COMPUTER GRAPHICS LAB

CSE 3222

Submitted By:

MD. AL SHAHRIAR

ROLL: **1611076145**

Submitted To:

Dr. Abu Raihan Shoyeb Ahmed Siddique

DECEMBER 13, 2018

COMPUTER SCIENCE & ENGINEERING

UNIVERSITY of RAJSHAHI

//A java program to draw the flag of BANGLADESH:

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;

class Fun extends Frame implements WindowListener {

    public Fun() {

        setSize(300, 300);

        setTitle("Draw Flag");
        show();
        this.addWindowListener(this);

    }

    public void paint(Graphics g) {

        g.setColor(new Color(46, 139, 89));
        g.fillRect(80, 80, 130, 80);
        g.setColor(Color.RED);
        g.fillOval(120, 100, 40, 40);
        g.setColor(new Color(139, 69, 19));
        g.fillRect(75, 70, 5, 170);
        g.setFont(new Font("", Font.BOLD, 20));
        g.drawString("MD. AL SHAHRIAR", 100, 270);
    }

    public void windowClosing(WindowEvent event) {

        dispose();
        System.exit(0);
    }

    public void windowActivated(WindowEvent event) {
    };

    public void windowClosed(WindowEvent event) {
    };

    public void windowDeactivated(WindowEvent event) {
    };

    public void windowDeiconified(WindowEvent event) {
    };

    public void windowIconified(WindowEvent event) {
    };

    public void windowOpened(WindowEvent event) {
    };
}

public class BDflag{

    public static void main(String[] args) {

        new Fun();

    }

}
```

//A java program to make digital clock:

```
import java.awt.Font;
import java.awt.Color;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.Timer;
import javax.swing.SwingConstants;
import java.util.*;
import java.text.*;

public class DigitalClock {

  public static void main(String[] arguments) {

    ClockLabel dateLable = new ClockLabel("date");
    ClockLabel timeLable = new ClockLabel("time");
    ClockLabel dayLable = new ClockLabel("day");

    JFrame.setDefaultLookAndFeelDecorated(true);
    JFrame f = new JFrame("Digital Clock");
    f.setSize(300,150);
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f.setLayout(new GridLayout(3, 1));

    f.add(dateLable);
    f.add(timeLable);
    f.add(dayLable);

    f.getContentPane().setBackground(Color.black);
    f.setVisible(true);
  }
}

class ClockLabel extends JLabel implements ActionListener {

  String type;
  SimpleDateFormat sdf;

  public ClockLabel(String type) {
    this.type = type;
    setForeground(Color.green);

    switch (type) {
      case "date" : sdf = new SimpleDateFormat("  MMMM dd yyyy");
                    setFont(new Font("sans-serif", Font.PLAIN, 12));
                    setHorizontalAlignment(SwingConstants.LEFT);
                    break;
      case "time" : sdf = new SimpleDateFormat("hh:mm:ss a");
                    setFont(new Font("sans-serif", Font.PLAIN, 40));
                    setHorizontalAlignment(SwingConstants.CENTER);
                    break;
      case "day"  : sdf = new SimpleDateFormat("EEEE  ");
                    setFont(new Font("sans-serif", Font.PLAIN, 16));
                    setHorizontalAlignment(SwingConstants.RIGHT);
                    break;
      default     : sdf = new SimpleDateFormat();
                    break;
    }

    Timer t = new Timer(1000, this);
    t.start();
  }

  public void actionPerformed(ActionEvent ae) {
    Date d = new Date();
    setText(sdf.format(d));
  }
}
```

//A java program to implement Translation:

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Graphics;
import java.awt.Graphics2D;
import javax.swing.JFrame;
import javax.swing.JPanel;
```

```
class Surface extends JPanel {

    private void doDrawing(Graphics g) {

        Graphics2D g2d = (Graphics2D) g.create();

        g2d.setPaint(new Color(150, 150, 150));
        g2d.fillRect(20, 20, 80, 50);
        g2d.translate(150, 50);
        g2d.fillRect(20, 20, 80, 50);

        g2d.dispose();
    }

    @Override
    public void paintComponent(Graphics g) {

        super.paintComponent(g);
        doDrawing(g);
    }
}

public class TranslationEx extends JFrame {

    public TranslationEx() {

        initUI();
    }

    private void initUI() {

        add(new Surface());

        setTitle("Translation");
        setSize(300, 200);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {

                TranslationEx ex = new TranslationEx();
                ex.setVisible(true);
            }
        });
    }
}
```

//A java program to implement Rotation:

