Федеральное агентство связи

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра математической кибернетики и информационных технологий

Отчет по лабораторной работе №6

по дисциплине «Технологии разработки программного обеспечения»

Выполнил: студент группы БВТ1801

Протасова Елена Сергеевна

Руководитель:

Мосева Марина Сергеевна

Москва 2020

Цель работы: необходимо реализовать возможность рисования фрактала с несколькими фоновыми потоками.

Выполнение:

FractalExplorer:

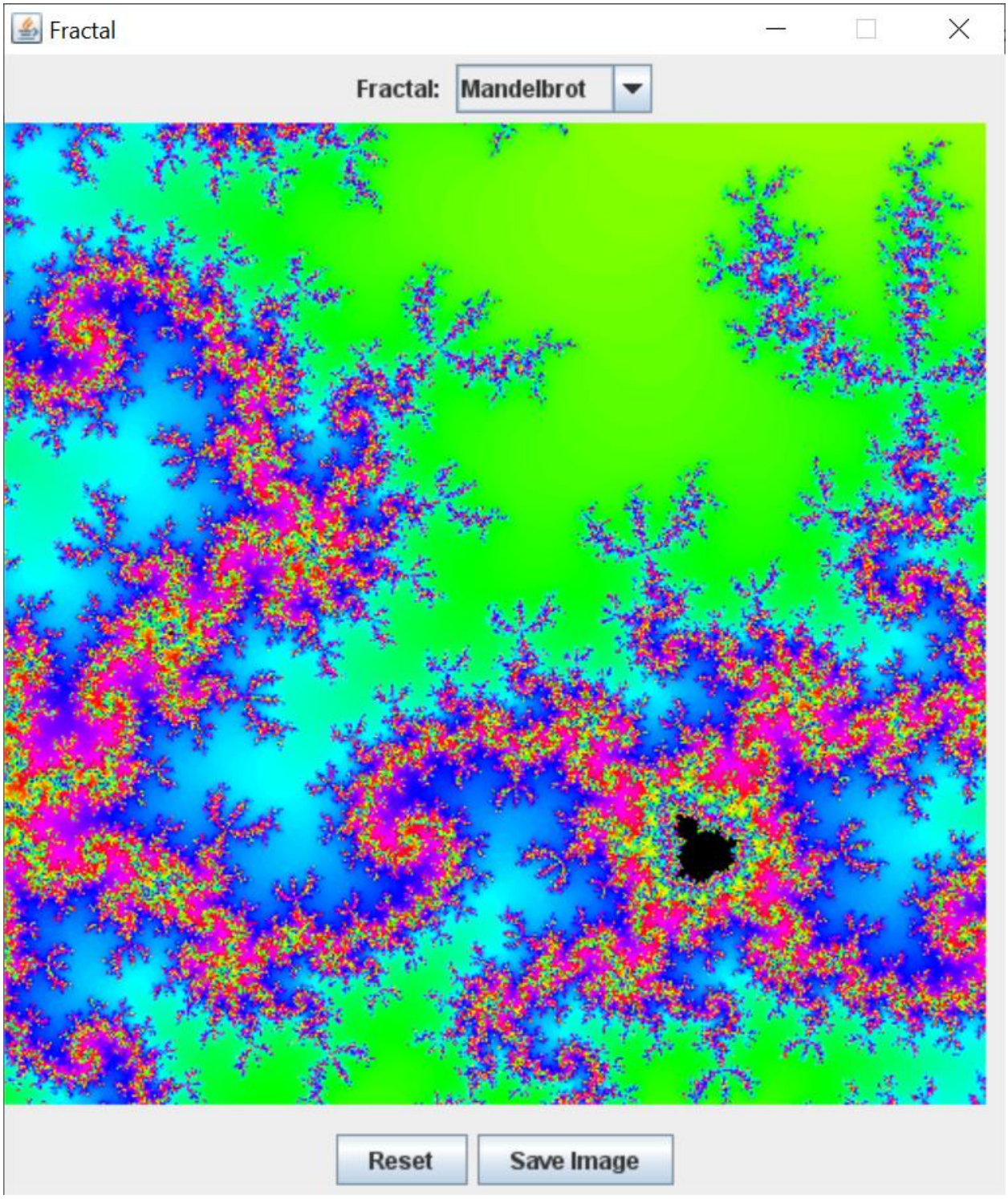
```
import javax.imageio.ImageIO;
import javax.swing.*;
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileNameExtensionFilter;
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.Rectangle2D;
import java.awt.image.BufferedImage;
import java.io.File;

public class FractalExplorer {
    private int size;
    private JImageDisplay jImageDisplay;
    private  FractalGenerator fractalGenerator;
    private Rectangle2D.Double rectangle2D;
    private JComboBox jComboBox;
    private JButton jButton;
    private JButton sButton;
    private int rowsremaning;
    public FractalExplorer(int size){
        this.size = size;
        rectangle2D = new Rectangle2D.Double();
        fractalGenerator = new Mandelbrot();
        fractalGenerator.getInitialRange(rectangle2D);
        jImageDisplay = new JImageDisplay(size,size);
    }

    public static void main(String[] args) {
        FractalExplorer fractalExplorer = new FractalExplorer(500);
        fractalExplorer.createAndShowGUI();

    }
    public void createAndShowGUI() {
        EventButton eventButton = new EventButton();
        EventMouse eventMouse = new EventMouse();

        JFrame jFrame = new JFrame("Fractal");
        jImageDisplay.addMouseListener(eventMouse);
        jFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        jFrame.add(jImageDisplay,BorderLayout.CENTER );

        JPanel SaveReset = new JPanel();
        jButton = new JButton("Reset");
        sButton = new JButton("Save Image");
        SaveReset.add(jButton);
        SaveReset.add(sButton);
        jButton.addActionListener(eventButton);
        sButton.addActionListener(eventButton);


        jFrame.add(SaveReset, BorderLayout.SOUTH);

        jComboBox = new JComboBox();
        jComboBox.addActionListener(eventButton);
```

```
        jComboBox.addItem(new Mandelbrot());
        jComboBox.addItem(new BurningShip());
        jComboBox.addItem(new Tricorn());
        JPanel jPanel = new JPanel();
        JLabel jLabel = new JLabel();
        jLabel.setText("Fractal: ");
        jPanel.add(jLabel);
        jPanel.add(jComboBox);
        jFrame.add(jPanel, BorderLayout.NORTH);

        jFrame.pack ();
        jFrame.setVisible (true);
        jFrame.setResizable (false);

    }
    private void drawFractal() {
        enableUI(false);
        rowsremaning = size;

        for (int x=0; x<size; x++){
            FractalWorker drawfractal = new FractalWorker(x);
            drawfractal.execute();
        }
    }
    private void enableUI(boolean var){
        jComboBox.setEnabled(var);
        jButton.setEnabled(var);
        sButton.setEnabled(var);
    }

    private class EventButton implements ActionListener{
        @Override
        public void actionPerformed(ActionEvent e) {
            //fractalGenerator.getInitialRange(rectangle2D);
            //drawFractal();
            if (e.getSource() instanceof JComboBox){
                JComboBox jcomboBox = (JComboBox) e.getSource();
                fractalGenerator = (FractalGenerator) jcomboBox.getSelectedItem();
