Ордена Трудового Красного Знамени

федеральное государственное бюджетное

образовательное учреждение высшего

образования «Московский технический

университет связи и информатики»

Кафедра информатики

## Лабораторная работа № 5 «Multithreaded Fractal Explorer»

Выполнил:

Студент 2 курса

Факультета «ОТФ-2»

Группы БСТ-1602

Марченко Дмитрий

Вариант № 11

Москва, 2018

**Цель работы:** В данной задаче мы закончим с генератором фракталов с еще одной функцией – возможность нарисовать фрактал с многопоточными фоновыми процессами. Это приведет к следующим улучшениям: во-первых, пользовательский интерфейс не зависнет во время рисования нового фрактала, а во-вторых, если ваш компьютер многоядерный, ваш рисунок будет нарисован намного быстрее. Вносить изменения будем, используя встроенную поддержку Swing для фоновых потоков.

**Задание:**

1. Вы должны создать подкласс SwingWorker под названием FractalWorker, который является внутренним классом FractalExplorer. Это самый простой способ написать этот код, так как ему потребуется доступ к нескольким внутренним членам FractalExplorer.
2. Класс FractalWorker будет отвечать за вычисление значений цвета для одного строки/ряда row фрактала, поэтому ему понадобятся два поля: целочисленная y-координата строки/ряда row, которая будет вычислена, и массив значений int для хранения вычисленных значений RGB Для каждого пикселя в этой строке
3. Метод done () вызывается, когда фоновая задача завершена, и этот метод вызывается из ядра обработки событий Swing. Это означает, что вы можете модифицировать компоненты Swing в соответствии с содержанием вашего ядра heart.
4. добавить поле «оставшиеся строки» в наш класс Fractal Explorer 31 и использовать его, чтобы узнать, когда будет завершена перерисовка. Мы будем читать и записывать это значение только из ядра обработки событий, чтобы мы никогда не вводили какие-либо параллельные обращения.
5. Создайте функцию void enableUI (boolean val), которая будет включать или отключать кнопки и выпадающий список вашего интерфейса на основе указанного значения. Для включения или отключения этих компонентов можно использовать метод Swing setEnabled (boolean).
6. В вашем рабочем методе done () уменьшите значение «rows rows» на 1 в качестве последнего шага этой операции. Затем, если после уменьшения оставшихся строк не осталось, вызовите enableUI (true).
7. измените имплементацию вашего приемника-мышки (mouselistener), чтобы немедленно возвращаться, если значение «rows rows» не равно нулю.

**Анализ предметной области и инструментарий:** для разработки программы обработки информации, поступающей от пользователя в циклах и ее вывода взят пакетjava development kit-9.0.4 с классом Scanner для организации ввода и стандартный редактор тхт.

**Код программы:**

Большинство элементов описано в предыдущей работе, описание новых элементов:

**Описание класса** **FractalExplorer:**

|  |  |
| --- | --- |
|  | import java.awt.BorderLayout; |
|  | import javax.imageio.ImageIO; |
|  | import java.awt.\*; |
|  | import java.awt.event.\*; |
|  | import javax.swing.\*; |
|  | import java.awt.event.\*; |
|  | import java.io.\*; |
|  | import javax.swing.filechooser.FileNameExtensionFilter; |
|  | public class FractalExplorer |
|  | { |
|  | public static void main(String[] args) |
|  | { |
|  | FractalExplorer MyExplorer = new FractalExplorer(500); |
|  | MyExplorer.createAndShowGUI(); |
|  | MyExplorer.drawFractal(); |
|  | } |
|  | private int MyDispalySize; |
|  | private int MySize; |
|  | private int RowsRem; |
|  | private JImageDisplay MyImage; |
|  | private FractalGenerator MyGenerator; |
|  | private FractalGenerator MyGeneratorT; |
|  | private FractalGenerator MyGeneratorM; |
|  | private FractalGenerator MyGeneratorB; |
|  | private Rectangle2D.Double MyRange= new Rectangle2D.Double(0,0,0,0); |
|  | private JComboBox Box =new JComboBox(); |
|  | private Button B=new Button("Reset"); |
|  | private Button S=new Button("Save"); |
|  | FractalExplorer(int NewSize) |
|  | { |
|  | MySize=NewSize; |
|  | MyImage = new JImageDisplay(NewSize,NewSize); |
|  | MyGenerator= new Tricorn(); |
|  | MyGeneratorT= new Tricorn(); |
|  | MyGeneratorM= new Mandelbrot(); |
|  | MyGeneratorB= new BurningShip(); |
|  | MyGenerator.getInitialRange(MyRange); |
|  | } |
|  | private void enableUI(boolean val) { //метод устанавливает возможность //взаимодействия с элементами управления через встроенный метод .swing setEnabled() |
|  | B.setEnabled(val); |
|  | S.setEnabled(val); |
|  | Box.setEnabled(val); |
|  | } |
|  | public void createAndShowGUI() |
|  | { |
|  | JFrame MyFrame =new JFrame(); |
|  | MyFrame.setLayout(new BorderLayout()); |
|  |
|  | JLabel MyLable=new JLabel("FRACTAL"); |
|  | JPanel MyPanel=new JPanel(); |
|  | JPanel MyPanel2=new JPanel(); |
|  | MyPanel.add(MyLable); |
|  | MyPanel.add(Box); |
|  | MyFrame.add(MyPanel, BorderLayout.NORTH); |
|  | Box.addItem(MyGeneratorT); |
|  | Box.addItem(MyGeneratorM); |
|  | Box.addItem(MyGeneratorB); |
|  | MyPanel2.add(B); |
|  | MyPanel2.add(S); |
|  | MyFrame.add(MyPanel2, BorderLayout.SOUTH); |
|  | MyFrame.add(MyImage, BorderLayout.CENTER); |
|  | MyFrame.setDefaultCloseOperation(MyFrame.EXIT\_ON\_CLOSE); |
|  | MyFrame.pack(); |
|  | MyFrame.setVisible(true); |
|  | MyEventListener handler2=new MyEventListener(); |
|  | MyFrame.setResizable(false); |
|  | MouseMyAdapter handler=new MouseMyAdapter(); |
|  | MyImage.addMouseListener(handler); |
|  | B.addActionListener(handler2); |
|  | S.addActionListener(handler2); |
|  | Box.addActionListener(handler2); |
|  | } |
|  | public void drawFractal() |
|  | { |
|  | enableUI(false); //блокировка управления |
|  | RowsRem=MySize; //счетчик окончания |
|  | for(int y=0;y<MySize;y++) //построчный запуск |
|  | { |
|  | FractalWorker MyWorker = new FractalWorker(y); |
|  | MyWorker.execute(); |
|  | } |
|  | } |
|  | public class MouseMyAdapter extends MouseAdapter |
|  | { |
|  | public void mouseClicked(MouseEvent e) |
|  | { |
|  | if (RowsRem!=0) |
|  | { |
|  | return; |
|  | } |
|  | MyGenerator.recenterAndZoomRange(MyRange,FractalGenerator.getCoord(MyRange.x,MyRange.x+MyRange.width,MySize,e.getX()),FractalGenerator.getCoord(MyRange.y,MyRange.y+MyRange.height,MySize,e.getY()),0.2); |
|  | drawFractal(); |
|  | } |
|  | } |
|  | public class MyEventListener implements ActionListener |
|  | { |
|  | public void actionPerformed(ActionEvent e) |
|  | { |
|  | if(e.getSource().equals(B)) |
|  | { |
|  | MyGenerator.getInitialRange(MyRange); |
|  | } |
|  | if(e.getSource().equals(S)) |
|  | { |
|  | JFileChooser ChooseFile = new JFileChooser(); |
|  | FileNameExtensionFilter FileFilter = new FileNameExtensionFilter("PNG Images","png"); |
|  | ChooseFile.setFileFilter(FileFilter); |
|  | ChooseFile.setAcceptAllFileFilterUsed(false); |
|  | if(ChooseFile.showSaveDialog(null)==JFileChooser.APPROVE\_OPTION) |
|  | { |
|  | try |
|  | { |
|  | File file = ChooseFile.getSelectedFile(); |
|  | String PathM = file.getPath(); |
|  | file=new File(PathM+ ".png"); |
|  | ImageIO.write(MyImage.getImage(),"png",file); |
|  | } |
|  | catch(IOException err) |
|  | { |
|  | JOptionPane.showMessageDialog(null,"error"); |
|  | } |
|  |  |
|  | } |
|  | } |
|  | if(e.getSource().equals(Box)) |
|  | { |
|  | String H=Box.getSelectedItem().toString(); |
|  | if(H.equals("Mandelbrot")) |
|  | { |
|  | MyGenerator=MyGeneratorM; |
|  | } |
|  | if(H.equals("Tricorn")) |
|  | { |
|  | MyGenerator=MyGeneratorT; |
|  | } |
|  | if(H.equals("BurningShip")) |
|  | { |
|  | MyGenerator=MyGeneratorB; |
|  | } |
|  | MyGenerator.getInitialRange(MyRange); |
|  | } |
|  | drawFractal(); |
|  | } |
|  | } |
|  | public class FractalWorker extends SwingWorker<Object,Object> |
|  | { |
|  | int y2; //координата строки по У |
|  | int[] S; //массив цветов строки |
|  | public FractalWorker(int a) //конст. инициализации |
|  | { |
|  | y2=a; сохранение номера строки |
|  | } |
|  | public Object doInBackground() |
|  | { //фоновый режим расчет цветов фрактала в строке как и прежде |
|  | S= new int[MySize]; |
|  | double Y = FractalGenerator.getCoord(MyRange.y,MyRange.y+MyRange.height,MySize,y2); |
|  | for(int x=0;x<MySize;x++) |
|  | { |
|  | double X = FractalGenerator.getCoord(MyRange.x,MyRange.x+MyRange.width,MySize,x); |
|  | int iteration = MyGenerator.numIterations(X,Y); |
|  | int ColoR=0; |
|  | if(iteration!=-1) |
|  | { |
|  | float hue = 0.7f + (float)iteration/200f; |
|  | ColoR = Color.HSBtoRGB(hue,1f,1f); |
|  | } |
|  | S[x]=ColoR; |
|  | } |
|  | return null; |
|  | } |
|  | public void done() |
|  | { //операции по завершении вычислений |
|  | for(int x=0;x<MySize;x++) |
|  | { |
|  | MyImage.drawPixel(x,y2,S[x]); //перерисовка фрактала |
|  | } |
|  | MyImage.repaint(0,0,y2,MySize,1); //обновление изображения |
|  | RowsRem--; //счетчик -1 |
|  | if (RowsRem<1) //если счетик обнулен |
|  | { |
|  | enableUI(true); //активация элем. управления |
|  | } |
|  | } |

|  |
| --- |
|  |
|  |

**Классы не подвергшиеся изменениям:**

FractalGenerator.java – обеспечивает обработку координат а также приближение.

Mandelbrot.java – обеспечивает расчет точек фрактала

Tricorn.java и BurningShip.java – производные от предыдущего имеют различные формулы расчета точек

**Выводы:** построены алгоритмы работы с объектами swing и работы со списками, сравнение элементов управления, обработки изображения, работы с интерфейсами и устройствами.