Animated Dice Roller Application

A Java Swing application that simulates rolling dice with beautiful 3D animation effects, perfect for board games like Ludo or Monopoly.

🎲

Realistic 3D Dice

Custom-designed dice buttons with proper dot patterns that visually represent each possible roll (1-6).

🔄

Smooth Animation

Rotating animation effect that makes the dice appear to tumble before settling on their final values.

➕

Automatic Total Calculation

The application automatically calculates and displays the sum of both dice for easy game play.

🎨

Visual Appeal

Clean, modern UI with anti-aliased graphics, 3D effects, and responsive design elements.

⚙️

Game-Ready

Perfect for integrating into board game applications or as a standalone dice-rolling utility.

💻

Java Swing

Built with standard Java libraries for maximum compatibility across different operating systems.

DiceRoller.java - Main Application Code

Copy Code

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Random;
import javax.swing.Timer;

public class DiceRoller extends JFrame {
    private DiceButton diceButton1;
    private DiceButton diceButton2;
    private JButton rollButton;
    private JLabel totalLabel;
    private Timer animationTimer;
    private int animationFrame = 0;
    private final int ANIMATION_FRAMES = 20;
    private final int ANIMATION_DELAY = 50;

    public DiceRoller() {
        setTitle("Animated Dice Roller");
        setSize(500, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());
        setLocationRelativeTo(null);

        // Create components
        JPanel dicePanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 30, 30));
        dicePanel.setBorder(BorderFactory.createEmptyBorder(20, 0, 20, 0));

        // Create dice buttons
        diceButton1 = new DiceButton(1);
        diceButton2 = new DiceButton(1);
        dicePanel.add(diceButton1);
        dicePanel.add(diceButton2);

        rollButton = new JButton("Roll Dice");
        rollButton.setFont(new Font("Arial", Font.BOLD, 18));
        rollButton.setPreferredSize(new Dimension(150, 50));

        totalLabel = new JLabel("Total: 0", SwingConstants.CENTER);
        totalLabel.setFont(new Font("Arial", Font.BOLD, 28));

        // Add action listener to roll button
        rollButton.addActionListener(e -> startRollAnimation());
        // Setup animation timer
        animationTimer = new Timer(ANIMATION_DELAY, e -> {
            if (animationFrame < ANIMATION_FRAMES) {
                // Rotate and show random values during animation
                diceButton1.rotate();
                diceButton2.rotate();
                diceButton1.setValue(new Random().nextInt(6) + 1);
                diceButton2.setValue(new Random().nextInt(6) + 1);
                animationFrame++;
            } else {
                // Animation complete
                animationTimer.stop();
                rollButton.setEnabled(true);
                int val1 = new Random().nextInt(6) + 1;
                int val2 = new Random().nextInt(6) + 1;
                diceButton1.setValue(val1);
                diceButton2.setValue(val2);
                diceButton1.resetRotation();
                diceButton2.resetRotation();
                totalLabel.setText("Total: " + (val1 + val2));
            }
        });

        // Add components to frame
        add(dicePanel, BorderLayout.CENTER);
        add(rollButton, BorderLayout.SOUTH);
        add(totalLabel, BorderLayout.NORTH);
    }

    private void startRollAnimation() {
        rollButton.setEnabled(false);
        animationFrame = 0;
        animationTimer.start();
    }

    class DiceButton extends JButton {
        private int value;
        private double rotation = 0;
```

```java
    public DiceButton(int value) {
        super(String.valueOf(value));
        this.value = value;
        setPreferredSize(new Dimension(100, 100));
        setFont(new Font("Arial", Font.BOLD, 48));
        setBorder(BorderFactory.createRaisedBevelBorder());
        setBackground(Color.WHITE);
        setFocusPainted(false);
        setContentAreaFilled(false);
        setOpaque(true);
    }

    public void setValue(int value) {
        this.value = value;
        setText(String.valueOf(value));
    }

    public void rotate() {
        rotation += 45;
        repaint();
    }

    public void resetRotation() {
        rotation = 0;
        repaint();
    }

    @Override
    protected void paintComponent(Graphics g) {
        Graphics2D g2d = (Graphics2D) g.create();

        // Enable anti-aliasing for smoother graphics
        g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                         RenderingHints.VALUE_ANTIALIAS_ON);

        // Clear the background
        g2d.setColor(getBackground());
        g2d.fillRect(0, 0, getWidth(), getHeight());

        // Draw 3D border
        g2d.setColor(Color.LIGHT_GRAY);
        g2d.fill3DRect(0, 0, getWidth(), getHeight(), true);

        // Apply rotation transformation
        g2d.rotate(Math.toRadians(rotation), getWidth()/2, getHeight()/2);

        // Draw dice face
        g2d.setColor(Color.BLACK);
        g2d.drawRoundRect(5, 5, getWidth()-10, getHeight()-10, 20, 20);

        // Draw dots based on value
        int dotSize = 15;
        int centerX = getWidth()/2;
        int centerY = getHeight()/2;

        g2d.setColor(Color.BLACK);
        switch(value) {
            case 1:
                g2d.fillOval(centerX-dotSize/2, centerY-dotSize/2, dotSize, dotSize);
                break;
            case 2:
                g2d.fillOval(centerX-dotSize-10, centerY-dotSize-10, dotSize, dotSize);
                g2d.fillOval(centerX+10, centerY+10, dotSize, dotSize);
                break;
            case 3:
                g2d.fillOval(centerX-dotSize-10, centerY-dotSize-10, dotSize, dotSize);
                g2d.fillOval(centerX-dotSize/2, centerY-dotSize/2, dotSize, dotSize);
                g2d.fillOval(centerX+10, centerY+10, dotSize, dotSize);
                break;
            case 4:
                g2d.fillOval(centerX-dotSize-10, centerY-dotSize-10, dotSize, dotSize);
                g2d.fillOval(centerX+10, centerY-dotSize-10, dotSize, dotSize);
                g2d.fillOval(centerX-dotSize-10, centerY+10, dotSize, dotSize);
                g2d.fillOval(centerX+10, centerY+10, dotSize, dotSize);
                break;
            case 5:
                g2d.fillOval(centerX-dotSize-10, centerY-dotSize-10, dotSize, dotSize);
                g2d.fillOval(centerX+10, centerY-dotSize-10, dotSize, dotSize);
                g2d.fillOval(centerX-dotSize/2, centerY-dotSize/2, dotSize, dotSize);
                g2d.fillOval(centerX-dotSize-10, centerY+10, dotSize, dotSize);
                g2d.fillOval(centerX+10, centerY+10, dotSize, dotSize);
                break;
            case 6:
                g2d.fillOval(centerX-dotSize-10, centerY-dotSize-10, dotSize, dotSize);
                g2d.fillOval(centerX+10, centerY-dotSize-10, dotSize, dotSize);
                g2d.fillOval(centerX-dotSize-10, centerY-dotSize/2, dotSize, dotSize);
                g2d.fillOval(centerX+10, centerY-dotSize/2, dotSize, dotSize);
                g2d.fillOval(centerX-dotSize-10, centerY+10, dotSize, dotSize);
                g2d.fillOval(centerX+10, centerY+10, dotSize, dotSize);
                break;
        }
        g2d.dispose();
      }
    }

    public static void main(String[] args) {
      SwingUtilities.invokeLater(() -> {
        DiceRoller roller = new DiceRoller();
        roller.setVisible(true);
      });
    }
}
```

Ready to Roll?

Copy this complete Java source code and integrate this beautiful dice roller into your board game application today!

Copy Code to Clipboard