Федеральное агентство связи

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра Информатики

Отчет по лабораторной работе №6

по предмету «КТП»:

Выполнил: студент группы БВТ1802

Самаков Владислав Владимирович

Руководитель:

Ксения Андреевна Полянцева

Москва 2020

1 Цель работы

Цель работы: изучить алгоритм расчета фрактала, научиться исользовать java.swing и класс java.swing.worker.

2 Задание

Распараллелить отрисовку фрактала, сделав ее более быстрой.

3 Текст программы

FractalExplorer.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.geom.Rectangle2D;
import java.awt.event.*;
import java.io.IOException;
import javax.swing.filechooser.*;
import javax.imageio.ImageIO;

public class FractalExplorer {
    private int displaySize;
    private FractalGenerator generator;
    private JComboBox switchButton;
    private JButton saveButton;
    private JFrame frame;
    private int remainingRows;
    private Rectangle2D.Double range;
    private JImageDisplay display;
    private JButton resetButton;




    private class actionListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent actionEvent) {
            if (actionEvent.getSource() == switchButton) {
                generator = (FractalGenerator) switchButton.getSelectedItem();
                generator.getInitialRange(range);
                drawFractal();
            }
            else if (actionEvent.getSource() == resetButton) {
                generator.getInitialRange(range);
                drawFractal();
            }
            else if (actionEvent.getSource() == saveButton) {
                JFileChooser chooser = new JFileChooser();
                FileFilter filter = new FileNameExtensionFilter("PNG Images", "png");
                chooser.setFileFilter(filter);
                chooser.setAcceptAllFileFilterUsed(false);
                if (chooser.showSaveDialog(null) == JFileChooser.APPROVE_OPTION) {
                    try {
                        ImageIO.write(display.img, "png", chooser.getSelectedFile());
                    } catch (IOException e) {
                        JOptionPane.showMessageDialog(frame, e.getMessage(), "Cannot
```

```
Save Image",
                                    JOptionPane.ERROR_MESSAGE);
                    }
                }
            }
        }
    }

    private class MouseListener extends MouseAdapter {
        @Override
        public void mouseClicked(MouseEvent e) {
            if (remainingRows != 0) return;
            int x = e.getX();
            int y = e.getY();

            double xCoord = generator.getCoord(range.x, range.x + range.width,
displaySize,x);
            double yCoord = generator.getCoord(range.y, range.y + range.height,
displaySize,y);
            generator.recenterAndZoomRange(range, xCoord, yCoord, 0.5);
            drawFractal();
        }
    }
    public FractalExplorer(int ScreenSize) {
        displaySize = ScreenSize;
        range = new Rectangle2D.Double();
        generator = new Mandelbrot();
        generator.getInitialRange(range);
    }

    private class FractalWorker extends SwingWorker<Object, Object> {

        private int m_Ycoord;
        private int[] m_Xcoords;

        private FractalWorker(int yCoord) {
            m_Ycoord = yCoord;
        }
        @Override
        protected Object doInBackground() throws Exception {
            m_Xcoords = new int[displaySize];
            double yCoord = FractalGenerator.getCoord
                    (range.y, range.y + range.height, displaySize, m_Ycoord);
            for (int x = 0; x < displaySize; x++)
            {
                double xCoord = FractalGenerator.getCoord
                        (range.x, range.x + range.width, displaySize, x);
                int IterNum = generator.numIterations(xCoord, yCoord);
                if (IterNum == -1) m_Xcoords[x] = 0;
                else {
                    float hue = 0.7f + (float) IterNum / 200f;
                    int rgbColor = Color.HSBtoRGB(hue, 1f, 1f);
                    m_Xcoords[x] = rgbColor;
                }
            }
            return null;
        }

        @Override
        protected void done() {
            for (int x = 0; x < displaySize; x++) {
                display.drawPixel(x, m_Ycoord, m_Xcoords[x]);
```

```
            }
            display.repaint(0, m_Ycoord, displaySize, 1);
            remainingRows--;
            if (remainingRows == 0) {
                enableUI(true);
            }
        }
    }

    public void createAndShowGUI() {

        JPanel panel = new JPanel();
        switchButton = new JComboBox();
        switchButton.addItem(new Mandelbrot());
        switchButton.addItem(new Tricorn());
        switchButton.addItem(new BurningShip());
        switchButton.addActionListener(new actionListener());
        JLabel label = new JLabel("Fractal type:");
        panel.add(label);
        panel.add(switchButton);


        display = new JImageDisplay(displaySize, displaySize);
        display.addMouseListener(new MouseListener());


        resetButton = new JButton("Reset Image");
        resetButton.addActionListener(new actionListener());
        saveButton = new JButton("Save Image");
        saveButton.addActionListener(new actionListener());
        JPanel panel2 = new JPanel();
        panel2.add(resetButton);
        panel2.add(saveButton);

