7. Handling Keyboard Inputs with GLUT

We can register callback functions to handle keyboard inputs for normal and special keys, respectively.

- `glutKeyboardFunc`: registers callback handler for keyboard event.

```
void glutKeyboardFunc (void (*func)(unsigned char key, int x, int y)
   // key is the char pressed, e.g., 'a' or 27 for ESC
      // (x, y) is the mouse location in Windows' coordinates
```

- `glutSpecialFunc`: registers callback handler for special key (such as arrow keys and function keys).

```
void glutSpecialFunc (void (*func)(int specialKey, int x, int y)
   // specialKey: GLUT_KEY_* (* for LEFT, RIGHT, UP, DOWN, HOME, END, PAGE_UP,
   PAGE_DOWN, F1,...F12).
      // (x, y) is the mouse location in Windows' coordinates
```

7.1 Example 8: Switching between Full-Screen and Windowed-mode (GL08FullScreen.cpp)

For the bouncing ball program, the following special-key handler toggles between *full-screen* and *windowed modes* using F1 key.

```
/*
 * GL08FullScreen.cpp: Switching between full-screen mode and windowed-mode
 */
#include <windows.h>  // for MS Windows
#include <GL/glut.h>  // GLUT, includes glu.h and gl.h
#include <Math.h>     // Needed for sin, cos
#define PI 3.14159265f

// Global variables
char title[] = "Full-Screen & Windowed Mode";  // Windowed mode's title
int windowWidth  = 640;     // Windowed mode's width
int windowHeight = 480;     // Windowed mode's height
int windowPosX   = 50;      // Windowed mode's top-left corner x
int windowPosY   = 50;      // Windowed mode's top-left corner y

GLfloat ballRadius = 0.5f;   // Radius of the bouncing ball
GLfloat ballX = 0.0f;         // Ball's center (x, y) position
GLfloat ballY = 0.0f;
GLfloat ballXMax, ballXMin, ballYMax, ballYMin; // Ball's center (x, y) bounds
GLfloat xSpeed = 0.02f;      // Ball's speed in x and y directions
GLfloat ySpeed = 0.007f;
int refreshMillis = 30;      // Refresh period in milliseconds

// Projection clipping area
```

```
25GLdouble clipAreaXLeft, clipAreaXRight, clipAreaYBottom, clipAreaYTop;
26
27bool fullScreenMode = true; // Full-screen or windowed mode?
28
29/* Initialize OpenGL Graphics */
30void initGL() {
31   glClearColor(0.0, 0.0, 0.0, 1.0); // Set background (clear) color to black
32}
33
34/* Callback handler for window re-paint event */
35void display() {
36   glClear(GL_COLOR_BUFFER_BIT);  // Clear the color buffer
37   glMatrixMode(GL_MODELVIEW);    // To operate on the model-view matrix
38   glLoadIdentity();              // Reset model-view matrix
39
40   glTranslatef(ballX, ballY, 0.0f);  // Translate to (xPos, yPos)
41   // Use triangular segments to form a circle
42   glBegin(GL_TRIANGLE_FAN);
43      glColor3f(0.0f, 0.0f, 1.0f);  // Blue
44      glVertex2f(0.0f, 0.0f);       // Center of circle
45      int numSegments = 100;
46      GLfloat angle;
47      for (int i = 0; i <= numSegments; i++) { // Last vertex same as first vertex
48         angle = i * 2.0f * PI / numSegments;  // 360 deg for all segments
49         glVertex2f(cos(angle) * ballRadius, sin(angle) * ballRadius);
50      }
51   glEnd();
52
53   glutSwapBuffers();  // Swap front and back buffers (of double buffered mode)
54
55   // Animation Control - compute the location for the next refresh
56   ballX += xSpeed;
57   ballY += ySpeed;
58   // Check if the ball exceeds the edges
59   if (ballX > ballXMax) {
60      ballX = ballXMax;
61      xSpeed = -xSpeed;
62   } else if (ballX < ballXMin) {
63      ballX = ballXMin;
64      xSpeed = -xSpeed;
65   }
66   if (ballY > ballYMax) {
67      ballY = ballYMax;
68      ySpeed = -ySpeed;
69   } else if (ballY < ballYMin) {
```

```
 70      ballY = ballYMin;
 71      ySpeed = -ySpeed;
