ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ

Ордена Трудового Красного Знамени

федеральное государственное бюджетное

образовательное учреждение высшего образования

«Московский Технический Университет Связи и Информатики» (МТУСИ)

Кафедра МКиИТ

Лабораторная работа по технологиям программирования №6

«Многоядерный исследователь/генератор фракталов»

Выполнил:

Студент 2 курса

Группы БСТ-1602

Пащенко Алексей

Вариант №12

Москва, 2018

Оглавление

[1. Цель и задачи. 3](#_Toc514608113)

[2. Анализ предметной области и выбор инструментария. 3](#_Toc514608114)

[3. Код программы и объяснение функций. 3](#_Toc514608115)

[3.1. Файл BurningShip.java 3](#_Toc514608116)

[3.2. Файл Tricorn.java 4](#_Toc514608117)

[3.3. Класс act\_listener 5](#_Toc514608118)

[4. Пример работы программы. 6](#_Toc514608119)

[5. Заключение. 6](#_Toc514608120)

1. Цель и задачи.

Оптимизировать программу, добавив ей возможность проводить расчёты не в один поток, а в несколько.

1. Анализ предметной области и выбор инструментария.

В текущей работе я использовал бесплатно распространяемый пакет разработчика Java Development Kit. Однако, этот пакет не предусматривает интегрированную среду разработки, поэтому для написания непосредственно кода мною была выбрана программа Visual Studio Code, так как она обеспечивает подсветку интерфейса и автоподстановку кода.

1. Код программы и объяснение функций.
   1. Файл FractalExplorer.java

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.awt.geom.Rectangle2D;

import javax.imageio.ImageIO;

import javax.swing.filechooser.\*;

import java.io.IOException;

import javax.swing.SwingWorker;

public class FractalExplorer {

private int remLines;

private JImageDisplay display;

private int dSize;

private FractalGenerator fGenerator;

private Rectangle2D.Double range = new Rectangle2D.Double(0, 0, 0, 0);

JComboBox box = new JComboBox();

JButton buttonR;

JButton buttonS;

// Create the Window

JFrame frame = new JFrame("Fractal Explorer");

public static void main(String[] args) {

FractalExplorer expl = new FractalExplorer(600);

expl.createAndShowGUI();

expl.drawFractal();

}

private void enableUI(boolean val){

buttonR.setEnabled(val);

buttonS.setEnabled(val);

box.setEnabled(val);

}

/\*\*

\* @param size - The height and width of the window

\*/

public FractalExplorer(int size) {

dSize = size;

display = new JImageDisplay(dSize, dSize);

fGenerator = new Mandelbrot();

fGenerator.getInitialRange(range);

}

public void createAndShowGUI() {

//Add fractal display

display.addMouseListener(new mouse\_listener());

frame.add(display, BorderLayout.CENTER);

//Add buttons panel

JPanel buttonsPanel = new JPanel();

//Add the reset buttons

buttonR = new JButton("Reset");

buttonR.addActionListener(new act\_listener());

buttonsPanel.add(buttonR, BorderLayout.CENTER);

//Add the save button

buttonS = new JButton("Save");

buttonS.addActionListener(new act\_listener());

buttonsPanel.add(buttonS, BorderLayout.CENTER);

frame.add(buttonsPanel, BorderLayout.SOUTH);

//Add the combobox

JLabel comboboxLabel = new JLabel("Select the fractal:");

JPanel comboboxPanel = new JPanel();

box.addItem(new Mandelbrot());

box.addItem(new Tricorn());

box.addItem(new BurningShip());

box.addActionListener(new act\_listener());

comboboxPanel.add(comboboxLabel, BorderLayout.CENTER);

comboboxPanel.add(box, BorderLayout.CENTER);

frame.add(comboboxPanel, BorderLayout.NORTH);

// Set the default close action to exit

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

//The installation default settings

frame.pack();

frame.setVisible(true);

frame.setResizable(false);

}

private class FractalWorker extends SwingWorker<Object, Object> {

int row;

int[] colors;

public FractalWorker(int y) {

row = y;

}

public Object doInBackground(){

colors = new int[dSize];

double yCoord = FractalGenerator.getCoord(range.y, range.y + range.height, dSize, row);

for (int i=0;i<dSize;i++){

double xCoord = FractalGenerator.getCoord(range.x, range.x + range.width, dSize, i);

// Compute the number of iterations

int iterations = fGenerator.numIterations(xCoord, yCoord);

//If the color is not default

if(iterations!=-1){

float hue = 0.7f + (float) iterations / 200f;

colors[i] = Color.HSBtoRGB(hue, 1f, 1f);

}

// If color is default

else colors[i]=0;

}

return null;

}

public void done(){

for (int i=0; i<dSize; i++)

display.drawPixel(i,row,colors[i]);

display.repaint(0,0,row,dSize,1);

remLines--;

if(remLines==0){

enableUI(true);

}

}

}

public void drawFractal() {

enableUI(false);

remLines=dSize;

for (int j = 0; j < dSize; j++) {

FractalWorker fWork = new FractalWorker(j);

fWork.execute();

}

// Display the image on the screen

display.repaint();

}

private class act\_listener implements ActionListener {

public void actionPerformed(ActionEvent e) {

String command = e.getActionCommand();

if (command.equals("Reset")) {

fGenerator.getInitialRange(range);

display.clearImage();

drawFractal();

}

if (command.equals("Save")) {

JFileChooser chooser = new JFileChooser();

FileFilter filter = new FileNameExtensionFilter("PNG Images", "png");

chooser.setFileFilter(filter);

chooser.setAcceptAllFileFilterUsed(false);

if (chooser.showSaveDialog(frame) == JFileChooser.APPROVE\_OPTION) {

try {

ImageIO.write(display.getImage(), "png", chooser.getSelectedFile());

} catch (IOException ex) {

JOptionPane.showMessageDialog(frame, ex.getMessage(), "Unable to save image",

JOptionPane.ERROR\_MESSAGE);

}

} else {

return;

}

}

if ((JComboBox) e.getSource() == box) {

fGenerator = (FractalGenerator) box.getSelectedItem();

fGenerator.getInitialRange(range);

display.clearImage();

drawFractal();

}

}

}

private class mouse\_listener extends MouseAdapter {

public void mouseClicked(MouseEvent e) {

if(remLines>0)

return;

// Get mouse's coordinates

super.mouseClicked(e);

int mouseX = e.getX();

int mouseY = e.getY();

double xCoord = FractalGenerator.getCoord(range.x, range.x + range.width, dSize, mouseX);

double yCoord = FractalGenerator.getCoord(range.y, range.y + range.height, dSize, mouseY);

fGenerator.recenterAndZoomRange(range, xCoord, yCoord, 0.5);

display.clearImage();

drawFractal();

}

}

}

В ходе этой работы был изменён файл FractalExplorer.java. Из новшеств, добавлен класс FractalWorker, который отвечает за прорисовку одной строки фрактала, и изменено несколько отдельных строк кода для интеграции этого класса. То есть, теперь фрактал прорисовывается не сразу целиком, а построчно, что позволяет задействовать сразу несколько потоков в прорисовке одного изображения. Такой подход значительно уменьшает время, требуемое компьютеру для расчётов.

1. Заключение.

В ходе работы была модифицирована уже имеющаяся программа. Теперь она производит вычисления с использованием многопоточности, что значительно увеличивает производительность.