**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 дисциплине «КТП»

на тему:

«**Многопоточный генератор фракталов»**

Выполнила: студентка группы БВТ1802

Лаврухина Елена Павловна

Руководитель:

Полянцева Ксения Андреевна

Москва 2020

Цель работы

Реализовать возможность рисования фрактала с несколькими фоновыми потоками.

Выполнение

Код программы

1. JImageDisplay

import javax.swing.\*;  
import java.awt.image.\*;  
import java.awt.\*;  
*// наследование от класса JComponent*public class JImageDisplay extends JComponent {  
 *// для управления изображением* private BufferedImage image;  
 *// получение изображения из другого класса* public BufferedImage getImage() {  
 return image;  
 }  
 *// новые значения длины и ширины изображения и его тип* public JImageDisplay(int width, int height) {  
 image = new BufferedImage(width, height, BufferedImage.*TYPE\_INT\_RGB*);  
 Dimension imageDimension = new Dimension(width, height);  
 *// метод, который включит наш компонент в пользовательский интерфейс* super.setPreferredSize(imageDimension);  
 }  
 *// отрисовка* public void paintComponent (Graphics g) {  
 super.paintComponent(g);  
 g.drawImage (image, 0, 0, image.getWidth(), image.getHeight(), null);  
 }  
 *// установка всех пикселей изображения в черный цвет* public void clearImage() {  
 for (int i=0;i<image.getWidth();i++)  
 for (int j=0;j<image.getHeight();j++)  
 image.setRGB(i,j,0);  
 }  
 *// установка пикселя в определенный цвет* public void drawPixel(int x, int y, int rgbColor) {  
 image.setRGB(x,y,rgbColor);  
 }  
}

1. Mandelbrot

import java.awt.geom.Rectangle2D;  
*// подкласс FractalGenerator*public class Mandelbrot extends FractalGenerator {  
 *// константа с максимальным количеством итераций* public static final int *MAX\_ITERATIONS* = 2000;  
 *// начальный диапазон* public void getInitialRange(Rectangle2D.Double range) {  
 range.x = -2;  
 range.y = -1.5;  
 range.width = 3;  
 range.height = 3;  
 }  
 *//Вычисляем количество итераций с помощью мнимой и реальной части* public int numIterations(double x, double y) {  
 int iterations = 0;  
 double zreal = 0;  
 double zimaginary = 0;  
  
 while (iterations < *MAX\_ITERATIONS* && zreal \* zreal + zimaginary \* zimaginary < 4) {  
 double zreal1 = zreal \* zreal - zimaginary \* zimaginary + x;  
 double zimaginary1 = 2 \* zreal \* zimaginary + y;  
 zreal = zreal1;  
 zimaginary = zimaginary1;  
 iterations += 1;  
 } *// Если достигнут максимум, вернуть -1 (точка не вышла за пределы границы)* if (iterations == *MAX\_ITERATIONS*) {  
 return -1;  
 }  
 return iterations;  
 }  
 *// возвращаем имя фрактала* public String toString() {  
 return "Mandelbrot";  
 }  
}

1. FractalExplorer

import java.awt.\*;  
import javax.swing.\*;  
import javax.imageio.ImageIO;  
import java.awt.geom.Rectangle2D;  
import java.awt.event.\*;  
import java.io.File;  
import javax.swing.filechooser.\*;  
import java.awt.image.\*;  
  
public class FractalExplorer {  
 private JButton saveButton;  
 private JButton resetButton;  
 private JComboBox ComboBox;  
 private int rowsRemaining;  
 private int displaySize;  
 private JImageDisplay display;  
 private FractalGenerator fractal;  
 private Rectangle2D.Double range;  
  
 public FractalExplorer(int size) {  
 displaySize = size;  
 fractal = new Mandelbrot();  
 range = new Rectangle2D.Double();  
 fractal.getInitialRange(range);  
 display = new JImageDisplay(displaySize, displaySize);  
 }  
 public void createAndShowGUI() {  
 display.setLayout(new BorderLayout());  
 JFrame frame = new JFrame("Fractal Explorer");  
 *// позиция center* frame.add(display, BorderLayout.*CENTER*);  
 *// кнопка сброса* resetButton = new JButton("Reset");  
 *// обработчик кнопки сброс* ButtonHandler resetHandler = new ButtonHandler();  
 resetButton.addActionListener(resetHandler);  
 *// обработчик для мыши* MouseHandler click = new MouseHandler();  
 display.addMouseListener(click);  
 *// закрытие окна* frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 *// поле со списком* ComboBox = new JComboBox();  
 *// добавляем объекты фрактала в поле со списком* FractalGenerator mandelbrotFractal = new Mandelbrot();  
 ComboBox.addItem(mandelbrotFractal);  
 FractalGenerator tricornFractal = new Tricorn();  
 ComboBox.addItem(tricornFractal);  
 FractalGenerator burningShipFractal = new BurningShip();  
 ComboBox.addItem(burningShipFractal);  
 *// обработчик кнопок в поле со списком* ButtonHandler fractalChooser = new ButtonHandler();  
 ComboBox.addActionListener(fractalChooser);  
 *// верхняя панель* JPanel Panel = new JPanel();  
 JLabel Label = new JLabel("Fractal:");  
 Panel.add(Label);  
 Panel.add(ComboBox);  
 frame.add(Panel, BorderLayout.*NORTH*);  
 *// нижняя панель* saveButton = new JButton("Save Image");  
 JPanel BottomPanel = new JPanel();  
 BottomPanel.add(saveButton);  
 BottomPanel.add(resetButton);  
 frame.add(BottomPanel, BorderLayout.*SOUTH*);  
 *// обработчик кнопки save* ButtonHandler saveHandler = new ButtonHandler();  
 saveButton.addActionListener(saveHandler);  
 frame.pack();  
 *// видимость изображения* frame.setVisible(true);  
 *// запрет изменения размера* frame.setResizable(false);  
 }  
 private void drawFractal() {  
 *// отключаем все элементы управления пользовательского интерфейса во время рисования* enableUI(false);  
 rowsRemaining = displaySize;  
 *// перебираем каждую строку на дисплее и рисуем* for (int x=0; x < displaySize; x++) {  
 FractalWorker drawRow = new FractalWorker(x);  
 drawRow.execute();  
 }  
 }  
 *// Включаем или отключаем кнопки интерфейса и поля со списком* private void enableUI(boolean val) {  
 ComboBox.setEnabled(val);  
 resetButton.setEnabled(val);  
 saveButton.setEnabled(val);  
 }  
 private class ButtonHandler implements ActionListener {  
 public void actionPerformed(ActionEvent e) {  
 String command = e.getActionCommand();  
 if (e.getSource() instanceof JComboBox) {  
 *// выводим фракталы* JComboBox Source = (JComboBox) e.getSource();  
 fractal = (FractalGenerator) Source.getSelectedItem();  
 fractal.getInitialRange(range);  
 drawFractal();  
 }  
 *// сбрасываем* else if (command.equals("Reset")) {  
 fractal.getInitialRange(range);  
 drawFractal();  
 }  
 *// сохраняем* else if (command.equals("Save Image")) {  
 *// выбираем файл для сохранения* JFileChooser FileChooser = new JFileChooser();  
 *// только png* FileFilter Filter = new FileNameExtensionFilter("PNG Images", "png");  
 FileChooser.setFileFilter(Filter);  
 FileChooser.setAcceptAllFileFilterUsed(false);  
 *// выбор директории* int userSelection = FileChooser.showSaveDialog(display);  
 if (userSelection == JFileChooser.*APPROVE\_OPTION*) {  
 *// доступ к файлу и имени файла* File file = FileChooser.getSelectedFile();  
 String file\_name = file.toString();  
 *// сохраняем фрактальное изображение* try {  
 BufferedImage image = display.getImage();  
 ImageIO.*write*(image, "png", file);  
 }  
 *// исключения* catch (Exception exception) {  
 JOptionPane.*showMessageDialog*(display,  
 exception.getMessage(), "Cannot Save Image",  
 JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 else return;  
 }  
 }  
 }  
 private class MouseHandler extends MouseAdapter {  
 public void mouseClicked(MouseEvent e) {  
 if (rowsRemaining != 0) {  
 return;  
 }  
 *// получение координат в области щелчка мыши* int x = e.getX();  
 double xCoord = fractal.*getCoord*(range.x, range.x + range.width, displaySize, x);  
 int y = e.getY();  
 double yCoord = fractal.*getCoord*(range.y, range.y + range.height, displaySize, y);  
 fractal.recenterAndZoomRange(range, xCoord, yCoord, 0.5);  
 *// перерисовываем фрактал* drawFractal();  
 }  
 }  
 *// вычисляем значения цвета для одного ряда фрактала.* private class FractalWorker extends SwingWorker<Object, Object> {  
 int yCoordinate;  
 int[] computedRGBValues;  
 private FractalWorker(int row) {  
 yCoordinate = row;  
 }  
 protected Object doInBackground() {  
 computedRGBValues = new int[displaySize];  
 *// перебираем все пиксели в строке* for (int i = 0; i < computedRGBValues.length; i++) {  
 *// Находим соответствующие координаты x и y* double xCoord = fractal.*getCoord*(range.x, range.x + range.width, displaySize, i);  
 double yCoord = fractal.*getCoord*(range.y, range.y + range.height, displaySize, yCoordinate);  
 *// вычисляем количество итераций* int iterations = fractal.numIterations(xCoord, yCoord);  
 *// вычисляем значения RGB массива для черного* if (iterations == -1) {  
 computedRGBValues[i] = 0;  
 }  
 else {  
 *// выбираем значение цвета в зависимости от итерации* float hue = 0.7f + (float) iterations / 200f;  
 int rgbColor = Color.*HSBtoRGB*(hue, 1f, 1f);  
 *// обновляем массив* computedRGBValues[i] = rgbColor;  
 }  
 }  
 return null;  
 }  
 protected void done() {  
 */\*Перебираем массив, рисуем в пикселях, которые были вычислены в doInBackground().  
 Перерисовываем строку, которая была изменена.\*/* for (int i = 0; i < computedRGBValues.length; i++) {  
 display.drawPixel(i, yCoordinate, computedRGBValues[i]);  
 }  
 display.repaint(0, 0, yCoordinate, displaySize, 1);  
 *// уменьшаем оставшиеся строки* rowsRemaining--;  
 if (rowsRemaining == 0) {  
 enableUI(true);  
 }  
 }  
 }  
 public static void main(String[] args) {  
 FractalExplorer displayExplorer = new FractalExplorer(600);  
 displayExplorer.createAndShowGUI();  
 displayExplorer.drawFractal();  
 }  
}

