Лабораторная работа № 4: Рисование фракталов

В следующих нескольких лабораторных работ вы создадите небольшое JAVA-приложение, которое сможет рисовать фракталы. Если вы никогда не играли с фракталами раньше, вы будете удивлены тем, как просто можно создать красивые изображения. Это будет сделано с помощью фреймворка Swing и Java API, который позволяет создавать графические пользовательские интерфейсы.

Начальная версия приложения будет довольно проста, но в следующих лабораторных работах будут добавлены некоторые полезные функции такие как, как сохранение сгенерированных изображений, возможность переключения между различными видами фракталов. И графический интерфейс (GUI), и механизм для поддержки различных фракталов будут зависеть от иерархий классов.

Вот простой пример графического интерфейса в его начальном состоянии:

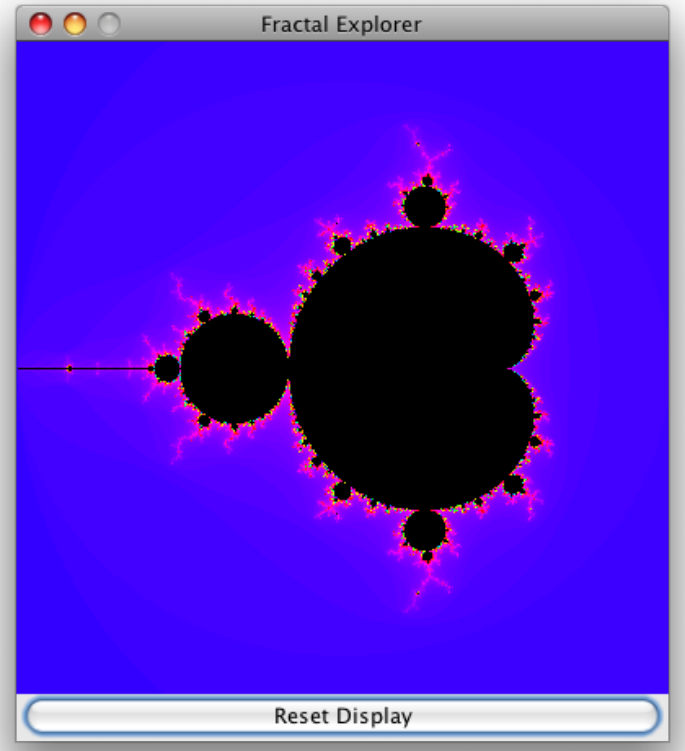

Рисунок 5.2. Пример графического интерфейса

И, вот некоторые интересные области фрактала: слоны и морские коньки!

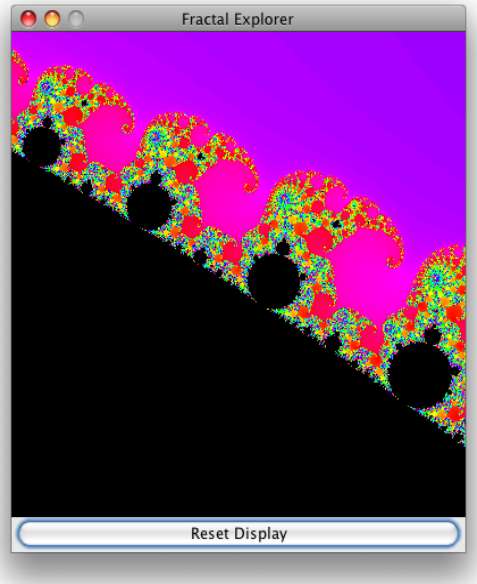

Рисунок 5.3. Фрактал «Слоны»

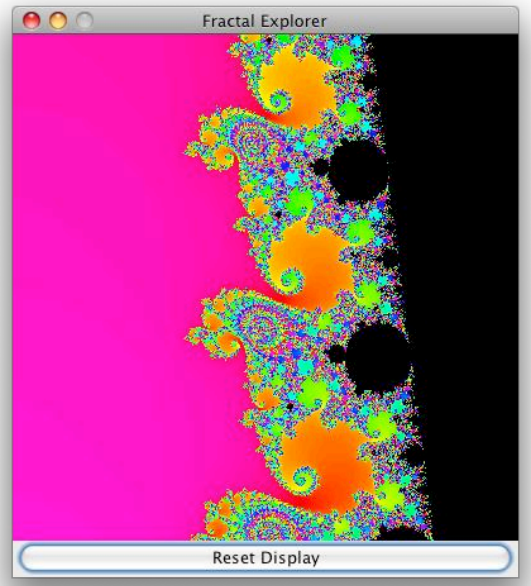

Рисунок 5.4. Фрактал «Морские коньки»