 72   }
 73}
 74
 75/* Call back when the windows is re-sized */
 76void reshape(GLsizei width, GLsizei height) {
 77   // Compute aspect ratio of the new window
 78   if (height == 0) height = 1;                // To prevent divide by 0
 79   GLfloat aspect = (GLfloat)width / (GLfloat)height;
 80
 81   // Set the viewport to cover the new window
 82   glViewport(0, 0, width, height);
 83
 84   // Set the aspect ratio of the clipping area to match the viewport
 85   glMatrixMode(GL_PROJECTION);  // To operate on the Projection matrix
 86   glLoadIdentity();             // Reset the projection matrix
 87   if (width >= height) {
 88      clipAreaXLeft   = -1.0 * aspect;
 89      clipAreaXRight  = 1.0 * aspect;
 90      clipAreaYBottom = -1.0;
 91      clipAreaYTop    = 1.0;
 92   } else {
 93      clipAreaXLeft   = -1.0;
 94      clipAreaXRight  = 1.0;
 95      clipAreaYBottom = -1.0 / aspect;
 96      clipAreaYTop    = 1.0 / aspect;
 97   }
 98   gluOrtho2D(clipAreaXLeft, clipAreaXRight, clipAreaYBottom, clipAreaYTop);
 99   ballXMin = clipAreaXLeft + ballRadius;
100   ballXMax = clipAreaXRight - ballRadius;
101   ballYMin = clipAreaYBottom + ballRadius;
102   ballYMax = clipAreaYTop - ballRadius;
103}
104
105/* Called back when the timer expired */
106void Timer(int value) {
107   glutPostRedisplay();    // Post a paint request to activate display()
108   glutTimerFunc(refreshMillis, Timer, 0); // subsequent timer call at milliseconds
109}
110
111/* Callback handler for special-key event */
112void specialKeys(int key, int x, int y) {
113   switch (key) {
114      case GLUT_KEY_F1:    // F1: Toggle between full-screen and windowed mode
```

```
115         fullScreenMode = !fullScreenMode;         // Toggle state
116         if (fullScreenMode) {                     // Full-screen mode
117            windowPosX   = glutGet(GLUT_WINDOW_X); // Save parameters for restoring later
118            windowPosY   = glutGet(GLUT_WINDOW_Y);
119            windowWidth  = glutGet(GLUT_WINDOW_WIDTH);
120            windowHeight = glutGet(GLUT_WINDOW_HEIGHT);
121            glutFullScreen();                      // Switch into full screen
122         } else {                                         // Windowed mode
123            glutReshapeWindow(windowWidth, windowHeight); // Switch into windowed mode
124            glutPositionWindow(windowPosX, windowPosX);   // Position top-left corner
125         }
126         break;
127   }
128}
129
130/* Main function: GLUT runs as a console application starting at main() */
131int main(int argc, char** argv) {
132   glutInit(&argc, argv);            // Initialize GLUT
133   glutInitDisplayMode(GLUT_DOUBLE); // Enable double buffered mode
134   glutInitWindowSize(windowWidth, windowHeight);  // Initial window width and height
135   glutInitWindowPosition(windowPosX, windowPosY); // Initial window top-left corner (x,
136   glutCreateWindow(title);      // Create window with given title
137   glutDisplayFunc(display);     // Register callback handler for window re-paint
138   glutReshapeFunc(reshape);     // Register callback handler for window re-shape
139   glutTimerFunc(0, Timer, 0);   // First timer call immediately
140   glutSpecialFunc(specialKeys); // Register callback handler for special-key event
141   glutFullScreen();             // Put into full screen
142   initGL();                     // Our own OpenGL initialization
143   glutMainLoop();               // Enter event-processing loop
144   return 0;
145}
```

[TODO] Explanation

[TODO] Using `glVertex` to draw a Circle is inefficient (due to the compute-intensive `sin()` and `cos()` functions). Try using GLU's quadric.

