randomstuff\SandSimulation.java

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.List;

public class SandSimulation extends JPanel implements ActionListener {

    private static final int WIDTH = 800;
    private static final int HEIGHT = 600;
    private static final int SAND_SIZE = 10;
    private static final int GRAB_RADIUS = 10 * SAND_SIZE;

    private boolean[][] sandGrid;
    private List<Particle> particles;
    private boolean isLeftMouseDown = false;

    private Point grabOffset = new Point(0, 0);

    public SandSimulation() {
        sandGrid = new boolean[WIDTH / SAND_SIZE][HEIGHT / SAND_SIZE];
        particles = new ArrayList<>();

        setPreferredSize(new Dimension(WIDTH, HEIGHT));
        setBackground(Color.WHITE);

        addMouseListener(new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                if (e.getButton() == MouseEvent.BUTTON1) {
                    isLeftMouseDown = true;
                }
            }

            @Override
            public void mouseReleased(MouseEvent e) {
                if (e.getButton() == MouseEvent.BUTTON1) {
                    isLeftMouseDown = false;
                } else if (e.getButton() == MouseEvent.BUTTON3) {

                    releaseParticles();
                }
            }
        });

        addMouseMotionListener(new MouseAdapter() {
            @Override
            public void mouseDragged(MouseEvent e) {
                if (isLeftMouseDown) {
                    addSandParticle(e.getX(), e.getY());
                }
            }
        });
```

```
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
98          // Collect particles within the grab radius
99          for (int x = centerX - 10; x <= centerX + 10; x++) {
100             for (int y = centerY - 10; y <= centerY + 10; y++) {
101                 if (x >= 0 && x < WIDTH / SAND_SIZE && y >= 0 && y < HEIGHT / SAND_SIZE) {
102                     if (sandGrid[x][y]) {
103                         particles.add(new Particle(x, y, true));
104                     }
105                 }
106             }
107         }
108
109         // Calculate the offset of the grab position from the first grabbed particle
```

```
        if (!particles.isEmpty()) {
            Particle firstParticle = particles.get(0);
            grabOffset.setLocation(centerX - firstParticle.x, centerY - firstParticle.y);
        }
    }

    private void releaseParticles() {
        for (Particle particle : particles) {
            particle.isGrabbed = false;
        }

    }

    private void moveGrabbedParticles(int mouseX, int mouseY) {
        int centerX = mouseX / SAND_SIZE - grabOffset.x;
        int centerY = mouseY / SAND_SIZE - grabOffset.y;

        // Move the grabbed particles
        for (Particle particle : particles) {
            int newX = centerX + particle.x - particles.get(0).x;
            int newY = centerY + particle.y - particles.get(0).y;

            if (newX >= 0 && newX < WIDTH / SAND_SIZE && newY >= 0 && newY < HEIGHT /
SAND_SIZE) {
                sandGrid[newX][newY] = true;
                sandGrid[particle.x][particle.y] = false;
                particle.setLocation(newX, newY);
            }
        }
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        for (int x = 0; x < WIDTH / SAND_SIZE; x++) {
            for (int y = 0; y < HEIGHT / SAND_SIZE; y++) {
                if (sandGrid[x][y]) {
                    g.setColor(Color.YELLOW);
                    g.fillRect(x * SAND_SIZE, y * SAND_SIZE, SAND_SIZE, SAND_SIZE);
                }
            }
        }

        // Draw a circle to indicate the grab radius when right-clicking

    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            JFrame frame = new JFrame("Sand Simulation");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.getContentPane().add(new SandSimulation());
            frame.pack();
            frame.setLocationRelativeTo(null);
            frame.setVisible(true);
```

```
        });
    }

    private class Particle extends Point {
        boolean isGrabbed;

        Particle(int x, int y, boolean isGrabbed) {
            super(x, y);
            this.isGrabbed = isGrabbed;
        }
    }
}
```