```
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  attribute vec4 aPosition; ₩n₩
8  attribute vec4 aColor; ₩n₩
9  varying 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 varying vec4 vColor; ₩n₩
19 void main(void) { ₩n₩
20   gl_FragColor = vColor; ₩n₩
21 }";
22
23 static char* vsSource2 = "#version 120 ₩n₩
24 attribute vec4 aPosition; ₩n₩
25 attribute vec4 aColor; ₩n₩
26 varying vec4 vColor; ₩n₩
27 uniform float udist; ₩n₩
28 void main(void) { ₩n₩
29   gl_Position.x = aPosition.x - udist; ₩n₩
30   gl_Position.yzw = aPosition.yzw; ₩n₩
31   vColor = aColor; ₩n₩
32 }";
33
34 static char* fsSource2 = "#version 120 ₩n₩
35 varying vec4 vColor; ₩n₩
36 void main(void) { ₩n₩
37   gl_FragColor = vColor; ₩n₩
38 }";
39
40 GLuint vs[2] = { 0,0 };
41 GLuint fs[2] = { 0, 0 };
42 GLuint prog[2] = { 0,0 };
43
44 char buf[1024];
45 GLuint vbo[2], vao[2];
46
47 GLfloat vertices[] = {
48     -0.5, -0.5, 0.0, 1.0,
49     +0.5, -0.5, 0.0, 1.0,
50     -0.5, +0.5, 0.0, 1.0,
51 };
52
53 GLfloat colors[] = {
54     1.0, 0.0, 0.0, 1.0,
55     0.0, 1.0, 0.0, 1.0,
56     0.0, 0.0, 1.0, 1.0,
```

```
57  };
58
59
60  GLfloat vertices2[] = {
61      -0.8, -0.8, 0.0, 1.0,
62      +0.2, -0.8, 0.0, 1.0,
63      -0.8, +0.2, 0.0, 1.0,
64  };
65
66  GLfloat colors2[] = {
67      1.0, 0.0, 0.0, 1.0,
68      1.0, 0.0, 0.0, 1.0,
69      1.0, 0.0, 0.0, 1.0,
70
71  };
72
73
74  void myinit(void) {
75      GLuint status;
76
77      printf("***** Your student number and name *****₩n");
78
79      for (int i = 0; i < 2; i++) {
80          vs[i] = glCreateShader(GL_VERTEX_SHADER);
81          if (i== 0)
82              glShaderSource(vs[i], 1, &vsSource, NULL);
83          else
84              glShaderSource(vs[i], 1, &vsSource2, NULL);
85          glCompileShader(vs[i]); // compile to get .OBJ
86          glGetShaderiv(vs[i], GL_COMPILE_STATUS, &status);
87          printf("vs compile status = %s₩n", (status == GL_TRUE) ? "true" :
              "false");
88          glGetShaderInfoLog(vs[i], sizeof(buf), NULL, buf);
89          printf("vs log = [%s]₩n", buf);
90
91          fs[i] = glCreateShader(GL_FRAGMENT_SHADER);
92          if (i == 0)
93              glShaderSource(fs[i], 1, &fsSource, NULL);
94          else
95              glShaderSource(fs[i], 1, &fsSource2, NULL);
96
97          glCompileShader(fs[i]); // compile to get .OBJ
98          glGetShaderiv(fs[i], GL_COMPILE_STATUS, &status);
99          printf("fs compile status = %s₩n", (status == GL_TRUE) ? "true" :
              "false");
100         glGetShaderInfoLog(fs[i], sizeof(buf), NULL, buf);
101         printf("fs log = [%s]₩n", buf);
102         // prog: program
103         prog[i] = glCreateProgram();
104         glAttachShader(prog[i], vs[i]);
105         glAttachShader(prog[i], fs[i]);
106         glLinkProgram(prog[i]); // link to get .EXE
107         glGetProgramiv(prog[i], GL_LINK_STATUS, &status);
108         printf("program link status = %s₩n", (status == GL_TRUE) ? "true" :
              "false");
109         glGetProgramInfoLog(prog[i], sizeof(buf), NULL, buf);
```

```
110         printf("link log = [%s]₩n", buf);
111         glValidateProgram(prog[i]);
112         glGetProgramiv(prog[i], GL_VALIDATE_STATUS, &status);
113         printf("program validate status = %s₩n", (status == GL_TRUE) ?
