**Федеральное агентство связи**

**Ордена Трудового Красного Знамени федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

**“Московский технический университет связи и информатики”**

**Кафедра МКиИТ**

**ЛАБОРАТОРНАЯ РАБОТА №5**

**Выбор и сохранение фракталов**

Выполнил: студент гр. БСУ1901 Панов Д. А.

Проверил: ст.преп. Мосева М.С.

Москва 2021 г.

ОГЛАВЛЕНИЕ

I ХОД РАБОТЫ

Выполнение работы3-9

I. ЗАДАНИЕ

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);  
 }  
  
  
 // позволяет генератору фракталов определить наиболее «интересную»  
 // область комплексной плоскости для конкретного фрактала  
  
  
 public abstract void getInitialRange(Rectangle2D.Double range);  
  
  
 // Обновляет текущий диапазон, чтобы он был центрирован по указанным координатам,  
 // и для увеличения или уменьшения с указанным коэффициентом масштабирования.  
  
 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;  
 }  
  
  
 // реализует итеративную функцию для фрактала Мандельброта  
 //Учитывая координату <em> x </em> + <em>iy</em> на комплексной плоскости,  
 //вычисляет и возвращает количество итераций перед фракталом функция выходит за пределы ограничивающей  
 // области для этой точки. Дело в том, что не исчезает, пока не будет указан предел итераций с результатом -1.  
  
 public abstract int numIterations(double x, double y);  
}

Рисунок 1 – Код файла FractalGenerator.java

// Этот класс позволяет исследовать различные части фрактала с помощью  
// создания и отображения графического интерфейса Swing и обработки событий, вызванных различными  
// взаимодействиями с пользователем.  
  
import java.awt.BorderLayout;  
import java.awt.event.ActionListener;  
import java.awt.event.MouseListener;  
import javax.swing.JFrame;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
import java.awt.\*;  
import java.awt.geom.Rectangle2D;  
import java.awt.event.ItemEvent;  
import java.awt.event.ActionEvent;  
import java.awt.event.ItemListener;  
import java.awt.event.ActionListener;  
import java.awt.image.BufferedImage;  
  
import javax.swing.\*;  
import javax.swing.event.ChangeEvent;  
import javax.swing.JFileChooser;  
import javax.swing.filechooser.FileNameExtensionFilter;  
import javax.swing.filechooser.FileFilter;  
import javax.swing.event.ChangeListener;  
  
public class FractalExplorer  
{  
 // Целочисленный размер отображения - это ширина и высота отображения в пикселях.  
private int displaySize;  
  
 // Ссылка JImageDisplay для обновления отображения с помощью различных методов по мере вычисления фрактала.  
  
private JImageDisplay display;  
  
 // Объект FractalGenerator, использующий ссылку базового класса для отображения других типов фракталов в будущем.  
  
private FractalGenerator fractal;  
  
 // указывает диапазон комплексной плоскости, которая выводится на экран.  
private Rectangle2D.Double range;  
  
 // конструктор, который принимает значение  
 //размера отображения в качестве аргумента, затем сохраняет это значение в  
 //соответствующем поле, а также инициализирует объекты диапазона и  
 //фрактального генератора  
  
  
public FractalExplorer(int size)  
 {  
displaySize = size;  
fractal = new Mandelbrot();  
range = new Rectangle2D.Double();  
fractal.getInitialRange(range);  
display = new JImageDisplay(displaySize, displaySize);  
  
 }  
  
 // инициализирует графический интерфейс Swing  
  
public void createAndShowGUI()  
 {  
display.setLayout(new BorderLayout());  
JFrame myframe = new JFrame("Fractal Explorer");  
myframe.add(display, BorderLayout.CENTER);  
  
 // Параметры кнопки  
JButton resetButton = new JButton("Reset Display");  
ButtonHandler handler = new ButtonHandler();  
resetButton.addActionListener(handler);  
MouseHandler click = new MouseHandler();  
display.addMouseListener(click);  
  
 // Операция закрытия окна по умолчанию  
myframe.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
  
JComboBox 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);  
  
JButton 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);  
 // Данные операции правильно разметят содержимое окна, сделают его видимым  
  
myframe.pack();  
myframe.setVisible(true);  
myframe.setResizable(false);  
 }  
  
 // вывод на экран фрактала  
  
private void drawFractal()  
 {  
 // Просмотр каждого пикселя на дисплее  
for (int x=0; x<displaySize; x++){  
for (int y=0; y<displaySize; y++){  
  
