**Федеральное государственное общеобразовательное учреждение высшего образования**

**Московский государственный технический университет**

**им. Н. Э. Баумана**

Факультет "Информатика и системы управления"

Кафедра "Информационные системы и телекоммуникации"

**Отчет по лабораторной работе №5**

**"Выбор и сохранение фракталов"**

Студент: Задубин А.А.

Группа: ИУ3-42Б

Преподаватель: Руденкова Ю.С.

**Задание:**

Необходимо было добавить: Во первых, возможность отображения нескольких фракталов и выбор фрактала с помощью выпадающего списка. Во вторых, возможность сохранить текущее изображение фрактала в файл.

Создали новые реализации FractalGenerator. Первая – фрактал треуголка. Сохранили ее в файле Tricorn.java. Сделали класс наследник FractalGenerator, и реализация его была почти такая же, как реализация Мандельброта, за исключением двух отличий. Фактически, мы могли просто скопировать исходный код фрактала Мандельброта, и внести следующие изменения. Уравнение теперь такое zn = zn-1^2 + c. Разница только в том, что на каждой итерации вычисляется комплексно сопряженное значение zn-1.Начальный диапазон фрактала треуголка должен был быть от (-2, -2) до (2, 2). Второй фрактал, который надо было сделать называется "Горящий Корабль", который в действительности выглядит как горящий корабль. Вот детали его реализации: Уравнение для вычисления фрактала zn = (|Re(zn-1)| + i |Im(zn-1)|) ^2 + c. Начальный диапазон фрактала от (-2, -2.5) до (2, 1.5).

Использовали конструктор без аргументов, и метод addItem(Object) для добавления каждой реализации абстрактного класса FractalGenerator. Выпадающий список использует метод toString() объектов генераторов фракталов для отображения их имен.

Добавили к пользовательскому интерфейсу, перед выпадающим списком, текстовое поле поясняющим роль выпадающего списка. Для этого создали новый объект JPanel, добавили в него объекты JLabel и JComboBox, и затем поместили панель в позицию NORTH содержимого JFrame. Наконец, добавили обработку событий выпадающего списка к реализации ActionListener.

Далее добавили на экран программы кнопку "Сохранить изображение". Обе кнопки Сохранить и Сброс можно было поместить в новый объект JPanel, и затем добавить его в область SOUTH объекта JFrame программы, аналогично тому как раньше мы поступили с текстовым полем и выпадающим списком.

Для кнопки Сохранить также надо было предусмотреть обработку событий в ActionListener. Кнопки Сохранить и Сброс должны были иметь собственные значения команд, для того чтобы обработчик мог их различить. В обработчике кнопки "Сохранить ", надо было добавить выбор имени файла в котором будет сохранено изображение. Это можно было очень просто сделать с помощью класса javax.swing.JFileChooser. Этот класс имеет метод showSaveDialog(), который вызывает диалоговое окно "Сохранить файл " в котором пользователь может указать имя сохраняемого файла. Методу нужно было передать ссылку на объект Component который будет родительским окном диалога выбора файла; это нужно для того чтобы диалог центрировался относительно родительского окна. Подставили в этот параметр ссылку на объект JFrame. Метод возвращает значение типа int, которое определяет результат операции выбора файла. Если метод возвращает константу JFileChooser.APPROVE\_OPTION, можно будет продолжить операцию записи файла; иначе, пользователь отменил запрос, и надо просто вернуть управление.

