**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

**Кафедра Информатики**

****

**Отчет по лабораторной работе №6**

по предмету «КТП»:

Выполнил: студент группы БВТ1802

Каринов Евгений Александрович

Руководитель:

Ксения Андреевна Полянцева

Москва 2020

**1 Цель работы**

Цель работы: изучить алгоритм расчета фрактала, научиться исользовать java.swing и класс java.swing.worker.

**2 Задание**

Распараллелить отрисовку фрактала, сделав ее более быстрой.

**3 Текст программы**

**Class FractalExplorer**

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.geom.Rectangle2D;  
import java.awt.event.\*;  
import java.awt.image.BufferedImage;  
import java.awt.image.RenderedImage;  
import java.io.IOException;  
import java.nio.Buffer;  
import javax.swing.filechooser.\*;  
import javax.imageio.ImageIO;  
  
public class FractalExplorer {  
 private int m\_DisplaySize;  
 private FractalGenerator m\_Generator;  
 private Rectangle2D.Double m\_Range;  
 private JImageDisplay m\_Display;  
 private JButton m\_ResetButton;  
 private JComboBox m\_Switch;  
 private JButton m\_SaveButton;  
 private JFrame m\_Frame;  
 private int m\_RowsRemaining;  
  
  
  
 private class actionListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent actionEvent) {  
 if (actionEvent.getSource() == m\_Switch) {  
 m\_Generator = (FractalGenerator) m\_Switch.getSelectedItem();  
 m\_Generator.getInitialRange(m\_Range);  
 drawFractal();  
 }  
 else if (actionEvent.getSource() == m\_ResetButton) {  
 m\_Generator.getInitialRange(m\_Range);  
 drawFractal();  
 }  
 else if (actionEvent.getSource() == m\_SaveButton) {  
 JFileChooser chooser = new JFileChooser();  
 FileFilter filter = new FileNameExtensionFilter("PNG Images", "png");  
 chooser.setFileFilter(filter);  
 chooser.setAcceptAllFileFilterUsed(false);  
 if (chooser.showSaveDialog(null) == JFileChooser.*APPROVE\_OPTION*) {  
 try {  
 ImageIO.*write*(m\_Display.img, "png", chooser.getSelectedFile());  
 } catch (IOException e) {  
 JOptionPane.*showMessageDialog*(m\_Frame, e.getMessage(), "Cannot Save Image",  
 JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 }  
 }  
 }  
  
 private class MouseListener extends MouseAdapter {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 if (m\_RowsRemaining != 0) return;  
 int x = e.getX();  
 int y = e.getY();  
  
 double xCoord = m\_Generator.*getCoord*(m\_Range.x, m\_Range.x + m\_Range.width, m\_DisplaySize,x);  
 double yCoord = m\_Generator.*getCoord*(m\_Range.y, m\_Range.y + m\_Range.height, m\_DisplaySize,y);  
 m\_Generator.recenterAndZoomRange(m\_Range, xCoord, yCoord, 0.5);  
 drawFractal();  
 }  
 }  
 public FractalExplorer(int ScreenSize) {  
 m\_DisplaySize = ScreenSize;  
 m\_Range = new Rectangle2D.Double();  
 m\_Generator = new Mandelbrot();  
 m\_Generator.getInitialRange(m\_Range);  
 }  
  
 private class FractalWorker extends SwingWorker<Object, Object> {  
  
 private int m\_Ycoord;  
 private int[] m\_Xcoords;  
  
 private FractalWorker(int yCoord) {  
 m\_Ycoord = yCoord;  
 }  
 @Override  
 protected Object doInBackground() throws Exception {  
 m\_Xcoords = new int[m\_DisplaySize];  
 double yCoord = FractalGenerator.*getCoord* (m\_Range.y, m\_Range.y + m\_Range.height, m\_DisplaySize, m\_Ycoord);  
 for (int x = 0; x < m\_DisplaySize; x++)  
 {  
 double xCoord = FractalGenerator.*getCoord* (m\_Range.x, m\_Range.x + m\_Range.width, m\_DisplaySize, x);  
 int IterNum = m\_Generator.numIterations(xCoord, yCoord);  
 if (IterNum == -1) m\_Xcoords[x] = 0;  
 else {  
 float hue = 0.7f + (float) IterNum / 200f;  
 int rgbColor = Color.*HSBtoRGB*(hue, 1f, 1f);  
 m\_Xcoords[x] = rgbColor;  
 }  
 }  
 return null;  
 }  
  
 @Override  
 protected void done() {  
 for (int x = 0; x <m\_DisplaySize; x++) {  
 m\_Display.drawPixel(x, m\_Ycoord, m\_Xcoords[x]);  
 }  
 m\_Display.repaint(0, m\_Ycoord, m\_DisplaySize, 1);  
 m\_RowsRemaining--;  
 if (m\_RowsRemaining == 0) {  
 enableUI(true);  
 }  
 }  
 }  
  
 public void createAndShowGUI() {  
  
 // setting up a panel with switch and label  
 JPanel panel = new JPanel();  
 m\_Switch = new JComboBox();  
 m\_Switch.addItem(new Mandelbrot());  
 m\_Switch.addItem(new Tricorn());  
 m\_Switch.addItem(new BurningShip());  
 m\_Switch.addActionListener(new actionListener());  
 JLabel label = new JLabel("Fractal type:");  
 panel.add(label);  
 panel.add(m\_Switch);  
  
 // setting up display  
 m\_Display = new JImageDisplay(m\_DisplaySize, m\_DisplaySize);  
 m\_Display.addMouseListener(new MouseListener());  
  
 // setting up a panel with reset and save buttons  
 m\_ResetButton = new JButton("Reset Image");  
 m\_ResetButton.addActionListener(new actionListener());  
 m\_SaveButton = new JButton("Save Image");  
 m\_SaveButton.addActionListener(new actionListener());  
 JPanel panel2 = new JPanel();  
 panel2.add(m\_ResetButton);  
 panel2.add(m\_SaveButton);  
  
 // setting up the whole frame  
 m\_Frame = new JFrame();  
 m\_Frame.getContentPane().add(panel, BorderLayout.*NORTH*);  
 m\_Frame.getContentPane().add(m\_Display, BorderLayout.*CENTER*);  
 m\_Frame.getContentPane().add(panel2, BorderLayout.*SOUTH*);  
 m\_Frame.setDefaultCloseOperation(JFrame.*DISPOSE\_ON\_CLOSE*);  
  
  
 // showing gui  
 m\_Frame.pack();  
 m\_Frame.setVisible(true);  
 m\_Frame.setResizable(true);  
 }  
  
 void enableUI(boolean val) {  
 m\_ResetButton.setEnabled(val);  
 m\_SaveButton.setEnabled(val);  
 m\_Switch.setEnabled(val);  
 }  
  
 private void drawFractal() {  
 enableUI(false);  
 m\_RowsRemaining = m\_DisplaySize;  
 for (int y = 0; y < m\_DisplaySize; y++) {  
 FractalWorker worker = new FractalWorker(y);  
 worker.execute();  
 }  
 }  
  
 public static void main(String args[]) {  
 FractalExplorer explorer = new FractalExplorer(1000);  
 explorer.createAndShowGUI();  
 explorer.drawFractal();  
 }  
  
}

**4 Работа программы**

Время до появления фрактала на экране значительно уменьшилось, и теперь отрисовку фрактала можно наблюдать в процессе ее выполнения, чего нельзя сказать об однопоточной программе, где приходилось несколько секунд наблюдать черный экран.