1. BurningShip

import java.awt.geom.Rectangle2D;  
*// подкласс FractalGenerator*public class BurningShip extends FractalGenerator {  
 *// константа с максимальным количеством итераций* public static final int *MAX\_ITERATIONS* = 2000;  
 *// начальный диапазон* public void getInitialRange(Rectangle2D.Double range) {  
 range.x = -2;  
 range.y = -2.5;  
 range.width = 3;  
 range.height = 3;  
 }  
 *//Вычисляем количество итераций с помощью мнимой и реальной части* public int numIterations(double x, double y) {  
 int iterations = 0;  
 double zreal = 0;  
 double zimaginary = 0;  
  
 while (iterations < *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;  
 iterations += 1;  
 } *// Если достигнут максимум, вернуть -1 (точка не вышла за пределы границы)* if (iterations == *MAX\_ITERATIONS*) {  
 return -1;  
 }  
 return iterations;  
 }  
 *// возвращаем имя фрактала* public String toString() {  
 return "Burning Ship";  
 }  
}

1. Tricorn

import java.awt.geom.Rectangle2D;  
*// подкласс FractalGenerator*public class Tricorn extends FractalGenerator {  
 *// константа с максимальным количеством итераций* public static final int *MAX\_ITERATIONS* = 2000;  
 *// начальный диапазон* public void getInitialRange(Rectangle2D.Double range) {  
 range.x = -2;  
 range.y = -2;  
 range.width = 4;  
 range.height = 4;  
 }  
 *//Вычисляем количество итераций с помощью мнимой и реальной части* public int numIterations(double x, double y) {  
 int iterations = 0;  
 double zreal = 0;  
 double zimaginary = 0;  
  
 while (iterations < *MAX\_ITERATIONS* && zreal \* zreal + zimaginary \* zimaginary < 4) {  
 double zrealUpdated = zreal \* zreal - zimaginary \* zimaginary + x;  
 double zimaginaryUpdated = -2 \* zreal \* zimaginary + y;  
 zreal = zrealUpdated;  
 zimaginary = zimaginaryUpdated;  
 iterations += 1;  
 } *// Если достигнут максимум, вернуть -1 (точка не вышла за пределы границы)* if (iterations == *MAX\_ITERATIONS*) {  
 return -1;  
 }  
 return iterations;  
 }  
 *// возвращаем имя фрактала* public String toString() {  
 return "Tricorn";  
 }  
}

