

randomstuff\SandSimulation.java

```
1 import javax.swing.*;
2 import java.awt.*;
3 import java.awt.event.*;
4 import java.util.ArrayList;
5 import java.util.List;
6
7 public class SandSimulation extends JPanel implements ActionListener {
8
9     private static final int WIDTH = 800;
10    private static final int HEIGHT = 600;
11    private static final int SAND_SIZE = 10;
12    private static final int GRAB_RADIUS = 10 * SAND_SIZE;
13
14    private boolean[][][] sandGrid;
15    private List<Particle> particles;
16    private boolean isLeftMouseDown = false;
17
18    private Point grabOffset = new Point(0, 0);
19
20    public SandSimulation() {
21        sandGrid = new boolean[WIDTH / SAND_SIZE][HEIGHT / SAND_SIZE];
22        particles = new ArrayList<>();
23
24        setPreferredSize(new Dimension(WIDTH, HEIGHT));
25        setBackground(Color.WHITE);
26
27        addMouseListener(new MouseAdapter() {
28            @Override
29            public void mousePressed(MouseEvent e) {
30                if (e.getButton() == MouseEvent.BUTTON1) {
31                    isLeftMouseDown = true;
32                }
33            }
34
35            @Override
36            public void mouseReleased(MouseEvent e) {
37                if (e.getButton() == MouseEvent.BUTTON1) {
38                    isLeftMouseDown = false;
39                } else if (e.getButton() == MouseEvent.BUTTON3) {
40
41                    releaseParticles();
42                }
43            }
44        });
45
46        addMouseMotionListener(new MouseAdapter() {
47            @Override
48            public void mouseDragged(MouseEvent e) {
49                if (isLeftMouseDown) {
50                    addSandParticle(e.getX(), e.getY());
51                }
52            }
53        });
54    }
55
56    private void addSandParticle(int x, int y) {
57        Particle particle = new Particle(x, y);
58        particles.add(particle);
59
60        int sandX = x / SAND_SIZE;
61        int sandY = y / SAND_SIZE;
62
63        sandGrid[sandX][sandY] = true;
64
65        int radius = GRAB_RADIUS / 2;
66
67        for (int i = -radius; i <= radius; i++) {
68            for (int j = -radius; j <= radius; j++) {
69                int nx = sandX + i;
70                int ny = sandY + j;
71
72                if (nx < 0 || ny < 0 || nx >= sandGrid.length || ny >= sandGrid[0].length)
73                    continue;
74
75                sandGrid[nx][ny] = true;
76            }
77        }
78    }
79
80    private void releaseParticles() {
81        for (Particle particle : particles) {
82            particle.setVelocity(0, 0);
83        }
84    }
85
86    @Override
87    public void actionPerformed(ActionEvent e) {
88        repaint();
89    }
90
91    protected void paintComponent(Graphics g) {
92        super.paintComponent(g);
93
94        for (int x = 0; x < sandGrid.length; x++) {
95            for (int y = 0; y < sandGrid[0].length; y++) {
96                if (sandGrid[x][y]) {
97                    g.fillRect(x * SAND_SIZE, y * SAND_SIZE, SAND_SIZE, SAND_SIZE);
98                }
99            }
100        }
101    }
102}
```

```

54
55     Timer timer = new Timer(5, this);
56     timer.start();
57 }
58
59 @Override
60 public void actionPerformed(ActionEvent e) {
61     simulateSand();
62     repaint();
63 }
64
65 private void addSandParticle(int mouseX, int mouseY) {
66     int x = mouseX / SAND_SIZE;
67     int y = mouseY / SAND_SIZE;
68
69     if (x >= 0 && x < WIDTH / SAND_SIZE && y >= 0 && y < HEIGHT / SAND_SIZE) {
70         sandGrid[x][y] = true;
71     }
72 }
73
74 private void simulateSand() {
75     boolean[][] newSandGrid = new boolean[WIDTH / SAND_SIZE][HEIGHT / SAND_SIZE];
76
77     for (int x = 0; x < WIDTH / SAND_SIZE; x++) {
78         for (int y = 0; y < HEIGHT / SAND_SIZE; y++) {
79             if (sandGrid[x][y] && (y == HEIGHT / SAND_SIZE - 1 || sandGrid[x][y + 1])) {
80                 // Particle has reached the bottom or landed on another particle
81                 newSandGrid[x][y] = true;
82             } else if (sandGrid[x][y] && !sandGrid[x][y + 1]) {
83                 // Particle can fall one step down
84                 newSandGrid[x][y + 1] = true;
85             }
86         }
87     }
88
89     sandGrid = newSandGrid;
90 }
91
92 private void grabParticles(int mouseX, int mouseY) {
93     int centerX = mouseX / SAND_SIZE;
94     int centerY = mouseY / SAND_SIZE;
95
96
97     // Collect particles within the grab radius
98     for (int x = centerX - 10; x <= centerX + 10; x++) {
99         for (int y = centerY - 10; y <= centerY + 10; y++) {
100            if (x >= 0 && x < WIDTH / SAND_SIZE && y >= 0 && y < HEIGHT / SAND_SIZE) {
101                if (sandGrid[x][y]) {
102                    particles.add(new Particle(x, y, true));
103                }
104            }
105        }
106    }
107 }
108
109 // Calculate the offset of the grab position from the first grabbed particle

```

```
110     if (!particles.isEmpty()) {
111         Particle firstParticle = particles.get(0);
112         grabOffset.setLocation(centerX - firstParticle.x, centerY - firstParticle.y);
113     }
114 }
115
116 private void releaseParticles() {
117     for (Particle particle : particles) {
118         particle.isGrabbed = false;
119     }
120 }
121
122
123 private void moveGrabbedParticles(int mouseX, int mouseY) {
124     int centerX = mouseX / SAND_SIZE - grabOffset.x;
125     int centerY = mouseY / SAND_SIZE - grabOffset.y;
126
127     // Move the grabbed particles
128     for (Particle particle : particles) {
129         int newX = centerX + particle.x - particles.get(0).x;
130         int newY = centerY + particle.y - particles.get(0).y;
131
132         if (newX >= 0 && newX < WIDTH / SAND_SIZE && newY >= 0 && newY < HEIGHT / SAND_SIZE) {
133             sandGrid[newX][newY] = true;
134             sandGrid[particle.x][particle.y] = false;
135             particle.setLocation(newX, newY);
136         }
137     }
138 }
139
140 @Override
141 protected void paintComponent(Graphics g) {
142     super.paintComponent(g);
143
144     for (int x = 0; x < WIDTH / SAND_SIZE; x++) {
145         for (int y = 0; y < HEIGHT / SAND_SIZE; y++) {
146             if (sandGrid[x][y]) {
147                 g.setColor(Color.YELLOW);
148                 g.fillRect(x * SAND_SIZE, y * SAND_SIZE, SAND_SIZE, SAND_SIZE);
149             }
150         }
151     }
152
153     // Draw a circle to indicate the grab radius when right-clicking
154 }
155
156
157 public static void main(String[] args) {
158     SwingUtilities.invokeLater(() -> {
159         JFrame frame = new JFrame("Sand Simulation");
160         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
161         frame.getContentPane().add(new SandSimulation());
162         frame.pack();
163         frame.setLocationRelativeTo(null);
164         frame.setVisible(true);
165     });
166 }
```

```
165     });
166 }
167
168 private class Particle extends Point {
169     boolean isGrabbed;
170
171     Particle(int x, int y, boolean isGrabbed) {
172         super(x, y);
173         this.isGrabbed = isGrabbed;
174     }
175 }
176 }
```