```
1  #include <GL/glew.h>
2  #include <GL/glut.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  static char* vsSource = "#version 120 ₩n₩
7  in vec4 aPosition; ₩n₩
8  in vec4 aColor; ₩n₩
9  out vec4 vColor; ₩n₩
10 uniform float udist; ₩n₩
11 void main(void) { ₩n₩
12   gl_Position.x = aPosition.x + udist; ₩n₩
13   gl_Position.yzw = aPosition.yzw; ₩n₩
14   vColor = aColor; ₩n₩
15 }";
16
17 static char* fsSource = "#version 120 ₩n₩
18 in vec4 vColor; ₩n₩
19 void main(void) { ₩n₩
20   gl_FragColor = vColor; ₩n₩
21 }";
22
23 GLuint vs = 0;
24 GLuint fs = 0;
25 GLuint prog = 0;
26
27 char buf[1024];
28 float dist = 0.0f;
29 GLuint vbo[2];
30
31 GLfloat vertices[] = {
32     -0.5, -0.5, 0.0, 1.0,
33     +0.5, -0.5, 0.0, 1.0,
34     -0.5, +0.5, 0.0, 1.0,
35 };
36
37 GLfloat colors[] = {
38     1.0, 0.0, 0.0, 1.0, // red
39     0.0, 1.0, 0.0, 1.0, // green
40     0.0, 0.0, 1.0, 1.0, // blue
41 };
42
43 GLfloat vertices2[] = {
44     -0.8, -0.8, 0.0, 1.0,
45     +0.2, -0.8, 0.0, 1.0,
46     -0.8, +0.2, 0.0, 1.0,
47 };
48
49 GLfloat colors2[] = {
50     1.0, 0.0, 0.0, 1.0,
51     1.0, 0.0, 0.0, 1.0,
52     1.0, 0.0, 0.0, 1.0,
53 };
54
55 void myinit(void) {
56     GLuint status;
```

```
57
58      printf("***** Your student number and name *****₩n");
59      vs = glCreateShader(GL_VERTEX_SHADER);
60      glShaderSource(vs, 1, &vsSource, NULL);
61      glCompileShader(vs);
62      glGetShaderiv(vs, GL_COMPILE_STATUS, &status);
63      printf("vs compile status = %s₩n", (status == GL_TRUE) ? "true" :
          "false");
64      glGetShaderInfoLog(vs, sizeof(buf), NULL, buf);
65      printf("vs log = [%s]₩n", buf);
66
67      fs = glCreateShader(GL_FRAGMENT_SHADER);
68      glShaderSource(fs, 1, &fsSource, NULL);
69      glCompileShader(fs);
70      glGetShaderiv(fs, GL_COMPILE_STATUS, &status);
71      printf("fs compile status = %s₩n", (status == GL_TRUE) ? "true" :
          "false");
72      glGetShaderInfoLog(fs, sizeof(buf), NULL, buf);
73      printf("fs log = [%s]₩n", buf);
74
75      prog = glCreateProgram();
76      glAttachShader(prog, vs);
77      glAttachShader(prog, fs);
78      glLinkProgram(prog);
79      glGetProgramiv(prog, GL_LINK_STATUS, &status);
80      printf("program link status = %s₩n", (status == GL_TRUE) ? "true" :
          "false");
81      glGetProgramInfoLog(prog, sizeof(buf), NULL, buf);
82      printf("link log = [%s]₩n", buf);
83      glValidateProgram(prog);
84      glGetProgramiv(prog, GL_VALIDATE_STATUS, &status);
85      printf("program validate status = %s₩n", (status == GL_TRUE) ? "true" :
          "false");
86      glGetProgramInfoLog(prog, sizeof(buf), NULL, buf);
87      printf("validate log = [%s]₩n", buf);
88      glUseProgram(prog);
89
90
91      // using vertex buffer object
92      glGenBuffers(2, vbo);
93
94  }
95
96  void mykeyboard(unsigned char key, int x, int y) {
97      switch (key) {
98      case 27: // ESCAPE
99          exit(0);
100         break;
101     }
102 }
103
104
105
106 void myidle(void) {
107     dist += 0.0001f;
108     if (dist > 1.5)
```

```
109         dist = 0.0f;
110
111     // redisplay
112     glutPostRedisplay();
113 }
114
115
116 void mydisplay(void) {
117     GLuint loc;
118
119     glClearColor(0.7f, 0.7f, 0.7f, 1.0f); // gray
120     glClear(GL_COLOR_BUFFER_BIT);
121
122
123     glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
124     glBufferData(GL_ARRAY_BUFFER, 2 * 3 * 4 * sizeof(GLfloat), NULL,
        GL_STATIC_DRAW);
125     glBufferSubData(GL_ARRAY_BUFFER, 0, 3 * 4 * sizeof(GLfloat), vertices);
126     glBufferSubData(GL_ARRAY_BUFFER, 3 * 4 * sizeof(GLfloat), 3 * 4 * sizeof
        (GLfloat),
127         colors);
128
129     loc = glGetAttribLocation(prog, "aPosition");
130     glEnableVertexAttribArray(loc);
131     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid *)0);
132
133     loc = glGetAttribLocation(prog, "aColor");
134     glEnableVertexAttribArray(loc);
135     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid *)(3 * 4 *
        sizeof(GLfloat)));
136
137     loc = glGetUniformLocation(prog, "udist");
138     glUniform1f(loc, dist);
139
140     glDrawArrays(GL_TRIANGLES, 0, 3);
141
142     glBindBuffer(GL_ARRAY_BUFFER, vbo[1]);
143     glBufferData(GL_ARRAY_BUFFER, 2 * 3 * 4 * sizeof(GLfloat), NULL,
        GL_STATIC_DRAW);
144     glBufferSubData(GL_ARRAY_BUFFER, 0, 3 * 4 * sizeof(GLfloat), vertices2);
145     glBufferSubData(GL_ARRAY_BUFFER, 3 * 4 * sizeof(GLfloat), 3 * 4 * sizeof
        (GLfloat),
146         colors2);
147     loc = glGetAttribLocation(prog, "aPosition");
148     glEnableVertexAttribArray(loc);
149     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid *)0);
150
151     loc = glGetAttribLocation(prog, "aColor");
152     glEnableVertexAttribArray(loc);
153     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid *)(3 * 4 *
        sizeof(GLfloat)));
154
155     loc = glGetUniformLocation(prog, "udist");
156     glUniform1f(loc, dist);
157
158     glDrawArrays(GL_TRIANGLES, 0, 3);
```

```
159
160     glFlush();
161     glutSwapBuffers();
162 }
163
164 int main(int argc, char* argv[]) {
165     glutInit(&argc, argv);
166     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
167     glutInitWindowSize(500, 500);
168     glutInitWindowPosition(0, 0);
169     glutCreateWindow("*** Your Student Number and Name ***");
170     glutDisplayFunc(mydisplay);
171     glutIdleFunc(myidle);
172     glutKeyboardFunc(mykeyboard);
173     glewInit();
174     myinit();
175     glutMainLoop();
176     return 0;
177 }
178
```