**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**

по предмету «Кроссплатформенные технологии программирования»

на тему:

«Выбор и сохранение фракталов»

Выполнил: студент группы

Митрохин Ярослав Игоревич

Руководитель:

Мосева Марина Сергеевна

Москва 2020

*Цель работы:* расширить генератор фракталов. Добавить поддержку нескольких фракталов и реализовать возможность выбирать нужный фрактал из выпадающего списка.

*Ход работы:*

*Реализация класса FractalExplorer:*

import java.awt.\*;  
import javax.swing.\*;  
import javax.swing.filechooser.FileNameExtensionFilter;  
import java.awt.BorderLayout;  
import java.awt.event.\*;  
import java.awt.geom.Rectangle2D;  
import java.awt.image.BufferedImage;  
import java.io.FileFilter;  
  
public class FractalExplorer {  
 private int displaySize;  
 private JimageDisplay upDisplay;//Ссылка,для отображения в разных методах  
 private FractalGenerator fractalF;//Ссылка на базовый класс  
 private Rectangle2D.Double range;//Диапазон комплексной плоскости  
 private JComboBox comboBox;  
 private JButton saveBut;  
 public FractalExplorer(int size)  
 {  
 displaySize = size;  
 range = new Rectangle2D.Double();  
 fractalF = new Mandelbrot();  
 fractalF.getInitialRange(range);//Инициализация объекты диапазона и фрактального генератора  
 upDisplay = new JimageDisplay(displaySize,displaySize);  
 }  
 public void createAndShowGUI()  
 {  
 JFrame frame = new JFrame("Fractal");  
 frame.add(upDisplay,BorderLayout.*CENTER*);  
 JButton button = new JButton("Reset");  
 ResetDis reset = new ResetDis();  
 button.addActionListener(reset);  
 //frame.add(button,BorderLayout.SOUTH);  
  
 MousePiPi mouse = new MousePiPi();  
 upDisplay.addMouseListener(mouse);  
  
 JComboBox comboBox = new JComboBox();  
 FractalGenerator mandelbrotFrac = new Mandelbrot();  
 comboBox.addItem(mandelbrotFrac);  
 FractalGenerator tricornFrac = new Tricorn();  
 comboBox.addItem(tricornFrac);  
 FractalGenerator burningshipFrac = new BurningShip();  
 comboBox.addItem(burningshipFrac);  
  
 ResetDis fracCh = new ResetDis();  
 comboBox.addActionListener(fracCh);  
  
 JPanel panel = new JPanel();  
 JLabel label = new JLabel("Fractal ");  
 panel.add(label);  
 panel.add(comboBox);  
 frame.add(panel,BorderLayout.*NORTH*);//обавляем label в интерфейс перед списком  
  
 saveBut = new JButton("Save");  
 JPanel BotPan = new JPanel();  
 BotPan.add(saveBut);  
 BotPan.add(button);  
 frame.add(BotPan,BorderLayout.*SOUTH*);  
  
 ResetDis saveH = new ResetDis();  
 saveBut.addActionListener(saveH);  
  
 frame.pack();  
 frame.setVisible(true);  
 frame.setResizable(false);  
 //Правильно разместят содержимое окна  
 }  
 private void drawFractal()  
 {  
 for (int i = 0; i<displaySize;i++){  
 for (int j = 0; j<displaySize;j++){  
 double xCoord = FractalGenerator.*getCoord*(range.x, range.x+range.width,displaySize, i);//Пиксельная и координата в пространстве  
 double yCoord = FractalGenerator.*getCoord*(range.y, range.y+range.height,displaySize, j);  
 int iter = fractalF.numIterations(xCoord,yCoord);  
 if (iter == -1)  
 {  
 upDisplay.drawPixel(i,j,0);  
 }  
 else  
 {  
 float hue = 0.7f + (float) iter/200f;  
 int rgbColor = Color.*HSBtoRGB*(hue,1f,1f);  
 upDisplay.drawPixel(i,j,rgbColor);  
 }  
 }  
 }  
 upDisplay.repaint();//Обновление JimageDisplay в соответствии с цветом для каждого пикселя  
 }  
  
 private class ResetDis implements ActionListener  
 {  
 public void actionPerformed(ActionEvent e)  
 {  
 if (e.getSource() instanceof JComboBox)  
 {  
 JComboBox cB = (JComboBox) e.getSource();  
 fractalF = (FractalGenerator) cB.getSelectedItem();  
 fractalF.getInitialRange(range);  
 drawFractal();  
 } else if (e.getActionCommand().equals("Reset"))  
 {  
 fractalF.getInitialRange(range);  
 drawFractal();  
 } else if (e.getActionCommand().equals("Save"))  
 {  
 JFileChooser fileChooser = new JFileChooser();  
  
