Федеральное агентство связи

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 «Объектно ориентированное программирование»

на тему «Рисование фракталов»

Выполнил

Студент группы БФИ1701

Неретин И.Г.

Проверила:

Мосеева М.С.

Москва 2020

1. **Цель работы**

Цель работы – научиться работать с интерфейсами фреймворка Swing и Java API, а также изучить способ построения фрактала Мандельброта.

1. **Задание**

Цель работы определила следующие задачи:

1. Разработать систему графического отображения фракталов, а также интерфейс взаимодействия;
2. Создать функции, вычисляющие фрактал Мандельброта;
3. Инициализировать и запустить систему, отображающую фрактал Мандельброта.
4. **Порядок выполнения работы**

Для вычисления фракталов был создан вспомогательный класс Complex, обладающий всеми необходимыми методами.

Листинг – Complex.java

|  |
| --- |
| public class Complex {  public double x;  public double y;  Complex(){  x = 0;  y = 0;  }  Complex (double x, double y){  this.x = x;  this.y = y;  }  public Complex sum(Complex b){  Complex a = new Complex();  a.x = this.x + b.x;  a.y = this.y + b.y;  return a;  }  public Complex step2(){  double buf = x;  x = x\*x - y\*y;  y = 2\*buf\*y;  return this;  }  public boolean isMoreThan(double a){  return (x\*x + y\*y) > a\*a;  } } |

Был создан класс для управления системой в целом, а также вспомогательные подклассы, наследующие классы слушателей.

Листинг 2 – FractalExplorer.java

|  |
| --- |
| import javax.swing.\*; import javax.swing.plaf.basic.BasicButtonListener; import javax.swing.plaf.basic.BasicOptionPaneUI; import java.awt.\*; import java.awt.color.ICC\_ProfileRGB; import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.awt.event.MouseAdapter; import java.awt.event.MouseEvent; import java.awt.geom.Rectangle2D;  public class FractalExplorer {  private int screenSize;  private JImageDisplay display;  private FractalGenerator fractalGenerator;  private Rectangle2D.Double range;  FractalExplorer(int size){  screenSize = size;  display = new JImageDisplay(screenSize, screenSize);  range = new Rectangle2D.Double();  FractalGenerator.Mandelbrot.*getInitialRange*(range);  }  public void createAndShowGUI(){  //кнопка сброса  Button reset = new Button("Reset");  reset.setSize(screenSize, 50);  reset.addActionListener(new JButtonClick());  reset.setVisible(true);   //панель вывода компонентов интерфейса  BorderLayout layout = new BorderLayout(screenSize, screenSize);  display.clearImage();   //создание и настройка окна  JFrame frame = new JFrame();  frame.add(reset, BorderLayout.*SOUTH*);  frame.add(display, BorderLayout.*CENTER*);  //frame.setLayout(layout);  frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  frame.setPreferredSize(new Dimension(screenSize, screenSize));  frame.pack();  frame.setVisible(true);  frame.setResizable(false);  frame.addMouseListener(new JMouseAdapter());  this.drawFractal();  }   private void drawFractal(){  int i, j;  double xCoord, yCoord;  int iterations;  float hue;  int rgbColor;  for (i = 0; i < display.getWidth(); i++){  for (j = 0; j < display.getHeight(); j++){  xCoord = FractalGenerator.Mandelbrot.*getCoord*(range.x, range.x + range.width, screenSize, i);  yCoord = FractalGenerator.Mandelbrot.*getCoord*(range.y, range.y + range.height, screenSize, j);  iterations = FractalGenerator.Mandelbrot.*numIterations*(xCoord, yCoord);  if (iterations == -1){  rgbColor = 0;  }  else{  hue = 0.7f + (float) iterations / 200f;  rgbColor = Color.*HSBtoRGB*(hue, 1f, 1f);  }  display.drawPixel(i, j, rgbColor);  display.repaint();  }  }   }   private class JMouseAdapter extends java.awt.event.MouseAdapter {  @Override  public void mouseClicked(MouseEvent e) {  super.mouseClicked(e);  display.drawPixel(e.getX(), e.getY(), ICC\_ProfileRGB.*icSigGreenColorantTag*);  double xCoord = FractalGenerator.Mandelbrot.*getCoord*(range.x, range.x + range.width, screenSize, e.getX());;  double yCoord = FractalGenerator.Mandelbrot.*getCoord*(range.y, range.y + range.height, screenSize, e.getY());  FractalGenerator.Mandelbrot.*recenterAndZoomRange*(range, xCoord, yCoord, 0.5);  display.repaint();  drawFractal();  }  }   private class JButtonClick implements ActionListener {  @Override  public void actionPerformed(ActionEvent e) {  FractalGenerator.Mandelbrot.*getInitialRange*(range);  drawFractal();  }  } } |

