```java
import javax.swing.JApplet;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.ImageIcon;
import javax.swing.SwingUtilities;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
//import java.awt.geom.Point2D;


public class AnimatedImage extends JApplet implements Runnable
{

    private JPanel canvas, buttonPanel;
    private JButton stopButton, startButton;
    private JComboBox<String> modePicker;
    private JLabel modeLabel;

    private int width,
                height,
                mesh,
                steps,
                deltaT,
                deltaW,
                deltaH;
    private boolean runThread = true,
                    doAnimate = false,
                    toggle = true,
                    animationThreadRunning = false;
    private Image blueSphere;
    private String mode;

    @Override
    public void init()
    {
        width   = Integer.parseInt(getParameter("width"));
        height  = Integer.parseInt(getParameter("height")) - 40;
        mesh    = Integer.parseInt(getParameter("mesh"));
        steps   = Integer.parseInt(getParameter("steps"));
        deltaT  = Integer.parseInt(getParameter("deltaT"));
        deltaW  = width  / mesh;
        deltaH  = height / mesh;
```

```java
        mode    = getParameter("mode");
        blueSphere = new ImageIcon(getClass().getResource("sphere.jpg")).getImage();
        blueSphere = blueSphere.getScaledInstance(width, height, Image.SCALE_DEFAULT);

        canvas = new JPanel()
            {
                @Override
                public void paintComponent(Graphics g)
                {
                    super.paintComponent(g);
                    setPreferredSize(new Dimension(width, height));
                    Graphics2D g2D = (Graphics2D)g;
                    /*
                    g2D.setPaint(new GradientPaint(
                            new Point2D.Double(width / 8, height / 8), Color.GREEN,
                            new Point2D.Double(width / 4, height / 4), Color.CYAN,
                            true));
                    g2D.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, .5f));
                    g2D.fillRect(0, 0, width, height);
                    */
                    g2D.drawImage(blueSphere, 0, 0, width, height, Color.BLACK, this);
                    g2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                                         RenderingHints.VALUE_ANTIALIAS_ON);
                    g2D.setStroke(new BasicStroke(1));
                    for (int i = 0; i < mesh; i++)
                    {

                        if (mode.equals("curtains"))
                        {
                            g2D.setColor(Color.BLUE);
                            g2D.drawLine(0, i * deltaH, (i + 1)  * deltaW, height);
                            g2D.setColor(Color.RED);
                            g2D.drawLine(width, height - i * deltaH, width - (i + 1) * deltaW, 0);
                            g2D.setColor(Color.MAGENTA);
                            g2D.drawLine(0, height - (i * deltaH), (i + 1) * deltaW, 0);
                            g2D.setColor(Color.MAGENTA);
                            g2D.drawLine(width, i * deltaH, width - ((i + 1) * deltaW), height);
                        }
                        else
                        if (mode.equals("cradle") ||  toggle)
                        {
                            g2D.setColor(Color.BLUE);
                            g2D.drawLine(0, i * deltaH, (i + 1)  * deltaW, height);
                            g2D.setColor(Color.RED);
                            g2D.drawLine(width, height - i * deltaH, width - (i + 1) * deltaW, 0);
```

```
                            g2D.setColor(Color.MAGENTA);
                            g2D.drawLine(width - i * deltaW, 0, 0, (i + 1) * deltaH);
                            g2D.setColor(Color.MAGENTA);
                            g2D.drawLine(i * deltaW , height, width, height - i * deltaH);
                        }
                        else
                        {
                            g2D.setColor(Color.BLUE);
                            g2D.drawLine(width - i * deltaW, height, 0, height - (i + 1) * deltaH);
                            g2D.setColor(Color.RED);
                            g2D.drawLine(i * deltaW, 0, width, (i + 1) * deltaH);
                            g2D.setColor(Color.MAGENTA);
                            g2D.drawLine(0, height - (i * deltaH), (i + 1) * deltaW, 0);
                            g2D.setColor(Color.MAGENTA);
                            g2D.drawLine(width, i * deltaH, width - ((i + 1) * deltaW), height);
                        }
                    }
                }
            };
        add(canvas, BorderLayout.CENTER);
        buttonPanel = new JPanel();
        buttonPanel.setPreferredSize(new Dimension(width, 40));
        buttonPanel.setBackground(Color.WHITE);
        modeLabel = new JLabel("Mode:");
        buttonPanel.add(modeLabel);
        modePicker = new JComboBox<String>(new String[] {"curtains", "cradle", "wheel"});
        modePicker.setSelectedItem(mode);
        modePicker.addActionListener(
                new ActionListener()
                {
                    public void actionPerformed(ActionEvent e)
                    {
                        destroy();
                        stopButton.setEnabled(true);
                        startButton.setEnabled(false);
                        new Thread()
                        {
                            public void run()
                            {