Создание пользовательского интерфейса

Прежде чем рисовать фракталы, необходимо создать графический виджет, который будет их отображать. Swing не предоставляет такой компонент, но его очень просто создать самостоятельно. Обратите внимание на то, что в этой лабораторной работе мы будем использовать широкий спектр классов Java AWT и Swing, детали которых здесь будут не раскрыты. Для более подробной информации вы можете воспользоваться онлайн-документами по API Java. Просто перейдите к пакету данного класса Java, выберите сам класс, а затем прочтите подробную информацию о том, как использовать класс.

- Создайте класс JImageDisplay, производный от javax.swing.JComponent. Класса должен иметь одно поле с типом доступа private, экземпляп java.awt.image.BufferedImage. Класс BufferedImage управляет изображением, содержимое которого можно записать.
- Конструктор JImageDisplay должен принимать целочисленные значения ширины и высоты, и инициализировать объект BufferedImage новым изображением с этой шириной и высотой, и типом изображения TYPE_INT_RGB. Тип определяет, как цвета каждого пикселя будут

представлены в изображении; значение TYPE_INT_RGB обозначает, что красные, зеленые и синие компоненты имеют по 8 битов, представленные в формате int в указанном порядке.

Конструктор также должен вызвать метод setPreferredSize() родительского класса метод с указанной шириной и высотой. (Вы должны будете передать эти значения в объект java.awt.Dimension) Таким образом, когда ваш компонент будет включен в пользовательский интерфейс, он отобразит на экране все изображение.

- Пользовательские компоненты Swing должны предоставлять свой собственный код для отрисовки, переопределяя защищенный метод JComponent paintComponent (Graphics g). Так как наш компонент просто выводит на экран данные изображения, реализация будет очень проста! Во-первых, нужно всегда вызывать метод суперкласса paintComponent (g) так, чтобы объекты отображались правильно. После вызова версии суперкласса, вы можете нарисовать изображение в компоненте, используя следующую операцию:

g.drawImage (image, 0, 0, image.getWidth(), image.getHeight(), null);

(Мы передаем значение null для ImageObserver, поскольку данная функциональность не требуется.)

- Вы также должны создать два метода с доступом public для записи данных в изображение: метод clearImage (), который устанавливает все пиксели изображения в черный цвет (значение RGB 0), и метод drawPixel (int x, int y, int rgbColor), который устанавливает пиксель в определенный цвет. Оба метода будут необходимы для использования в методе setRGB () класса BufferedImage.

Не забывайте про комментарии!

Вычисления фрактала Мандельброта

Следующая ваша задача: написать код для вычисления фрактала Мандельброта.

Для создания фракталов используйте следующий исходный файл FractalGenerator.java, от которого будут унаследованы все ваши фрактальные

генераторы. Как вы могли заметить данный файл предоставляет также некоторые полезные операции для перевода из экранных координат в систему координат вычисляемого фрактала.

Виды фракталов, с которыми нужно будет работать, вычисляются в комплексном виде и включают в себя простые математические функции, которые выполняются многократно, пока не выполнится определенное условие. Функция для фрактала Мандельброта имеет вид: $z_n = z_{n-1}^2 + c$, где все значения — это комплексные числа, $z_0 = 0$, и c - определенная точка фрактала, которую мы отображаем на экране. Вычисления повторяются до тех пор, пока $|z| > 2$ (в данной ситуации точка находится не во множестве Мандельброта), или пока число итераций не достигнет максимального значения, например, 2000 (в этом случае делается предположение, что точка находится в наборе).

Процесс построения фрактала Мандельброта прост: необходимо перебрать все пиксели изображения, рассчитать количество итераций для соответствующей координаты, и затем установить пиксель в цвет, основанный на количестве рассчитанных итераций. Все это будет сделано позже, на данном этапе необходимо реализовать приведенные выше вычисления.

- Создайте подкласс FractalGenerator с именем Mandelbrot. в нем вам необходимо будет обеспечить только два метода: getInitialRange() и numIterations().
- getInitialRange (Rectangle2D.Double) - метод позволяет генератору фракталов определить наиболее «интересную» область комплексной плоскости для конкретного фрактала. Обратите внимание на то, что методу в качестве аргумента передается прямоугольный объект, и метод должен изменить поля прямоугольника для отображения правильного начального диапазона для фрактала. (Пример можно увидеть в методе FractalGenerator.recenterAndZoomRange().) В классе Mandelbrot этот метод должен установить начальный диапазон в *(-2 - 1.5i) - (1 + 1.5i)*. Т.е. значения x и y будут равны -2 и -1.5 соответственно, а ширина и высота будут равны 3.

- Метод numIterations(double, double) реализует итеративную функцию для фрактала Мандельброта. Константу с максимальным количеством итераций можно определить следующим образом:

public static final int MAX_ITERATIONS = 2000;

Затем вы сможете ссылаться на эту переменную в вашей реализации.

Обратите внимание на то, что у Java нет подходящего типа данных для комплексных чисел, поэтому необходимо будет реализовать итеративную функцию, используя отдельные переменные для действительной и мнимой частей. (Вы можете реализовать отдельный класс для комплексных чисел.) Ваш алгоритм должен обладать быстродействием, например, не стоит сравнивать $|z|$ с 2; сравните $|z|^2$ с 2^2 для того, чтобы избежать сложных и медленных вычислений квадратного корня. Также не стоит использовать метод Math.pow () для вычисления небольших степеней, лучше перемножьте значение, иначе ваш быстродействие вашего кода сильно упадет.

В случае, если алгоритм дошел до значения MAX_ITERATIONS нужно вернуть -1, чтобы показать, что точка не выходит за границы.

Ваши задачи

Создайте класс FractalExplorer, который позволит вам исследовать различные области фрактала, путем его создания, отображения через графический интерфейс Swing и обработки событий, вызванных взаимодействием приложения с пользователем.

Как видно их приведенных выше изображений пользовательского интерфейса, FractalExplorer очень прост, он состоит из JFrame, который в свою очередь содержит объект JImageDisplay, который отображает фрактал,и объект JButton для сброса изображения, необходимый для отображения целого фрактала. Данный макет можно создать, установив для фрейма BorderLayout, затем поместив отображение в центр макета и кнопку сброса в "южной" части макета.

- Класс FractalExplorer должен отслеживать несколько важных полей для состояния программы:

1) Целое число «размер экрана», которое является шириной и высотой отображения в пикселях. (Отображение фрактала будет квадратным.)

2) Ссылка JImageDisplay, для обновления отображения в разных методах в процессе вычисления фрактала.

3) Объект FractalGenerator. Будет использоваться ссылка на базовый класс для отображения других видов фракталов в будущем.

4) Объект Rectangle2D.Double, указывающий диапазона комплексной плоскости, которая выводится на экран.

Все вышеприведенные поля будут иметь тип доступа private.

- У класса должен быть конструктор, который принимает значение размера отображения в качестве аргумента, затем сохраняет это значение в соответствующем поле, а также инициализирует объекты диапазона и фрактального генератора. Данный конструктор не должен устанавливать какие-либо компоненты Swing; они будут установлены в следующем методе.

- Создайте метод createAndShowGUI (), который инициализирует графический интерфейс Swing: JFrame, содержащий объект JimageDisplay, и кнопку для сброса отображения. Используйте java.awt.BorderLayout для содержимого окна; добавьте объект отображения изображения в позицию BorderLayout.CENTER и кнопку в позицию BorderLayout.SOUTH.

Вам необходимо дать окну подходящий заголовок и обеспечить операцию закрытия окна по умолчанию (см. метод JFrame.setDefaultCloseOperation ()).

После того, как компоненты пользовательского интерфейса инициализированы и размещены, добавьте следующую последовательность операций:

```
frame.pack ();
frame.setVisible (true);
```

frame.setResizable (false);

Данные операции правильно разметят содержимое окна, сделают его видимым (окна первоначально не отображаются при их создании для того, чтобы можно было сконфигурировать их прежде, чем выводить на экран), и затем запретят изменение размеров окна.

