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 №5

по дисциплине «Технологии разработки программного обеспечения»

Выполнил: студент группы БВТ1801

Протасова Елена Сергеевна

Руководитель:

Мосева Марина Сергеевна

Москва 2020

Цель работы: добавить поддержку нескольких фракталов и реализовать возможность выбирать нужный фрактал из выпадающего списка, добавить поддержку сохранения текущего изображения в файл.

Выполнение работы:

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


        jFrame.add(SaveReset, BorderLayout.SOUTH);

        JComboBox 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() {
        for (int x=0; x<size; x++) {
            for (int y=0; y<size; y++){
                double xCoord = FractalGenerator.getCoord(rectangle2D.x,
rectangle2D.x + rectangle2D.width, size, x);
                double yCoord = FractalGenerator.getCoord(rectangle2D.y,
rectangle2D.y + rectangle2D.height, size, y);
                int numI = fractalGenerator.numIterations(xCoord,yCoord);
                if (numI==-1) jImageDisplay.drawPixel(x,y,0);
                else {
                    float hue = 0.7f + (float)numI / 200f;
                    int rgbColor = Color.HSBtoRGB(hue, 1f, 1f);
                    jImageDisplay.drawPixel(x,y,rgbColor);
                }
            }
        }
        jImageDisplay.repaint();
    }
    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) {
            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();
        }
    }

}
```

BurningShip:

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

public class BurningShip extends FractalGenerator {
    @Override
    public void getInitialRange(Rectangle2D.Double range) {
        range.x = -2;
        range.y = -2.5;
        range.width = 4;
        range.height = 3.5;
    }
    public static final int MAX_ITERATIONS = 2000;
    @Override
    public int numIterations(double x, double y) {
        double rez = 0, imz = 0;
        int n = 0;
        while (((rez*rez)+(imz*imz))<4 && n<2000){
            double ri = rez;
            rez = rez*rez - imz * imz + x;
            imz = 2*Math.abs(ri*imz) + y;
            n++;
        }
        if (n>=2000) return -1;
        return n;
    }
    public String toString(){
        return "Burning Ship";
    }
}
```

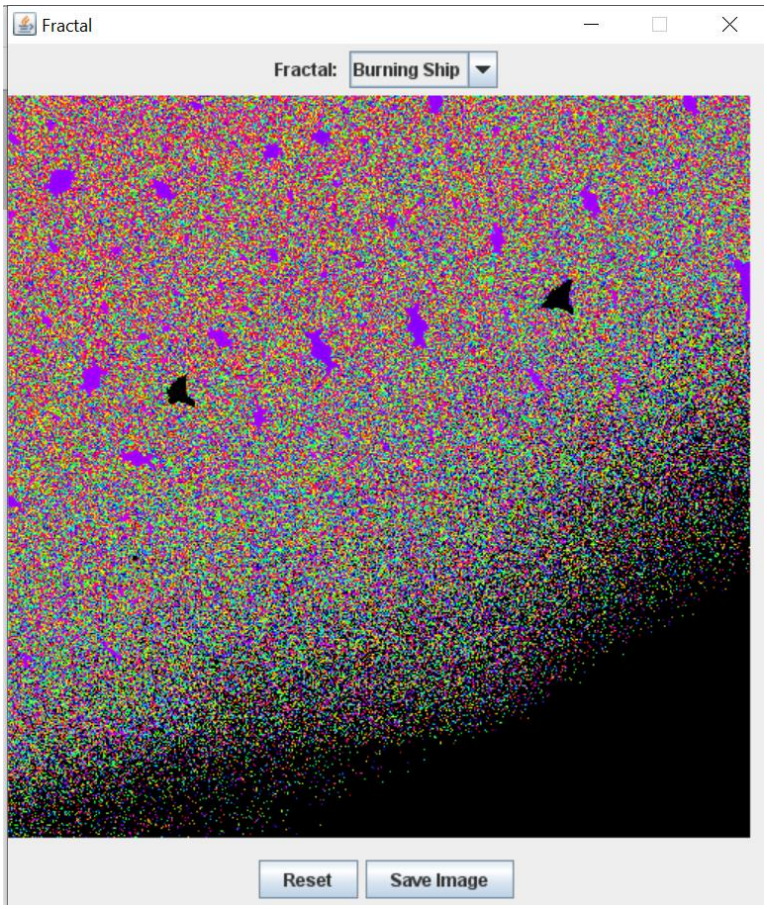
Mandelbrot:

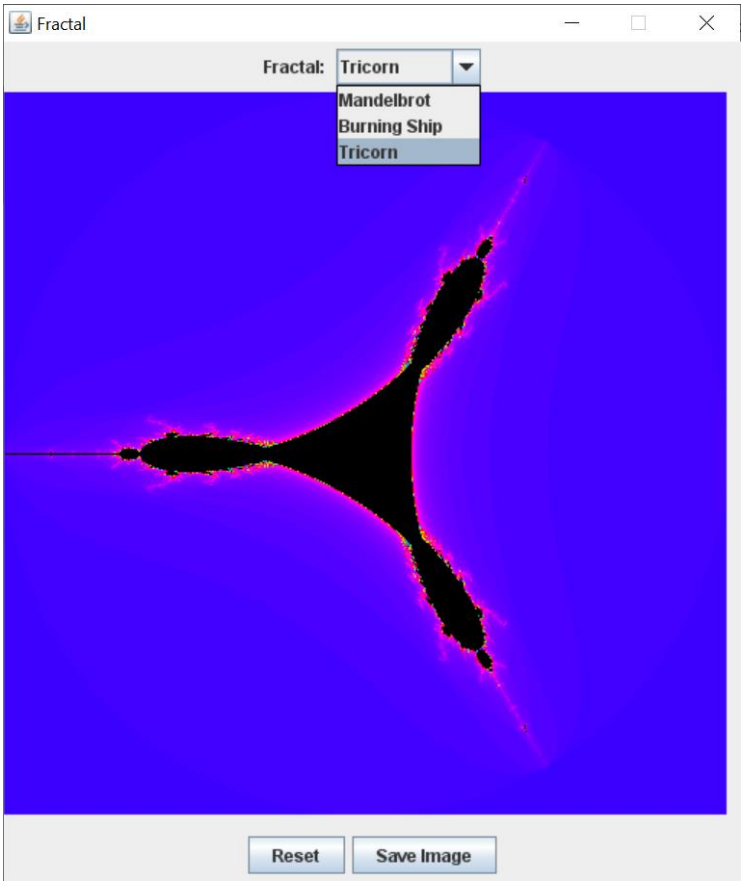
```
import java.awt.geom.Rectangle2D;

public class Mandelbrot extends FractalGenerator {
    @Override
    public void getInitialRange(Rectangle2D.Double range) {
        range.x = -2;
        range.y = -1.5;
        range.width = 3;
        range.height = 3;
    }
    public static final int MAX_ITERATIONS = 2000;
    @Override
    public int numIterations(double x, double y) {
        double rez = 0, imz = 0;
        int n = 0;
        while (((rez*rez)+(imz*imz))<4 && n<2000){
            double ri = rez;
            rez = rez*rez - imz * imz + x;
            imz = 2*ri*imz + y;
            n++;
        }
        if (n>=2000) return -1;
        return n;
    }
    public String toString(){
        return "Mandelbrot";
    }
}
```

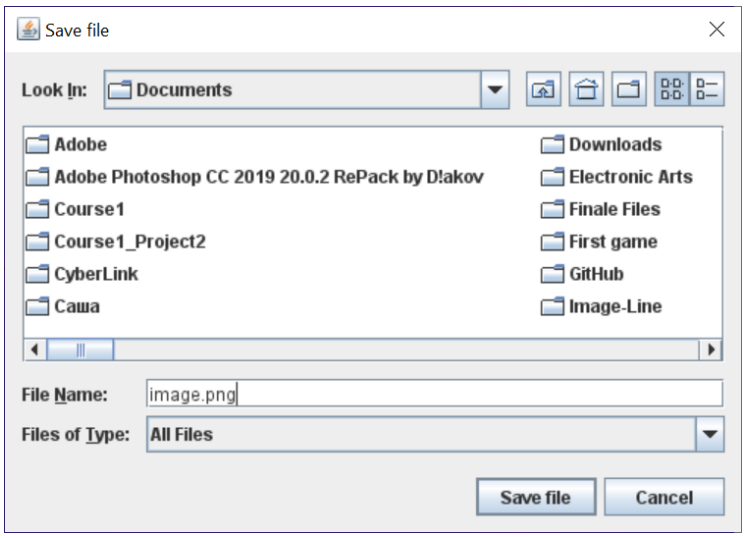
Tricorn:

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

public class Tricorn extends FractalGenerator{
    @Override
    public void getInitialRange(Rectangle2D.Double range) {
        range.x = -2;
        range.y = -2;
        range.width = 4;
        range.height = 4;
    }
    public static final int MAX_ITERATIONS = 2000;
    @Override
    public int numIterations(double x, double y) {
        double rez = 0, imz = 0;
        int n = 0;
        while (((rez*rez)+(imz*imz))<4 && n<2000){
            double ri = rez;
            rez = rez*rez - imz * imz + x;
            imz = -2*ri*imz + y;
            n++;
        }
        if (n>=2000) return -1;
        return n;
    }
    public String toString(){
        return "Tricorn";
    }
}
```

Скрины работы:







Выводы: была изучена работа с ComboBox, рассмотрены 2 новые реализации генерации фракталов.