Adding Scoring and Speed Increase

Time required: 45 minutes

Contents

Every game needs a measurement of success. In our case we will include in the top left corner of the screen the score, which will be the number of times we are able to hit the ball with the racquet. On the other hand, the game should be a bit more complicated each time, so that the player doesn't get bored.

SimplePong Class

The moving objects of the game are the ball and the paddles. Changing the speed of these objects, we will modify the speed of the game. We are going to include a property called "gameSpeed" in the SimplePong class to keep the speed of the game. The property "gameSpeed" will be 1 initially, and it will increase each time we hit the ball with the racquet.

To keep score, we need a couple of new properties to increase each time we hit the ball.

Let's see what modifications need to be made to the SimplePong class:

```
20 public class SimplePong extends JPanel {
21    private static final long serialVersionUID = 1L;
22
23    // Constants for the JFrame size
24    final static int GAME_WIDTH = 800;
25    final static int GAME_HEIGHT = 500;
26
27    // How many rounds it takes to win
28    final static int WIN = 4;
29
30    // Paddle size for player and computer
31    final static int PADDLE_WIDTH = 10;
32    final static int PADDLE_HEIGHT = 100;
33
34    // Speed of the game loop
35    // Decrease for faster, increase for slower
36    final static int GAME_LOOP_SPEED = 17;
37
38    // Variable for the speed of the game
39    static int gameSpeed = 1;
40
41    // Keep track of score
42    int playerScore = 0;
43    int computerScore = 0;
```

To paint the fonts, at the top of the class, we need to add a reference to

```
10 import java.awt.Font;
```

```
82    @Override // Override the default paint method
83    public void paint(Graphics g) {
84       super.paint(g);            // Clear the window
85       setBackground(Color.WHITE); // Set window background to White
86       Graphics2D g2d = (Graphics2D) g;
87       g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
88                            RenderingHints.VALUE_ANTIALIAS_ON);
89
90       // Override the game objects paint methods
91       ball.paint(g2d);
92       player.paint(g2d);
93       computer.paint(g2d);
94
95       // Display score
96       g2d.setColor(Color.GRAY);
97       g2d.setFont(new Font("Verdana", Font.BOLD, 14));
98       g2d.drawString("Player Score: " + String.valueOf(playerScore), 40, 15);
99       g2d.drawString("Computer Score: " + String.valueOf(computerScore),
100                     GAME_WIDTH - 200, 15);
101   }
102
103   // Game Over called from Ball object
104   public void gameOver() {
105      Sound.BACKGROUND.stop();
106      Sound.GAMEOVER.play();
107      String msg = "Player score is: " + String.valueOf(playerScore)
108                 + "\nComputer score is: " + String.valueOf(computerScore);
109      JOptionPane.showMessageDialog(this, msg, "Game Over", JOptionPane.YES_NO_OPTION);
110      System.exit(ABORT);
111   }
```

To paint the score in the top left corner, we add the following code at the end of the paint method:

```
g2d.setColor(Color.GRAY);
g2d.setFont(new Font("Verdana", Font.BOLD, 30));
g2d.drawString(String.valueOf(getScore()), 10, 30);
```

In the first line we choose the color; grey, in the second line the type of letter; Verdana, bold type of 30 pixels and finally the position (x, y) = (10, 30), where we paint the punctuation.

In the gameOver() method, we modify the second parameter to show the score:

```
JOptionPane.showMessageDialog(this,
"Your score is: " + getScore(), "Game Over",
JOptionPane.YES_NO_OPTION);
```

The variable in line 26 needs to be changed to GAME_LOOP_SPEED

```
122         // Game loop, loops forever
123         while (true) {
124            simplePong.move();             // Call the move methods
125            simplePong.repaint();          // Repaint the application screen
126            Thread.sleep(GAME_LOOP_SPEED);  // Pause thread to let JFrame redraw
127         }
128      }
129   }
```