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Graphics;
import java.awt.Graphics2D;
import javax.swing.JFrame;
import javax.swing.JPanel;

class Surface extends JPanel {

    private void doDrawing(Graphics g) {

        Graphics2D g2d = (Graphics2D) g.create();

        g2d.setPaint(new Color(150, 150, 150));
        g2d.fillRect(20, 20, 80, 50);
        g2d.translate(180, -50);
        g2d.rotate(Math.PI/4);
        g2d.fillRect(80, 80, 80, 50);

        g2d.dispose();
    }

    @Override
    public void paintComponent(Graphics g) {

        super.paintComponent(g);
        doDrawing(g);
    }
}

public class RotationEx extends JFrame {

    public RotationEx() {

        initUI();
    }

    private void initUI() {

        setTitle("Rotation");

        add(new Surface());

        setSize(300, 200);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {

                RotationEx ex = new RotationEx();
                ex.setVisible(true);
            }
        });
    }
}
```

//A java program to implement Scaling:

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import javax.swing.JFrame;
import javax.swing.JPanel;

class Surface extends JPanel {

    private void doDrawing(Graphics g) {

        Graphics2D g2d = (Graphics2D) g.create();

        g2d.setColor(new Color(150, 150, 150));
        g2d.fillRect(20, 20, 80, 50);

        AffineTransform tx1 = new AffineTransform();
        tx1.translate(110, 22);
        tx1.scale(0.5, 0.5);

        g2d.setTransform(tx1);
        g2d.fillRect(0, 0, 80, 50);

        AffineTransform tx2 = new AffineTransform();
        tx2.translate(170, 20);
        tx2.scale(1.5, 1.5);

        g2d.setTransform(tx2);
        g2d.fillRect(0, 0, 80, 50);

        g2d.dispose();
    }

    @Override
    public void paintComponent(Graphics g) {

        super.paintComponent(g);
        doDrawing(g);
    }
}

public class ScalingEx extends JFrame {

    public ScalingEx() {

        initUI();
    }

    private void initUI() {

        add(new Surface());

        setTitle("Scaling");
        setSize(330, 160);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {

                ScalingEx ex = new ScalingEx();
                ex.setVisible(true);
            }
        });
    }
}
```

//A java program to implement Self Squaring Fractals:

```
import javax.swing.*;
import java.awt.*;
import java.awt.geom.*;

public class squareFractal {
        public static void main(String[] args)
        {
                FractalFrame frame = new FractalFrame();
                frame.setTitle("Square Fractal Generator 1.0");
                frame.setVisible(true);
        }
}

class FractalFrame extends JFrame
{
        private JPanel panel;
    private int width, height;
    private int x = 200;
    private int limit = 15;
```

```
        public FractalFrame()
        {
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                final int defaultWidth  = 800;
                final int defaultHeight = 800;
                setSize(defaultWidth, defaultHeight);
                setLocation(300,50);

                panel = new JPanel();
                Container contentPane = getContentPane();
                contentPane.add(panel, "Center");
        }

        public void paint(Graphics g) {
        super.paint(g);
        Graphics2D g2 = (Graphics2D) g;
        //Gets the size of the frame
        width  = getWidth();
        height = getHeight();
        g.setColor(Color.BLACK);

        //Creates the starting position of the first square to fit EXACTLY
center to the frame.
        int w = (width/2)-(x/2);
        int h = (height/2)-(x/2);
        //Draws the first square
        g.fillRect(w, h, x, x);
        //Starts the recursive fractal generation if the complexity is
greater than 0.
        if(limit > 0) drawSquare(w,h,g);
        //When complete, notifies the user.
        JOptionPane.showMessageDialog(null, "Generating Fractal =
Complete!");
    }

    // Draws the first level of squares and then recursively draws the
remaining amount (Stops when N == Limit).
    // Each concurrent square will be exactly half that of its parent. Thus:
    // --> fillRect( [X-Coordinate], [Y-Coordinate], [Width of Square],
[Height of Square])
    private void drawSquare(int w, int h, Graphics g) {
        // N will be used as a checker to count how many levels of recursion
have occured.
        int n = 0;
        //Draws Pos 0 (Top-Left)
        g.fillRect(w-(x/2), h-(x/2), x/2, x/2);
        drawSquare(w-(x/2), h-(x/2), g, n, 0, x/2);
        //Draws Pos 1 (Top-Right)
        g.fillRect(w+x, h-(x/2), x/2, x/2);
        drawSquare(w+x, h-(x/2), g, n, 1, x/2);
        //Draws pos 2 (Bottom-Right)
        g.fillRect(w+x, h+x, x/2, x/2);
        drawSquare(w+x, h+x, g, n, 2, x/2);
        //Draws pos 3 (Bottom-Left)
        g.fillRect(w-(x/2), h+x, x/2, x/2);
        drawSquare(w-(x/2), h+x, g, n, 3, x/2);
    }

    // Recursive function to draw squares
    private void drawSquare(int w, int h, Graphics g, int n, int origin, int
size) {
        // Stops the recursion loop when it has reached its complexity limit.
        if(n == limit) return;
        n++;
        // This bit of math is key. It prevents the program from drawing a
square at the location of its parent.
        origin = (origin+2)%4;
        //Draws Pos 0
        if(origin != 0) {
            g.fillRect(w-(size/2), h-(size/2), size/2, size/2);
            //Recursive Call
            drawSquare(w-(size/2), h-(size/2),g,n,0,size/2);
        }
        //Draws Pos 1
        if(origin != 1) {
            g.fillRect(w+size, h-(size/2), size/2, size/2);
            //Recursive Call
            drawSquare(w+size, h-(size/2),g,n,1,size/2);
        }
        //Draws pos 2
        if(origin != 2) {
            g.fillRect(w+size, h+size, size/2, size/2);
            //Recursive Call
            drawSquare(w+size, h+size,g,n,2,size/2);
        }
        //Draws pos 3
        if(origin != 3) {
            g.fillRect(w-(size/2), h+size, size/2, size/2);
            //Recursive Call
            drawSquare(w-(size/2), h+size,g,n,3,size/2);
        }
    }
}
```

