**Федеральное агентство связи**

**Ордена Трудового Красного Знамени федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

**“Московский технический университет связи и информатики”**

**Кафедра МКиИТ**

**ЛАБОРАТОРНАЯ РАБОТА №6**

**Многопоточный генератор фракталов**

Выполнил: студент гр. БСУ1901 Панов Д. А.

Проверил: ст.преп. Мосева М.С.

Москва 2021 г.

ОГЛАВЛЕНИЕ

I ХОД РАБОТЫ

Выполнение работы3-11

I. ЗАДАНИЕ

import java.awt.geom.Rectangle2D;  
  
  
public abstract class FractalGenerator {  
  
  
 public static double getCoord(double rangeMin, double rangeMax, int size, int coord) {  
  
 assert size > 0;  
 assert coord >= 0 && coord < size;  
  
 double range = rangeMax - rangeMin;  
 return rangeMin + (range \* (double) coord / (double) size);  
 }  
  
  
 // позволяет генератору фракталов определить наиболее «интересную»  
 // область комплексной плоскости для конкретного фрактала  
  
  
 public abstract void getInitialRange(Rectangle2D.Double range);  
  
  
 // Обновляет текущий диапазон, чтобы он был центрирован по указанным координатам,  
 // и для увеличения или уменьшения с указанным коэффициентом масштабирования.  
  
 public void recenterAndZoomRange(Rectangle2D.Double range, double centerX, double centerY, double scale) {  
  
 double newWidth = range.width \* scale;  
 double newHeight = range.height \* scale;  
  
 range.x = centerX - newWidth / 2;  
 range.y = centerY - newHeight / 2;  
 range.width = newWidth;  
 range.height = newHeight;  
 }  
  
  
 // реализует итеративную функцию для фрактала Мандельброта  
 //Учитывая координату <em> x </em> + <em>iy</em> на комплексной плоскости,  
 //вычисляет и возвращает количество итераций перед фракталом функция выходит за пределы ограничивающей  
 // области для этой точки. Дело в том, что не исчезает, пока не будет указан предел итераций с результатом -1.  
  
 public abstract int numIterations(double x, double y);  
}

Рисунок 1 – Код файла FractalGenerator.java

import javax.swing.\*;  
import java.awt.\*;  
import java.io.IOException;  
import javax.imageio.ImageIO;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
import javax.swing.filechooser.FileNameExtensionFilter;  
import java.awt.geom.Rectangle2D;  
import javax.swing.\*;  
import javax.swing.event.ChangeEvent;  
import javax.swing.JFileChooser;  
import javax.swing.filechooser.FileNameExtensionFilter;  
import javax.swing.filechooser.FileFilter;  
import javax.swing.event.ChangeListener;  
import javax.swing.JFrame;  
  
public class FractalExplorer {  
 private int displaySize;  
 private JImageDisplay imageDisplay;  
 private FractalGenerator fractalGenerator;  
 private Rectangle2D.Double range;  
 private JComboBox comboBox;  
 private int rowsRemaining;  
 private JButton buttonReset;  
 private JButton buttonSave;  
  
 private FractalExplorer (int displaySize) {  
 this.displaySize = displaySize;  
 this.fractalGenerator = new Mandelbrot();  
 this.range = new Rectangle2D.Double(0,0,0,0);  
 fractalGenerator.getInitialRange(this.range);  
 }  
  
 // точка входа  
 public static void main(String[] args) {  
 FractalExplorer fractalExplorer = new FractalExplorer(600);  
 fractalExplorer.setGUI();  
 fractalExplorer.drawFractal();  
 }  
  
 // задание интерфейса  
 public void setGUI() {  
 JFrame frame = new JFrame("Fractal Generator");  
 JPanel jPanel\_1 = new JPanel();  
 JPanel jPanel\_2 = new JPanel();  
 JLabel label = new JLabel("Fractal:");  
  
 imageDisplay = new JImageDisplay(displaySize, displaySize);  
 imageDisplay.addMouseListener(new MouseListener());  
  