Simple Pong Code Complete

```
 9 import java.awt.Color;
10 import java.awt.Font;
11 import java.awt.Graphics;
12 import java.awt.Graphics2D;
13 import java.awt.RenderingHints;
14 import java.awt.event.KeyEvent;
15 import java.awt.event.KeyListener;
16 import javax.swing.JFrame;
17 import javax.swing.JOptionPane;
18 import javax.swing.JPanel;
19
20 public class SimplePong extends JPanel {
21    private static final long serialVersionUID = 1L;
22
23    // Constants for the JFrame size
24    final static int GAME_WIDTH = 800;
25    final static int GAME_HEIGHT = 500;
26
27    // How many rounds it takes to win
28    final static int WIN = 4;
29
30    // Paddle size for player and computer
31    final static int PADDLE_WIDTH = 10;
32    final static int PADDLE_HEIGHT = 100;
33
34    // Speed of the game loop
35    // Decrease for faster, increase for slower
36    final static int GAME_LOOP_SPEED = 17;
37
38    // Variable for the speed of the game
39    static int gameSpeed = 1;
40
41    // Keep track of score
42    int playerScore = 0;
43    int computerScore = 0;
44
45    // Create Ball and Paddle objects
46    Ball ball = new Ball(this);
47    Player player = new Player(this);
48    Computer computer = new Computer(this);
49
50    // Construct the Game application
51    public SimplePong() {
```

```
52          Sound.init();               // Load all sound files in memory
53          Sound.volume = Sound.Volume.LOW; // Set sound volume
54
55          // Add KeyListener to the application
56          addKeyListener(new KeyListener() {
57             @Override
58             public void keyTyped(KeyEvent e) {
59             }
60
61             @Override
62             public void keyReleased(KeyEvent e) {
63                player.keyReleased(e);
64             }
65
66             @Override
67             public void keyPressed(KeyEvent e) {
68                player.keyPressed(e);
69             }
70          });
71          setFocusable(true);   // Allow keyboard events to be captured from JFrame
72          Sound.BACKGROUND.loop(); // Loop background sound
73       }
74
75       // Move the Ball and Paddles
76       private void move() {
77          ball.move();
78          player.move();
79          computer.move();
80       }
81
82       @Override // Override the default paint method
83       public void paint(Graphics g) {
84          super.paint(g);            // Clear the window
85          setBackground(Color.WHITE); // Set window background to White
86          Graphics2D g2d = (Graphics2D) g;
87          g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
88                               RenderingHints.VALUE_ANTIALIAS_ON);
89
90          // Override the game objects paint methods
91          ball.paint(g2d);
92          player.paint(g2d);
93          computer.paint(g2d);
94
95          // Display score
96          g2d.setColor(Color.GRAY);
97          g2d.setFont(new Font("Verdana", Font.BOLD, 14));
98          g2d.drawString("Player Score: " + String.valueOf(playerScore), 40, 15);
99          g2d.drawString("Computer Score: " + String.valueOf(computerScore),
```

```
100                     GAME_WIDTH - 200, 15);
101    }
102
103    // Game Over called from Ball object
104    public void gameOver() {
105       Sound.BACKGROUND.stop();
106       Sound.GAMEOVER.play();
107       String msg = "Player score is: " + String.valueOf(playerScore)
108                  + "\nComputer score is: " + String.valueOf(computerScore);
109       JOptionPane.showMessageDialog(this, msg, "Game Over", JOptionPane.YES_NO_OPTION);
110       System.exit(ABORT);
111    }
112
113    public static void main(String[] args) throws InterruptedException {
114       JFrame frame = new JFrame("Simple Pong");
115       SimplePong simplePong = new SimplePong();
116       frame.add(simplePong);
117       frame.setSize(GAME_WIDTH, GAME_HEIGHT);
118       frame.setVisible(true);
119       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
120
121       // Game loop, loops forever
122       while (true) {
123          simplePong.move();            // Call the move methods
124          simplePong.repaint();         // Repaint the application screen
125          Thread.sleep(GAME_LOOP_SPEED);  // Pause thread to let JFrame redraw
126       }
127    }
128 }
```

Ball Class

The move() method of the Ball class has been modified to take into account the new property “simplePong.gameSpeed". When the ball changed direction, the properties of speed "MoveX" and "MoveY" were changed to 1 or -1. Now, taking into account speed, these properties, change to + simplePong.speed or - simplePong.gameSpeed.

```
24    // Move the ball, check for collision
25    void move() {
26
27       // Move the ball by adding x, y integers to current location
28       BallX = BallX + MoveX;
29       BallY = BallY + MoveY;
30
31       // If the ball hits either paddle, reverse direction, play sound
32       if (simplePong.player.getBounds().intersects(getBounds())
33           || simplePong.computer.getBounds().intersects(getBounds()))
34       {
35          Sound.BALL.play();
36          MoveX = -MoveX;   // Reverse horizontal direction
37       }
38
39       // If the ball runs into the top or botton border, reverse direction
40       if (BallY < 0 || BallY + BALL_DIAMETER > simplePong.getHeight())
41       {
42          MoveY = -MoveY;   // Reverse the vertical direction of the ball
43       }
44
45       // If the ball runs into the left border, Computer wins
46       if (BallX + MoveX < 0)
47       {
48          MoveX = -MoveX;    // Reverse the horizontal direction of the ball
49          MoveX = MoveX + simplePong.gameSpeed;
50          increaseMoveY();
51          BallX = simplePong.getWidth() /2;
52          simplePong.computerScore++;
53       }
54
55       // If the ball runs into the right border, Player wins
56       if (BallX + BALL_DIAMETER > simplePong.getWidth())
57       {
58          MoveX = -MoveX;    // Reverse the horizontal direction of the ball
59          MoveX = MoveX - simplePong.gameSpeed;
60          increaseMoveY();
61          BallX = simplePong.getWidth() /2;
62          simplePong.playerScore++;
63       }
64
65       if(simplePong.playerScore >= simplePong.WIN)
66       {
67          simplePong.gameOver();
68       }
69
70       if(simplePong.computerScore >= simplePong.WIN)
71       {
72          simplePong.gameOver();
73       }
74    }
```

```
76    void increaseMoveY()
77    {
78       if(MoveY < 0)
79       {
80          MoveY = MoveY - simplePong.gameSpeed;
81       }
82       else
83       {
84          MoveY = MoveY + simplePong.gameSpeed;
85       }
86    }
```

Our game is complete.

Challenge

What can you add to this game to make it more your own and more interesting?

Assignment Submission

Attach the .java files to the assignment in Blackboard.