ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ

Ордена Трудового Красного Знамени

федеральное государственное бюджетное

образовательное учреждение высшего образования

«Московский Технический Университет Связи и Информатики» (МТУСИ)

Кафедра МКиИТ

Лабораторная работа по технологиям программирования №5

«Выбор и сохранение фракталов»

Лабораторная работа по технологиям программирования №6

«Многоядерный генератор фракталов»

Выполнил:

Студент 2 курса

Группы БСТ-1602

Абросимов Данил

Вариант №1

Москва, 2018

Оглавление

[1. Цель и задачи. 3](#_Toc510541482)

[2. Анализ предметной области и выбор инструментария. 3](#_Toc510541484)

[3. Код программы и объяснение функций. 3](#_Toc510541485)

[3.1. Класс FractalExplorer. 3](#_Toc510541486)

[3.2. Класс BurningShip. 3](#_Toc510541488)

[3.3. Класс Tricorn. 3](#_Toc510541489)

[4. Заключение. 4](#_Toc510541490)

1. Цель и задачи.

Доработать класс FractalExplorer, реализовав проверку точек на идентичность и реализовав метод кучи. Добавить 2 фрактала: Burning Ship, Tricorn. Создать выпадающий список, где можно будет выбрать нужный фрактал, реализовать рисование в фоновом режиме с помощью SwingWorker, создать возможность сохранения фракталов в виде .png.

1. Анализ предметной области и выбор инструментария.

В текущей работе я использовал бесплатно распространяемый пакет разработчика Java Development Kit. Однако, этот пакет не предусматривает интегрированную среду разработки, поэтому для написания непосредственно кода мною была выбрана программа Notepad++, так как она обеспечивает достаточно гибкий интерфейс и предоставляет возможность подсветки синтаксиса языка.

1. Код программы и объяснение функций.
   1. Класс FractalExplorer.

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Container;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseAdapter;

import java.awt.geom.Rectangle2D;

import javax.imageio.ImageIO;

import javax.swing.filechooser.FileNameExtensionFilter;

import java.io.\*;

import javax.swing.\*;

public class FractalExplorer {

private int display\_size;

private JImageDisplay mImage;

private JComboBox<String> choose\_fractal;

private JButton bu\_save;

private JButton bu\_reset;

private FractalGenerator generate;

private Rectangle2D.Double range;

private int need\_rows;

private void enableUI(boolean val) {

choose\_fractal.setEnabled(val);

bu\_save.setEnabled(val);

bu\_reset.setEnabled(val);

}

private class FractalWorker extends SwingWorker<Object, Object> {

private int mY;

private int[] RGBValue;

public FractalWorker(int y) {

mY = y;

}

public Object doInBackground() {

RGBValue = new int[display\_size];

double yCoord = FractalGenerator.getCoord(range.y, range.y + range.height, display\_size, mY);

for (int x = 0; x < display\_size; x++) {

double xCoord = FractalGenerator.getCoord(range.x, range.x + range.width, display\_size, x);

int n\_iters;

int colors = 0;

float hue;

n\_iters = generate.numIterations(xCoord, yCoord);

if (n\_iters >= 0) {

hue = 0.7f + n\_iters / 200f;

colors = Color.HSBtoRGB(hue, 1f, 1f);

}

RGBValue[x] = colors;

}

return null;

}

public void done() {

for (int x = 0; x < display\_size; x++) {

mImage.drawPixel(x, mY, RGBValue[x]);

}

mImage.repaint(0, 0, mY, display\_size, 1);

if (need\_rows-- < 1) {

enableUI(true);

}

}

}

private class FractalHandler implements ActionListener {

public void actionPerformed(ActionEvent e) {

String cmd = e.getActionCommand();

if (e.getSource() == choose\_fractal) {

String selectedItem = choose\_fractal.getSelectedItem().toString();

if (selectedItem.equals("Mandelbrot")) {

generate = new Mandelbrot();

}

else if (selectedItem.equals("Tricorn")) {

generate = new Tricorn();

}

else if (selectedItem.equals("BurningShip")) {

generate = new BurningShip();

}

else {

JOptionPane.showMessageDialog(null, "Error: Couldn't recognize choice");

return;

}

range = new Rectangle2D.Double();

generate.getInitialRange(range);

drawFractal();

}

else if (cmd.equals("reset")) {

range = new Rectangle2D.Double();

generate.getInitialRange(range);

drawFractal();

}

else if (cmd.equals("save")) {

JFileChooser chooser = new JFileChooser();

FileNameExtensionFilter filter = new FileNameExtensionFilter("png", "png");

chooser.setFileFilter(filter);

chooser.setAcceptAllFileFilterUsed(false);

if (chooser.showSaveDialog(null) == JFileChooser.APPROVE\_OPTION) {

try {

File f = chooser.getSelectedFile();

String filePath = f.getPath();

if (!filePath.toLowerCase().endsWith(".png")) {

f = new File(filePath + ".png");

}

ImageIO.write(mImage.getImage(), "png", f);

