**Федеральное агентство связи**

**Ордена трудового Красного Знамени федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

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

Кафедра Математическая кибернетика и информационные технологии

Лабораторная работа № 6

Выполнил: студент группы БФИ2002

Ковачев В.Е.

Проверила:

Москва, 2021

**Задание:**

В данной лабораторной работе необходимо будет реализовать возможность рисования фрактала с несколькими фоновыми потоками. Два преимущества данного подхода: первое - пользовательский интерфейс не будет зависать в процессе рисования нового фрактала, а второе - если у вас компьютер многоядерный, то процесс рисования будет намного быстрее.

• doInBackground() - метод, который фактически выполняет фоновые операции. Swing вызывает этот метод в фоновом потоке, а не в потоке обработке событий.

• done() - этот метод вызывается, когда фоновая задача завершена. Он вызывается в потоке обработки событий, поэтому данному методу разрешено взаимодействовать с пользовательским интерфейсом.

Класс SwingWorker имеет запутанную спецификацию, на самом деле это SwingWorker . Тип T - это тип значения, возвращаемого функцией doInBackground(), когда задача полностью выполнена. Тип V используется, когда фоновая задача возвращает промежуточные значения во время выполнения; эти промежуточные значения будут доступны при использовании методов publish () и process (). Оба типа могут не использоваться, в таких случаях необходимо указать Object для неиспользуемого типа.

Рисование в фоновом режиме

В данной лабораторной работе в основном необходимо будет работать в классе FractalExplorer. Часть кода будет новой, но некоторые части будут представлять из себя модифицированный код, который вы уже написали.

1) Создайте подкласс SwingWorker с именем FractalWorker, который будет внутренним классом FractalExplorer. Это необходимо для того, чтобы у него был доступ к нескольким внутренним членам FractalExplorer. Помните, что класс SwingWorker является универсальным, поэтому нужно указать параметры - можно просто указать Object для двух параметров, потому что в данной реализации эти параметры не будут использоваться. В результате у вас должна получиться следующая строчка кода:

private class FractalWorker extends SwingWorker

2) Класс FractalWorker будет отвечать за вычисление значений цвета для одной строки фрактала, поэтому ему потребуются два поля: целочисленная yкоордината вычисляемой строки, и массив чисел типа int для хранения вычисленных значений RGB для каждого пикселя в этой строке. Конструктор должен будет получать y-координату в качестве параметра и сохранять это. (На данном этапе не надо выделять память под целочисленный массив, так как он не потребуется, пока строка не будет вычислена.)

3) Метод doInBackground() вызывается в фоновом потоке и отвечает за выполнение длительной задачи. Поэтому в вашей реализации вам нужно будет взять часть кода из вашей предыдущей функции «draw fractal» и поместить ее в этот метод. Вместо того, чтобы рисовать изображение в окне, цикл должен будет сохранить каждое значение RGB в соответствующем элементе целочисленного массива. Вы не сможете изменять отображение из этого потока, потому что вы нарушите ограничения ограничения потоков Swing.

4) Вместо этого выделите память для массив целых чисел в начале реализации этого метода (массив должен быть достаточно большим для хранения целой строки значений цвета), а затем сохраните цвет каждого пикселя в этом массиве. Единственные различия между настоящим и предыдущим кодом в том, что вам нужно будет вычислить фрактал для указанной строки, и что вы на данном этапе не обновляете отображение. Метод doInBackground() должен возвращать объект типа Object, так как это указано в объявлении SwingWorker . Просто верните null.

5) Метод done() вызывается, когда фоновая задача завершена, и этот метод вызывается из потока обработки событий Swing. Это означает, что вы можете модифицировать компоненты Swing на ваш вкус. Поэтому в этом методе вы можете перебирать массив строк данных, рисуя пиксели, которые были вычислены в doInBackground ().

После того, как строка будет вычислена, вам нужно будет сообщить Swing, перерисовать часть изображения, которая была изменена. Поскольку вы изменили только одну строку, перерисовывать изображение целиком будет затратно, поэтому вы можете использовать метод JComponent.repaint(), который позволит вам указать область для перерисовки. У данного метода есть неиспользуемый параметр типа long, вы можете просто указать 0 для этого аргумента. В качестве остальных параметров укажите вычисленную строку, значения начала фрагмента для перерисовки (0, y) и конечные значения фрагмента (displaySize, 1). После того, как вы завершили класс для фоновой задачи, следующим шагом нужно будет привязать его к процессу рисования фракталов. Так как часть кода из функции «draw fractal» уже задействована в разрабатываемом классе, на данном этапе можно изменить функцию «draw fractal», а именно, для каждой строки в отображении создать отдельный рабочий объект, а затем вызвать для него метод execute (). Это действие запустит фоновый поток и запустит задачу в фоновом режиме. Помните, что класс FractalWorker отвечает за генерацию данных строки и за рисование этой строки, поэтому функция «draw fractal» должна быть простой.

