Лабораторная работа №5

по дисциплине

“Объектно-ориентированное программирование”

Выполнил студент

группы БВТ2005

Нурмагомедов Магомед

Москва 2020

**Цель:**

Добавить в 4-ую лабораторную работу поддержку нескольких фракталов и реализовать сохранение текущего изображения в файл.

**Кодпрограммы:**

public class BurningShipextends FractalGenerator {  
public static final int*LIMIT* = 2000;  
  
 public void getInitialRange(Rectangle2D.Double range) {  
range.x= -2;  
range.y= -2.5;  
range.width= 4;  
range.height= 4;  
}  
  
public intnumIterations(double x, double y) {  
ComplexNumcmplx = new ComplexNum(0, 0);  
intiterator = 0;  
  
 while (iterator <*LIMIT* &&cmplx.getSquaredModule() <4) {  
cmplx.makeSquaredWithAbsInPoint(x, y);  
  
iterator++;  
}  
  
if (iterator == *LIMIT*) return -1;  
  
 return iterator;  
}  
  
@Override  
public String toString() { return "Burning Ship"; }  
}

public class Tricorn extends FractalGenerator {  
public static final int*LIMIT* = 2000;  
  
 public void getInitialRange(Rectangle2D.Double range) {  
range.x= -2;  
range.y= -2;  
range.width= 4;  
range.height= 4;  
}  
  
public intnumIterations(double x, double y) {  
ComplexNumcmplx = new ComplexNum(0, 0);  
intiterator = 0;  
  
 while (iterator <*LIMIT* &&cmplx.getSquaredModule() <4) {  
cmplx.makeSquaredWithConjInPoint(x, y);  
  
iterator++;  
}  
  
if (iterator == *LIMIT*) return -1;  
  
 return iterator;  
}  
  
@Override  
public String toString() { return "Tricorn"; }  
}

public class ComplexNum {  
public double rl;  
 public double im;  
  
 public ComplexNum(double rl, double im){  
this.rl= rl;  
 this.im = im;  
}  
  
public double getSquaredModule() {  
return (this.rl\* this.rl+ this.im \* this.im);  
}  
  
public void makeSquaredInPoint(double x, double y) {  
double real = (rl\* rl) - (im\* im) + x;  
 double imagine = 2 \* rl\* im+ y;  
  
rl= real;  
im= imagine;  
}  
  
public void makeSquaredWithConjInPoint(double x, double y) {  
double real = (rl\* rl) - (im\* im) + x;  
 double imagine = - 2 \* rl\* im+ y;  
  
rl= real;  
im= imagine;  
}  
  
public void makeSquaredWithAbsInPoint(double x, double y) {  
double real = (rl\* rl) - (im\* im) + x;  
 double imagine = 2 \* Math.*abs*(rl) \* Math.*abs*(im) + y;  
  
rl= real;  
im= imagine;  
}  
}

public class FractalExplorer {  
private intdisplaySize;  
 private JImageDisplaydisplay;  
 private FractalGeneratorfractal;  
 private Rectangle2D.Double range;  
  
 public FractalExplorer(intsize) {  
displaySize= size;  
  
fractal = new Mandelbrot();  
range = new Rectangle2D.Double();  
  
fractal.getInitialRange(range);  
display = new JImageDisplay(displaySize, displaySize);  
}  
  
public void createAndShowGUI() {  
display.setLayout(new BorderLayout());  
  
JButtonresetButton = new JButton("Reset");  
ResetterresetHandler = new Resetter();  
resetButton.addActionListener(resetHandler);  
  
JButtonsaveButton = new JButton("Save");  
Saver saveHandler = new Saver();  
saveButton.addActionListener(saveHandler);  
  
Clicker click = new Clicker();  
display.addMouseListener(click);  
  
FractalGeneratormandelbrotFractal = new Mandelbrot();  
FractalGeneratortricornFractal = new Tricorn();  
FractalGeneratorburningShipFractal = new BurningShip();  
  