**Код программы:**

BurningShip.java

**import** **java.awt.geom.Rectangle2D.Double**;

**public** **class** **BurningShip** **extends** FractalGenerator {

**public** **static** **final** **int** MAX\_ITERATIONS = **500**;

**@Override**

**public** **void** **getInitialRange**(Double range) {

range.x = -**2**;

range.y = -**2.5**;

range.width = **4**;

range.height = **4**;

}

**@Override**

**public** **int** **numIterations**(**double** x, **double** y) {

// xn = Re, yn = lm;

**double** xn = **0**, yn = **0**;

**double** zk = **0**, tmp = **0**;

**int** i;

**for** (i = **0**; i < MAX\_ITERATIONS; i++) {

tmp = xn \* xn - yn \* yn + x;

yn = **2** \* Math.abs(xn) \* Math.abs(yn) + y;

xn = tmp;

zk = xn \* xn + yn \* yn;

**if** (zk > **4**) **return** i;

}

**return** (i == MAX\_ITERATIONS) ? -**1** : i;

}

**@Override**

**public** String **toString**() {

**return** "Горящий корабль";

}

}

FractalExplorer.java

**import** **java.awt.BorderLayout**;

**import** **java.awt.Color**;

**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**;

**import** **java.awt.image.RenderedImage**;

**import** **javax.imageio.ImageIO**;

**import** **javax.swing.JButton**;

**import** **javax.swing.JComboBox**;

**import** **javax.swing.JFileChooser**;

**import** **javax.swing.JFrame**;

**import** **javax.swing.JLabel**;

**import** **javax.swing.JOptionPane**;

**import** **javax.swing.JPanel**;

**import** **javax.swing.filechooser.FileFilter**;

**import** **javax.swing.filechooser.FileNameExtensionFilter**;

**public** **class** **FractalExplorer** {

**private** **int** side;

**private** JImageDisplay jpgImg;

// private FractalGenerator geni = new Mandelbrot();

// private FractalGenerator geni = new Tricorn();

**private** FractalGenerator geni = **new** BurningShip();

**private** Rectangle2D.Double vsblRange;

**public** **FractalExplorer**(**int** x) {

side = x;

vsblRange = **new** Rectangle2D.Double();

geni.getInitialRange(vsblRange);

jpgImg = **new** JImageDisplay(x, x);

}

**class** **ActiveLisner** **implements** ActionListener{

**@Override**

**public** **void** **actionPerformed**(ActionEvent e) {

String cmd = e.getActionCommand();

**if** (cmd.equals("Сброс")) {

jpgImg.clearImage();

geni.getInitialRange(vsblRange);

drawFractal();

}

**else** **if** (cmd.equals("comboBoxChanged")) {

JComboBox<FractalGenerator> combo = (JComboBox<FractalGenerator>)e.getSource();

geni = (FractalGenerator)combo.getSelectedItem();

geni.getInitialRange(vsblRange);

drawFractal();

}

**else** **if** (cmd.equals("Скрин")) {

JFileChooser choozer = **new** JFileChooser();

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

choozer.setFileFilter(filter);

choozer.setAcceptAllFileFilterUsed(**false**);

**int** a = choozer.showSaveDialog(choozer.getParent());

System.out.println(e);

**if** (a == JFileChooser.APPROVE\_OPTION) {

String png = "png";

**try** {

ImageIO.write((RenderedImage)jpgImg.getImg(), png, choozer.getSelectedFile());

}

**catch** (Exception ex) {

JOptionPane.showMessageDialog(choozer.getParent(), ex.getMessage(), "Ты чё за лал ваще", JOptionPane.ERROR\_MESSAGE);

}

}

}

}

}

**class** **MouseClass** **extends** MouseAdapter{

**public** **void** **mouseClicked**(MouseEvent e) {

System.out.println(e);

**double** xCoord = FractalGenerator.getCoord(vsblRange.x, vsblRange.x + vsblRange.width, side, e.getX());

**double** yCoord = FractalGenerator.getCoord(vsblRange.y, vsblRange.y + vsblRange.height, side, e.getY());

geni.recenterAndZoomRange(vsblRange, xCoord, yCoord, **0.3**);

drawFractal();

}

}

**private** **void** **drawFractal**() {

**int** a;

**for** (**int** i = **0**; i < side; i++) {

**for** (**int** j = **0**; j < side; j++) {

**double** xCoord = FractalGenerator.getCoord(vsblRange.x, vsblRange.x + vsblRange.width, side, i);

**double** yCoord = FractalGenerator.getCoord(vsblRange.y, vsblRange.y + vsblRange.height, side, j);

a = geni.numIterations(xCoord, yCoord);

**float** hue = **0.7f** + (**float**) a / **200**f;

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

**if** (a == -**1**) {

jpgImg.drawPixel(i, j, **0**);

} **else** {

jpgImg.drawPixel(i, j, rgbColor);

}

}

}

jpgImg.repaint();

}

**private** **void** **createAndShowGUI**() {

JFrame f = **new** JFrame("Лабораторная работа 5");

JComboBox<FractalGenerator> combo = **new** JComboBox<FractalGenerator>();

f.setLayout(**new** BorderLayout());

f.add(jpgImg, BorderLayout.CENTER);

JButton b = **new** JButton("Сброс");

JPanel p2 = **new** JPanel();

JButton b2 = **new** JButton("Скрин");

f.add(b, BorderLayout.SOUTH);

p2.add(b2);

p2.add(b);

f.add(p2, BorderLayout.SOUTH);

b.addActionListener(**new** ActiveLisner());

jpgImg.addMouseListener(**new** MouseClass());

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

combo.addItem(**new** BurningShip());

combo.addItem(**new** Tricorn());

combo.addItem(**new** Mandelbrot());

JPanel p = **new** JPanel();

p.add(**new** JLabel("Фрактал "));

p.add(combo);

f.add(p, BorderLayout.NORTH);

combo.addActionListener(**new** ActiveLisner());

b2.addActionListener(**new** ActiveLisner());

f.pack();

f.setVisible(**true**);

f.setResizable(**false**);

}

**public** **static** **void** **main**(String[] args) {

FractalExplorer frac = **new** FractalExplorer(**800**);

frac.createAndShowGUI();

frac.drawFractal();

}

}

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);

}

JImageDisplay.java

**import** **java.awt.Dimension**;

**import** **java.awt.Graphics**;

**import** **java.awt.image.BufferedImage**;

**import** **javax.swing.JComponent**;

**public** **class** **JImageDisplay** **extends** JComponent {

**private** BufferedImage img;

**public** **JImageDisplay**(**int** width, **int** height) {

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

setPreferredSize(**new** Dimension(width, height));

}

**public** BufferedImage **getImg**() {

**return** img;

}

**@Override**

**protected** **void** **paintComponent**(Graphics g) {

g.drawImage(img, **0**, **0**, img.getWidth(), img.getHeight(), **null**);

}

**public** **void** **clearImage**() {

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

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

img.setRGB(i, j, **0**);

}

}

}

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

img.setRGB(x, y, rgbColor);

}

}

Mandelbrot.java

**import** **java.awt.geom.Rectangle2D.Double**;

**public** **class** **Mandelbrot** **extends** FractalGenerator {

**public** **static** **final** **int** MAX\_ITERATIONS = **1000**;

**@Override**

**public** **void** **getInitialRange**(Double range) {

range.x = -**2**;

range.y = -**1.5**;

range.width = **3**;

range.height = **3**;

}

**@Override**

**public** **int** **numIterations**(**double** x, **double** y) {

**double** xn = **0**, yn = **0**;

**double** zk = **0**, tmp = **0**;

**int** i;

**for** (i = **0**; i < MAX\_ITERATIONS; i++) {

tmp = xn \* xn - yn \* yn + x;

yn = **2** \* xn \* yn + y;

xn = tmp;

zk = xn \* xn + yn \* yn;

**if** (zk > **4**) **return** i;

}

**return** (i == MAX\_ITERATIONS) ? -**1** : i;

}

**@Override**

**public** String **toString**() {

**return** "Мандельброта";

}

}

Tricorn.java

**import** **java.awt.geom.Rectangle2D.Double**;

**public** **class** **Tricorn** **extends** FractalGenerator {

**public** **static** **final** **int** MAX\_ITERATIONS = **1000**;

**@Override**

**public** **void** **getInitialRange**(Double range) {

range.x = -**2**;

range.y = -**2**;

range.width = **4**;

range.height = **4**;

}

**@Override**

**public** **int** **numIterations**(**double** x, **double** y) {

**double** xn = **0**, yn = **0**;

**double** zk = **0**, tmp = **0**;

**int** i;

**for** (i = **0**; i < MAX\_ITERATIONS; i++) {

tmp = xn \* xn - yn \* yn + x;

yn = - **2** \* xn \* yn + y;

xn = tmp;

zk = xn \* xn + yn \* yn;

**if** (zk > **4**) **return** i;

}

**return** (i == MAX\_ITERATIONS) ? -**1** : i;

}

**@Override**

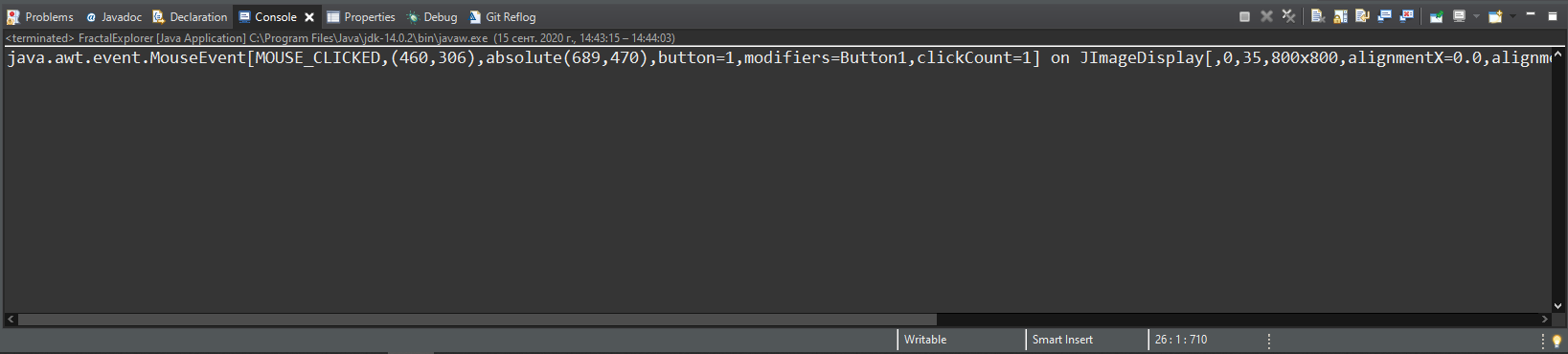
**public** String **toString**() {

**return** "Треуголка";

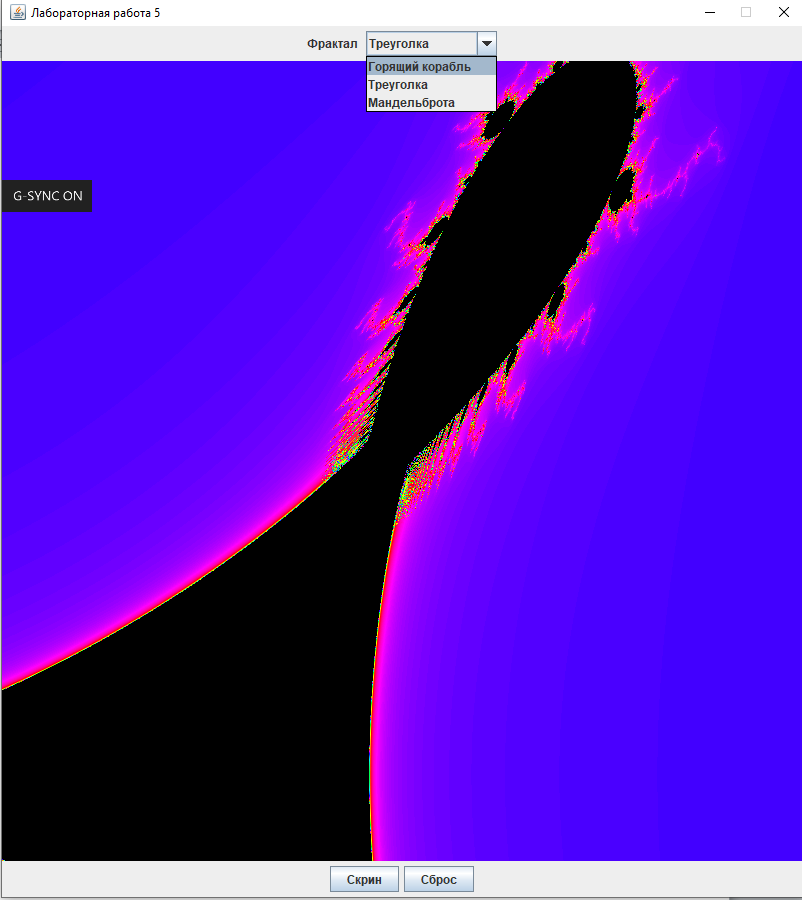
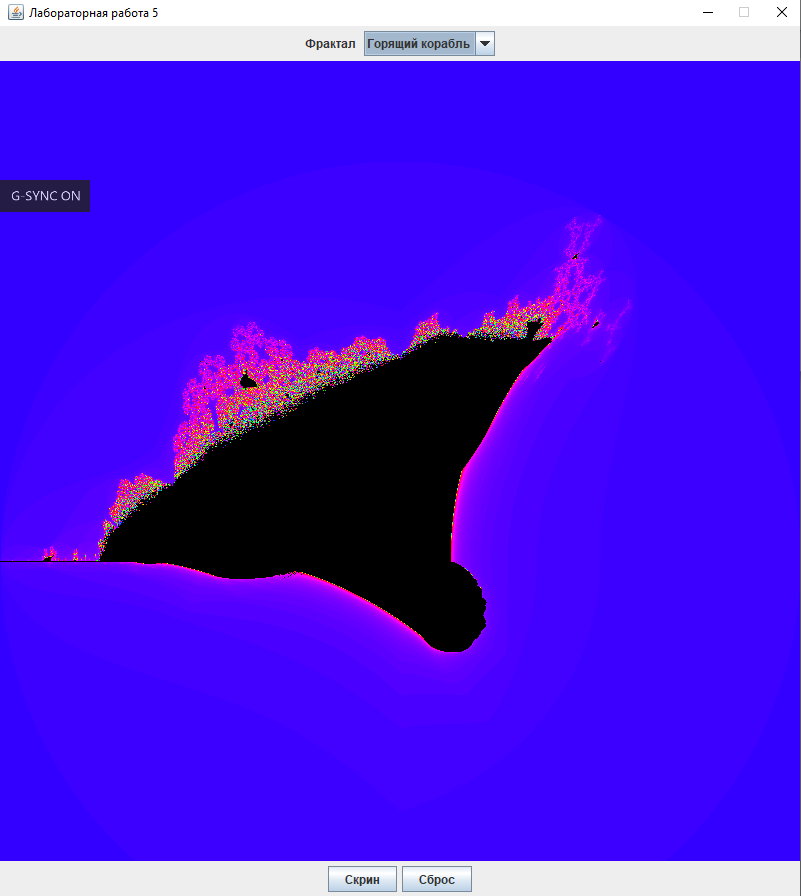
}

}

**Вывод в консоль:**

****

**Результат работы программы:**



**Вывод:** в данной лабораторной работе мы изучили разные типы фракталов, мы также можете сохранить то, что видим на диск. Научились создавать списки и выбирать из этих списков нужные фракталы.