7.2 Example 9: Key-Controlled (`GL09KeyControl.cpp`)

For the bouncing ball program, the following key and special-key handlers provide exits with ESC (27), increase/decrease y speed with up-/down-arrow key, increase/decrease x speed with left-/right-arrow key, increase/decrease ball's radius with PageUp/PageDown key.

```
1/*
2 * GL09KeyControl.cpp: A key-controlled bouncing ball
3 */
4#include <windows.h>  // for MS Windows
```

```
 5#include <GL/glut.h>  // GLUT, include glu.h and gl.h
 6#include <Math.h>     // Needed for sin, cos
 7#define PI 3.14159265f
 8
 9// Global variables
10char title[] = "Full-Screen & Windowed Mode";  // Windowed mode's title
11int windowWidth  = 640;     // Windowed mode's width
12int windowHeight = 480;     // Windowed mode's height
13int windowPosX   = 50;      // Windowed mode's top-left corner x
14int windowPosY   = 50;      // Windowed mode's top-left corner y
15
16GLfloat ballRadius = 0.5f;   // Radius of the bouncing ball
17GLfloat ballX = 0.0f;        // Ball's center (x, y) position
18GLfloat ballY = 0.0f;
19GLfloat ballXMax, ballXMin, ballYMax, ballYMin; // Ball's center (x, y) bounds
20GLfloat xSpeed = 0.02f;      // Ball's speed in x and y directions
21GLfloat ySpeed = 0.007f;
22int refreshMillis = 30;      // Refresh period in milliseconds
23
24// Projection clipping area
25GLdouble clipAreaXLeft, clipAreaXRight, clipAreaYBottom, clipAreaYTop;
26
27bool fullScreenMode = true; // Full-screen or windowed mode?
28
29/* Initialize OpenGL Graphics */
30void initGL() {
31   glClearColor(0.0, 0.0, 0.0, 1.0); // Set background (clear) color to black
32}
33
34/* Callback handler for window re-paint event */
35void display() {
36   glClear(GL_COLOR_BUFFER_BIT);  // Clear the color buffer
37   glMatrixMode(GL_MODELVIEW);    // To operate on the model-view matrix
38   glLoadIdentity();              // Reset model-view matrix
39
40   glTranslatef(ballX, ballY, 0.0f);  // Translate to (xPos, yPos)
41   // Use triangular segments to form a circle
42   glBegin(GL_TRIANGLE_FAN);
43      glColor3f(0.0f, 0.0f, 1.0f);  // Blue
44      glVertex2f(0.0f, 0.0f);       // Center of circle
45      int numSegments = 100;
46      GLfloat angle;
47      for (int i = 0; i <= numSegments; i++) { // Last vertex same as first vertex
48         angle = i * 2.0f * PI / numSegments;  // 360 deg for all segments
49         glVertex2f(cos(angle) * ballRadius, sin(angle) * ballRadius);
```

```
50      }
51   glEnd();
52
53   glutSwapBuffers();  // Swap front and back buffers (of double buffered mode)
54
55   // Animation Control - compute the location for the next refresh
56   ballX += xSpeed;
57   ballY += ySpeed;
58   // Check if the ball exceeds the edges
59   if (ballX > ballXMax) {
60      ballX = ballXMax;
61      xSpeed = -xSpeed;
62   } else if (ballX < ballXMin) {
63      ballX = ballXMin;
64      xSpeed = -xSpeed;
65   }
66   if (ballY > ballYMax) {
67      ballY = ballYMax;
68      ySpeed = -ySpeed;
69   } else if (ballY < ballYMin) {
70      ballY = ballYMin;
71      ySpeed = -ySpeed;
72   }
73}
74
75/* Call back when the windows is re-sized */
76void reshape(GLsizei width, GLsizei height) {
77   // Compute aspect ratio of the new window
78   if (height == 0) height = 1;                // To prevent divide by 0
79   GLfloat aspect = (GLfloat)width / (GLfloat)height;
80
81   // Set the viewport to cover the new window
82   glViewport(0, 0, width, height);
83
84   // Set the aspect ratio of the clipping area to match the viewport
85   glMatrixMode(GL_PROJECTION);  // To operate on the Projection matrix
86   glLoadIdentity();             // Reset the projection matrix
87   if (width >= height) {
88      clipAreaXLeft   = -1.0 * aspect;
89      clipAreaXRight  = 1.0 * aspect;
90      clipAreaYBottom = -1.0;
91      clipAreaYTop    = 1.0;
92   } else {
93      clipAreaXLeft   = -1.0;
94      clipAreaXRight  = 1.0;
```

```
 95      clipAreaYBottom = -1.0 / aspect;