В Листинге ниже описан подкласс класса «FractalGenerator», выполняющий функции построения фрактала Мандельброта. Класс объявлен статическим для удобства его использования.

Листинг 3 – FractalGenerator.java

|  |
| --- |
| import java.awt.geom.Rectangle2D;  */\*\*  \* Этот класс предоставляет общий интерфейс и операции для генераторов фракталов,   \* которые можно просматривать в Fractal Explorer.  \*/* public abstract class FractalGenerator {   */\*\*  \* Эта статическая вспомогательная функция принимает целочисленную координату   \* и преобразует ее в значение двойной точности, соответствующее определенному диапазону.   \* Он используется для преобразования координат пикселей в значения   \* двойной точности для вычисления фракталов и т. Д.  \*  \** ***@param*** *rangeMin минимальное значение диапазона с плавающей запятой  \** ***@param*** *rangeMax максимальное значение диапазона с плавающей запятой  \*  \** ***@param*** *size размер измерения, из которого берется пиксельная координата.  \* Например, это может быть ширина изображения или высота изображения.  \*  \** ***@param*** *coord координата, чтобы вычислить значение двойной точности  \* Координата должна находиться в диапазоне [0, размер].  \*/* 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);  }    */\*\*  \* Устанавливает указанный прямоугольник, чтобы содержать начальный диапазон, подходящий  \* для генерируемого фрактала.  \*/* public static void getInitialRange(Rectangle2D.Double range) {};  public static int numIterations(double x, double y) { return 0; };   */\*\*  \* Обновляет текущий диапазон с центром в указанных координатах,   \* а также для увеличения или уменьшения с помощью указанного коэффициента масштабирования.  \*/* public static 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;  }   */\*\*  \* Созданный класс для фрактала Мандельброта  \* \*\*/* public static class Mandelbrot extends FractalGenerator {  public static final int *MAX\_ITERATIONS* = 2000;  public static void getInitialRange(Rectangle2D.Double rect){  rect.x = -2;  rect.y = -1.5;  rect.height = 3;  rect.width = 3;  }   //Возвращает количество итераций для точки (x, y), при которых очевидно, что точка не принадлежит  // набору. Возвращает -1, если точка находится во множестве Мандельброта  public static int numIterations(double x, double y){  int iterations = 0;  Complex c = new Complex(x, y);  Complex z = new Complex(0, 0);  while (iterations < *MAX\_ITERATIONS*){  iterations++;  z = z.step2().sum(c);  if (z.isMoreThan(2)){  return iterations;  }  }  return -1;  }  } } |

Листинг 4 – JavaFractalMain.java

|  |
| --- |
| public class JavaFractalMain {  public static void main(String[] args){  FractalExplorer explorer = new FractalExplorer(500);  explorer.createAndShowGUI();  } } |

Листинг 5 – JImageDisplay.java

|  |
| --- |
| import java.awt.\*; import java.awt.image.BufferedImage;  import static java.awt.image.BufferedImage.*TYPE\_INT\_RGB*;  public class JImageDisplay extends javax.swing.JComponent{  private java.awt.image.BufferedImage bufferedImage;  JImageDisplay(int width, int height){  bufferedImage = new BufferedImage(width, height, *TYPE\_INT\_RGB*);  super.setPreferredSize(new Dimension(width, height));  }   @Override  protected void paintComponent(Graphics g) {  super.paintComponent(g);  g.drawImage (bufferedImage, 0, 0, bufferedImage.getWidth(), bufferedImage.getHeight(), null);  }   public void clearImage(){  int[] rgbData = new int[bufferedImage.getWidth()\*bufferedImage.getHeight()];  //создаем массив нулевых значений, чтобы окрасить все полотно в черный цвет  for (int i=0; i < rgbData.length; i++){  rgbData[i] = 0;  }  bufferedImage.setRGB(0, 0, bufferedImage.getWidth(),  bufferedImage.getHeight(), rgbData, 0, bufferedImage.getWidth());  }   public void drawPixel(int x, int y, int rgbColor){  bufferedImage.setRGB(x, y, rgbColor);  } } |

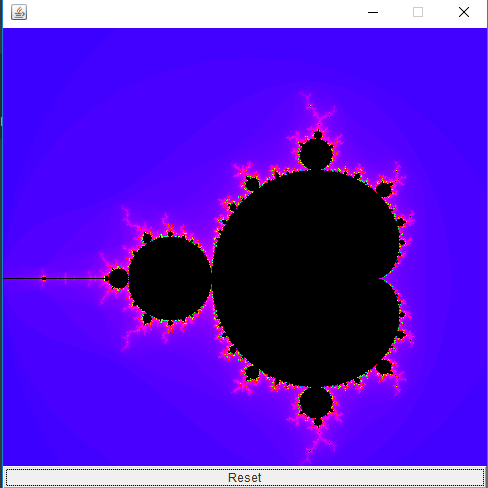


Рисунок – Фрактал Мандельброта в начальном положении отображения

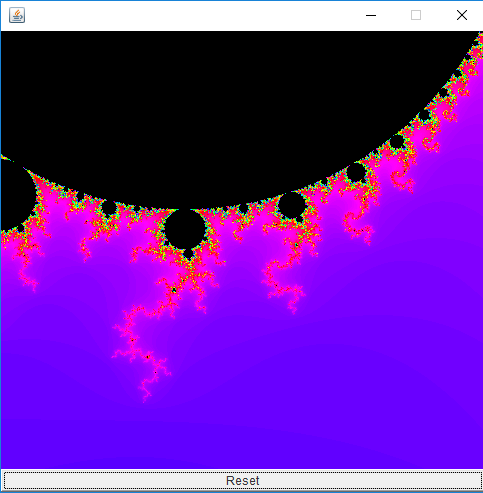


Рисунок – Фрактал Мандельброта в увеличенном отображении

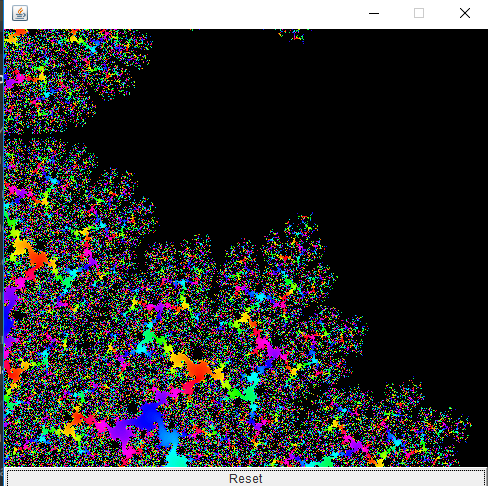


Рисунок – Фрактал Мандельброта при максимальном приближении

1. **Заключение**

В ходе данной лабораторной работы было разработано небольшое приложение, отображающее на графическом интерфейсе фрактал Мандельброта и позволяющее рассматривать его с различных ракурсов. Однако приложение обладает недостаточным быстродействием, поскольку очередное отображение фрактала занимает значительное время.

Цели и задачи данной лабораторной работы успешно выполнены, но приложение подвергнется дальнейшей разработке, направленной на устранение недостатков, в следующих работах.

1. **Литература**
2. Камаев В.А., Костерин В.В. Технологии программирования. М.: Высшая школа, 2006;
3. Жоголев Е.А.Технология программирования. – М.: Научный мир, 2004;
4. <https://www.google.ru/>