 FileNameExtensionFilter eFilter = new FileNameExtensionFilter("PNG IMAGES","png");  
 fileChooser.setFileFilter(eFilter);  
   
 fileChooser.setAcceptAllFileFilterUsed(false);  
  
 int sel = fileChooser.showSaveDialog(upDisplay);  
  
 if(sel == JFileChooser.*APPROVE\_OPTION*)  
 {  
 java.io.File file = fileChooser.getSelectedFile();  
 String filename = file.toString();  
 try {  
 BufferedImage bufferedImage = upDisplay.bufferedImage;  
 javax.imageio.ImageIO.*write*(bufferedImage,"png",file);  
 }  
 catch(Exception exception) {  
 JOptionPane.*showMessageDialog*(upDisplay,exception.getMessage(),"Can't save file",JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 else return;  
 }  
 }  
 }  
 private class MousePiPi extends MouseAdapter  
 {  
 public void mouseClicked(MouseEvent e)  
 {  
 int x = e.getX();  
 double xCoord = fractalF.*getCoord*(range.x, range.x+range.width,displaySize, x);  
 int y = e.getY();  
 double yCoord = fractalF.*getCoord*(range.y, range.y+range.height,displaySize, y);  
 fractalF.recenterAndZoomRange(range, xCoord,yCoord,0.5);  
 drawFractal();  
 }  
 }  
  
 public static void main(String[] args)  
 {  
 FractalExplorer display1 = new FractalExplorer(500);  
 display1.createAndShowGUI();  
 display1.drawFractal();  
 }  
}

*Реализация класса Mandelbrot:*

import java.awt.geom.Rectangle2D;  
  
public class Mandelbrot extends FractalGenerator {  
 public static final int *MAX\_ITERATIONS* = 2000;  
 @Override  
 public void getInitialRange(Rectangle2D.Double range) {  
 range.x = -2;  
 range.y = -1.5;  
 range.width = 4;  
 range.height = 4;  
 }  
  
 @Override  
 public int numIterations(double x, double y) {  
 int iter = 0;  
 double real = 0;  
 double notreal = 0;  
  
 while (iter < *MAX\_ITERATIONS* && real\*real+notreal\*notreal<4)  
 {  
 double real2 = real\*real - notreal\*notreal + x;  
 double notreal2 = 2 \* real \* notreal + y;  
 real = real2;  
 notreal = notreal2;  
 iter+=1;  
 }  
 if (iter == *MAX\_ITERATIONS*)  
 return -1;  
  
 return iter;  
 }  
 public String toString()  
 {  
 return "Mandelbrot";  
 }  
}

*Реализация класса Tricon:*

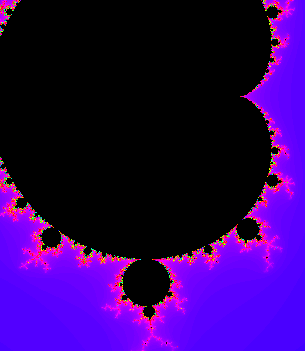
public static final int *MAX\_ITERATIONS* = 2000;  
 @Override  
 public void getInitialRange(Rectangle2D.Double range) {  
 range.x = -2;  
 range.y = -2;  
 range.width = 4;  
 range.height = 4;  
 }  
  