        frame = new JFrame();
        frame.getContentPane().add(panel, BorderLayout.NORTH);
        frame.getContentPane().add(display, BorderLayout.CENTER);
        frame.getContentPane().add(panel2, BorderLayout.SOUTH);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);


        frame.pack();
        frame.setVisible(true);
        frame.setResizable(true);
    }

    void enableUI(boolean val) {
        resetButton.setEnabled(val);
        saveButton.setEnabled(val);
        switchButton.setEnabled(val);
    }

    private void drawFractal() {
        enableUI(false);
        remainingRows = displaySize;
        for (int y = 0; y < displaySize; y++) {
            FractalWorker worker = new FractalWorker(y);
            worker.execute();
        }
    }

    public static void main(String args[]) {
```

```
        FractalExplorer explorer = new FractalExplorer(1000);
        explorer.createAndShowGUI();
        explorer.drawFractal();
    }

}
```

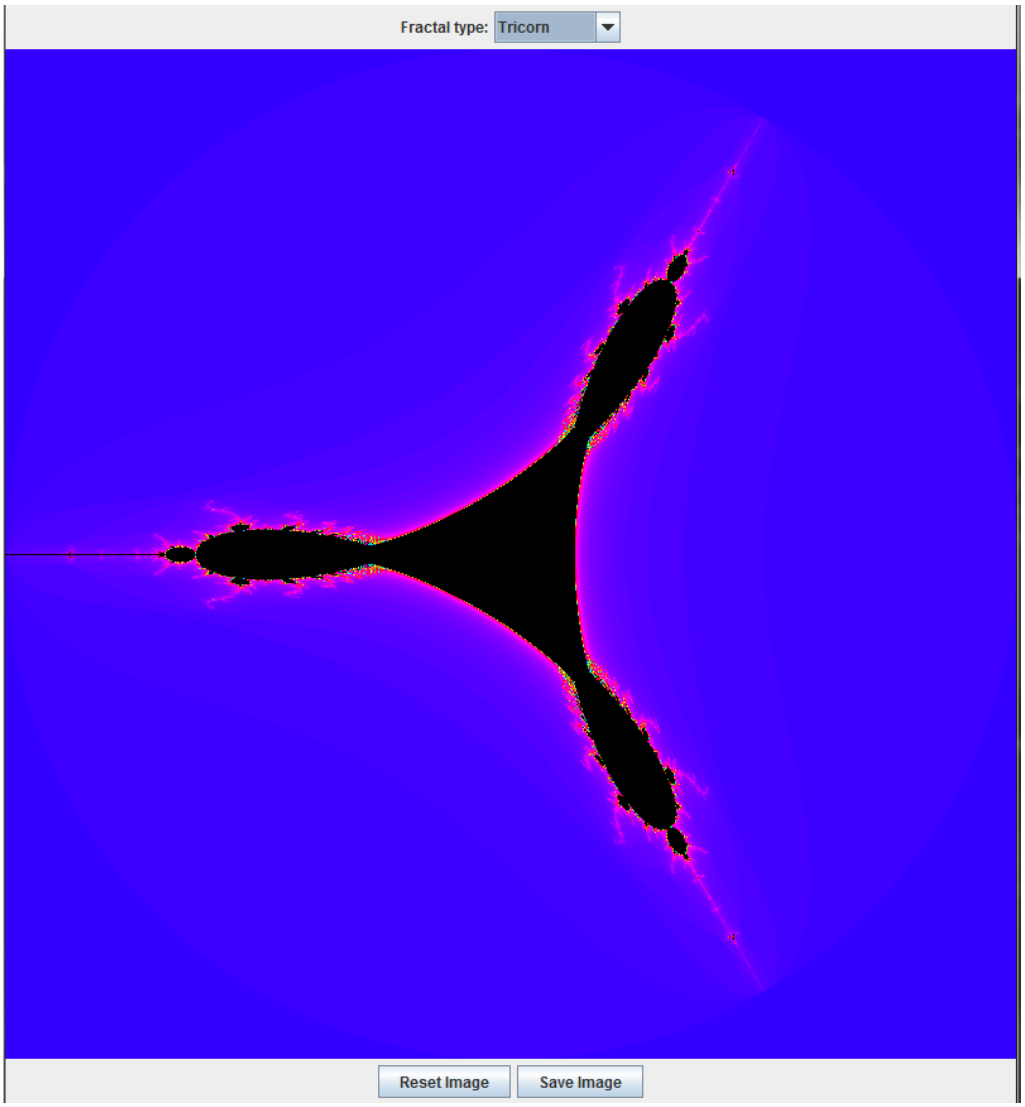
Mandelbrot.java

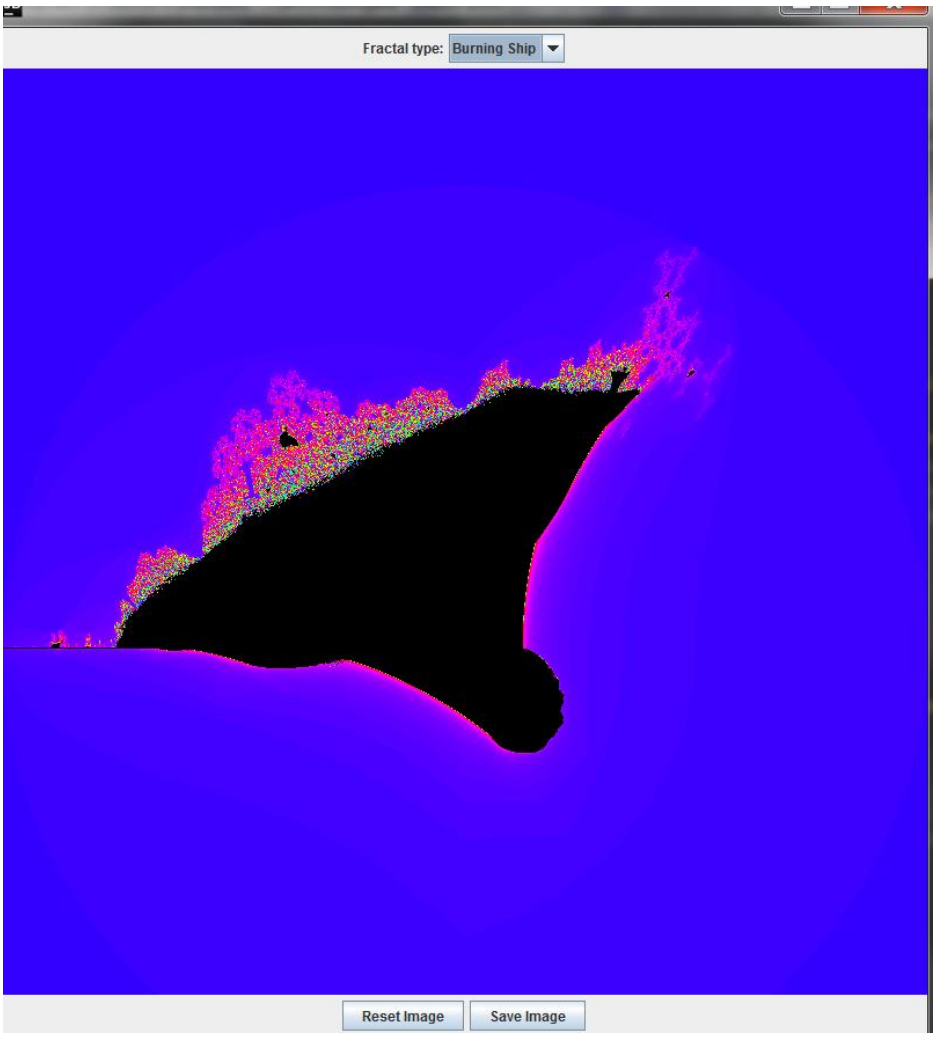
```
import java.awt.geom.Rectangle2D;