Игнорирование событий во время перерисовки

Самый простой способ решить проблему игнорирования событий во время перерисовки - отслеживать количество оставшихся строк, которые должны быть завершены, и игнорировать или отключать взаимодействие с пользователем до тех пор, пока не будут нарисованы все строки. Для нужно добавить поле «rows remaining» в класс Fractal Explorer и использовать его, чтобы узнать, когда будет завершена перерисовка. Чтение и запись этого значения будет происходить в потоке обработки событий, чтобы не было параллельного доступа к этому элементу. Если взаимодействие с ресурсом будет происходить только из одного потока, то не возникнет ошибок параллелизма. Для этого:

• Создайте функцию void enableUI(boolean val), которая будет включать или отключать кнопки с выпадающим списком в пользовательском интерфейсе на основе указанного параметра. Для включения или отключения этих компонентов можно использовать метод Swing setEnabled(boolean). Убедитесь, что ваш метод обновляет состояние кнопки сохранения, кнопки сброса и выпадающего списка.

• Функция «draw fractal» должна сделать еще две вещи. Первая - она должна вызвать метод enableUI (false), чтобы отключить все элементы пользовательского интерфейса во время рисования. Вторая - она должна установить значение «rows remaining» равным общему количеству строк, которые нужно нарисовать. Эти действия должны быть сделаны перед выполнением каких-либо рабочих задач, иначе это может привести к некорректной работе алгоритма.

• В методе done(), уменьшите значение «rows remaining» на 1, как последний шаг данной операции. Затем, если после уменьшения значение «rows remaining» равно 0, вызовите метод enableUI (true).

• Наконец, измените реализацию mouse-listener для того, чтобы она сразу возвращалась в предыдущее состояние, если значение «rows remaining» не равно нулю. Другими словами, приложение будет реагировать на щелчки мышью, только в том случае, если больше нет строк, которые должны быть нарисованы. (Обратите внимание, что также не нужно вносить аналогичные изменения в обработчике событий, потому что все эти компоненты будут отключены с помощью метода enableUI .

**Ход работы:**

**Для реализации многопоточности был расширен уже существующий класс FractalExplorer:**

import javax.swing.\*;

import java.awt.\*;

import java.io.IOException;

import javax.imageio.ImageIO;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import javax.swing.filechooser.FileNameExtensionFilter;

import java.awt.geom.Rectangle2D;

public class FractalExplorer {

private int displaySize;

private JImageDisplay imageDisplay;

private FractalGenerator fractalGenerator;

private Rectangle2D.Double range;

private JComboBox comboBox;

private int rowsRemaining;

private JButton buttonReset;

private JButton buttonSave;

private FractalExplorer (int displaySize) {

this.displaySize = displaySize;

this.fractalGenerator = new Mandelbrot();

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

fractalGenerator.getInitialRange(this.range);

}

**// точка входа**

public static void main(String[] args) {

FractalExplorer fractalExplorer = new FractalExplorer(600);

fractalExplorer.setGUI();

fractalExplorer.drawFractal();

}

**// задание интерфейса**

public void setGUI() {

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

JPanel jPanel\_1 = new JPanel();

JPanel jPanel\_2 = new JPanel();

JLabel label = new JLabel("Fractal:");

imageDisplay = new JImageDisplay(displaySize, displaySize);

imageDisplay.addMouseListener(new MouseListener());

**// выпадающий список**

comboBox = new JComboBox();

comboBox.addItem(new Mandelbrot());

comboBox.addItem(new Tricorn());

comboBox.addItem(new BurningShip());

comboBox.addActionListener(new ActionHandler());

**// кнопка reset**

buttonReset = new JButton("Reset");

buttonReset.setActionCommand("Reset");

buttonReset.addActionListener(new ActionHandler());

