INSTRUCTIONS

EXERCISE 1 : Walkthrough Navigation

1. Create a new GLUT project.
2. Write down the following codes:

```
#include <windows.h>
#include <GL/glut.h>
#include <math.h>

// angle of rotation for the camera direction
float angle = 0.0f;

// actual vector representing the camera's direction
float lx=0.0f,lz=-1.0f;

// XZ position of the camera
float x=0.0f, z=5.0f;

// the key states. These variables will be zero
// when no key is being presses
float deltaAngle = 0.0f;
float deltaMove = 0;
int xOrigin = -1;

```

```
20  void changeSize(int w, int h)
21  {
22      // Prevent a divide by zero, when window is too short
23      // (you cant make a window of zero width).
24      if (h == 0)
25          h = 1;
26
27      float ratio =  w * 1.0 / h;
28
29      // Use the Projection Matrix
30      glMatrixMode(GL_PROJECTION);
31
32      // Reset Matrix
33      glLoadIdentity();
34
35      // Set the viewport to be the entire window
36      glViewport(0, 0, w, h);
37
38      // Set the correct perspective.
39      gluPerspective(45.0f, ratio, 0.1f, 100.0f);
40
41      // Get Back to the Modelview
42      glMatrixMode(GL_MODELVIEW);
43  }
44
45  void drawSnowMan()
46  {
47      glColor3f(1.0f, 1.0f, 1.0f);
48
49      // Draw Body
50      glTranslatef(0.0f ,0.75f, 0.0f);
51      glutSolidSphere(0.75f,20,20);
52
53      // Draw Head
54      glTranslatef(0.0f, 1.0f, 0.0f);
55      glutSolidSphere(0.25f,20,20);
56
57      // Draw Eyes
58      glPushMatrix();
59      glColor3f(0.0f,0.0f,0.0f);
60      glTranslatef(0.05f, 0.10f, 0.18f);
61      glutSolidSphere(0.05f,10,10);
62      glTranslatef(-0.1f, 0.0f, 0.0f);
63      glutSolidSphere(0.05f,10,10);
64      glPopMatrix();
65
66      // Draw Nose
```

```
67      glColor3f(1.0f, 0.5f , 0.5f);
68      glRotatef(0.0f,1.0f, 0.0f, 0.0f);
69      glutSolidCone(0.08f,0.5f,10,2);
70  }
71
72  void computePos(float deltaMove)
73  {
74      x += deltaMove * lx * 0.1f;
75      z += deltaMove * lz * 0.1f;
76  }
77
78  void renderScene(void)
79  {
80      if (deltaMove)
81          computePos(deltaMove);
82
83      // Clear Color and Depth Buffers
84      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
85
86      // Reset transformations
87      glLoadIdentity();
88      // Set the camera
89      gluLookAt(  x,      1.0f,   z,
90                  x+lx,   1.0f,   z+lz,
91                  0.0f,   1.0f,   0.0f);
92
93      // Draw ground
94      glColor3f(0.9f, 0.9f, 0.9f);
95      glBegin(GL_QUADS);
96          glVertex3f(-100.0f, 0.0f, -100.0f);
97          glVertex3f(-100.0f, 0.0f,  100.0f);
98          glVertex3f( 100.0f, 0.0f,  100.0f);
99          glVertex3f( 100.0f, 0.0f, -100.0f);
100     glEnd();
101
102     // Draw 36 SnowMen
103     for(int i = -3; i < 3; i++)
104         for(int j=-3; j < 3; j++)
105         {
106             glPushMatrix();
107             glTranslatef(i*10.0,0,j * 10.0);
108             drawSnowMan();
109             glPopMatrix();
110         }
111         glutSwapBuffers();
112 }
113
```

```
114 void processNormalKeys(unsigned char key, int xx, int yy)
115 {
116     if (key == 27)
117         exit(0);
118 }
119
120 void pressKey(int key, int xx, int yy)
121 {
122     switch (key)
123     {
124         case GLUT_KEY_UP : deltaMove = 0.5f; break;
125         case GLUT_KEY_DOWN : deltaMove = -0.5f; break;
126     }
127 }
128
129 void releaseKey(int key, int x, int y)
130 {
131     switch (key)
132     {
133          case GLUT_KEY_UP :
134          case GLUT_KEY_DOWN :
135              deltaMove = 0;
136              break;
137      }
138 }
139
140 void mouseMove(int x, int y)
141 {
142     // this will only be true when the left button is down
143     if (xOrigin >= 0)
144     {
145         // update deltaAngle
146         deltaAngle = (x - xOrigin) * 0.001f;
147
148         // update camera's direction
149         lx = sin(angle + deltaAngle);
150         lz = -cos(angle + deltaAngle);
151     }
152 }
153
```

```
154 void mouseButton(int button, int state, int x, int y)
155 {
156     // only start motion if the left button is pressed
157     if (button == GLUT_LEFT_BUTTON)
158     {
159         // when the button is released
160         if (state == GLUT_UP)
161         {
162             angle += deltaAngle;
163             xOrigin = -1;
164         }
165         else  // state = GLUT_DOWN
166         {
167             xOrigin = x;
168         }
169     }
170 }
171
172 int main(int argc, char **argv)
173 {
174     // initialize GLUT and create window
175     glutInit(&argc, argv);
176     glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
177     glutInitWindowPosition(200,100);
178     glutInitWindowSize(600,400);
179     glutCreateWindow("Walkthrough Navigation");
180
181     // register callbacks
182     glutDisplayFunc(renderScene);
183     glutReshapeFunc(changeSize);
184     glutIdleFunc(renderScene);
185
186     glutIgnoreKeyRepeat(1);
187     glutKeyboardFunc(processNormalKeys);
188     glutSpecialFunc(pressKey);
189     glutSpecialUpFunc(releaseKey);
190
191     // here are the two new functions
192     glutMouseFunc(mouseButton);
193     glutMotionFunc(mouseMove);
194
```

```
195         // OpenGL init
196         glEnable(GL_DEPTH_TEST);
197
198         // enter GLUT event processing cycle
199         glutMainLoop();
200
201         return 0;
202     }
```

4. Sample output: