Лабораторная работа №6

по дисциплине

“Объектно-ориентированное программирование”

Выполнил студент

группы БВТ2005

Нурмагомедов Магомед

Москва 2020

**Цель:**

Реализовать возможность рисования фрактала с несколькими фоновыми потоками, и блокирование интерфейса при отрисовке фрактала.

**Кодпрограммы:**

public class FractalExplorer {  
private int displaySize;  
  
 private JImageDisplay display;  
  
 private FractalGenerator fractal;  
  
 private Rectangle2D.Double range;  
  
 private int rowRemaining;  
  
 private JButton saveButton;  
 private JButton resetButton;  
 private JComboBox comboBox;  
  
 public FractalExplorer(int size) {  
displaySize = size;  
  
fractal = new Mandelbrot();  
range = new Rectangle2D.Double();  
  
fractal.getInitialRange(range);  
display = new JImageDisplay(displaySize, displaySize);  
}  
  
public void createAndShowGUI() {  
display.setLayout(new BorderLayout());  
  
resetButton = new JButton("Reset");  
Resetter resetHandler = new Resetter();  
resetButton.addActionListener(resetHandler);  
  
saveButton = new JButton("Save");  
Saver saveHandler = new Saver();  
saveButton.addActionListener(saveHandler);  
  
Clicker click = new Clicker();  
display.addMouseListener(click);  
  
FractalGenerator mandelbrotFractal = new Mandelbrot();  
FractalGenerator tricornFractal = new Tricorn();  
FractalGenerator burningShipFractal = new BurningShip();  
  
comboBox = 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);  
  
JPanel myBottomPanel = new JPanel();  
myBottomPanel.add(saveButton);  
myBottomPanel.add(resetButton);  
  
JFrame myFrame = 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() {  
 enableUI(false);  
  
rowRemaining = displaySize;  
  
 for (int y = 0; y <displaySize; y++){  
 FractalWorker drawRow = new FractalWorker(y);  
drawRow.execute();  
}  
 }  
  
private void enableUI(boolean value) {  
comboBox.setEnabled(value);  
resetButton.setEnabled(value);  
saveButton.setEnabled(value);  
}  
  
private class Resetter implements 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 instanceof JComboBox) {  
 JComboBox comboBox = (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")) {  
 JFileChooser fileChooser = new JFileChooser();  
  
FileFilter extensionFilter = new FileNameExtensionFilter(  
"PNG",  
"png"  
);  
  
fileChooser.setFileFilter(extensionFilter);  
  
fileChooser.setAcceptAllFileFilterUsed(false);  
  
 int userSelection = 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 {  
 BufferedImage displayImage = 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) {  
if (rowRemaining != 0) return;  
  
 int x = e.getX();  
 double xCoord = FractalGenerator.*getCoord*(range.x,  
range.x + range.width, displaySize, x);  
  
 int y = e.getY();  
 double yCoord = FractalGenerator.*getCoord*(range.y,  
range.y + range.height, displaySize, y);  
  
fractal.recenterAndZoomRange(range, xCoord, yCoord, 0.5);  
  
drawFractal();  
}  
 }  
  
private class FractalWorker extends SwingWorker<Object, Object> {  
int row;  
 int[] rgbValues;  
  
 private FractalWorker(int row) { this.row = row; }  
  
@Override  
protected Object doInBackground() {  
rgbValues = new int[displaySize];  
  
 for (int x = 0; x <rgbValues.length; x++) {  
double xCoord = FractalGenerator.*getCoord*(range.x,  
range.x + range.width, displaySize, x);  
  
 double yCoord = FractalGenerator.*getCoord*(range.y,  
range.y + range.height, displaySize, row);  
  
 int iteration = fractal.numIterations(xCoord, yCoord);  
  
 if (iteration == -1) {  
display.drawPixel(x, row, 0);  
} else {  
float hue = 0.5f + (float) iteration / 50;  
 int rgbColor = Color.*HSBtoRGB*(hue, 1f, 1f);  
  
rgbValues[x] = rgbColor;  
}  
 }  
return null;  
}  
  
protected void done() {  
for (int x = 0; x <rgbValues.length; x++) {  
display.drawPixel(x, row, rgbValues[x]);  
}  
  
display.repaint(0, 0, row, displaySize, 1);  
  
rowRemaining--;  
 if (rowRemaining == 0) enableUI(true);  
}  
 }  
  
public static void main(String[] args)  
 {  
 FractalExplorer displayExplorer = new FractalExplorer(800);  
displayExplorer.createAndShowGUI();  
displayExplorer.drawFractal();  
}  
}

**Примерыработыпрограммы:**

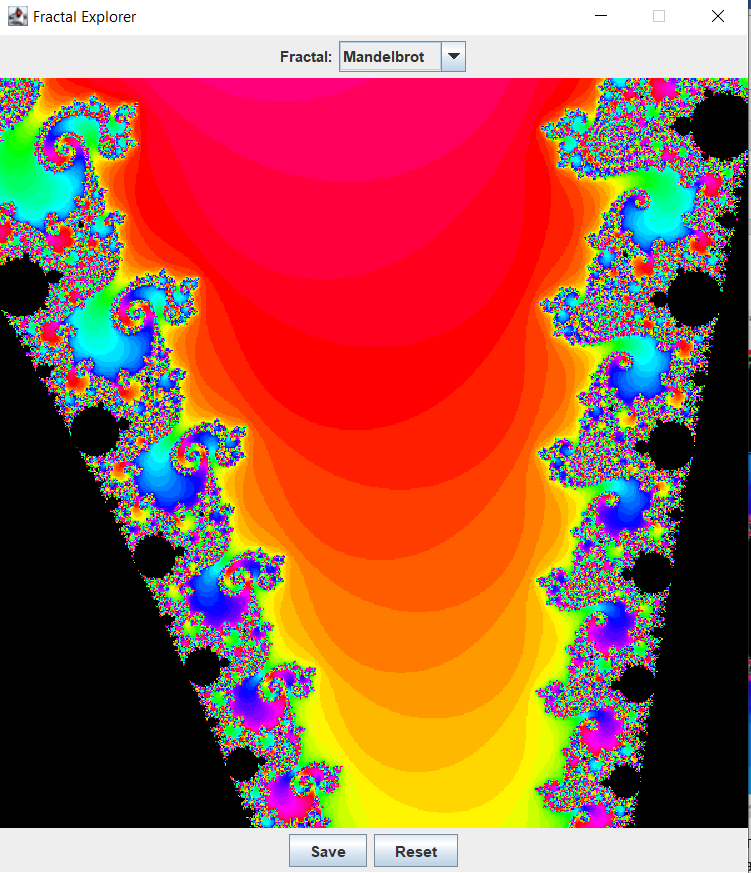


Рисунок1–Наглядное представление построчного вычисления и блокирования интерфейса