- Реализуйте вспомогательный метод с типом доступа private для вывода на экран фрактала, можете дать ему имя drawFractal (). Этот метод должен циклически проходить через каждый пиксель в отображении (т.е. значения x и y будут меняться от 0 до размера отображения), и сделайте следующее:

  - Вычислите количество итераций для соответствующих координат в области отображения фрактала. Вы можете определить координаты с плавающей точкой для определенного набора координат пикселей, используя вспомогательный метод FractalGenerator.getCoord (); например, чтобы получить координату x, соответствующую координате пикселя X, сделайте следующее:

```
//x - пиксельная координата; xCoord - координата в пространстве фрактала
double xCoord = FractalGenerator.getCoord (range.x, range.x + range.width, displaySize, x);
```

  - Если число итераций равно -1 (т.е. точка не выходит за границы, установите пиксель в черный цвет (для rgb значение 0). Иначе выберите значение цвета, основанное на количестве итераций. Можно также для этого использовать цветовое пространство HSV: поскольку значение цвета варьируется от 0 до 1, получается плавная последовательность цветов от красного к желтому, зеленому, синему, фиолетовому и затем обратно к красному! Для этого вы можете использовать следующий фрагмент:

```
float hue = 0.7f + (float) numIters / 200f;
int rgbColor = Color.HSBtoRGB(hue, 1f, 1f);
```

Если вы придумали другой способ отображения пикселей в зависимости от количества итераций, попробуйте реализовать его!

- Отображение необходимо обновлять в соответствии с цветом для каждого пикселя.
- После того, как вы закончили отрисовывать все пиксели, вам необходимо обновить JimageDisplay в соответствии с текущим изображением. Для этого вызовите функцию repaint() для компонента. В случае, если вы не воспользуетесь данным методом, изображение на экране не будет обновляться!

- Создайте внутренний класс для обработки событий java.awt.event.ActionListener от кнопки сброса. Обработчик должен сбросить диапазон к начальному, определенному генератором, а затем перерисовать фрактал.

После того, как вы создали этот класс, обновите метод createAndShowGUI ().

- Создайте другой внутренний класс для обработки событий java.awt.event.MouseListener с дисплея. Вам необходимо обработать события от мыши, поэтому вы должны унаследовать этот внутренний класс от класса MouseAdapterAWT. При получении события о щелчке мышью, класс должен отобразить пиксельные коорината щелчка в область фрактала, а затем вызвать метод генератора recenterAndZoomRange() с координатами, по которым щелкнули, и масштабом 0.5. Таким образом, нажимая на какое-либо место на фрактальном отображении, вы увеличиваете его!

Не забывайте перерисовывать фрактал после того, как вы меняете область фрактала.

Далее обновите метод createAndShowGUI (), чтобы зарегистрировать экземпляр этого обработчика в компоненте фрактального отображения.

- В заключении, вам необходимо создать статический метод main() для FractalExplorer так, чтобы можно было его запустить. В main необходимо будет сделать:

- Инициализировать новый экземпляр класса FractalExplorer с размером отображения 800.
- Вызовите метод createAndShowGUI () класса FractalExplorer.
- Вызовите метод drawFractal() класса FractalExplorer для отображения начального представления.

После выполнения приведенных выше действий, вы сможете детально рассмотреть фрактал Мандельброта. Если вы увеличите масштаб, то вы можете столкнуться с двумя проблемами:

- Во-первых, вы сможете заметить, что в конечном итоге уровень детализации заканчивается; это вызвано тем, что в таком случае необходимо более 2000 итераций для поиска точки во множестве Мандельброта! Можно увеличить максимальное количество итераций, но это приведет к замедлению работы алгоритма.
- Во-вторых, при сильном увеличении масштаба, вы столкнетесь с пиксельным выводом отображения! Это вызвано тем, что вы работаете в пределе того, что могут предоставить значения с плавающей запятой с двойной точностью.

При рисовании фрактала экран ненадолго зависает. Следующая лабораторная работа будет направлена на решение данной проблемы.