Федеральное агентство связи

Ордена Трудового Красного знамени

Федеральное государственное бюджетное образовательное учреждение

Высшего образования

«Московский технический университет связи и информатики»

Кафедра Математической кибернетики и информационных технологий

Отчет по лабораторной работе №6

по дисциплине «Объектно ориентированное программирование»

на тему «Многопоточный генератор фракталов»

Выполнил

Студент группы БФИ1701

Неретин И.Г.

Проверила:

Мосеева М.С.

Москва 2020

1. **Цель работы**

Цель данной лабораторной работы – научиться работать с несколькими потоками в фоновом режиме при использовании фреймфорка Swing.

1. **Задание**

Цель работы определила следующие задачи:

1. Разработать подкласс потока «FractalWorker» для вычисления цвета строки;
2. Запустить несколько потоков, вычисляющих значения цветов для строк, для каждой строки изображения и вывести их на изображение;
3. Создать ограничения для пользовательского интерфейса во время рендеринга изображения.
4. **Порядок выполнения работы**

Изменения были произведены только в файле «FractalExplorer.java», соответственно прикреплен код только этого файла. Остальные файлы находятся в репозитории лабораторной работы.

Листинг – FractalExplorer.java

|  |
| --- |
| import javax.swing.\*; import javax.swing.filechooser.FileFilter; import javax.swing.filechooser.FileNameExtensionFilter; import javax.swing.plaf.basic.BasicButtonListener; import javax.swing.plaf.basic.BasicOptionPaneUI; import java.awt.\*; import java.awt.color.ICC\_ProfileRGB; 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.io.File;  public class FractalExplorer {  private int screenSize;  private JImageDisplay display;  private FractalGenerator CurrentFractalGenerator;  private Rectangle2D.Double range;  private JFrame frame;  private int rowsRemaining;  private boolean interfaceEnabled;   //interface  private Button reset;  private Button saveImg;  private JComboBox<FractalGenerator> comboBox;   FractalExplorer(int size){  screenSize = size;  display = new JImageDisplay(screenSize, screenSize);  range = new Rectangle2D.Double();  CurrentFractalGenerator = new Mandelbrot();  CurrentFractalGenerator.getInitialRange(range);  }  public void createAndShowGUI(){  //кнопка сброса  reset = new Button("Reset");  reset.setSize(screenSize / 3, 50);  reset.addActionListener(new JButtonClick());  reset.setVisible(true);   //кнопка сохранения фрактала  saveImg = new Button("Save");  saveImg.setSize(screenSize / 3, 50);  saveImg.addActionListener(new SaveButtonClick());  saveImg.setVisible(true);   //создание фракталов  BurningShip BurningShipFractal = new BurningShip();  Tricorn TricornFractal = new Tricorn();   //комбобокс выбора фракталов  comboBox = new JComboBox<>();  comboBox.addItem(CurrentFractalGenerator);  comboBox.addItem(BurningShipFractal);  comboBox.addItem(TricornFractal);  comboBox.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  FractalGenerator newFractal = (FractalGenerator) comboBox.getSelectedItem();  if (newFractal != null){  CurrentFractalGenerator = newFractal;  newFractal.getInitialRange(range);  drawFractal();  }  }  });   //панель вывода компонентов интерфейса  JPanel panel = new JPanel();  GroupLayout layout = new GroupLayout(panel);  panel.setLayout(layout);   //настройка размещения элементов интерфейса на форме  layout.setHorizontalGroup(layout.createParallelGroup()  .addComponent(display)  .addComponent(comboBox)  .addGroup(layout.createSequentialGroup()  .addComponent(reset)  .addComponent(saveImg))   );  layout.setVerticalGroup(layout.createSequentialGroup()  .addComponent(comboBox)  .addComponent(display)  .addGroup(layout.createParallelGroup()  .addComponent(reset)  .addComponent(saveImg)  )  );   display.clearImage();   //создание и настройка окна  frame = new JFrame();  frame.setContentPane(panel);  frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  frame.setPreferredSize(new Dimension(screenSize, screenSize));  frame.pack();  frame.setVisible(true);  frame.setResizable(false);  frame.addMouseListener(new JMouseAdapter());  this.drawFractal();  }   private void drawFractal(){  int j;  rowsRemaining = display.getHeight();  enableUI(false);  for (j = 0; j < display.getHeight(); j+=4){  FractalWorker worker1 = new FractalWorker(j);  FractalWorker worker2 = new FractalWorker(j+1);  FractalWorker worker3 = new FractalWorker(j+2);  FractalWorker worker4 = new FractalWorker(j+3);  worker1.execute();  worker2.execute();  worker3.execute();  worker4.execute();  }   }   private class JMouseAdapter extends java.awt.event.MouseAdapter {  @Override  public void mouseClicked(MouseEvent e) {  super.mouseClicked(e);  if (interfaceEnabled){  display.drawPixel(e.getX(), e.getY(), ICC\_ProfileRGB.*icSigGreenColorantTag*);  double xCoord = CurrentFractalGenerator.getCoord(range.x, range.x + range.width, screenSize, e.getX());;  double yCoord = CurrentFractalGenerator.getCoord(range.y, range.y + range.height, screenSize, e.getY());  FractalGenerator.*recenterAndZoomRange*(range, xCoord, yCoord, 0.5);  display.repaint();  drawFractal();  }  }  }   private class JButtonClick implements ActionListener {  @Override  public void actionPerformed(ActionEvent e) {  CurrentFractalGenerator.getInitialRange(range);  drawFractal();  }  }   private class SaveButtonClick implements ActionListener {  @Override  public void actionPerformed(ActionEvent e) {  JFileChooser saveDlg = new JFileChooser();  if (saveDlg.showSaveDialog(frame) == JFileChooser.*APPROVE\_OPTION*){  File file = saveDlg.getSelectedFile();  FileFilter filter = new FileNameExtensionFilter("PNG Images", "png");  saveDlg.setFileFilter(filter);  saveDlg.setAcceptAllFileFilterUsed(false);  try{  javax.imageio.ImageIO.*write*(display.getBufferedImage(), "png", file);  }  catch (Exception Exc){  JOptionPane.*showMessageDialog*(frame, "Saving image failed. " + Exc.getMessage(), "error", JOptionPane.*ERROR\_MESSAGE*);  }  }  }  }    private class FractalWorker extends SwingWorker<Object, Object>{  int y;  int[] colorArray;  FractalWorker(int y){  this.y = y;  }  @Override  protected Object doInBackground() throws Exception {  colorArray = new int[display.getWidth()];  int i, j;  double xCoord, yCoord;  int iterations;  float hue;  int rgbColor;  for (i = 0; i < display.getWidth(); i++){  xCoord = CurrentFractalGenerator.getCoord(range.x, range.x + range.width, screenSize, i);  yCoord = CurrentFractalGenerator.getCoord(range.y, range.y + range.height, screenSize, y);  iterations = CurrentFractalGenerator.numIterations(xCoord, yCoord);  if (iterations == -1){  rgbColor = 0;  }  else{  hue = 0.7f + (float) iterations / 200f;  rgbColor = Color.*HSBtoRGB*(hue, 1f, 1f);  }  colorArray[i] = rgbColor;  }  return null;  }   @Override  protected void done() {  for (int i = 0; i < display.getWidth(); i++){  display.drawPixel(i, y, colorArray[i]);  }  display.repaint(0, 0, y, display.getWidth(), 1);  rowsRemaining--;  if (rowsRemaining == 0){  enableUI(true);  }  }  }   public void enableUI(boolean a){  saveImg.setEnabled(a);  reset.setEnabled(a);  comboBox.setEnabled(a);  interfaceEnabled = a;  } } |