**// кнопка сохранить**

buttonSave = new JButton("Save image");

buttonSave.setActionCommand("Save");

buttonSave.addActionListener(new ActionHandler());

jPanel\_1.add(label, BorderLayout.CENTER);

jPanel\_1.add(comboBox, BorderLayout.CENTER);

jPanel\_2.add(buttonReset, BorderLayout.CENTER);

jPanel\_2.add(buttonSave, BorderLayout.CENTER);

frame.setLayout(new BorderLayout());

frame.add(imageDisplay, BorderLayout.CENTER);

frame.add(jPanel\_1, BorderLayout.NORTH);

frame.add(jPanel\_2, BorderLayout.SOUTH);

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

frame.pack();

frame.setVisible(true);

frame.setResizable(false);

}

**// отрисовка фрактала в JImageDisplay**

private void drawFractal() {

**// отключаем интерфейс на момент рисования**

enableGUI(false);

rowsRemaining = displaySize;

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

FractalWorker drawRow = new FractalWorker(i);

drawRow.execute();

}

}

**// включение - отключение gui**

public void enableGUI(boolean b) {

buttonSave.setEnabled(b);

buttonReset.setEnabled(b);

comboBox.setEnabled(b);

}

**// обработчик кнопок**

public class ActionHandler implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

if (e.getActionCommand().equals("Reset")) {

**// перерисовка фрактала**

fractalGenerator.getInitialRange(range);

drawFractal();

} else if (e.getActionCommand().equals("Save")) {

**// сохранение**

JFileChooser fileChooser = new JFileChooser();

FileNameExtensionFilter fileFilter = new FileNameExtensionFilter("PNG Images", "png");

fileChooser.setFileFilter(fileFilter);

fileChooser.setAcceptAllFileFilterUsed(false);

int t = fileChooser.showSaveDialog(imageDisplay);

if (t == JFileChooser.APPROVE\_OPTION) {

try {

ImageIO.write(imageDisplay.getImage(), "png", fileChooser.getSelectedFile());

} catch (NullPointerException | IOException ee) {

JOptionPane.showMessageDialog(imageDisplay, ee.getMessage(), "Cannot save image", JOptionPane.ERROR\_MESSAGE);

}

}

} else {

fractalGenerator = (FractalGenerator) comboBox.getSelectedItem();

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

fractalGenerator.getInitialRange(range);

drawFractal();

}

}

}

public class MouseListener extends MouseAdapter {

@Override

public void mouseClicked(MouseEvent e) {

double x = FractalGenerator.getCoord(range.x, range.x + range.width, displaySize, e.getX());

double y = FractalGenerator.getCoord(range.y, range.y + range.width, displaySize, e.getY());

fractalGenerator.recenterAndZoomRange(range, x, y, 0.5);

drawFractal();

}

}

public class FractalWorker extends SwingWorker<Object, Object> {

private int y\_coord;

private int[] rgb;

public FractalWorker(int y\_coord) {

this.y\_coord = y\_coord;

}

@Override

protected Object doInBackground() throws Exception {

rgb = new int[displaySize];

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

int count = fractalGenerator.numIterations(FractalGenerator.getCoord(range.x, range.x + range.width, displaySize, i),

FractalGenerator.getCoord(range.y, range.y+range.width, displaySize, y\_coord));

if (count == -1)

rgb[i] = 0;

else {

double hue = 0.7f + (float) count / 200f;

int rgbColor = Color.HSBtoRGB((float) hue, 1f, 1f);

rgb[i] = rgbColor;

}

}

return null;

}

@Override

protected void done() {

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

imageDisplay.drawPixel(i, y\_coord, rgb[i]);

}

imageDisplay.repaint(0,0,y\_coord,displaySize,1);

rowsRemaining--;

if (rowsRemaining == 0)

enableGUI(true);

