Лабораторная работа №6. Многопоточный генератор фракталов

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. Несмотря на то, что многопоточное программирование может быть очень сложным, процесс изменения приложения будет прост, благодаря встроенной поддержке Swing фоновых потоков.

На данный момент приложение генератора фракталов выполнялось в одном потоке (Event-Dispatch Thread). Это поток, который обрабатывает все события Swing, такие как нажатие кнопок, перерисовка и т.д. Поэтому разработанный пользовательский интерфейс зависает во время вычисления фрактала; так как вычисление выполняется в потоке обработки событий, возникающие события не могут быть обработаны до завершения вычисления.

В этой лабораторной работе нужно изменить программу так, чтобы она использовала один или несколько фоновых потоков для вычисления фрактала. В частности, поток обработки событий не будет использоваться для вычисления фрактала. Теперь, если вычисление будет выполняться несколькими потоками, необходимо будет разбить его на несколько независимых частей. Например, при рисовании фракталов можно дать каждому потоку по одной строке фрактала для вычисления. Сложность заключается в том, что необходимо соблюдать важное ограничение Swing, а именно, что в потоке обработке событий происходит взаимодействие только с компонентами Swing. Для этого Swing предоставляет инструменты, что упрощает поставленную задачу.

В программировании часто возникают подобные проблемы, что через пользовательский интерфейс запускают длительную операцию, и данная операция должна выполняться в фоновом режиме для сохранения

работоспособности пользовательского интерфейса. Наиболее яркий пример — веб-браузеры; пока страница загружается и отображается, у пользователя должна быть возможность отменить операцию, щелкнуть ссылку или выполнить любое другое действие. Для такого рода проблем Swing предоставляет класс javax.swing.SwingWorker, который облегчает процесс организации фонового потока. SwingWorker - абстрактный класс, включающий в себя следующие важные методы:

- 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 <T, V>. 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<Object, Object>

2) Класс FractalWorker будет отвечать за вычисление значений цвета для одной строки фрактала, поэтому ему потребуются два поля: целочисленная y-координата 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 <T, V>. 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.

После завершения данной функции, вы сможете заметить отображения стало более быстрым, а пользовательский интерфейс стал более отзывчивым.

Также вы сможете заметите одну проблему в вашем пользовательском интерфейсе - если вы нажмете на экран или на кнопку во время перерисовки, программа обработает это событие, хотя оно должно быть проигнорировано до завершения операции.

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 ().)

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.