 // выпадающий список  
 comboBox = new JComboBox();  
 comboBox.addItem(new Mandelbrot());  
 comboBox.addItem(new Tricorn());  
 comboBox.addItem(new BurningShip());  
 comboBox.addActionListener(new ActionHandler());  
  
 // кнопка reset  
 buttonReset = new JButton("Reset");  
 buttonReset.setActionCommand("Reset");  
 buttonReset.addActionListener(new ActionHandler());  
  
 // кнопка сохранить  
 buttonSave = new JButton("Save image");  
 buttonSave.setActionCommand("Save");  
 buttonSave.addActionListener(new ActionHandler());  
  
 jPanel\_1.add(label, BorderLayout.CENTER);  
 jPanel\_1.add(comboBox, BorderLayout.CENTER);  
 jPanel\_2.add(buttonReset, BorderLayout.CENTER);  
 jPanel\_2.add(buttonSave, BorderLayout.CENTER);  
  
 frame.setLayout(new BorderLayout());  
 frame.add(imageDisplay, BorderLayout.CENTER);  
 frame.add(jPanel\_1, BorderLayout.NORTH);  
 frame.add(jPanel\_2, BorderLayout.SOUTH);  
 frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
  
 frame.pack();  
 frame.setVisible(true);  
 frame.setResizable(false);  
 }  
  
 // отрисовка фрактала в JImageDisplay  
 private void drawFractal() {  
 // отключаем интерфейс на момент рисования  
 enableGUI(false);  
 rowsRemaining = displaySize;  
 for (int i = 0; i < displaySize; i++) {  
 FractalWorker drawRow = new FractalWorker(i);  
 drawRow.execute();  
 }  
 }  
 // включение - отключение gui  
 public void enableGUI(boolean b) {  
 buttonSave.setEnabled(b);  
 buttonReset.setEnabled(b);  
 comboBox.setEnabled(b);  
 }  
  
 // обработчик кнопок  
 public class ActionHandler implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (e.getActionCommand().equals("Reset")) {  
 // перерисовка фрактала  
 fractalGenerator.getInitialRange(range);  
 drawFractal();  
 } else if (e.getActionCommand().equals("Save")) {  
 // сохранение  
 JFileChooser fileChooser = new JFileChooser();  
 FileNameExtensionFilter fileFilter = new FileNameExtensionFilter("PNG Images", "png");  
 fileChooser.setFileFilter(fileFilter);  
 fileChooser.setAcceptAllFileFilterUsed(false);  
 int t = fileChooser.showSaveDialog(imageDisplay);  
 if (t == JFileChooser.APPROVE\_OPTION) {  
 try {  
 ImageIO.write(imageDisplay.getImage(), "png", fileChooser.getSelectedFile());  
 } catch (NullPointerException | IOException ee) {  
 JOptionPane.showMessageDialog(imageDisplay, ee.getMessage(), "Cannot save image", JOptionPane.ERROR\_MESSAGE);  
 }  
 }  
 } else {  
 fractalGenerator = (FractalGenerator) comboBox.getSelectedItem();  
 range = new Rectangle2D.Double(0,0,0,0);  
 fractalGenerator.getInitialRange(range);  
 drawFractal();  
 }  
 }  
 }  
  
 public class MouseListener extends MouseAdapter {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 double x = FractalGenerator.getCoord(range.x, range.x + range.width, displaySize, e.getX());  
 double y = FractalGenerator.getCoord(range.y, range.y + range.width, displaySize, e.getY());  
 fractalGenerator.recenterAndZoomRange(range, x, y, 0.5);  
 drawFractal();  
 }  
 }  
  
 public class FractalWorker extends SwingWorker<Object, Object> {  
 private int y\_coord;  
 private int[] rgb;  
  