}

}

}

**Результат работы программы**

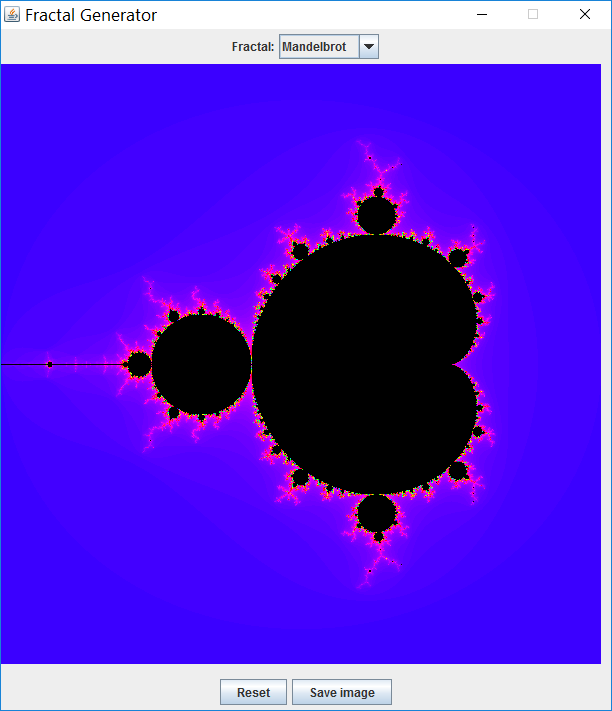


Рисунок №1- Фрактал Mandellbrot

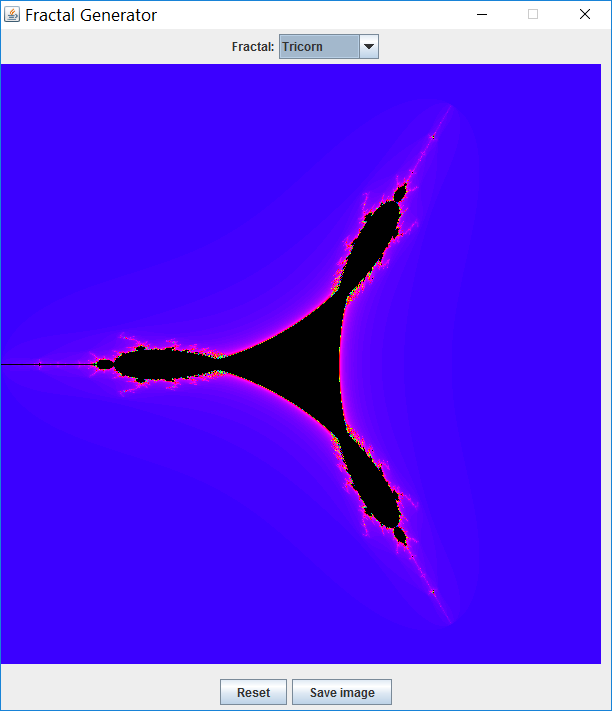


Рисунок №2- Фрактал Tricorn

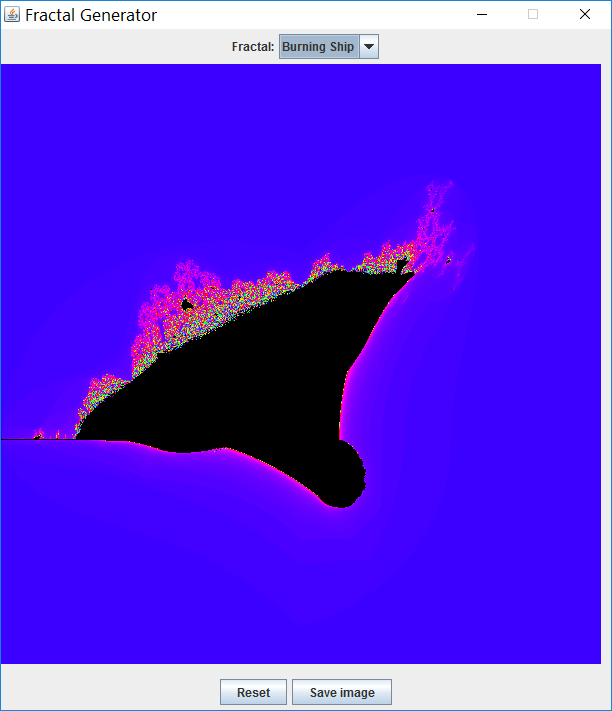


Рисунок №3- Фрактал Burning Ship

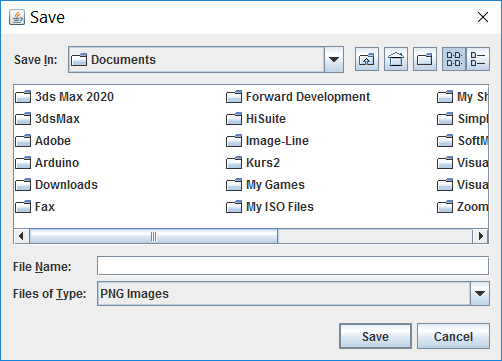


Рисунок №3- Диалоговое окно для сохранения фрактала

**Вывод**

В данной лабораторной работе мы изучили использование многопоточной обработки данных в фоновом режиме на языке java.

**Список использованной литературы**

1. Камаев В.А., Костерин В.В. Технологии программирования. М.: Высшая школа,

2006.

2. Жоголев Е.А.Технология программирования. – М.: Научный мир, 2004.