// Поиск соответствующих координат xCoord и yCoord в области отображения фрактала.  
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);  
  
// Если число итераций равно -1 (т.е. точка не выходит за границы, установите пиксель в черный цвет  
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);  
}  
  
 }  
 }  
  
// Обновление JimageDisplay в соответствии с текущим изображением.  
 display.repaint();  
 }  
  
 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("PNGImages", "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(), "CannotSaveImage",  
 JOptionPane.ERROR\_MESSAGE);  
 }  
 }  
 else return;  
 }  
 }  
 }  
  
 // класс для обработки событий java.awt.event.MouseListener с дисплея  
  
private class MouseHandler extends MouseAdapter  
 {  
 // При получении события о щелчке мышью, класс должен  
 //отобразить пиксельныекооринаты 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!  
  
  
@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(800);  
displayExplorer.createAndShowGUI();  
displayExplorer.drawFractal();  
 }  
}

Рисунок 2 – Код файла FractalExplorer.java

// Этот класс позволяет нам отображать наши фракталы.  
import javax.swing.JComponent;  
import java.awt.image.BufferedImage;  
import java.awt.Dimension;  
import java.awt.BorderLayout;  
import java.awt.event.ActionListener;  
import java.awt.event.MouseListener;  
import java.awt.\*;  
import java.awt.event.ItemEvent;  
import java.awt.event.ActionEvent;  
import java.awt.event.ItemListener;  
import java.awt.event.ActionListener;  
import java.awt.geom.Rectangle2D;  
import javax.swing.\*;  
import javax.swing.event.ChangeEvent;  
import javax.swing.event.ChangeListener;  
  
  
class JImageDisplay extends JComponent  
{  
 // управляет изображением, содержимое которого можно записать.  
  
  
private BufferedImage displayImage;  
  
public BufferedImage getImage() {  
 return displayImage;  
 }  
  
 // принимает целочисленные значения ширины и высоты, и инициализирует объект BufferedImage новым  
 //изображением с этой шириной и высотой, и типом изображения TYPE\_INT\_RGB.  
 public JImageDisplay (int width, int height)  
 {  
displayImage = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB);  
Dimension imageDimension = new Dimension(width, height);  
super.setPreferredSize(imageDimension);  
  
 }  
  
 // код для отрисовки  
  
 @Override  
public void paintComponent(Graphics g)  
 {  
super.paintComponent(g);  
g.drawImage(displayImage, 0, 0, displayImage.getWidth(), displayImage.getHeight(), null);  
 }  
  
 // устанавливает все пиксели изображения в черный цвет  
  
public void clearImage()  
 {  
int[] blankArray = new int[getWidth() \* getHeight()];  
displayImage.setRGB(0, 0, getWidth(), getHeight(), blankArray, 0, 1);  
 }  
  
 // устанавливает пиксель в определенный цвет  
  
public void drawPixel(int x, int y, int rgbColor)  
 {  
  
 displayImage.setRGB(x, y, rgbColor);  
}  
}

Рисунок 3 – Код файла JImageDisplay.java

import java.awt.geom.Rectangle2D;  
public class Mandelbrot extends FractalGenerator  
{  
  
public static final int MAX\_ITERATIONS = 2000;  
  
 // этот метод  
 //должен установить начальный диапазон в (-2 - 1.5i) - (1 + 1.5i)  
  
  
public void getInitialRange(Rectangle2D.Double range)  
 {  
 range.x = -2;  
 range.y = -1.5;  
 range.width = 3;  
 range.height = 3;  
 }  
  