JComboBoxcomboBox = new JComboBox();  
  
comboBox.addItem(mandelbrotFractal);  
comboBox.addItem(tricornFractal);  
comboBox.addItem(burningShipFractal);  
  
Chooser fractalChooser = new Chooser();  
comboBox.addActionListener(fractalChooser);  
  
JLabel label = new JLabel("Fractal:");  
  
JPanel panel = new JPanel();  
panel.add(label);  
panel.add(comboBox);  
  
JPanelmyBottomPanel = new JPanel();  
myBottomPanel.add(saveButton);  
myBottomPanel.add(resetButton);  
  
JFramemyFrame = new JFrame("Fractal Explorer");  
  
myFrame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
myFrame.add(myBottomPanel, BorderLayout.*SOUTH*);  
myFrame.add(display, BorderLayout.*CENTER*);  
myFrame.add(panel, BorderLayout.*NORTH*);  
  
myFrame.pack();  
myFrame.setVisible(true);  
myFrame.setResizable(false);  
}  
  
private void drawFractal() {  
for (intx = 0; x <displaySize; x++) {  
for (inty = 0; y <displaySize; y++) {  
  
double xCoord = FractalGenerator.*getCoord*(range.x,  
range.x+ range.width, displaySize, x);  
  
 double yCoord = FractalGenerator.*getCoord*(range.y,  
range.y+ range.height, displaySize, y);  
  
intiteration = fractal.numIterations(xCoord, yCoord);  
  
 if (iteration == -1) {  
display.drawPixel(x, y, 0);  
} else {  
float hue = 0.5f + (float) iteration / 50;  
intrgbColor = Color.*HSBtoRGB*(hue, 1f, 1f);  
  
display.drawPixel(x, y, rgbColor);  
}  
  
 }  
 }  
display.repaint();  
}  
  
private class Resetterimplements ActionListener {  
public void actionPerformed(ActionEvent e) {  
if (e.getActionCommand().equals("Reset")) {  
fractal.getInitialRange(range);  
drawFractal();  
}  
 }  
 }  
  
private class Chooser implements ActionListener {  
public void actionPerformed(ActionEvent e) {  
 Object source = e.getSource();  
 if (source instanceofJComboBox) {  
JComboBoxcomboBox = (JComboBox) source;  
  
fractal = (FractalGenerator) comboBox.getSelectedItem();  
 assert fractal != null;  
  
fractal.getInitialRange(range);  
drawFractal();  
}  
 }  
 }  
  
private class Saver implements ActionListener {  
public void actionPerformed(ActionEvent e) {  
if (e.getActionCommand().equals("Save")) {  
JFileChooserfileChooser = new JFileChooser();  
  
FileFilterextensionFilter = new FileNameExtensionFilter(  
"PNG",  
"png"  
);  
  
fileChooser.setFileFilter(extensionFilter);  
  
fileChooser.setAcceptAllFileFilterUsed(false);  
  
intuserSelection = fileChooser.showSaveDialog(display);  
  
 if (userSelection == JFileChooser.*APPROVE\_OPTION*) {  
java.io.File file = fileChooser.getSelectedFile();  
String filePath = file.getPath();  
  
 if (!filePath.contains(".png")) file = new File(filePath + ".png");  
 try {  
BufferedImagedisplayImage = display.getImage();  
javax.imageio.ImageIO.*write*(displayImage, "png", file);  
} catch (Exception exception) {  
JOptionPane.*showMessageDialog*(display,  
exception.getMessage(), "Cannot Save Image",  
JOptionPane.*ERROR\_MESSAGE*);  
}  
 }  
else return;  
}  
 }  
 }  
  
private class Clicker extends MouseAdapter {  
@Override  
public void mouseClicked(MouseEvent e) {  
intx = e.getX();  
 double xCoord = FractalGenerator.*getCoord*(range.x,  
range.x+ range.width, displaySize, x);  
  
inty = e.getY();  
 double yCoord = FractalGenerator.*getCoord*(range.y,  
range.y+ range.height, displaySize, y);  
  
fractal.recenterAndZoomRange(range, xCoord, yCoord, 0.5);  
  
drawFractal();  
}  
 }  
  
public static void main(String[] args)  
 {  
FractalExplorerdisplayExplorer = new FractalExplorer(800);  
displayExplorer.createAndShowGUI();  
displayExplorer.drawFractal();  
}  
}

**Примерыработыпрограммы:**

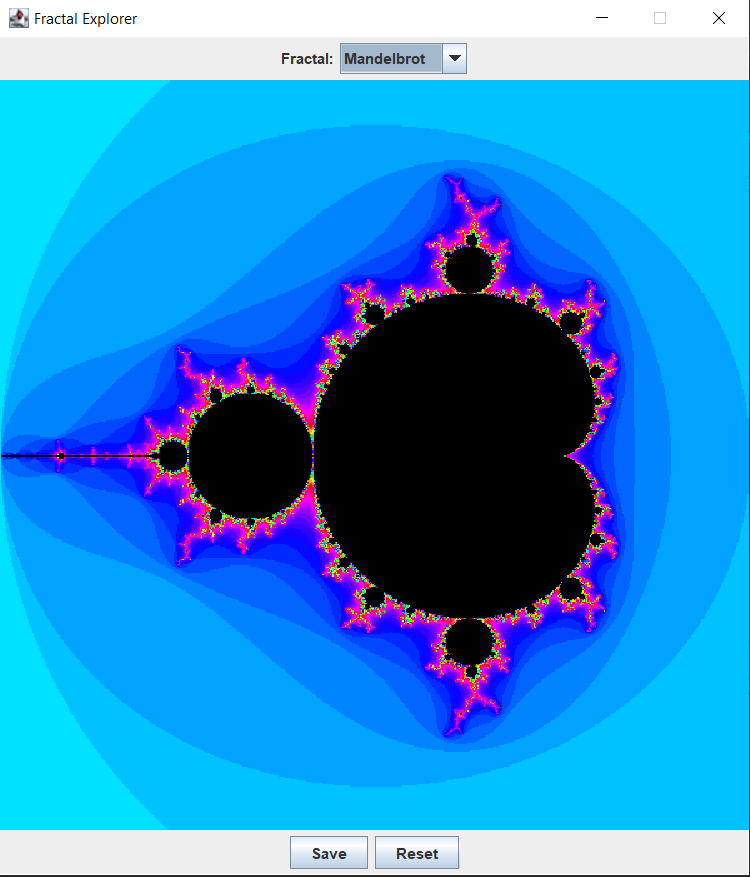


Рисунок1 - ВыводфракталаМандельброта

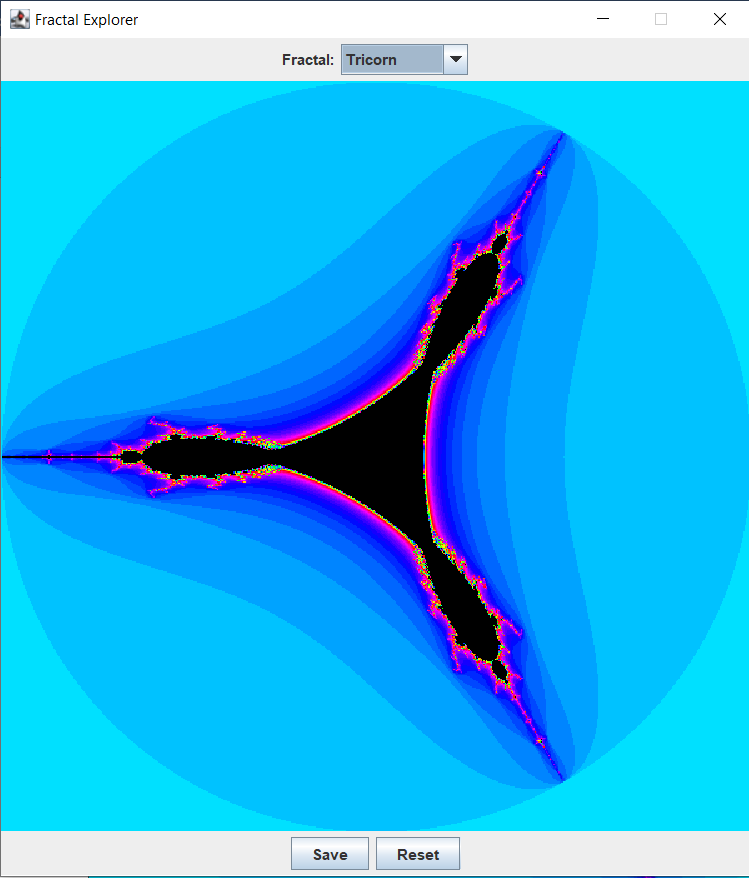


Рисунок2 - Выводфрактала «Трикорн»

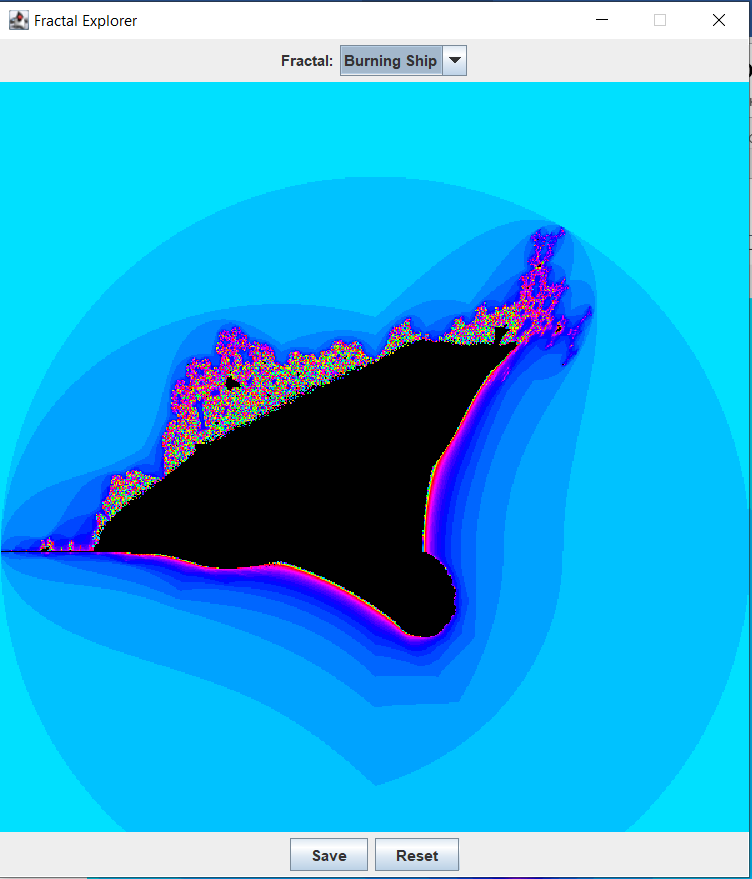


Рисунок 3 - ВыводфракталаПылающийкорабль

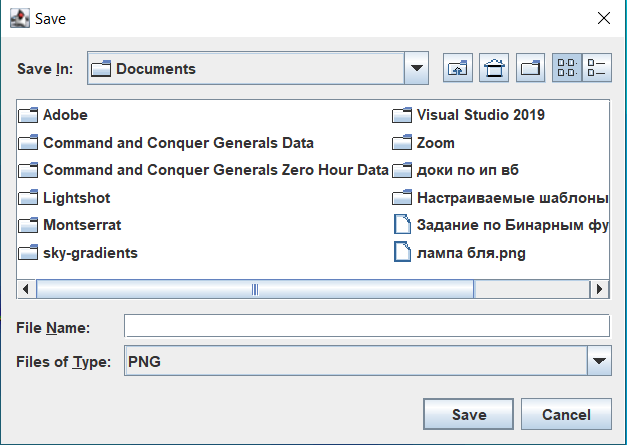


Рисунок 4 – Окносохранения