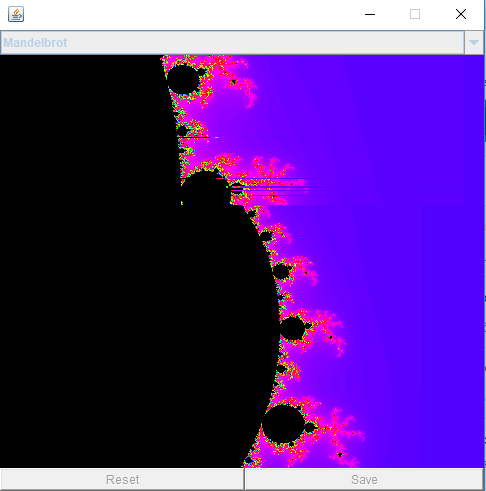


Рисунок – Фрактал во время рендеринга. Видны небольшие линии, свидетельствующие о работе нескольких параллельных потоков.

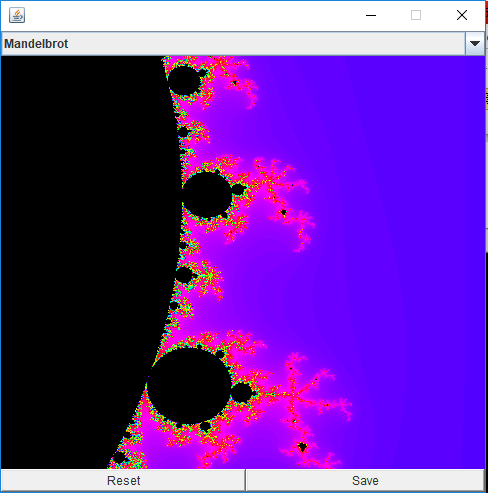


Рисунок – Фрактал завершил рендеринг

1. **Заключение**

Фреймворк Swing позволяет легко создавать потоки, исполняющиеся в фоновом режиме. Достаточно лишь определить класс потока с вычислительной функцией и запустить его. В целом, при использовании потоков, скорость отображения фрактала стала значительно больше.

1. **Литература**
2. Камаев В.А., Костерин В.В. Технологии программирования. М.: Высшая школа, 2006;
3. Жоголев Е.А.Технология программирования. – М.: Научный мир, 2004;
4. <https://www.google.ru/>