 public FractalWorker(int y\_coord) {  
 this.y\_coord = y\_coord;  
 }  
  
 @Override  
 protected Object doInBackground() throws Exception {  
 rgb = new int[displaySize];  
 for (int i = 0; i < displaySize; i++) {  
 int count = fractalGenerator.numIterations(FractalGenerator.getCoord(range.x, range.x + range.width, displaySize, i),  
 FractalGenerator.getCoord(range.y, range.y+range.width, displaySize, y\_coord));  
 if (count == -1)  
 rgb[i] = 0;  
 else {  
 double hue = 0.7f + (float) count / 200f;  
 int rgbColor = Color.HSBtoRGB((float) hue, 1f, 1f);  
 rgb[i] = rgbColor;  
 }  
 }  
 return null;  
 }  
 @Override  
 protected void done() {  
 for (int i = 0; i < displaySize; i++) {  
 imageDisplay.drawPixel(i, y\_coord, rgb[i]);  
 }  
 imageDisplay.repaint(0,0,y\_coord,displaySize,1);  
 rowsRemaining--;  
 if (rowsRemaining == 0)  
 enableGUI(true);  
 }  
 }  
}

Рисунок 2 – Код файла FractalExplorer.java

// Этот класс позволяет нам отображать наши фракталы.  
import javax.swing.JComponent;  
import java.awt.image.BufferedImage;  
import java.awt.Dimension;  
import java.awt.BorderLayout;  
import java.awt.event.ActionListener;  
import java.awt.event.MouseListener;  
import java.awt.\*;  
import java.awt.event.ItemEvent;  
import java.awt.event.ActionEvent;  
import java.awt.event.ItemListener;  
import java.awt.event.ActionListener;  
import java.awt.geom.Rectangle2D;  
import javax.swing.\*;  
import javax.swing.event.ChangeEvent;  
import javax.swing.event.ChangeListener;  
  
class JImageDisplay extends JComponent  
{  
 // управляет изображением, содержимое которого можно записать.  
private BufferedImage displayImage;  
  
public BufferedImage getImage() {  
 return displayImage;  
 }  
 // принимает целочисленные значения ширины и высоты, и инициализирует объект BufferedImage новым  
 //изображением с этой шириной и высотой, и типом изображения TYPE\_INT\_RGB.  
 public JImageDisplay (int width, int height)  
 {  
displayImage = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB);  
Dimension imageDimension = new Dimension(width, height);  
super.setPreferredSize(imageDimension);  
  
 }  
  
 // код для отрисовки  
  
 @Override  
public void paintComponent(Graphics g)  
 {  
super.paintComponent(g);  
g.drawImage(displayImage, 0, 0, displayImage.getWidth(), displayImage.getHeight(), null);  
 }  
 // устанавливает все пиксели изображения в черный цвет  
public void clearImage()  
 {  
int[] blankArray = new int[getWidth() \* getHeight()];  
displayImage.setRGB(0, 0, getWidth(), getHeight(), blankArray, 0, 1);  
 }  
  
 // устанавливает пиксель в определенный цвет  
  
public void drawPixel(int x, int y, int rgbColor)  
 {  
  
 displayImage.setRGB(x, y, rgbColor);  
}  
}

Рисунок 3 – Код файла JImageDisplay.java

import java.awt.geom.Rectangle2D;  
public class Mandelbrot extends FractalGenerator  
{  
  
public static final int MAX\_ITERATIONS = 2000;  
  
 // этот метод  
 //должен установить начальный диапазон в (-2 - 1.5i) - (1 + 1.5i)  
  
  
public void getInitialRange(Rectangle2D.Double range)  
 {  
 range.x = -2;  
 range.y = -1.5;  
 range.width = 3;  
 range.height = 3;  
 }  
  