                fractalGenerator.getInitialRange(rectangle2D);
                drawFractal();
            }
            else if (e.getActionCommand().equals("Reset")){
                fractalGenerator.getInitialRange(rectangle2D);
                drawFractal();
            }
            else if (e.getActionCommand().equals("Save Image")){
                JFileChooser jFileChooser = new JFileChooser();
                int showSelec = jFileChooser.showDialog(jImageDisplay,"Save file");
                FileFilter filter = new FileNameExtensionFilter("PNG Images", "png");
                jFileChooser.setFileFilter(filter);
                jFileChooser.setAcceptAllFileFilterUsed(false);
                if (showSelec==jFileChooser.APPROVE_OPTION) {
                    File file = jFileChooser.getSelectedFile();
                    try{
                        BufferedImage bufferedImage = jImageDisplay.image;
                        ImageIO.write(bufferedImage, "png", file);

                    }
                    catch(Exception ex){
                        JOptionPane.showMessageDialog(jImageDisplay,
ex.getMessage(),"Cannot Save Image",JOptionPane.ERROR_MESSAGE);
                    }
```

```
                }
            }
        }
    }
    private class EventMouse extends MouseAdapter {
        @Override
        public void mouseClicked(MouseEvent e) {
            if (rowsremaning != 0) {
              return;
            }
            int x = e.getX();
            int y = e.getY();
            double xCoord = FractalGenerator.getCoord(rectangle2D.x, rectangle2D.x +
rectangle2D.width, size, x);
            double yCoord = FractalGenerator.getCoord(rectangle2D.y, rectangle2D.y +
rectangle2D.height, size, y);
            fractalGenerator.recenterAndZoomRange(rectangle2D,xCoord, yCoord, 0.5);
            drawFractal();
        }
    }
    private class FractalWorker extends SwingWorker<Object, Object> {
        int ycoord=0;
        int[] rgbs;
        public FractalWorker(int ycoord){
            this.ycoord = ycoord;
        }
        @Override
        protected Object doInBackground() throws Exception {
            rgbs = new int[size];
            for (int x=0; x<size; x++) {

                double xCoord = FractalGenerator.getCoord(rectangle2D.x,
rectangle2D.x + rectangle2D.width, size, x);
                double yCoord = FractalGenerator.getCoord(rectangle2D.y,
rectangle2D.y + rectangle2D.height, size, ycoord);
                int numI = fractalGenerator.numIterations(xCoord, yCoord);
                if (numI == -1) rgbs[x] = 0;
                else {
                    float hue = 0.7f + (float) numI / 200f;
                    int rgbColor = Color.HSBtoRGB(hue, 1f, 1f);

                    rgbs[x] = rgbColor;
                }

            }
            return null;
        }
        protected void done() {
            // Iterate over the array of row-data, drawing in the pixels
            // that were computed in doInBackground().  Redraw the row
            // that was changed.
            for (int i = 0; i < size; i++) {
                jImageDisplay.drawPixel(i, ycoord, rgbs[i]);
            }
            jImageDisplay.repaint(0, 0, ycoord, size, 1);

            // Decrement rows remaining.  If 0, call enableUI(true)
            rowsremaning--;
            if (rowsremaning== 0) {
                enableUI(true);
            }
        }
```

```
    }
}
```

Результаты работы:



Выводы: была изучена реализация рисования фрактала с несколькими фоновыми потоками.