Скриншоты работы программы

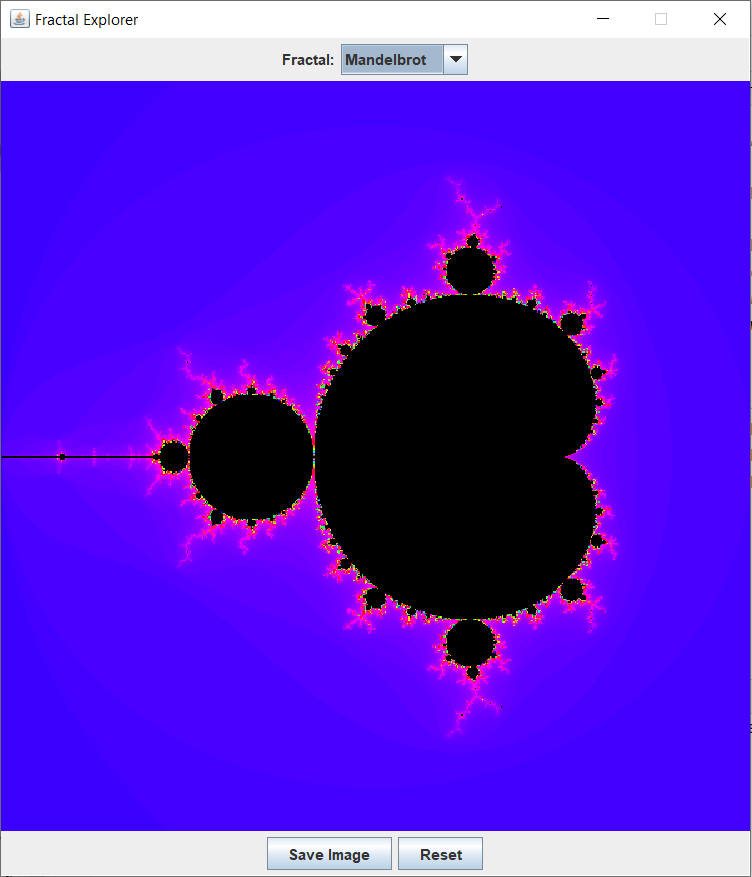


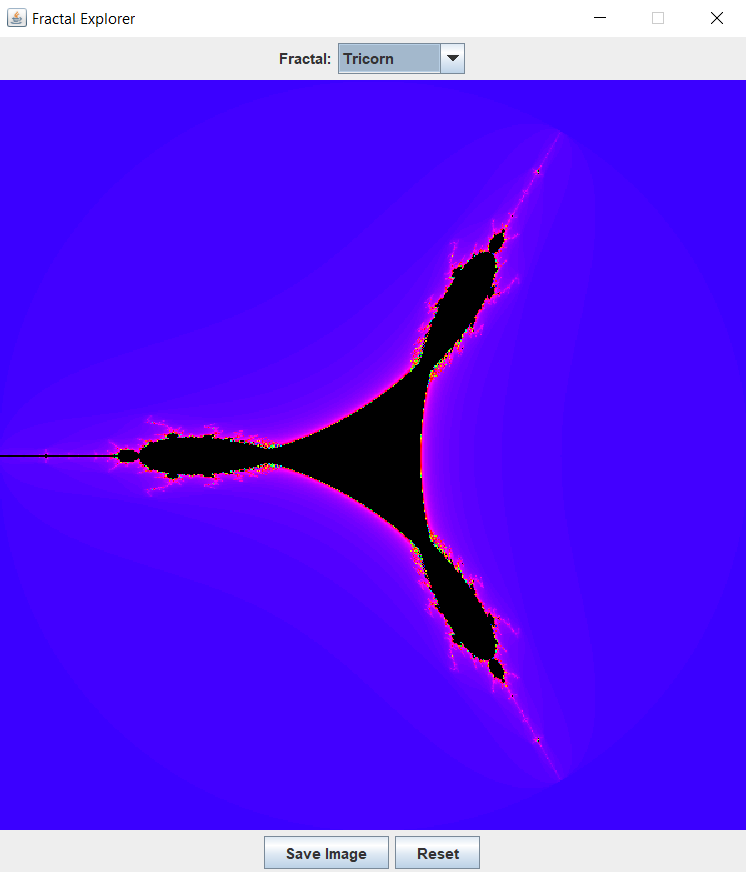
Рисунок 1 – Работа программы 

Рисунок 2 – Работа программы

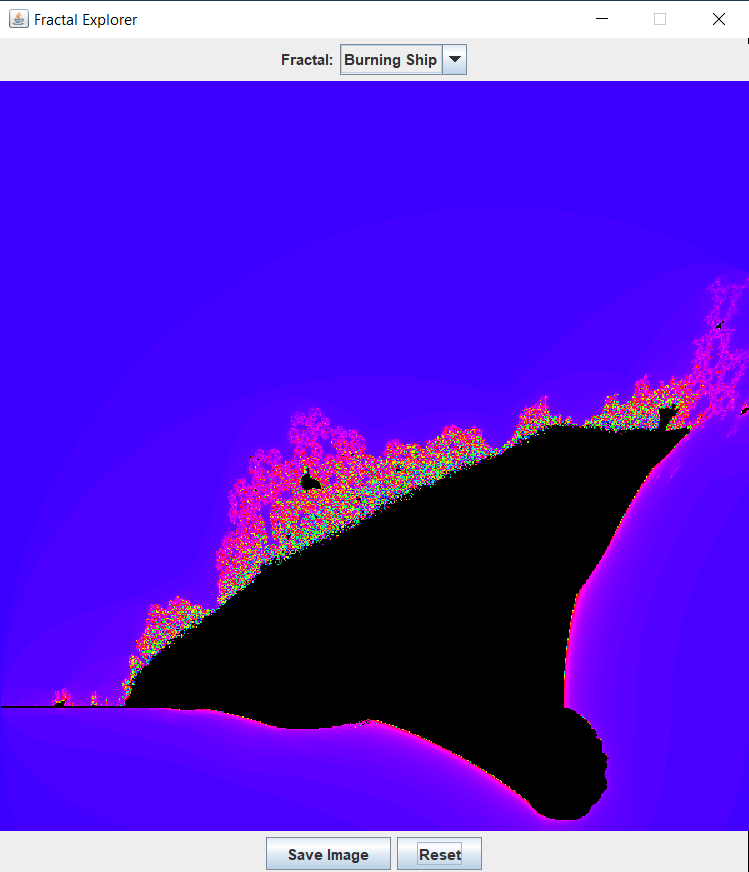


Рисунок 3 – Работа программы