//A java program to implement Julia Set:

```
import java.awt.*;
import java.awt.image.BufferedImage;
import javax.swing.*;

public class JuliaSet extends JPanel {
    private final int maxIter = 300;
    private final double zoom = 1;
    private double cY, cX;

    public JuliaSet() {
        setPreferredSize(new Dimension(800, 600));
        setBackground(Color.white);
    }

    void drawJuliaSet(Graphics2D g) {
        int w = getWidth();
        int h = getHeight();
        BufferedImage image = new BufferedImage(w, h,
                BufferedImage.TYPE_INT_RGB);

        cX = -0.7;
        cY = 0.27015;
        double moveX = 0, moveY = 0;
        double zx, zy;

        for (int x = 0; x < w; x++) {
            for (int y = 0; y < h; y++) {
                zx = 1.5 * (x - w / 2) / (0.5 * zoom * w) + moveX;
                zy = (y - h / 2) / (0.5 * zoom * h) + moveY;
                float i = maxIter;
                while (zx * zx + zy * zy < 4 && i > 0) {
                    double tmp = zx * zx - zy * zy + cX;
                    zy = 2.0 * zx * zy + cY;
                    zx = tmp;
                    i--;
                }
                int c = Color.HSBtoRGB((maxIter / i) % 1, 1, i > 0 ? 1 : 0);
                image.setRGB(x, y, c);
            }
        }
        g.drawImage(image, 0, 0, null);
    }

    @Override
    public void paintComponent(Graphics gg) {
        super.paintComponent(gg);
        Graphics2D g = (Graphics2D) gg;
        g.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                RenderingHints.VALUE_ANTIALIAS_ON);
        drawJuliaSet(g);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            JFrame f = new JFrame();
            f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            f.setTitle("Julia Set");
            f.setResizable(false);
            f.add(new JuliaSet(), BorderLayout.CENTER);
            f.pack();
            f.setLocationRelativeTo(null);
            f.setVisible(true);
        });
    }
}
```