 96      clipAreaYTop    = 1.0 / aspect;
 97   }
 98   gluOrtho2D(clipAreaXLeft, clipAreaXRight, clipAreaYBottom, clipAreaYTop);
 99   ballXMin = clipAreaXLeft + ballRadius;
100   ballXMax = clipAreaXRight - ballRadius;
101   ballYMin = clipAreaYBottom + ballRadius;
102   ballYMax = clipAreaYTop - ballRadius;
103}
104
105/* Called back when the timer expired */
106void Timer(int value) {
107   glutPostRedisplay();    // Post a paint request to activate display()
108   glutTimerFunc(refreshMillis, Timer, 0); // subsequent timer call at milliseconds
109}
110
111/* Callback handler for normal-key event */
112void keyboard(unsigned char key, int x, int y) {
113   switch (key) {
114      case 27:     // ESC key
115         exit(0);
116         break;
117   }
118}
119
120/* Callback handler for special-key event */
121void specialKeys(int key, int x, int y) {
122   switch (key) {
123      case GLUT_KEY_F1:    // F1: Toggle between full-screen and windowed mode
124         fullScreenMode = !fullScreenMode;         // Toggle state
125         if (fullScreenMode) {                     // Full-screen mode
126            windowPosX   = glutGet(GLUT_WINDOW_X); // Save parameters for restoring later
127            windowPosY   = glutGet(GLUT_WINDOW_Y);
128            windowWidth  = glutGet(GLUT_WINDOW_WIDTH);
129            windowHeight = glutGet(GLUT_WINDOW_HEIGHT);
130            glutFullScreen();                      // Switch into full screen
131         } else {                                          // Windowed mode
132            glutReshapeWindow(windowWidth, windowHeight); // Switch into windowed mode
133            glutPositionWindow(windowPosX, windowPosX);   // Position top-left corner
134         }
135         break;
136      case GLUT_KEY_RIGHT:    // Right: increase x speed
137         xSpeed *= 1.05f; break;
138      case GLUT_KEY_LEFT:     // Left: decrease x speed
139         xSpeed *= 0.95f; break;
```

```
140      case GLUT_KEY_UP:       // Up: increase y speed
141         ySpeed *= 1.05f; break;
142      case GLUT_KEY_DOWN:     // Down: decrease y speed
143         ySpeed *= 0.95f; break;
144      case GLUT_KEY_PAGE_UP:  // Page-Up: increase ball's radius
145         ballRadius *= 1.05f;
146         ballXMin = clipAreaXLeft + ballRadius;
147         ballXMax = clipAreaXRight - ballRadius;
148         ballYMin = clipAreaYBottom + ballRadius;
149         ballYMax = clipAreaYTop - ballRadius;
150         break;
151      case GLUT_KEY_PAGE_DOWN: // Page-Down: decrease ball's radius
152         ballRadius *= 0.95f;
153         ballXMin = clipAreaXLeft + ballRadius;
154         ballXMax = clipAreaXRight - ballRadius;
155         ballYMin = clipAreaYBottom + ballRadius;
156         ballYMax = clipAreaYTop - ballRadius;
157         break;
158   }
159}
160
161/* Main function: GLUT runs as a console application starting at main() */
162int main(int argc, char** argv) {
163   glutInit(&argc, argv);            // Initialize GLUT
164   glutInitDisplayMode(GLUT_DOUBLE); // Enable double buffered mode
165   glutInitWindowSize(windowWidth, windowHeight);  // Initial window width and height
166   glutInitWindowPosition(windowPosX, windowPosY); // Initial window top-left corner (x,
167   glutCreateWindow(title);      // Create window with given title
168   glutDisplayFunc(display);     // Register callback handler for window re-paint
169   glutReshapeFunc(reshape);     // Register callback handler for window re-shape
170   glutTimerFunc(0, Timer, 0);   // First timer call immediately
171   glutSpecialFunc(specialKeys); // Register callback handler for special-key event
172   glutKeyboardFunc(keyboard);   // Register callback handler for special-key event
173   glutFullScreen();             // Put into full screen
174   initGL();                     // Our own OpenGL initialization
175   glutMainLoop();               // Enter event-processing loop
176   return 0;
177}
```