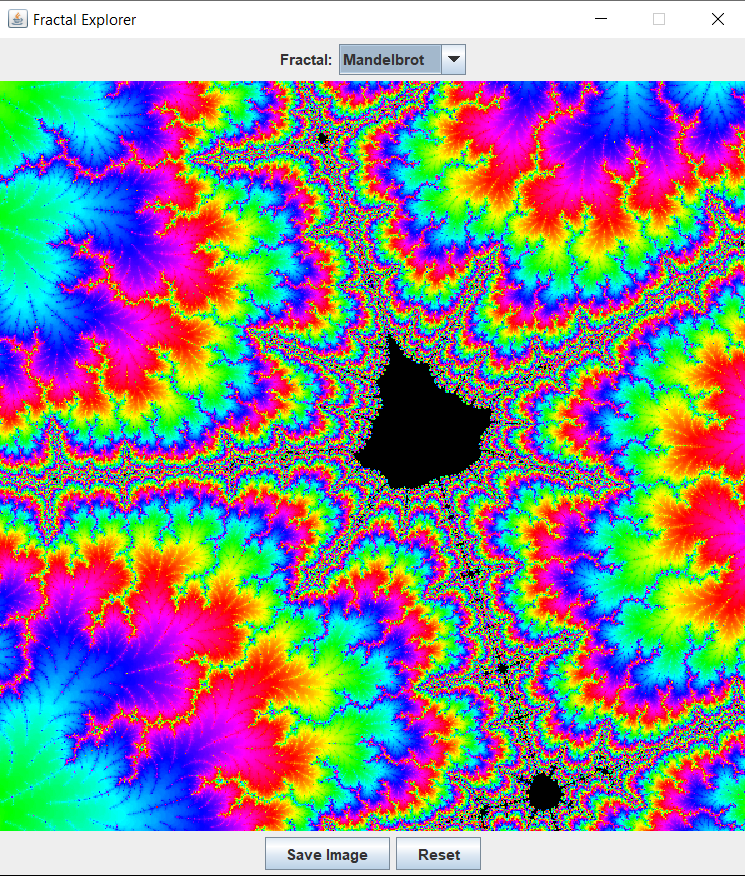


Рисунок 4 – Работа программы

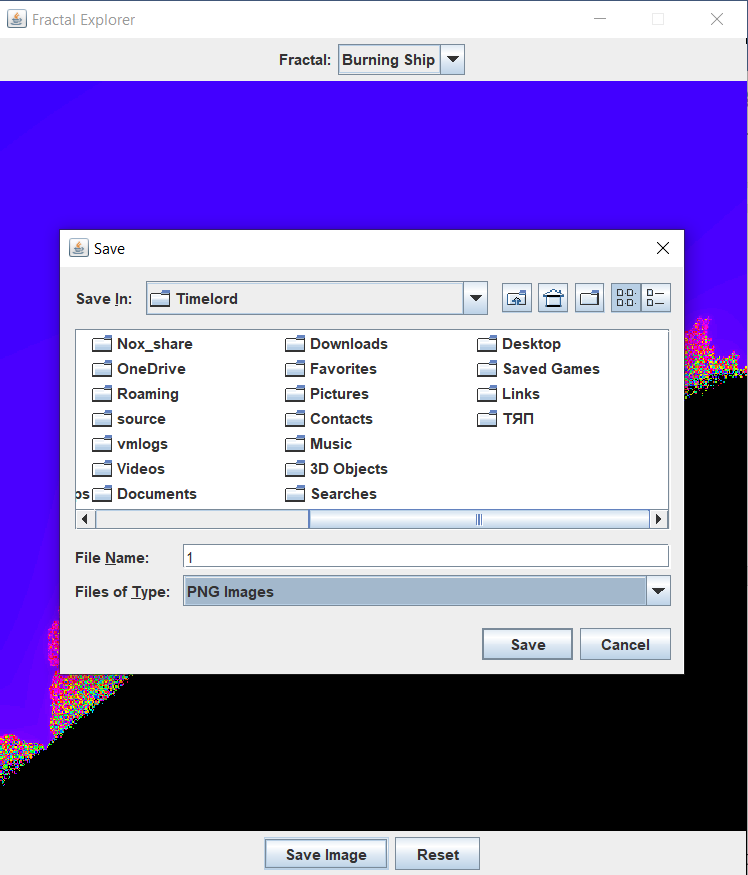


Рисунок 5 – Работа программы

Вывод

В ходе работы мы добавили в наше приложение возможность рисования фрактала с несколькими фоновыми потоками.