**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 предмету «Кроссплатформенные технологии программирования»

на тему:

«Многопоточный генератор фракталов»

Выполнил: студент группы БВТ 1802

Молчанова А.А.

Руководитель:

Полянцева К.А

Москва 2020

**Цель работы**

В данной лабораторной работе необходимо будет реализовать возможность рисования фрактала с несколькими фоновыми потоками.

**Задания**

В этой лабораторной работе нужно изменить программу так, чтобы она использовала один или несколько фоновых потоков для вычисления фрактала. В частности, поток обработки событий не будет использоваться для вычисления фрактала.

Необходимо соблюдать важное ограничение Swing, а именно, что в потоке обработке событий происходит взаимодействие только с компонентами Swing. Для этого Swing предоставляет инструменты, что упрощает поставленную задачу.

Создать подкласс SwingWorker с именем FractalWorker, который будет внутренним классом FractalExplorer. Это необходимо для того, чтобы у него был доступ к нескольким внутренним членам FractalExplorer. В результате должна получиться следующая строчка кода: private class FractalWorker extends SwingWorker<Object, Object>.

Класс FractalWorker будет отвечать за вычисление значений цвета для одной строки фрактала, поэтому ему потребуются два поля: целочисленная y-координата вычисляемой строки, и массив чисел типа int для хранения вычисленных значений RGB для каждого пикселя в этой строке. Конструктор должен будет получать y-координату в качестве параметра и сохранять это.

Метод doInBackground() вызывается в фоновом потоке и отвечает за выполнение длительной задачи. Поэтому в реализации нужно будет взять часть кода из предыдущей функции «draw fractal» и поместить ее в этот метод.

Выделить память для массив целых чисел в начале реализации этого метода (массив должен быть достаточно большим для хранения целой строки значений цвета), а затем сохранить цвет каждого пикселя в этом массиве.

Метод doInBackground() должен возвращать объект типа Object, так как это указано в объявлении SwingWorker <T, V>. Просто вернуть null.

Метод done() вызывается, когда фоновая задача завершена, и этот метод вызывается из потока обработки событий Swing. После того, как строка будет вычислена, нужно будет сообщить Swing. После того, как завершится класс для фоновой задачи, следующим шагом нужно будет привязать его к процессу рисования фракталов.

Создать функцию void enableUI(boolean val), которая будет включать или отключать кнопки с выпадающим списком в пользовательском интерфейсе на основе указанного параметра.

Функция «draw fractal» должна сделать еще две вещи. Первая - она должна вызвать метод enableUI (false), чтобы отключить все элементы пользовательского интерфейса во время рисования. Вторая - она должна установить значение «rows remaining» равным общему количеству строк, которые нужно нарисовать.

В методе done(), уменьшить значение «rows remaining» на 1, как последний шаг данной операции. Затем, если после уменьшения значение «rows remaining» равно 0, вызвать метод enableUI (true).

Изменить реализацию mouse-listener для того, чтобы она сразу возвращалась в предыдущее состояние.

**Коды программ**

import java.awt.\*;

import javax.swing.\*;

import java.awt.geom.Rectangle2D;

import java.awt.event.\*;

import javax.swing.filechooser.\*;

import java.awt.image.\*;

public class FractalExplorer{

private JButton saveButton;

private JButton resetButton;

private JComboBox myComboBox;

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 myFrame = new JFrame("Fractal Explorer");

myFrame.add(display, BorderLayout.CENTER);

resetButton = new JButton("Reset");

ButtonHandler resetHandler = new ButtonHandler();

resetButton.addActionListener(resetHandler);

MouseHandler click = new MouseHandler();

display.addMouseListener(click);

myFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

myComboBox = new JComboBox();

FractalGenerator mandelbrotFractal = new Mandelbrot();

myComboBox.addItem(mandelbrotFractal);

FractalGenerator tricornFractal = new Tricorn();

myComboBox.addItem(tricornFractal);

FractalGenerator burningShipFractal = new BurningShip();

myComboBox.addItem(burningShipFractal);

ButtonHandler fractalChooser = new ButtonHandler();

myComboBox.addActionListener(fractalChooser);

JPanel myPanel = new JPanel();

JLabel myLabel = new JLabel("Fractal:");

myPanel.add(myLabel);

myPanel.add(myComboBox);

myFrame.add(myPanel, BorderLayout.NORTH);

saveButton = new JButton("Save");

JPanel myBottomPanel = new JPanel();

myBottomPanel.add(saveButton);

myBottomPanel.add(resetButton);

myFrame.add(myBottomPanel, BorderLayout.SOUTH);

ButtonHandler saveHandler = new ButtonHandler();

saveButton.addActionListener(saveHandler);

myFrame.pack();

myFrame.setVisible(true);

myFrame.setResizable(false);}

private void drawFractal(){

enableUI(false);

rowsRemaining = displaySize;

for (int x=0; x<displaySize; x++){

FractalWorker drawRow = new FractalWorker(x);

drawRow.execute();}}

private void enableUI(boolean val) {

myComboBox.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) {

JComboBox mySource = (JComboBox) e.getSource();

fractal = (FractalGenerator) mySource.getSelectedItem();

fractal.getInitialRange(range);

drawFractal();}

else if (command.equals("Reset")) {

fractal.getInitialRange(range);

drawFractal();}

else if (command.equals("Save")) {

JFileChooser myFileChooser = new JFileChooser();

FileFilter extensionFilter =

new FileNameExtensionFilter("PNG Images", "png");

myFileChooser.setFileFilter(extensionFilter);

myFileChooser.setAcceptAllFileFilterUsed(false);

int userSelection = myFileChooser.showSaveDialog(display);

if (userSelection == JFileChooser.APPROVE\_OPTION) {

java.io.File file = myFileChooser.getSelectedFile();

