Министерство цифрового развития, связи и

Массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра «Математической кибернетики и информационных технологий»

Отчет по лабораторной работе №4

по дисциплине «Кроссплатформенные технологии программирования»

Выполнил: студент группы БСТ2001

Багдасарян Артём

Москва 2022

Оглавление

[**1.** **Цель и задание** 3](#_Toc97797644)

[**2.** **Ход выполнения лабораторной работы** 3](#_Toc97797645)

[**2.1** **Код программы** 3](#_Toc97797646)

[**2.2** **Результат работы программы** 4](#_Toc97797647)

[**3.** **Вывод** 5](#_Toc97797648)

[**Список использованных источников** 6](#_Toc97797649)

1. **Цель и задание**

В данной лабораторной работе мы изучим алгоритм вычисления фрактала Мандельброта, а также создадим программу, которая создает рисунки с помощью данного алгоритма.

1. **Ход выполнения лабораторной работы** 
   1. **Код программы**

Ниже представлен код FractalGenerator.java

package com.company;  
  
import java.awt.geom.Rectangle2D;  
  
  
*/\*\*  
 \* This class provides the common interface and operations for fractal  
 \* generators that can be viewed in the Fractal Explorer.  
 \*/*public abstract class FractalGenerator {  
  
 */\*\*  
 \* This static helper function takes an integer coordinate and converts it  
 \* into a double-precision value corresponding to a specific range. It is  
 \* used to convert pixel coordinates into double-precision values for  
 \* computing fractals, etc.  
 \*  
 \** ***@param*** *rangeMin the minimum value of the floating-point range  
 \** ***@param*** *rangeMax the maximum value of the floating-point range  
 \*  
 \** ***@param*** *size the size of the dimension that the pixel coordinate is from.  
 \* For example, this might be the image width, or the image height.  
 \*  
 \** ***@param*** *coord the coordinate to compute the double-precision value for.  
 \* The coordinate should fall in the range [0, size].  
 \*/* 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);  
 }  
  
  
 */\*\*  
 \* Sets the specified rectangle to contain the initial range suitable for  
 \* the fractal being generated.  
 \*/* public abstract void getInitialRange(Rectangle2D.Double range);  
  
  
 */\*\*  
 \* Updates the current range to be centered at the specified coordinates,  
 \* and to be zoomed in or out by the specified scaling factor.  
 \*/* 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;  
 }  
  
  
 */\*\*  
 \* Given a coordinate <em>x</em> + <em>iy</em> in the complex plane,  
 \* computes and returns the number of iterations before the fractal  
 \* function escapes the bounding area for that point. A point that  
 \* doesn't escape before the iteration limit is reached is indicated  
 \* with a result of -1.  
 \*/* public abstract int numIterations(double x, double y);  
}

Ниже представлен код FractalExplorer.java

package com.company;  
  
import java.awt.\*;  
import javax.swing.\*;  
import java.awt.geom.Rectangle2D;  
import java.awt.event.\*;  
  
public class FractalExplorer {  
  
 private int displaySize;  
 private JImageDisplay display;  
 private FractalGenerator fractal;  
 private Rectangle2D.Double range;  
 public FractalExplorer(int size) {  
 //Сохраняет размер дисплея  
 displaySize = size;  
 //Инициализирует фрактальный генератор и объекты диапазона  
 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*);  
 JButton resetButton = new JButton("Reset Display");  
 ResetHandler handler = new ResetHandler();  
 resetButton.addActionListener(handler);  
 myframe.add(resetButton, BorderLayout.*SOUTH*);  
 MouseHandler click = new MouseHandler();  
 display.addMouseListener(click);  
 myframe.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 myframe.setTitle("this is my fractal");  
 Toolkit toolkit = Toolkit.*getDefaultToolkit*();  
 Dimension dimension = toolkit.getScreenSize();  
 myframe.setBounds(dimension.width/2 - 300,dimension.height/2 - 300, 600, 600);  
 myframe.pack();  
 myframe.setVisible(true);  
 myframe.setResizable(false);  
 }  
  
 private void drawFractal()  
 {  
  
 for (int x=0; x<displaySize; x++){  
 for (int y=0; y<displaySize; y++){  
  
 double xCoord = fractal.*getCoord*(range.x, range.x + range.width, displaySize, x);  
 double yCoord = fractal.*getCoord*(range.y, range.y + range.height, displaySize, y);  
  