 // реализует итеративную  
 //функцию для фрактала Мандельброта  
  
  
 public int numIterations(double x, double y)  
 {  
 int iteration = 0;  
 double zreal = 0;  
 double zimaginary = 0;  
  
 while (iteration < MAX\_ITERATIONS &&  
 zreal \* zreal + zimaginary \* zimaginary < 4)  
 {  
 double zrealUpdated = zreal \* zreal - zimaginary \* zimaginary + x;  
 double zimaginaryUpdated = 2 \* zreal \* zimaginary + y;  
 zreal = zrealUpdated;  
 zimaginary = zimaginaryUpdated;  
 iteration += 1;  
 }  
  
 if (iteration == MAX\_ITERATIONS)  
 {  
 return -1;  
 }  
  
return iteration;  
 }  
 public String toString() {  
 return "Mandelbrot";  
 }  
}

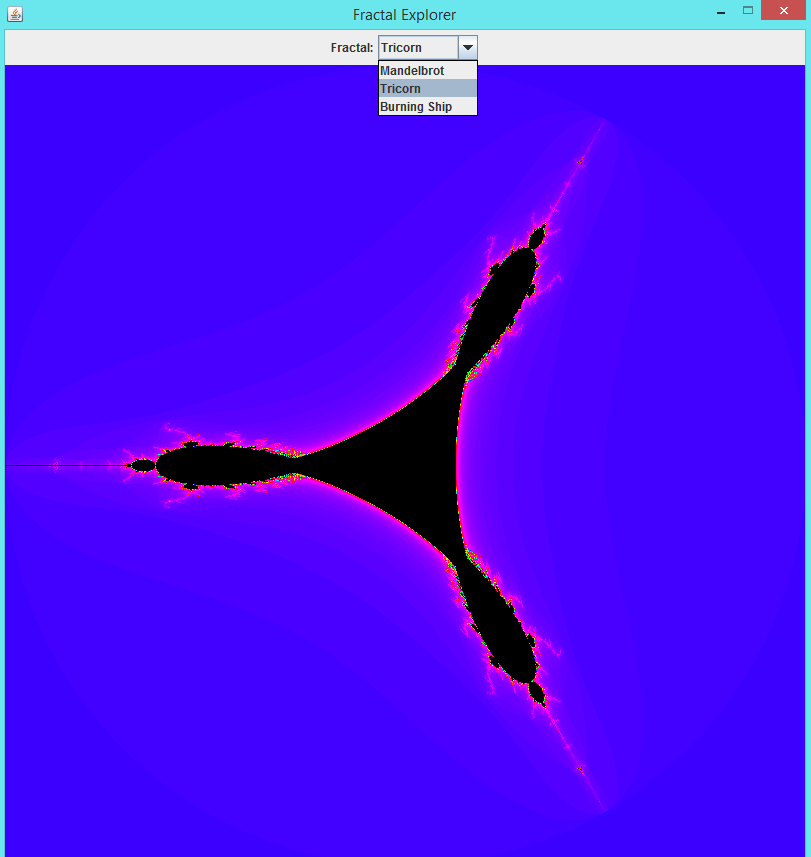
Рисунок 4 – Код файла Mandelbrot.java

import java.awt.geom.Rectangle2D;  
  
public class Tricorn extends FractalGenerator  
{  
public static final int MAX\_ITERATIONS = 2000;  
 public void getInitialRange(Rectangle2D.Double range)  
 {  
range.x= -2;  
range.y= -2;  
range.width= 4;  
range.height= 4;  
}  
public int numIterations(double x, double y)  
 {  
int iteration = 0;  
 double zreal = 0;  
 double zimaginary = 0;  
 while (iteration <MAX\_ITERATIONS &&  
zreal \* zreal + zimaginary \* zimaginary<4)  
 {  
double zrealUpdated = zreal \* zreal - zimaginary \* zimaginary + x;  
 double zimaginaryUpdated = -2 \* zreal \* zimaginary + y;  
zreal = zrealUpdated;  
zimaginary = zimaginaryUpdated;  
iteration += 1;  
}  
if (iteration == MAX\_ITERATIONS)  
 {  
return -1;  
}  
  
