Лабораторная работа №6

по дисциплине

“Объектно-ориентированное программирование”

Тема:

«Многопоточный генератор фракталов»

Выполнила студентка

группы БФИ1901

Киселева Анна

Москва 2020

**Цель:**

Реализовать возможность рисования фрактала с несколькими фоновыми потоками.

**Код программы:**

1. **Файл FractalExplorer.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.FileFilter;

import javax.swing.filechooser.FileNameExtensionFilter;

import java.awt.geom.Rectangle2D;

public class FractalExplorer {

private int sizeDisp;

private JImageDisplay image;

private FractalGenerator FGen;

private Rectangle2D.Double range;

private JComboBox box;

private int rowsRemaning;

JButton resetButton;

JButton saveButton;

private FractalExplorer (int sizeDisp){

this.sizeDisp=sizeDisp;

this.FGen = new Mandelbrot();

this.range = new Rectangle2D.Double(0, 0, 0, 0);

FGen.getInitialRange(this.range);

}

public void createAndShowGUI(){

JFrame frame = new JFrame("Fractal Explorer");

JPanel pan1 = new JPanel();

JPanel pan2 = new JPanel();

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

image = new JImageDisplay(sizeDisp, sizeDisp);

image.addMouseListener(new MouseListener());

box = new JComboBox();

box.addItem(new Mandelbrot());

box.addItem(new Tricorn());

box.addItem(new BurningShip());

box.addActionListener(new ActionHandler());

resetButton = new JButton("Reset");

resetButton.setActionCommand("Reset");

resetButton.addActionListener(new ActionHandler());

saveButton = new JButton("Save Image");

saveButton.setActionCommand("Save");

saveButton.addActionListener(new ActionHandler());

pan1.add(lbl, java.awt.BorderLayout.CENTER);

pan1.add(box, java.awt.BorderLayout.CENTER);

pan2.add(resetButton, java.awt.BorderLayout.CENTER);

pan2.add(saveButton, java.awt.BorderLayout.CENTER);

frame.setLayout(new java.awt.BorderLayout());

frame.add(image, java.awt.BorderLayout.CENTER);

frame.add(pan1, BorderLayout.NORTH);

frame.add(pan2, BorderLayout.SOUTH);

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

frame.pack();

frame.setVisible(true);

frame.setResizable(false);

}

public class ActionHandler implements ActionListener {

public void actionPerformed(ActionEvent e) {

if (e.getActionCommand().equals("Reset")) {

FGen.getInitialRange(range);

drawFractal();

} else if (e.getActionCommand().equals("Save")) {

JFileChooser chooser = new JFileChooser();

FileFilter filter = new FileNameExtensionFilter("PNG Images", "png");

chooser.setFileFilter(filter);

chooser.setAcceptAllFileFilterUsed(false);

int t = chooser.showSaveDialog(image);

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

try {

ImageIO.write(image.getBufferedImage(), "png", chooser.getSelectedFile());

} catch (NullPointerException | IOException e1) {

JOptionPane.showMessageDialog(image, e1.getMessage(), "Cannot Save Image", JOptionPane.ERROR\_MESSAGE);

}

}

} else {

FGen = (FractalGenerator) box.getSelectedItem();

range = new Rectangle2D.Double(0, 0, 0, 0);

FGen.getInitialRange(range);

drawFractal();

}

}

}

public class MouseListener extends MouseAdapter {

@Override

public void mouseClicked(MouseEvent e) {

if (rowsRemaning==0) {

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

double yCoord = FractalGenerator.getCoord(range.y, range.y + range.width, sizeDisp, e.getY());

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

drawFractal();

}

}

}

private void drawFractal(){

enableUI(false);

rowsRemaning = sizeDisp;

for (int i = 0; i < sizeDisp; i++)

{

FractalWorker rowDrawer = new FractalWorker(i);

rowDrawer.execute();

}

}

private class FractalWorker extends SwingWorker<Object, Object>{

private int yCoord;

private int[] rgb;

public FractalWorker(int yCoord)

{

this.yCoord = yCoord;

}

public Object doInBackground()

{

rgb = new int[sizeDisp];

for (int i = 0; i < sizeDisp; i++) {

int count = FGen.numIterations(FractalGenerator.getCoord(range.x, range.x + range.width, sizeDisp, i), FractalGenerator.getCoord(range.y, range.y + range.width, sizeDisp, yCoord));

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;

}

public void done()

{

for (int i = 0; i < sizeDisp; i++)

image.drawPixel(i,yCoord,rgb[i]);

image.repaint(0, 0, yCoord, sizeDisp, 1);

rowsRemaning--;

if (rowsRemaning==0)

enableUI(true);

}

}

public void enableUI(boolean var)

{

saveButton.setEnabled(var);

resetButton.setEnabled(var);

box.setEnabled(var);

}

public static void main(String[] args) {

FractalExplorer FExp = new FractalExplorer(600);

FExp.createAndShowGUI();

FExp.drawFractal();

}

}

1. **Файл BurningShip.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.height = 4;

range.width = 4;

}

public int numIterations(double x, double y) {

double Re = x;

double Im = y;

int counter = 0;

while ((counter < MAX\_ITERATIONS)) {

counter++;

double Re2 = Re \* Re - Im \* Im + x;

double Im2 = Math.abs(2 \* Re \* Im) + y;

Re = Re2;

Im = Im2;

if ((Re \* Re + Im \* Im) > 4)

break;

}

if (counter == MAX\_ITERATIONS)

return -1;

return counter;

}

public String toString()

{

return "Burning Ship";

}

}

1. **Файл FractalGenerator.java:**

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);

}

public abstract void getInitialRange(Rectangle2D.Double range);

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;

}

public abstract int numIterations(double x, double y);

}

1. **Файл JImageDisplay.java:**

import javax.swing.JComponent;

import java.awt.\*;

import java.awt.image.BufferedImage;

public class JImageDisplay extends JComponent {

private BufferedImage image;

public JImageDisplay(int width,int height) {

image = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB);

Dimension dim = new Dimension(width, height);

super.setPreferredSize(dim);

}

@Override

public void paintComponent(Graphics g){

g.drawImage(image,0,0, image.getWidth(), image.getHeight(), null);

}

public void clearImage (){

for(int i =0; i<image.getWidth(); i++)

for(int j =0; j<image.getHeight(); j++)

image.setRGB(i,j,0);

}

public void drawPixel(int x, int y, int rgbColor){

image.setRGB(x,y,rgbColor);

}

public BufferedImage getBufferedImage() {

return image;

}

}

1. **Файл Mandelbrot.java:**

import java.awt.geom.Rectangle2D;

public class Mandelbrot extends FractalGenerator{

public static final int MAX\_ITERATIONS = 2000;

public void getInitialRange (Rectangle2D.Double range){

range.x = -2;

range.y = -1.5;

range.height = 3;

range.width = 3;

}

public int numIterations(double x, double y) {

double Re = x;

double Im = y;

int counter = 0;

while ((counter < MAX\_ITERATIONS)) {

counter++;

double Re2 = Re \* Re - Im \* Im + x;

double Im2 = 2 \* Re \* Im + y;

Re = Re2;

Im = Im2;

if ((Re \* Re + Im \* Im) > 4)

break;

}

if (counter == MAX\_ITERATIONS)

return -1;

return counter;

}

public String toString()

{

return "Mandelbrot";

}

}

1. **Файл Tricorn.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.height = 4;

range.width = 4;

}

public int numIterations(double x, double y) {

double Re = x;

double Im = y;

int counter = 0;

while ((counter < MAX\_ITERATIONS)) {

counter++;

double Re2 = Re \* Re - Im \* Im + x;

double Im2 = (-2) \* Re \* Im + y;

Re = Re2;

Im = Im2;

if ((Re \* Re + Im \* Im) > 4)

break;

}

if (counter == MAX\_ITERATIONS)

return -1;

return counter;

}

public String toString()

{

return "Tricorn";

}

}

**Пример работы программы:**