              "true" : "false");
114         glGetProgramInfoLog(prog[i], sizeof(buf), NULL, buf);
115         printf("validate log = [%s]₩n", buf);
116
117     }
118
119     glGenVertexArrays(2, vao);
120     glBindVertexArray(vao[0]);
121
122     glGenBuffers(2,
          vbo);  ////////////////////////////////////////////////////////////
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     GLuint loc;
130     loc = glGetAttribLocation(prog[0], "aPosition");
131     glEnableVertexAttribArray(loc);
132     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid*)0);
133     loc = glGetAttribLocation(prog[0], "aColor");
134     glEnableVertexAttribArray(loc);
135     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid*)(3 * 4 *
          sizeof(GLfloat)));
136
137     glBindVertexArray(vao[1]);
138
139     glBindBuffer(GL_ARRAY_BUFFER, vbo[1]);
140     glBufferData(GL_ARRAY_BUFFER, 2 * 3 * 4 * sizeof(GLfloat), NULL,
          GL_STATIC_DRAW);
141     glBufferSubData(GL_ARRAY_BUFFER, 0, 3 * 4 * sizeof(GLfloat), vertices2);
142     glBufferSubData(GL_ARRAY_BUFFER, 3 * 4 * sizeof(GLfloat), 3 * 4 * sizeof
          (GLfloat),
143         colors2);
144
145     loc = glGetAttribLocation(prog[1], "aPosition");
146     glEnableVertexAttribArray(loc);
147     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid*)0);
148     loc = glGetAttribLocation(prog[1], "aColor");
149     glEnableVertexAttribArray(loc);
150     glVertexAttribPointer(loc, 4, GL_FLOAT, GL_FALSE, 0, (GLvoid*)(3 * 4 *
          sizeof(GLfloat)));
151
152
153 }
154
155 void mykeyboard(unsigned char key, int x, int y) {
156     switch (key) {
157     case 27: // ESCAPE
```

```
158             exit(0);
159             break;
160         }
161     }
162
163     float dist = 0;
164     void mydisplay(void) {
165
166         // clear
167         glClearColor(0.7f, 0.7f, 0.7f, 1.0f); // gray
168         glClear(GL_COLOR_BUFFER_BIT);
169         GLuint loc;
170
171         glUseProgram(prog[0]);
172         loc = glGetUniformLocation(prog[0], "udist");
173         glUniform1f(loc, dist);
174         glBindVertexArray(vao[0]);
175         glDrawArrays(GL_TRIANGLES, 0, 3);
176
177         glUseProgram(prog[1]);
178         loc = glGetUniformLocation(prog[1], "udist");
179         glUniform1f(loc, dist);
180         glBindVertexArray(vao[1]);
181         glDrawArrays(GL_TRIANGLES, 0, 3);
182
183         glFlush();
184         glutSwapBuffers();
185     }
186     void myreshape(int x, int y)
187     {
188         glViewport(0, 0, x, y);
189      }
190
191
192     void myidle(void)
193     {
194         dist += 0.0001f;
195         if (dist > 1.0)
196             dist = 0;
197     //  printf("dist  %f₩n", dist);
198         glutPostRedisplay();
199     }
200     int main(int argc, char* argv[]) {
201         glutInit(&argc, argv);
202         glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
203         glutInitWindowSize(500, 500);
204         glutInitWindowPosition(0, 0);
205         glutCreateWindow("*** Your Student Number and Name ***");
206         glutDisplayFunc(mydisplay);
207         glutKeyboardFunc(mykeyboard);
208         glutReshapeFunc(myreshape);
209         glutIdleFunc(myidle);
210
211         glewInit();
212         myinit();
213         glutMainLoop();
```

```
214         return 0;
215     }
216
```