return iteration;  
}  
  
public String toString() {  
return "Tricorn";  
}  
  
}

Рисунок 5 – Код файла Tricorn.java

import java.awt.geom.Rectangle2D;  
  
public class BurningShip extends FractalGenerator {  
public static final int MAX\_ITERATIONS = 2000;  
  
 public void getInitialRange(Rectangle2D.Double range) {  
range.x= -2;  
range.y= -2.5;  
range.width= 4;  
range.height= 4;  
}  
public int numIterations(double x, double y) {  
int iteration = 0;  
 double zreal = 0;  
 double zimaginary = 0;  
 while (iteration <MAX\_ITERATIONS &&  
zreal \* zreal + zimaginary \* zimaginary<4) {  
double zrealUpdated = zreal \* zreal - zimaginary \* zimaginary + x;  
 double zimaginaryUpdated = 2 \* Math.abs(zreal)  
 \* Math.abs(zimaginary) + y;  
  
zreal = zrealUpdated;  
zimaginary = zimaginaryUpdated;  
  
iteration += 1;  
}  
  
if (iteration == MAX\_ITERATIONS) {  
return -1;  
}  
  
return iteration;  
}  
public String toString() {  
return "Burning Ship";  
}  
  
}

Рисунок 6 – Код файла BurningShip.java



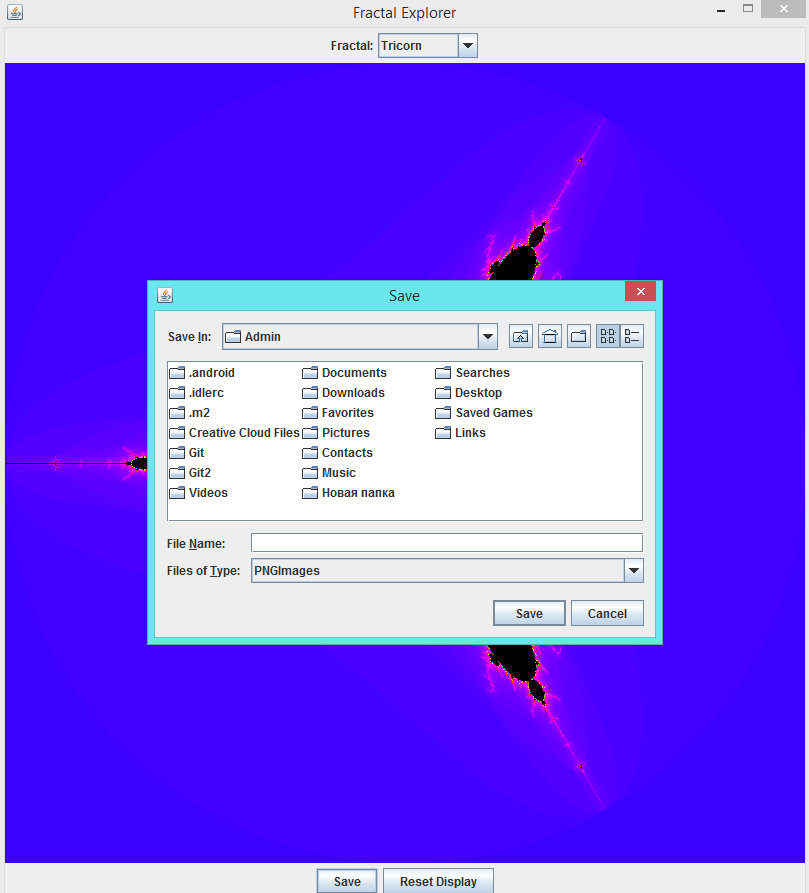


Рисунок 7 – Результат работы программы