ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ

Ордена Трудового Красного Знамени

федеральное государственное бюджетное

образовательное учреждение высшего образования

«Московский Технический Университет Связи и Информатики» (МТУСИ)

Кафедра Информатики

Лабораторная работа по технологиям программирования №5

«Выбор и сохранение фракталов»

Выполнил:

Студент 2 курса

Группы БСТ-1602

Тагунов Виталий

Вариант №16

Москва, 2018

**Цель работы:** основываясь на прошлой работе, добавить функцию выбора рисуемого фрактала и функцию сохранения изображения фрактала.

**Анализ предметной области:** в этой работе я использовал пакет разработчика Java Development Kit. Для написания кода была использована программа Блокнот, для того, чтобы можно было без проблем использовать виртуальную среду разработки Java и его компилятор.

**Программа:**

**BurningShip.java:**

import java.awt.geom.Rectangle2D;

public class BurningShip extends FractalGenerator{

// Maximum iterations:

public static final int MAX\_ITERATIONS = 2000;

/\*\*

\* @param range - The range borders of the fractal

\*/

public void getInitialRange (Rectangle2D.Double range){

range.x=-2;

range.y=-2.5;

range.width=range.height=4;

}

//Use formulas for fractal Mandelbrot

/\*\*

\* @param x - The abscissa of the point

\* @param y - The ordinate of the point

\*/

public int numIterations(double x, double y){

double re=0;

double im=0;

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

double re1=re\*re-im\*im+x;

double im1=Math.abs(2\*re\*im)+y;

//If we have reached the Mandelbrot condition, we will return the number of iterations

if ((im\*im+re\*re) > 4)

return i;

//The change the real and imaginary parts for the next iteration

re=re1;

im=im1;

}

return -1;

}

public String toString (){

return "BurningShip";

}

}

В классе **BurningShip** рассчитывается фрактал и здесь изменяется математическая функция и возвращается значение toString().

**Tricorn.java:**

import java.awt.geom.Rectangle2D;

public class Tricorn extends FractalGenerator{

// Maximum iterations:

public static final int MAX\_ITERATIONS = 2000;

/\*\*

\* @param range - The range borders of the fractal

\*/

public void getInitialRange (Rectangle2D.Double range){

range.x=-2;

range.y=-2;

range.width=range.height=4;

}

//Use formulas for fractal Mandelbrot

/\*\*

\* @param x - The abscissa of the point

\* @param y - The ordinate of the point

\*/

public int numIterations(double x, double y){

double re=0;

double im=0;

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

double re1=re\*re-im\*im+x;

double im1=-2\*re\*im+y;

//If we have reached the Mandelbrot condition, we will return the number of iterations

if ((im\*im+re\*re) > 4)

return i;

//The change the real and imaginary parts for the next iteration

re=re1;

im=im1;

}

return -1;

}

public String toString (){

return "Tricorn";

}

}

Еще один фрактал.

**FractalGenerator.java:**

**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.

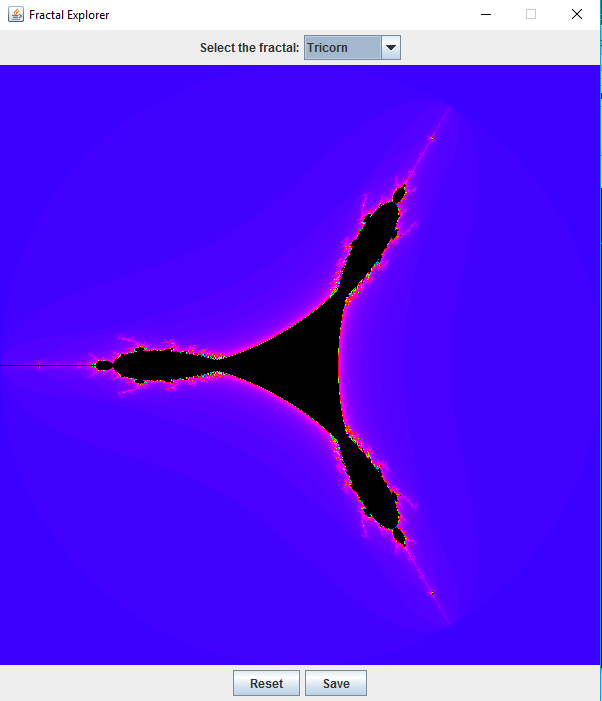
\*/

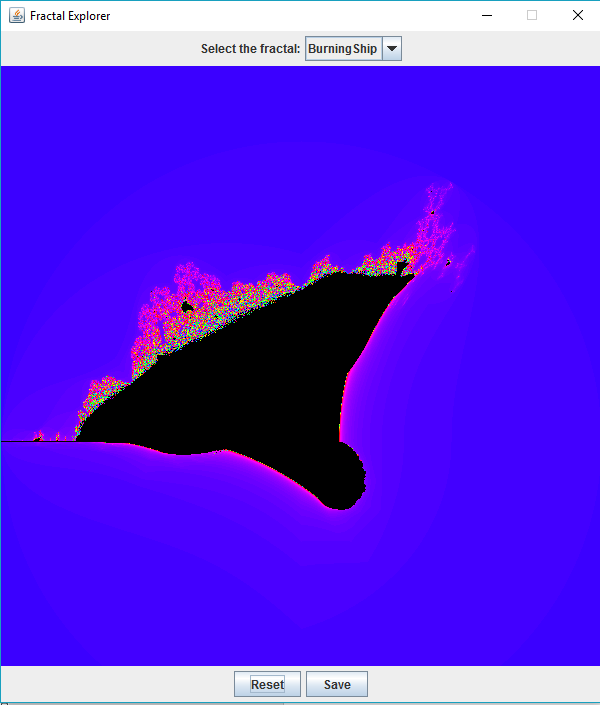
public abstract int numIterations(double x, double y);

}

В класс **FractalGenerator** добавляетсякласс **act\_listener**, который открывает новый фрактал и сохраняет текущий.

**Результат работы:**

****

****

**Вывод:** задача по графическому представлению фракталов и их сохранения выполнена.