**Федеральное агентство связи**

**Ордена трудового Красного Знамени федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра Математическая кибернетика и информационные технологии

Лабораторная работа № 4

Выполнил: студент группы БФИ2002

Ковачев В.Е.

Проверила:

Москва, 2021

**Задача:**

Создайте класс FractalExplorer, который позволит вам исследовать различные области фрактала, путем его создания, отображения через графический интерфейс Swing и обработки событий, вызванных взаимодействием приложения с пользователем.

• Класс FractalExplorer должен отслеживать несколько важных полей для состояния программы:

1) Целое число «размер экрана», которое является шириной и высотой отображения в пикселях. (Отображение фрактала будет квадратным.)

2) Ссылка JImageDisplay, для обновления отображения в разных методах в процессе вычисления фрактала.

3) Объект FractalGenerator. Будет использоваться ссылка на базовый класс для отображения других видов фракталов в будущем.

4) Объект Rectangle2D.Double, указывающий диапазона комплексной плоскости, которая выводится на экран.

Все вышеприведенные поля будут иметь тип доступа private.

• У класса должен быть конструктор, который принимает значение размера отображения в качестве аргумента, затем сохраняет это значение в соответствующем поле, а также инициализирует объекты диапазона и фрактального генератора. Данный конструктор не должен устанавливать какиелибо компоненты Swing; они будут установлены в следующем методе.

• Создайте метод createAndShowGUI (), который инициализирует графический интерфейс Swing: JFrame, содержащий объект JimageDisplay, и кнопку для сброса отображения. Используйте java.awt.BorderLayout для содержимого окна; добавьте объект отображения изображения в позицию BorderLayout.CENTER и кнопку в позицию BorderLayout.SOUTH.

Вам необходимо дать окну подходящий заголовок и обеспечить операцию закрытия окна по умолчанию (см. метод JFrame.setDefaultCloseOperation ()).

После того, как компоненты пользовательского интерфейса инициализированы и размещены, добавьте следующую последовательность операций:

frame.pack ();

frame.setVisible (true);

frame.setResizable (false);

Данные операции правильно разметят содержимое окна, сделают его видимым и затем запретят изменение размеров окна.

• Реализуйте вспомогательный метод с типом доступа private для вывода на экран фрактала, можете дать ему имя drawFractal (). Этот метод должен циклически проходить через каждый пиксель в отображении (т.е. значения x и y будут меняться от 0 до размера отображения), и сделайте следующее:

¬ Вычислите количество итераций для соответствующих координат в области отображения фрактала. Вы можете определить координаты с плавающей точкой для определенного набора координат пикселей, используя вспомогательный метод FractalGenerator.getCoord (); например, чтобы получить координату x, соответствующую координате пикселя X, сделайте следующее:

double xCoord = FractalGenerator.getCoord (range.x, range.x + range.width, displaySize, x);

¬ Если число итераций равно -1 (т.е. точка не выходит за границы, установите пиксель в черный цвет. Иначе выберите значение цвета, основанное на количестве итераций. Можно также для этого использовать цветовое пространство HSV: поскольку значение цвета варьируется от 0 до 1, получается плавная последовательность цветов от красного к желтому, зеленому, синему, фиолетовому и затем обратно к красному! Для этого вы можете использовать следующий фрагмент:

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

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

Если вы придумали другой способ отображения пикселей в зависимости от количества итераций, попробуйте реализовать его!

¬ Отображение необходимо обновлять в соответствии с цветом для каждого пикселя.

¬ После того, как вы закончили отрисовывать все пиксели, вам необходимо обновить JimageDisplay в соответствии с текущим изображением. Для этого вызовите функцию repaint() для компонента. В случае, если вы не воспользуетесь данным методом, изображение на экране не будет обновляться!

• Создайте внутренний класс для обработки событий java.awt.event.ActionListener от кнопки сброса. Обработчик должен сбросить диапазон к начальному, определенному генератором, а затем перерисовать фрактал. После того, как вы создали этот класс, обновите метод createAndShowGUI ().

• Создайте другой внутренний класс для обработки событий java.awt.event.MouseListener с дисплея. Вам необходимо обработать события от мыши, поэтому вы должны унаследовать этот внутренний класс от класса MouseAdapterAWT. При получении события о щелчке мышью, класс должен отобразить пиксельные кооринаты щелчка в область фрактала, а затем вызвать метод генератора recenterAndZoomRange() с координатами, по которым щелкнули, и масштабом 0.5. Таким образом, нажимая на какое-либо место на фрактальном отображении, вы увеличиваете его!

Далее обновите метод createAndShowGUI (), чтобы зарегистрировать экземпляр этого обработчика в компоненте фрактального отображения.

• В заключении, вам необходимо создать статический метод main() для FractalExplorer так, чтобы можно было его запустить. В main необходимо будет сделать:

¬ Инициализировать новый экземпляр класса FractalExplorer с размером отображения 800.

¬ Вызовите метод createAndShowGUI () класса FractalExplorer.

¬ Вызовите метод drawFractal() класса FractalExplorer для отображения начального представления.

**Ход работы:**

**Для выполнения первого задания был написан класс FractalExplorer.**

import java.awt.\*;

import javax.swing.\*;

import javax.swing.border.Border;

import java.awt.event.\*;

import java.awt.geom.Rectangle2D;

public class FractalExplorer {

private int displaySize;

private JImageDisplay imageDisplay;

private FractalGenerator fractalGenerator;

private Rectangle2D.Double range;

private FractalExplorer (int displaySize) {

this.displaySize = displaySize;

this.fractalGenerator = new Mandelbrot();

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

fractalGenerator.getInitialRange(this.range);

}

public static void main(String[] args) {

FractalExplorer fractalExplorer = new FractalExplorer(600);

fractalExplorer.setGUI();

fractalExplorer.drawFractal();

}

**// задание интерфейса**

public void setGUI() {

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

JButton button = new JButton("Reset");

imageDisplay = new JImageDisplay(displaySize, displaySize);

imageDisplay.addMouseListener(new MouseListener());

button.addActionListener(new ActionHandler());

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

frame.add(imageDisplay, BorderLayout.CENTER);

frame.add(button, BorderLayout.SOUTH);

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

frame.pack();

frame.setVisible(true);

frame.setResizable(false);

}

**// отрисовка фрактала в JImageDisplay**

private void drawFractal() {

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

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

int counter = fractalGenerator.numIterations(FractalGenerator.getCoord(range.x, range.x + range.width, displaySize, x),

fractalGenerator.getCoord(range.y, range.y + range.width, displaySize, y));

if (counter == -1) {

imageDisplay.drawPixel(x, y, 0);

}

else {

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

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

imageDisplay.drawPixel(x, y, rgbColor);

}

}

}

imageDisplay.repaint();

}

public class ActionHandler implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

fractalGenerator.getInitialRange(range);

drawFractal();

}

}

public class MouseListener extends MouseAdapter {

@Override

public void mouseClicked(MouseEvent e) {

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

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

fractalGenerator.recenterAndZoomRange(range, x, y, 0.5);

drawFractal();

}

}

}

**Для отрисовки изображения содержащего фрактал был реализован класс JImageDisplay:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.image.BufferedImage;

public class JImageDisplay extends JComponent {

private BufferedImage image;

**// объявление изображения и его параметров**

public JImageDisplay(int width, int height) {

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

Dimension dimension = new Dimension(width, height);

super.setPreferredSize(dimension);

}

**// метод отрисовки изображения**

public void paintComponent(Graphics graphics) {

super.paintComponent(graphics);

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

}

**// очистка изображения (устанавливает все пиксели**

**// в черный цвет)**

public void clearImage() {

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

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

drawPixel(i, j, 0);

}

}

}

**// задание цвета конкретному пикселю**

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

image.setRGB(x, y, rgbColor);

}

public BufferedImage getImage() {

return image;

}

}

Также для расчётов фрактала был создан класс Mandelbrot:

import java.awt.geom.Rectangle2D;

public class Mandelbrot extends FractalGenerator {

public static final int MAX\_ITERATIONS = 2000;

@Override

public void getInitialRange(Rectangle2D.Double range) {

range.x = -2;

range.y = -1.5;

range.height = 3;

range.width = 3;

}

@Override

public int numIterations(double x, double y) {

double r = x;

double i = y;

int counter = 0;

while (counter < MAX\_ITERATIONS) {

counter++;

double k = r \* r - i\*i+x;

double m = 2 \* r \* i + y;

r = k;

i = m;

if (r\*r+i\*i > 4)

break;

}

if (counter == MAX\_ITERATIONS)

return -1;

return counter;

}

}

**Результат работы программы:**

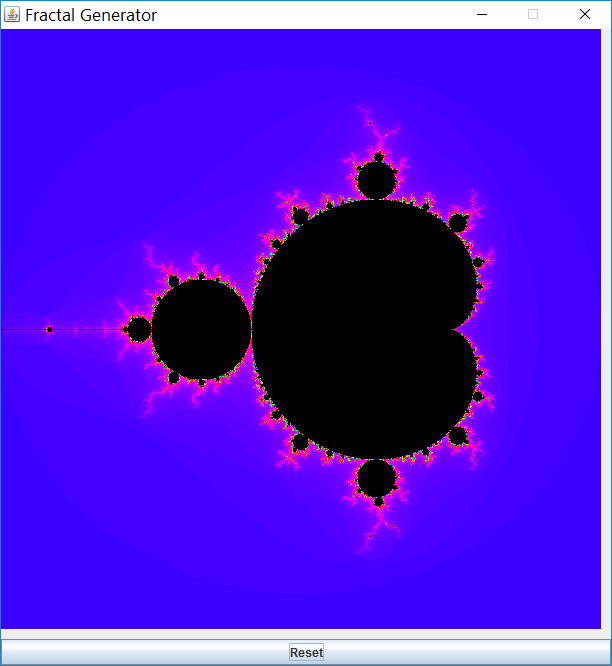


Рисунок №1- Результат работы программы

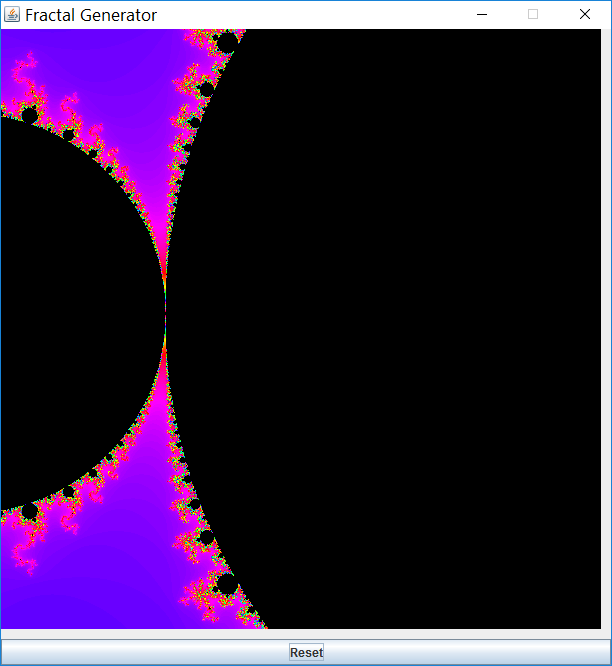


Рисунок №1- Приближение фрактала

**Вывод**

В результат е выполнения заданий было создано приложение на языке java, для отрисовки фракталов, а также было реализовано приближение фрактала.

**Список использованной литературы**

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

2006.

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