Министерство Цифрового Развития, Связи и Массовых Коммуникаций Российской Федерации Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования   
“Московский технический университет связи и информатики”

Кафедра «Информационные технологии»

Лабораторная работа №6:   
«Многопоточный генератор фракталов**»**

Выполнил: студент гр. БПЗ1902

Кварацхелия Д.Т.

Проверил: Харрасов К.Р.

Москва 2021 г.

Цель работы:

В данной лабораторной работе необходимо будет реализовать возможность рисования фрактала с несколькими фоновыми потоками.

Задание:

В этой лабораторной работе нужно изменить программу так, чтобы она использовала один или несколько фоновых потоков для вычисления фрактала.

Выполнение работы:

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 уже написан.

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. В результате должна получиться следующая строчка кода: private class FractalWorker extends SwingWorker

public class FractalWorker extends SwingWorker<Object, Object> {

        private int y\_coord;

        private int[] rgb;

        public FractalWorker(int y\_coord) {

            this.y\_coord = y\_coord;

        }

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.

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;

        }

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 ().

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)

                enableUI(true);

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 ().

// отрисовка фрактала в JImageDisplay

    private void drawFractal() {

        // отключаем интерфейс на момент рисования

        enableUI(false);

        rowsRemaining = displaySize;

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

            FractalWorker drawRow = new FractalWorker(i);

            drawRow.execute();

        }

    }

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.

• 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

 // включение - отключение

    public void enableUI(boolean b) {

        buttonSave.setEnabled(b);

        buttonReset.setEnabled(b);

        comboBox.setEnabled(b);

    }

• 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.

// отрисовка фрактала в JImageDisplay

    private void drawFractal() {

        // отключаем интерфейс на момент рисования

        enableUI(false);

        rowsRemaining = displaySize;

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

            FractalWorker drawRow = new FractalWorker(i);

            drawRow.execute();

        }

    }

• 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).

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)

                enableUI(true);

После выполнения данных шагов, должна получиться программа для рисования фракталов с несколькими потоками и, которая запретит действия пользователя, пока процесс рендеринга происходит в фоновом режиме.