[TODO] Explanation

8. Handling Mouse Inputs with GLUT

Similarly, we can register callback function to handle mouse-click and mouse-motion.

- `glutMouseFunc`: registers callback handler for mouse click.

```
• void glutMouseFunc(void (*func)(int button, int state, int x, int y)
•    // (x, y) is the mouse-click location.
•    // button: GLUT_LEFT_BUTTON, GLUT_RIGHT_BUTTON, GLUT_MIDDLE_BUTTON
```

```
    // state: GLUT_UP, GLUT_DOWN
```

- `glutMotionFunc`: registers callback handler for mouse motion (when the mouse is clicked and moved).

```
• void glutMotionFunc(void (*func)(int x, int y)
```

```
    // where (x, y) is the mouse location in Window's coordinates
```

8.1 Example 10: Mouse-Controlled (`GL10MouseControl.cpp`)

For the bouncing ball program, the following mouse handler pause the movement with left-mouse click, and resume with right-mouse click.

```
 1/*
 2 * GL10MouseControl.cpp: A mouse-controlled bouncing ball
 3 */
 4#include <windows.h>  // for MS Windows
 5#include <GL/glut.h>  // GLUT, include glu.h and gl.h
 6#include <Math.h>     // Needed for sin, cos
 7#define PI 3.14159265f
 8
 9// Global variables
10char title[] = "Full-Screen & Windowed Mode";  // Windowed mode's title
11int windowWidth  = 640;     // Windowed mode's width
12int windowHeight = 480;     // Windowed mode's height
13int windowPosX   = 50;      // Windowed mode's top-left corner x
14int windowPosY   = 50;      // Windowed mode's top-left corner y
15
16GLfloat ballRadius = 0.5f;   // Radius of the bouncing ball
17GLfloat ballX = 0.0f;        // Ball's center (x, y) position
18GLfloat ballY = 0.0f;
19GLfloat ballXMax, ballXMin, ballYMax, ballYMin; // Ball's center (x, y) bounds
20GLfloat xSpeed = 0.02f;      // Ball's speed in x and y directions
21GLfloat ySpeed = 0.007f;
22int refreshMillis = 30;      // Refresh period in milliseconds
23
24// Projection clipping area
25GLdouble clipAreaXLeft, clipAreaXRight, clipAreaYBottom, clipAreaYTop;
26
27bool fullScreenMode = true; // Full-screen or windowed mode?
28bool paused = false;         // Movement paused or resumed
29GLfloat xSpeedSaved, ySpeedSaved;  // To support resume
30
```

```
31/* Initialize OpenGL Graphics */
32void initGL() {
33   glClearColor(0.0, 0.0, 0.0, 1.0); // Set background (clear) color to black
34}
35
36/* Callback handler for window re-paint event */
37void display() {
38   glClear(GL_COLOR_BUFFER_BIT);  // Clear the color buffer
39   glMatrixMode(GL_MODELVIEW);    // To operate on the model-view matrix
40   glLoadIdentity();              // Reset model-view matrix
41
42   glTranslatef(ballX, ballY, 0.0f);  // Translate to (xPos, yPos)
43   // Use triangular segments to form a circle
44   glBegin(GL_TRIANGLE_FAN);
45      glColor3f(0.0f, 0.0f, 1.0f);  // Blue
46      glVertex2f(0.0f, 0.0f);       // Center of circle
47      int numSegments = 100;
48      GLfloat angle;
49      for (int i = 0; i <= numSegments; i++) { // Last vertex same as first vertex
50         angle = i * 2.0f * PI / numSegments;  // 360 deg for all segments