                                while (animationThreadRunning)
                                {
                                    try {
                                        Thread.sleep(deltaT);
                                    } catch (InterruptedException ex) { Thread.currentThread().interrupt(); }
                                }
```

```
                                runThread = true;
                                doAnimate = true;
                                deltaW  = width  / mesh;
                                deltaH  = height / mesh;
                                toggle = true;
                                mode    = (String)modePicker.getSelectedItem();
                                repaint();
                                new Thread(AnimatedImage.this).start();
                            }
                        }.start();
                    }
                }
            );
        buttonPanel.add(modePicker);
        startButton = new JButton("Start");
        startButton.addActionListener(
                new ActionListener()
                {
                    public void actionPerformed(ActionEvent e)
                    {
                        start();
                    }
                }
            );
        buttonPanel.add(startButton);
        stopButton = new JButton("Stop");
        stopButton.addActionListener(
                new ActionListener()
                {
                    public void actionPerformed(ActionEvent e)
                    {
                        stop();
                    }
                }
            );
        buttonPanel.add(stopButton);
        add(buttonPanel, BorderLayout.SOUTH);
        new Thread(this).start();
        setVisible(true);
    }

    @Override
    public void run()
    {
        synchronized(this)
```

```
        {
            animationThreadRunning = true;

            while(runThread)
            {

                while(runThread && !doAnimate)
                    try {
                        wait();
                    } catch (InterruptedException ex) { Thread.currentThread().interrupt(); }

                while(runThread && doAnimate)
                {

                    try {
                        wait(3 * deltaT);
                    } catch (InterruptedException ex) { Thread.currentThread().interrupt(); }

                    for (int i = steps - 1; runThread && i >= 0; i--)
                    {
                        if (doAnimate)
                        {
                            deltaW = (width  * i) / (mesh * steps);
                            deltaH = (height * i) / (mesh * steps);
                            repaint();
                            try {
                                wait(deltaT);
                            } catch (InterruptedException ex) { Thread.currentThread().interrupt(); }
                        }
                        else
                        {
                            try {
                                wait();
                            } catch (InterruptedException ex) { Thread.currentThread().interrupt(); }
                            i--;
                        }
                    }
                    toggle = !toggle;

                    try {
                        wait(3 * deltaT);
                    }  catch (InterruptedException ex) { Thread.currentThread().interrupt(); }

                    for (int i = 1; runThread && i <= steps; i++)
                    {
```

```
                        if (doAnimate)
                        {
                            deltaW = (width  * i) / (mesh * steps);
                            deltaH = (height * i) / (mesh * steps);
                            repaint();
                            try {
                                wait(deltaT);
                            } catch (InterruptedException ex) { Thread.currentThread().interrupt(); }
                        }
                        else
                        {
                            try {
                                wait();
                            } catch (InterruptedException ex) { Thread.currentThread().interrupt(); }
                            i--;
                        }
                    }
                    toggle = !toggle;
                }
            }
            animationThreadRunning = false;
        }
    }

    @Override
    public void start()
    {
        synchronized(this)
        {
            doAnimate = true;
            notifyAll();
        }
        SwingUtilities.invokeLater(
                new Runnable()
                    {
                        public void run()
                        {
                            stopButton.setEnabled(true);
                            startButton.setEnabled(false);
                        }
                    }
            );
    }

    @Override
```

```
    public void stop()
    {
        synchronized(this)
        {
            doAnimate = false;
            notifyAll();
        }
        SwingUtilities.invokeLater(
                new Runnable()
                    {
                        public void run()
                        {
                            stopButton.setEnabled(false);
                            startButton.setEnabled(true);
                        }
                    }
            );
    }

    @Override
    public void destroy()
    {
        synchronized(this)
        {
            runThread = false;
            doAnimate = true;
            notifyAll();
        }
    }

}
```