsSource = "#version 120 ₩n₩
in vec4 aPosition; ₩n₩
in vec4 aColor; ₩n₩
out vec4 vColor; ₩n₩
uniform float udist; ₩n₩
void main(void) { ₩n₩
  gl_Position.x = aPosition.x + udist; ₩n₩
  gl_Position.yzw = aPosition.yzw; ₩n₩
  vColor = aColor; ₩n₩
}";

static char* fsSource = "#version 120 ₩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];
float dist = 0.0f;

GLfloat vertices[] = {
    -0.5, -0.5, 0.0, 1.0,
    +0.5, -0.5, 0.0, 1.0,
    -0.5, +0.5, 0.0, 1.0,
};

GLfloat colors[] = {
    1.0, 0.0, 0.0, 1.0, // red
    0.0, 1.0, 0.0, 1.0, // green
    0.0, 0.0, 1.0, 1.0, // blue
};

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 vbo[1];
    // using vertex buffer object
    glGenBuffers(1, vbo);
    glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
    glBufferData(GL_ARRAY_BUFFER, 2 * 3 * 4 * sizeof(GLfloat), NULL, GL_STATIC_DRAW);
    glBufferSubData(GL_ARRAY_BUFFER, 0, 3 * 4 * sizeof(GLfloat), vertices);
    glBufferSubData(GL_ARRAY_BUFFER, 3 * 4 * sizeof(GLfloat), 3 * 4 * sizeof(GLfloat),
        colors);
}

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



void myidle(void) {
    dist += 0.0001f;
    if (dist > 1.5)
        dist = 0.0f;

    // redisplay
    glutPostRedisplay();
}


```

```c
void mydisplay(void) {
    GLuint loc;

    glClearColor(0.7f, 0.7f, 0.7f, 1.0f); // gray
    glClear(GL_COLOR_BUFFER_BIT);

    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 *)(3 * 4 * sizeof(GLfloat)));

    loc = glGetUniformLocation(prog, "udist");
    glUniform1f(loc, dist);

    glDrawArrays(GL_TRIANGLES, 0, 3);

    glFlush();
    glutSwapBuffers();
}

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

```