 @Override  
 public int numIterations(double x, double y) {  
 int iter = 0;  
 double real = 0;  
 double notreal = 0;  
  
 while (iter < *MAX\_ITERATIONS* && real\*real+notreal\*notreal<4)  
 {  
 double real2 = real\*real - notreal\*notreal + x;  
 double notreal2 = -2 \* real \* notreal + y;  
 real = real2;  
 notreal = notreal2;  
 iter+=1;  
 }  
 if (iter == *MAX\_ITERATIONS*)  
 return -1;  
  
 return iter;}  
 public String toString()  
 {  
 return "Tricorn";  
 }  
}

*Реализация класса Burning Ship:*

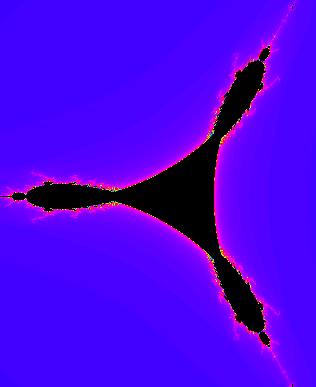
import java.awt.geom.Rectangle2D;  
  
public class BurningShip extends FractalGenerator {  
 public static final int *MAX\_ITERATIONS* = 2000;  
 @Override  
 public void getInitialRange(Rectangle2D.Double range) {  
 range.x = -1;  
 range.y = -1.5;  
 range.width = 2;  
 range.height = 2;  
 }  
  
 @Override  
 public int numIterations(double x, double y) {  
 int iter = 0;  
 double real = 0;  
 double notreal = 0;  
  
 while (iter < *MAX\_ITERATIONS* && real\*real+notreal\*notreal<4)  
 {  
 double real2 = real\*real - notreal\*notreal + x;  
 double notreal2 = 2 \* Math.*abs*(real) \* Math.*abs*(notreal) + y;  
 real = real2;  
 notreal = notreal2;  
 iter+=1;  
 }  
 if (iter == *MAX\_ITERATIONS*)  
 return -1;  
  
 return iter;  
 }  
 public String toString()  
 {  
 return "Burning Ship";  
 }  
}

*Результаты:*

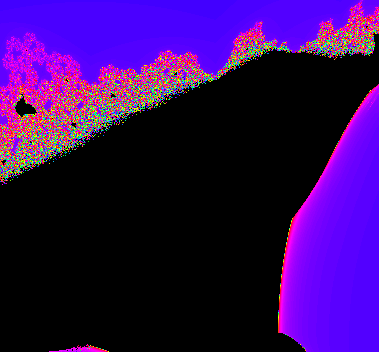
*Mandelbrot:*

**

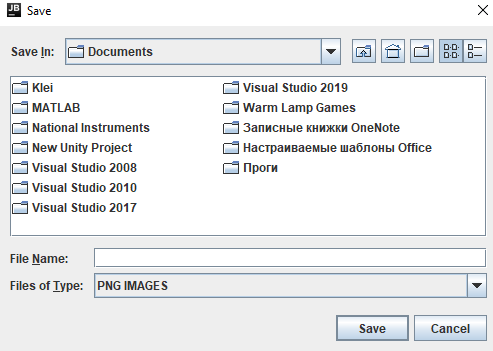
*Tricorn:*

**

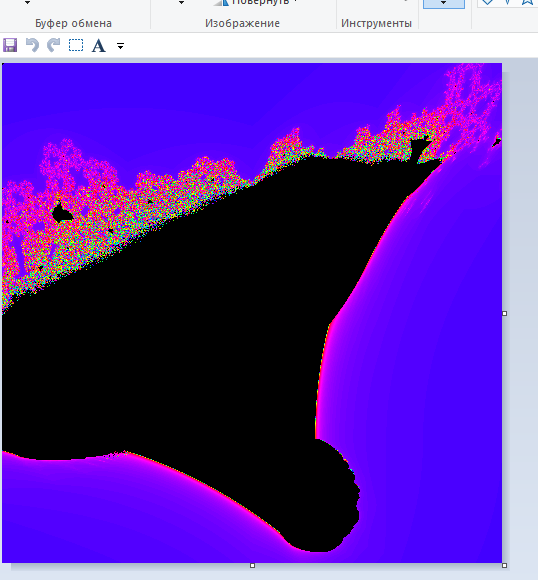
*Burning Ship:*



*Сохранение:*

**

*Файл:*

**

*Выводы:* расширил генератор фракталов. Добавил поддержку нескольких фракталов и реализовал возможность выбирать нужный фрактал из выпадающего списка.