 int iteration = fractal.numIterations(xCoord, yCoord);  
  
 if (iteration == -1){  
 display.drawPixel(x, y, 0);  
 } else {  
  
 float hue = 0.7f + (float) iteration / 200f;  
 int rgbColor = Color.*HSBtoRGB*(hue, 1f, 1f);  
  
 display.drawPixel(x, y, rgbColor);  
 }  
 }  
 }  
  
 display.repaint();  
 }  
  
 private class ResetHandler implements ActionListener  
 {  
  
 public void actionPerformed(ActionEvent e)  
 {  
 fractal.getInitialRange(range);  
 drawFractal();  
 }  
 }  
  
 private class MouseHandler extends MouseAdapter  
 {  
  
 @Override  
 public void mouseClicked(MouseEvent e)  
 {  
  
 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();  
 }  
 }  
  
 public static void main(String[] args)  
 {  
 FractalExplorer displayExplorer = new FractalExplorer(600);  
 displayExplorer.createAndShowGUI();  
 displayExplorer.drawFractal();  
 }  
}

Ниже представлен код JImageDisplay.java

package com.company;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.image.BufferedImage;  
  
public class JImageDisplay extends JComponent {  
  
 private BufferedImage displayImage;  
  
 public JImageDisplay(int height, int width){  
 displayImage = new BufferedImage(width, height, BufferedImage.*TYPE\_INT\_RGB*);  
 Dimension imageDimension = new Dimension(width, height);  
 super.setPreferredSize(imageDimension);  
 }  
  
 public void paintComponent(Graphics g){  
 super.paintComponent(g);  
 g.drawImage (displayImage, 0, 0, displayImage.getWidth(), displayImage.getHeight(), null);  
 }  
  
 public void clearImage(){  
 int[] blankArray = new int[getWidth() \* getHeight()];  
 displayImage.setRGB(0, 0, getWidth(), getHeight(), blankArray, 0, 1);  
 }  
  
 public void drawPixel(int x, int y, int rgbColor)  
 {  
 displayImage.setRGB(x, y, rgbColor);  
 }  
}

Ниже представлен код Mandelbort.java

package com.company;  
  
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.width = 3;  
 range.height = 3;  
 }  
  
 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;  
 }  
}

* 1. **Результат работы программы**

Результат работы программы показан на рисунке 1

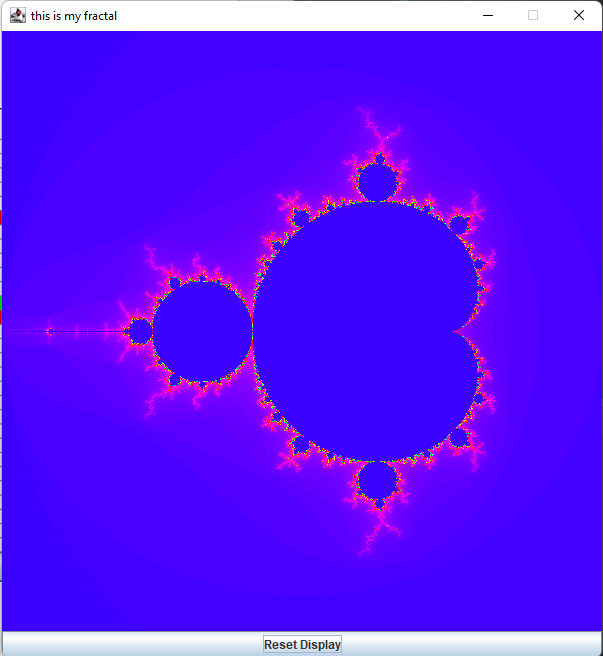


Рисунок 1 – Результат работы программы

* 1. **Программа на Git hub**

Ссылка на репозиторий на Git hub: https://github.com/Artem2406/KTP

1. **Вывод**

В данной лабораторной работе мы изучили алгоритм вычисления фрактала Мандельброта, а также создали программу, которая создает рисунки с помощью данного алгоритма.

# **Список использованных источников**

1. Камаев В.А., Костерин В.В. Технологии программирования. М.: Высшая школа, 2006.

2. Жоголев Е.А.Технология программирования. – М.: Научный мир, 2004.