51         glVertex2f(cos(angle) * ballRadius, sin(angle) * ballRadius);
52      }
53   glEnd();
54
55   glutSwapBuffers();  // Swap front and back buffers (of double buffered mode)
56
57   // Animation Control - compute the location for the next refresh
58   ballX += xSpeed;
59   ballY += ySpeed;
60   // Check if the ball exceeds the edges
61   if (ballX > ballXMax) {
62      ballX = ballXMax;
63      xSpeed = -xSpeed;
64   } else if (ballX < ballXMin) {
65      ballX = ballXMin;
66      xSpeed = -xSpeed;
67   }
68   if (ballY > ballYMax) {
69      ballY = ballYMax;
70      ySpeed = -ySpeed;
71   } else if (ballY < ballYMin) {
72      ballY = ballYMin;
73      ySpeed = -ySpeed;
74   }
75}
```

```
 76
 77/* Call back when the windows is re-sized */
 78void reshape(GLsizei width, GLsizei height) {
 79   // Compute aspect ratio of the new window
 80   if (height == 0) height = 1;                // To prevent divide by 0
 81   GLfloat aspect = (GLfloat)width / (GLfloat)height;
 82
 83   // Set the viewport to cover the new window
 84   glViewport(0, 0, width, height);
 85
 86   // Set the aspect ratio of the clipping area to match the viewport
 87   glMatrixMode(GL_PROJECTION);  // To operate on the Projection matrix
 88   glLoadIdentity();             // Reset the projection matrix
 89   if (width >= height) {
 90      clipAreaXLeft   = -1.0 * aspect;
 91      clipAreaXRight  = 1.0 * aspect;
 92      clipAreaYBottom = -1.0;
 93      clipAreaYTop    = 1.0;
 94   } else {
 95      clipAreaXLeft   = -1.0;
 96      clipAreaXRight  = 1.0;
 97      clipAreaYBottom = -1.0 / aspect;
 98      clipAreaYTop    = 1.0 / aspect;
 99   }
100   gluOrtho2D(clipAreaXLeft, clipAreaXRight, clipAreaYBottom, clipAreaYTop);
101   ballXMin = clipAreaXLeft + ballRadius;
102   ballXMax = clipAreaXRight - ballRadius;
103   ballYMin = clipAreaYBottom + ballRadius;
104   ballYMax = clipAreaYTop - ballRadius;
105}
106
107/* Called back when the timer expired */
108void Timer(int value) {
109   glutPostRedisplay();    // Post a paint request to activate display()
110   glutTimerFunc(refreshMillis, Timer, 0); // subsequent timer call at milliseconds
111}
112
113/* Callback handler for normal-key event */
114void keyboard(unsigned char key, int x, int y) {
115   switch (key) {
116      case 27:     // ESC key
117         exit(0);
118         break;
119   }
120}
```

```
121
122/* Callback handler for special-key event */
123void specialKeys(int key, int x, int y) {
124   switch (key) {
125      case GLUT_KEY_F1:    // F1: Toggle between full-screen and windowed mode
126         fullScreenMode = !fullScreenMode;         // Toggle state
127         if (fullScreenMode) {                     // Full-screen mode
128            windowPosX   = glutGet(GLUT_WINDOW_X); // Save parameters for restoring later
129            windowPosY   = glutGet(GLUT_WINDOW_Y);
130            windowWidth  = glutGet(GLUT_WINDOW_WIDTH);
131            windowHeight = glutGet(GLUT_WINDOW_HEIGHT);
132            glutFullScreen();                      // Switch into full screen
133         } else {                                         // Windowed mode