}

catch (IOException exc) {

}

}

}

}

}

private class MouseHandler extends MouseAdapter {

public void mouseClicked(MouseEvent e) {

if (need\_rows > 0) {

return;

}

double xCoord = FractalGenerator.getCoord(range.x, range.x + range.width, display\_size, e.getX());

double yCoord = FractalGenerator.getCoord(range.y, range.y + range.height, display\_size, e.getY());

generate.recenterAndZoomRange(range, xCoord, yCoord, 0.5);

drawFractal();

}

}

public FractalExplorer(int size) {

display\_size = size;

generate = new Mandelbrot();

range = new Rectangle2D.Double();

generate.getInitialRange(range);

}

public void createAndShowGUI() {

JFrame frame = new JFrame("Лабораторная работа №4");

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

frame.getContentPane().setLayout( new BorderLayout());

FractalHandler handler = new FractalHandler();

JPanel fractalPanel = new JPanel();

JLabel label = new JLabel("Фрактал: ");

fractalPanel.add(label);

choose\_fractal = new JComboBox<String>();

choose\_fractal.addItem("Mandelbrot");

choose\_fractal.addItem("Tricorn");

choose\_fractal.addItem("BurningShip");

choose\_fractal.addActionListener(handler);

fractalPanel.add(choose\_fractal);

frame.getContentPane().add(fractalPanel, BorderLayout.NORTH);

mImage = new JImageDisplay(display\_size, display\_size);

frame.getContentPane().add(mImage, BorderLayout.CENTER);

JPanel buttonsPanel = new JPanel();

bu\_save = new JButton("Save Image");

bu\_save.setActionCommand("save");

bu\_save.addActionListener(handler);

buttonsPanel.add(bu\_save);

bu\_reset = new JButton("Reset Display");

bu\_reset.setActionCommand("reset");

bu\_reset.addActionListener(handler);

buttonsPanel.add(bu\_reset);

frame.getContentPane().add(buttonsPanel, BorderLayout.SOUTH);

frame.getContentPane().addMouseListener(new MouseHandler());

frame.pack();

frame.setVisible(true);

frame.setResizable(false);

}

public void drawFractal() {

enableUI(false);

for (int y = 0; y < display\_size; y++) {

FractalWorker worker = new FractalWorker(y);

worker.execute();

}

mImage.repaint();

}

public static void main(String[] args) {

FractalExplorer explorer = new FractalExplorer(800);

explorer.createAndShowGUI();

explorer.drawFractal();

}

}

FractalWorker() Реализация SwingWorker, которая позволяет высчитывать нужную информацию в фоновом режиме, с помощью функции doInBackgroung().

drawFractal() Создает FractalWorker, который высчитывает сочетание RGB цветов и далее отрисовывает изображение.

doInBackground() Смысл этого метода заключается в том, что он выполняет фоновою операцию, метод вызывается в фоновом режиме и вычисляет все нужное для отрисовки фрактала.

createAndShowGUI() добавлена кнопка save и выпадающий список, с присвоением им команд.  
FractalHandler() добавлены реализация команд новых кнопок. При выборе нового фрактала картинка перерисовывается   
на новый выбранный фрактал. При сохранении, вызывается соответствующее окно сохранения с фильтром png, после выбора места, файл созраняется в формате .png

* 1. Класс BurningShip.

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 count = 0;

double real = 0;

double image = 0;

double z2 = 0;

while (count < MAX\_ITERATIONS && z2 < 4) {

count++;

double nextRe = Math.pow(real, 2) - Math.pow(image, 2) + x;

double nextIm = Math.abs(2 \* real \* image) + y;

z2 = Math.pow(nextRe,2) + Math.pow(nextIm, 2);

real = nextRe;

image = nextIm;

}

return count < MAX\_ITERATIONS ? count : -1;

}

}

getInitialRange() позволяет определять какая часть плоскости является самой перспективной для фрактала.

numIterations() Высчитывает число итераций для данного фрактала: на протяжении 2000 итераций или пока z1 < 4, считает переменную count, которая и является числом нужных итераций.

Различие у кода фракталов только в их особенности вычисления.

* 1. Класс Tricorn.

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 count = 0;

double real = 0;

double image = 0;

double z2 = 0;

while (count < MAX\_ITERATIONS && z2 < 4) {

count++;

double nextreal = Math.pow(real, 2) - Math.pow(image, 2) + x;

double nextimage = -2 \* real \* image + y;

z2 = Math.pow(nextreal, 2) + Math.pow(nextimage, 2);

real = nextreal;

image = nextimage;

}

return count < MAX\_ITERATIONS ? count : -1;

}

}

getInitialRange() позволяет определять какая часть плоскости является самой перспективной для фрактала.

numIterations() Высчитывает число итераций для данного фрактала: на протяжении 2000 итераций или пока z1 < 4, считает переменную count, которая и является числом нужных итераций.

Различие у кода фракталов только в их особенности вычисления.

1. Заключение.

Проделав всю работу, мы познакомились с java GUI, научились создавать кнопки, выпадающие списки, вставлять картинки, обрабатывать запросы и сохранять файлы. Научились создавать фракталы.