//A java program to implement Bresenham's Line Drawing Algorithm:

```java
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.SwingUtilities;
import javax.swing.WindowConstants;

public class Bresenham {

    public static void main(String[] args) {
        SwingUtilities.invokeLater(Bresenham::run);
    }

    private static void run() {
        JFrame f = new JFrame();
        f.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        f.setTitle("Bresenham");

        f.getContentPane().add(new BresenhamPanel());
        f.pack();

        f.setLocationRelativeTo(null);
        f.setVisible(true);
    }
}

class BresenhamPanel extends JPanel {

    private final int pixelSize = 10;

    BresenhamPanel() {
        setPreferredSize(new Dimension(600, 500));
        setBackground(Color.WHITE);
    }

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

        int w = (getWidth() - 1) / pixelSize;
        int h = (getHeight() - 1) / pixelSize;
        int maxX = (w - 1) / 2;
        int maxY = (h - 1) / 2;
        int x1 = -maxX, x2 = maxX * -2 / 3, x3 = maxX * 2 / 3, x4 = maxX;
        int y1 = -maxY, y2 = maxY * -2 / 3, y3 = maxY * 2 / 3, y4 = maxY;

        drawLine(g, 0, 0, x3, y1); // NNE
        drawLine(g, 0, 0, x4, y2); // ENE
        drawLine(g, 0, 0, x4, y3); // ESE
        drawLine(g, 0, 0, x3, y4); // SSE
        drawLine(g, 0, 0, x2, y4); // SSW
        drawLine(g, 0, 0, x1, y3); // WSW
        drawLine(g, 0, 0, x1, y2); // WNW
        drawLine(g, 0, 0, x2, y1); // NNW
    }

    private void plot(Graphics g, int x, int y) {
        int w = (getWidth() - 1) / pixelSize;
        int h = (getHeight() - 1) / pixelSize;
        int maxX = (w - 1) / 2;
        int maxY = (h - 1) / 2;

        int borderX = getWidth() - ((2 * maxX + 1) * pixelSize + 1);
        int borderY = getHeight() - ((2 * maxY + 1) * pixelSize + 1);
        int left = (x + maxX) * pixelSize + borderX / 2;
        int top = (y + maxY) * pixelSize + borderY / 2;

        g.setColor(Color.black);
        g.drawOval(left, top, pixelSize, pixelSize);
    }

    private void drawLine(Graphics g, int x1, int y1, int x2, int y2) {
        // delta of exact value and rounded value of the dependent variable
        int d = 0;

        int dx = Math.abs(x2 - x1);
        int dy = Math.abs(y2 - y1);

        int dx2 = 2 * dx; // slope scaling factors to
        int dy2 = 2 * dy; // avoid floating point

        int ix = x1 < x2 ? 1 : -1; // increment direction
        int iy = y1 < y2 ? 1 : -1;

        int x = x1;
        int y = y1;

        if (dx >= dy) {
            while (true) {
                plot(g, x, y);
                if (x == x2)
                    break;
                x += ix;
                d += dy2;
                if (d > dx) {
                    y += iy;
                    d -= dx2;
                }
            }
        } else {
            while (true) {
                plot(g, x, y);
                if (y == y2)
                    break;
                y += iy;
                d += dx2;
                if (d > dy) {
                    x += ix;
                    d -= dy2;
                }
            }
        }
    }
}
```

//A java program to implement Midpoint Circle Drawing Algorithm:

```
import java.awt.Color;

public class MidPointCircle {
        private BasicBitmapStorage image;

        public MidPointCircle(final int imageWidth, final int imageHeight) {
                this.image = new BasicBitmapStorage(imageWidth, imageHeight);
        }

        private void drawCircle(final int centerX, final int centerY, final
int radius) {
                int d = (5 - r * 4)/4;
                int x = 0;
                int y = radius;
                Color circleColor = Color.white;

                do {
                        image.setPixel(centerX + x, centerY + y, circleColor);
                        image.setPixel(centerX + x, centerY - y, circleColor);
                        image.setPixel(centerX - x, centerY + y, circleColor);
                        image.setPixel(centerX - x, centerY - y, circleColor);
                        image.setPixel(centerX + y, centerY + x, circleColor);
                        image.setPixel(centerX + y, centerY - x, circleColor);
                        image.setPixel(centerX - y, centerY + x, circleColor);
                        image.setPixel(centerX - y, centerY - x, circleColor);
                        if (d < 0) {
                                d += 2 * x + 1;
                        } else {
                                d += 2 * (x - y) + 1;
                                y--;
                        }
                        x++;
                } while (x <= y);

        }
}
```

//A java program to implement Ellipse Drawing Algorithm:

```
import java.lang.*;
import java.util.*;
import java.util.List;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.*;

public class MyDrawEllipse extends Frame {

  public void paint(Graphics g) {
     Graphics2D ga = (Graphics2D)g;
     ga.setPaint(Color.red);
     ga.drawArc(50,50,300,150, 0, 360);
     ga.setPaint(Color.blue);
     ga.drawArc(75,75,250,100, 0, 360);
  }

  public static void main(String args[])
  {
      MyDrawEllipse frame = new MyDrawEllipse();
      frame.addWindowListener(
      new WindowAdapter()
      {
         public void windowClosing(WindowEvent we)
         {
            System.exit(0);
         }
      }
      );

      frame.setSize(400, 400);
      frame.setVisible(true);
   }
}
```