134            glutReshapeWindow(windowWidth, windowHeight); // Switch into windowed mode
135            glutPositionWindow(windowPosX, windowPosX);   // Position top-left corner
136         }
137         break;
138      case GLUT_KEY_RIGHT:    // Right: increase x speed
139         xSpeed *= 1.05f; break;
140      case GLUT_KEY_LEFT:     // Left: decrease x speed
141         xSpeed *= 0.95f; break;
142      case GLUT_KEY_UP:       // Up: increase y speed
143         ySpeed *= 1.05f; break;
144      case GLUT_KEY_DOWN:     // Down: decrease y speed
145         ySpeed *= 0.95f; break;
146      case GLUT_KEY_PAGE_UP:  // Page-Up: increase ball's radius
147         ballRadius *= 1.05f;
148         ballXMin = clipAreaXLeft + ballRadius;
149         ballXMax = clipAreaXRight - ballRadius;
150         ballYMin = clipAreaYBottom + ballRadius;
151         ballYMax = clipAreaYTop - ballRadius;
152         break;
153      case GLUT_KEY_PAGE_DOWN: // Page-Down: decrease ball's radius
154         ballRadius *= 0.95f;
155         ballXMin = clipAreaXLeft + ballRadius;
156         ballXMax = clipAreaXRight - ballRadius;
157         ballYMin = clipAreaYBottom + ballRadius;
158         ballYMax = clipAreaYTop - ballRadius;
159         break;
160   }
161}
162
163/* Callback handler for mouse event */
164void mouse(int button, int state, int x, int y) {
165   if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) { // Pause/resume
```

```
166      paused = !paused;          // Toggle state
167      if (paused) {
168         xSpeedSaved = xSpeed;  // Save parameters for restore later
169         ySpeedSaved = ySpeed;
170         xSpeed = 0;            // Stop movement
171         ySpeed = 0;
172      } else {
173         xSpeed = xSpeedSaved;  // Restore parameters
174         ySpeed = ySpeedSaved;
175      }
176   }
177}
178
179/* Main function: GLUT runs as a console application starting at main() */
180int main(int argc, char** argv) {
181   glutInit(&argc, argv);            // Initialize GLUT
182   glutInitDisplayMode(GLUT_DOUBLE); // Enable double buffered mode
183   glutInitWindowSize(windowWidth, windowHeight);  // Initial window width and height
184   glutInitWindowPosition(windowPosX, windowPosY); // Initial window top-left corner (x,
185   glutCreateWindow(title);      // Create window with given title
186   glutDisplayFunc(display);     // Register callback handler for window re-paint
187   glutReshapeFunc(reshape);     // Register callback handler for window re-shape
188   glutTimerFunc(0, Timer, 0);   // First timer call immediately
189   glutSpecialFunc(specialKeys); // Register callback handler for special-key event
190   glutKeyboardFunc(keyboard);   // Register callback handler for special-key event
191   glutFullScreen();             // Put into full screen
192   glutMouseFunc(mouse);   // Register callback handler for mouse event
193   initGL();                     // Our own OpenGL initialization
194   glutMainLoop();               // Enter event-processing loop
195   return 0;
196}
```