String file\_name = file.toString();

try {

BufferedImage displayImage = display.getImage();

javax.imageio.ImageIO.write(displayImage, "png", file);}

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;}

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);

drawFractal();}}

private class FractalWorker extends SwingWorker<Object, Object>{

int yCoordinate;

int[] computedRGBValues;

private FractalWorker(int row) {

yCoordinate = row;}

// метод, выполняющий фоновые операции

protected Object doInBackground() {

computedRGBValues = new int[displaySize];

for (int i = 0; i < computedRGBValues.length; i++) {

double xCoord = fractal.getCoord(range.x,

range.x + range.width, displaySize, i);

double yCoord = fractal.getCoord(range.y,

range.y + range.height, displaySize, yCoordinate);

int iteration = fractal.numIterations(xCoord, yCoord);

if (iteration == -1){

computedRGBValues[i] = 0;}

else {

float hue = 0.7f + (float) iteration / 200f;

int rgbColor = Color.HSBtoRGB(hue, 1f, 1f);

computedRGBValues[i] = rgbColor;}}

return null;}

// вызывается при завершении фоновой задачи

protected void done() {

for (int i = 0; i < computedRGBValues.length; i++) {

display.drawPixel(i, yCoordinate, computedRGBValues[i]);}

display.repaint(0, 0, yCoordinate, displaySize, 1);

rowsRemaining--;

if (rowsRemaining == 0) {

enableUI(true);}}}

public static void main(String[] args){

FractalExplorer displayExplorer = new FractalExplorer(600);

displayExplorer.createAndShowGUI();

displayExplorer.drawFractal();}}

**Результат работы кода**

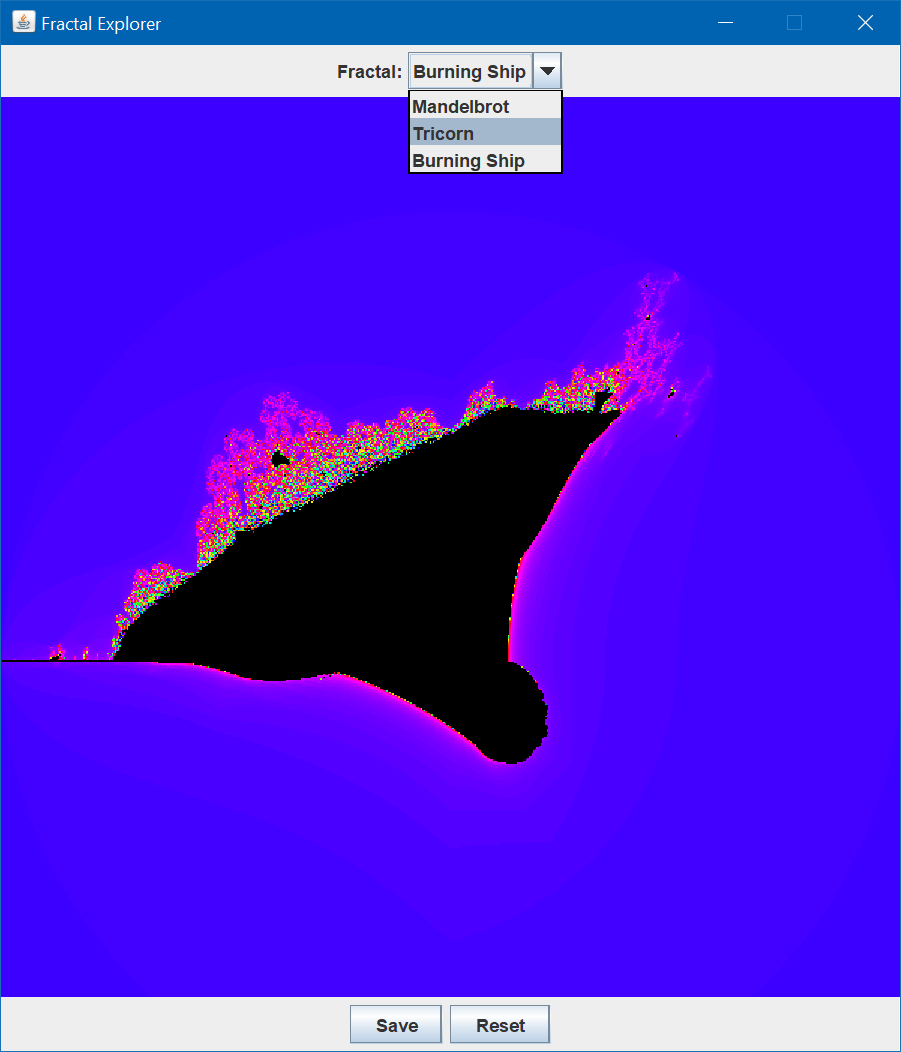


Рисунок 1 – Работа программы

**Вывод**

В ходе лабораторной работы изменили программу так, чтобы она использовала один или несколько фоновых потоков для вычисления фрактала. Создали подкласс SwingWorker с именем FractalWorker, который стал внутренним классом FractalExplorer. Создали функцию void enableUI(boolean val), которая стала включать или отключать кнопки с выпадающим списком в пользовательском интерфейсе на основе указанного параметра. Функция «draw fractal» делает две вещи: вызывает метод enableUI (false), чтобы отключить все элементы пользовательского интерфейса во время рисования, устанавливает значение «rows remaining» равным общему количеству строк, которые нужно нарисовать. Изменили реализацию mouse-listener для того, чтобы она сразу возвращалась в предыдущее состояние.