public class Mandelbrot extends FractalGenerator {

    public static final int MAX_ITERATIONS = 2000;


    public void getInitialRange(Rectangle2D.Double range) {
        range.setRect(-2, -1.5, 3,3);
    }

    public int numIterations(double x, double y) {
        Complex z = new Complex(0, 0);
        Complex c = new Complex(x, y);
        for (int IterNum = 0; IterNum < MAX_ITERATIONS; IterNum++) {
            z = z.mul(z).sum(c);
            if (z.abs() > 4) return IterNum;
        }
        return -1;
    }

    @Override
    public String toString() {
        return "Mandelbrot";
    }
}
```

Complex.java

```
public class Complex {
    public double real, imag;
    public Complex(double real, double imag) {
        this.real = real;
        this.imag = imag;
    }
    public double abs() { return real * real + imag * imag; }

    public Complex sum(Complex c) {
        return new Complex(this.real + c.real, this.imag + c.imag);
    }

    public Complex mul(Complex c) {
        double real = this.real * c.real - this.imag * c.imag;
        double imag = this.real * c.imag + this.imag * c.real;
        return new Complex(real,imag);
    }

    public Complex sopr() { return new Complex(this.real, -this.imag); }
}
```

4 Работа программы





Вывод

Я познакомился с многопоточностью в языке программирования java и применил ее в данной лабораторной работе. Время отрисовки фрактала значительно уменьшилось. Кроме того, теперь можно наблюдать отрисовку в реальном времени, а не смотреть несколько секунд на черный экран.