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

static char* vsSource = "#version 120 \n\
attribute vec4 aPosition; \n\
attribute vec4 aColor; \n\
varying 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\
varying vec4 vColor; \n\
void main(void) { \n\
  gl_FragColor = vColor; \n\
}";

GLuint vs = 0;
GLuint fs = 0;
GLuint prog = 0;

char buf[1024];
GLuint vbo[2], vao[2];

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,
    0.0, 1.0, 0.0, 1.0,
    0.0, 0.0, 1.0, 1.0,
};
GLfloat vertices2[] = {
    -0.8, -0.8, 0.0, 1.0,
    +0.2, -0.8, 0.0, 1.0,
    -0.8, +0.2, 0.0, 1.0,
};
GLfloat colors2[] = {
    1.0, 0.0, 0.0, 1.0,
    1.0, 0.0, 0.0, 1.0,
    1.0, 0.0, 0.0, 1.0,

};


void myinit(void) {
    GLuint status;

    printf("***** Your student number and name *****\n");
```

```
57      vs = glCreateShader(GL_VERTEX_SHADER);
58      glShaderSource(vs, 1, &vsSource, NULL);
59      glCompileShader(vs); // compile to get .OBJ
60      glGetShaderiv(vs, GL_COMPILE_STATUS, &status);
61      printf("vs compile status = %s\n", (status == GL_TRUE) ? "true" :
          "false");
62      glGetShaderInfoLog(vs, sizeof(buf), NULL, buf);
63      printf("vs log = [%s]\n", buf);
64
65      fs = glCreateShader(GL_FRAGMENT_SHADER);
66      glShaderSource(fs, 1, &fsSource, NULL);
67      glCompileShader(fs); // compile to get .OBJ
68      glGetShaderiv(fs, GL_COMPILE_STATUS, &status);
69      printf("fs compile status = %s\n", (status == GL_TRUE) ? "true" :
          "false");
70      glGetShaderInfoLog(fs, sizeof(buf), NULL, buf);
71      printf("fs log = [%s]\n", buf);
72      // prog: program
73      prog = glCreateProgram();
74      glAttachShader(prog, vs);
75      glAttachShader(prog, fs);
76      glLinkProgram(prog); // link to get .EXE
77      glGetProgramiv(prog, GL_LINK_STATUS, &status);
78      printf("program link status = %s\n", (status == GL_TRUE) ? "true" :
          "false");
79      glGetProgramInfoLog(prog, sizeof(buf), NULL, buf);
80      printf("link log = [%s]\n", buf);
81      glValidateProgram(prog);
82      glGetProgramiv(prog, GL_VALIDATE_STATUS, &status);
83      printf("program validate status = %s\n", (status == GL_TRUE) ? "true" :
          "false");
84      glGetProgramInfoLog(prog, sizeof(buf), NULL, buf);
85      printf("validate log = [%s]\n", buf);
86      glUseProgram(prog);
87
88      glGenVertexArrays(2, vao);
89      glBindVertexArray(vao[0]);
90
91      glGenBuffers(2,
          vbo);  ////////////////////////////////////////////////////////////
92      glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
93      glBufferData(GL_ARRAY_BUFFER, 2 * 3 * 4 * sizeof(GLfloat), NULL,
          GL_STATIC_DRAW);
94      glBufferSubData(GL_ARRAY_BUFFER, 0, 3 * 4 * sizeof(GLfloat), vertices);
95      glBufferSubData(GL_ARRAY_BUFFER, 3 * 4 * sizeof(GLfloat), 3 * 4 * sizeof
          (GLfloat),
96          colors);
97
98      GLuint loc;
99      loc = glGetAttribLocation(prog, "aPosition");
100     glEnableVertexAttribArray(loc);
101     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid*)0);
102     loc = glGetAttribLocation(prog, "aColor");
103     glEnableVertexAttribArray(loc);
104     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid*)(3 * 4 *
          sizeof(GLfloat)));
```

```
105
106     glBindVertexArray(vao[1]);
107
108     glBindBuffer(GL_ARRAY_BUFFER, vbo[1]);
109     glBufferData(GL_ARRAY_BUFFER, 2 * 3 * 4 * sizeof(GLfloat), NULL,
          GL_STATIC_DRAW);
110     glBufferSubData(GL_ARRAY_BUFFER, 0, 3 * 4 * sizeof(GLfloat), vertices2);
111     glBufferSubData(GL_ARRAY_BUFFER, 3 * 4 * sizeof(GLfloat), 3 * 4 * sizeof
          (GLfloat),
112         colors2);
113
114     loc = glGetAttribLocation(prog, "aPosition");
115     glEnableVertexAttribArray(loc);
116     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid*)0);
117     loc = glGetAttribLocation(prog, "aColor");
118     glEnableVertexAttribArray(loc);
119     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid*)(3 * 4 *
          sizeof(GLfloat)));
120
121
122 }
123
124 void mykeyboard(unsigned char key, int x, int y) {
125     switch (key) {
126     case 27: // ESCAPE
127         exit(0);
128         break;
129     }
130 }
131
132 float dist = 0;
133 void mydisplay(void) {
134
135     // clear
136     glClearColor(0.7f, 0.7f, 0.7f, 1.0f); // gray
137     glClear(GL_COLOR_BUFFER_BIT);
138     GLuint loc;
139
140     loc = glGetUniformLocation(prog, "udist");
141     glUniform1f(loc, dist);
142
143
144     glBindVertexArray(vao[0]);
145     glDrawArrays(GL_TRIANGLES, 0, 3);
146
147     glBindVertexArray(vao[1]);
148     glDrawArrays(GL_TRIANGLES, 0, 3);
149
150     glFlush();
151     glutSwapBuffers();
152 }
153 void myreshape(int x, int y)
154 {
155     glViewport(0, 0, x, y);
156  }
157
```

```
158
159 void myidle(void)
160 {
161     dist += 0.0001f;
162     if (dist > 1.0)
163         dist = 0;
164     printf("dist  %f\n", dist);
165     glutPostRedisplay();
166 }
167 int main(int argc, char* argv[]) {
168     glutInit(&argc, argv);
169     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
170     glutInitWindowSize(500, 500);
171     glutInitWindowPosition(0, 0);
172     glutCreateWindow("*** Your Student Number and Name ***");
173     glutDisplayFunc(mydisplay);
174     glutKeyboardFunc(mykeyboard);
175     glutReshapeFunc(myreshape);
176     glutIdleFunc(myidle);
177
178     glewInit();
179     myinit();
180     glutMainLoop();
181     return 0;
182 }
183
```