 // реализует итеративную  
 //функцию для фрактала Мандельброта  
  
  
 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;  
 }  
  
 public String toString() {  
 return "Mandelbrot";  
 }  
  
}

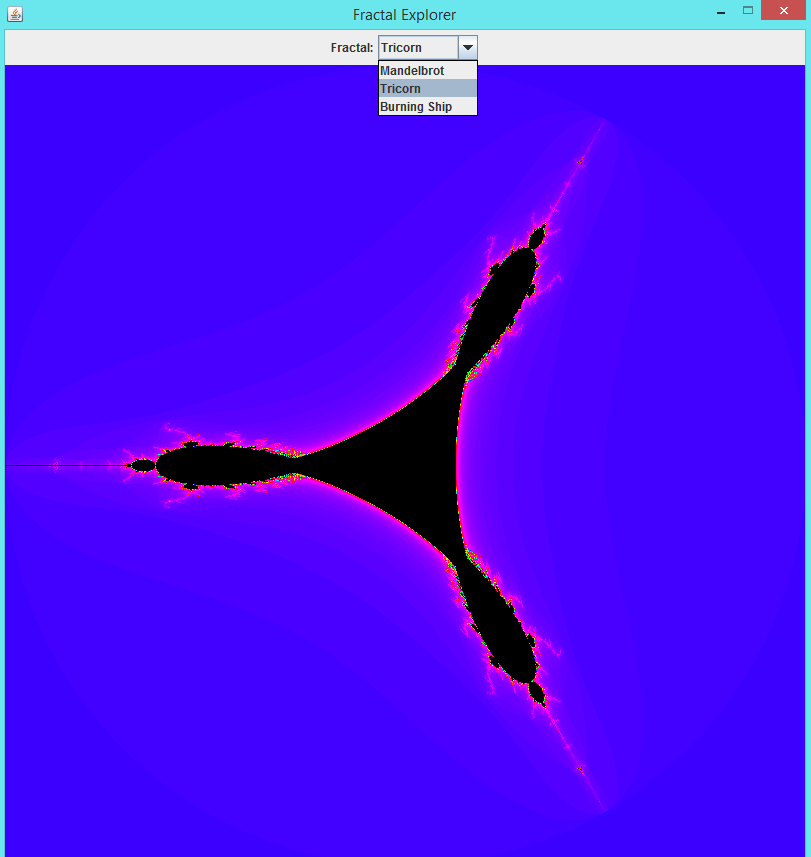
Рисунок 4 – Код файла Mandelbrot.java

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.width= 4;  
range.height= 4;  
}  
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;  
}  
  
public String toString() {  
return "Tricorn";  
}  
  
}

Рисунок 5 – Код файла Tricorn.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.width= 4;  
range.height= 4;  
}  
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 \* Math.abs(zreal)  
 \* Math.abs(zimaginary) + y;  
  
zreal = zrealUpdated;  
zimaginary = zimaginaryUpdated;  
  
iteration += 1;  
}  
  
if (iteration == MAX\_ITERATIONS) {  
return -1;  
}  
  
return iteration;  
}  
public String toString() {  
return "Burning Ship";  
}  
  
}

Рисунок 6 – Код файла BurningShip.java



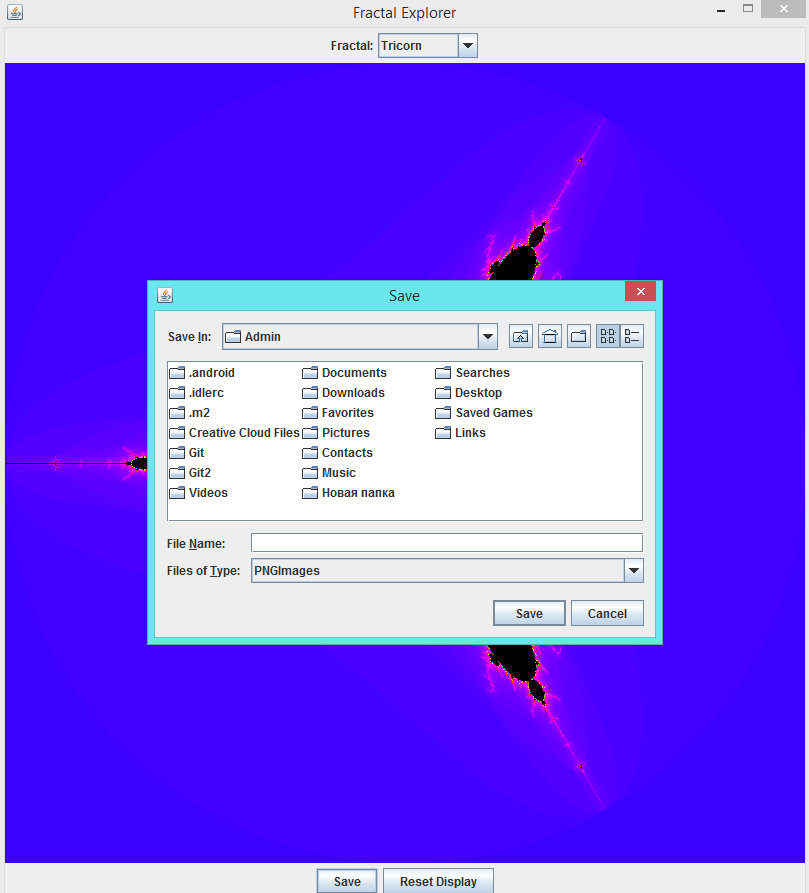


Рисунок 7 – Результат работы программы