Fractal Generator

General Introduction

The challenge exercises are intended for students who are done with the regular weekly homework exercises and seek an extra challenge. Only those who passed the challenge program entry test will be considered participants of the challenge program.

Your submission for all the parts of the exercise should be in Java. All the challenge exercises must be done individually. Feel free to send an e-mail to course2ip90+CP@gmail.com if you have any questions.

Assignment description

A fractal can be defined as an iterated, infinitely self-similar, and detailed mathematical construct and can also be seen as a form of art. Fractals are not necessarily limited on geometric patterns, and can also describe processes over time. Fractal patterns have been rendered and studied in multiple forms such as in images, structures (e.g. a snowflake), and sounds, but can be found everywhere in nature as you can see in Figure 1.

(a) Seoul, South Korea (b) Kuching, Malaysia

Figure 1: Fractals can also be found everywhere in nature

In this assignment we are going to focus mostly on the mathematical concept, in order to construct a java program capable of displaying the fractal art given a set of parameters. Throughout the assignment you will need to do your own research regarding the different kinds of fractals, and how they are constructed.

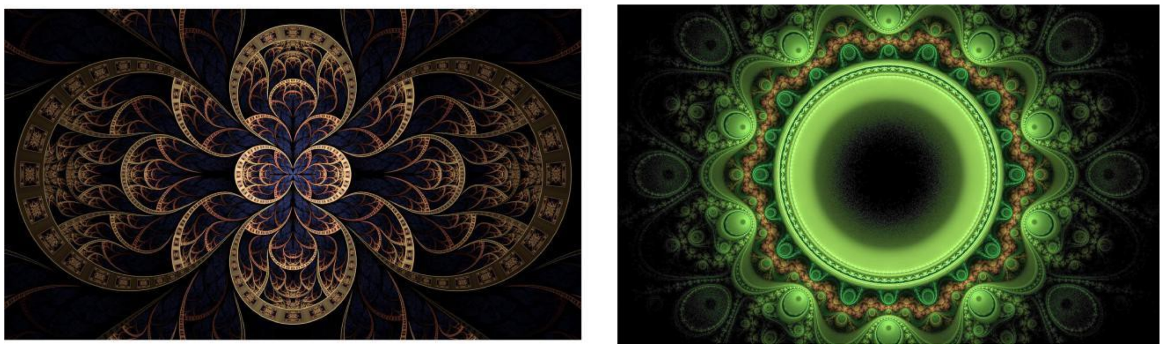

(a) Black-golden abstract eastern themed design. (b) Fluorescent green-black vector design.

Figure 2: More artistic fractals that are digitally generated. (beautified with multiple/gradient colours)

In this challenge we are going to develop a program that generates Fractals mathematically.

Exercise

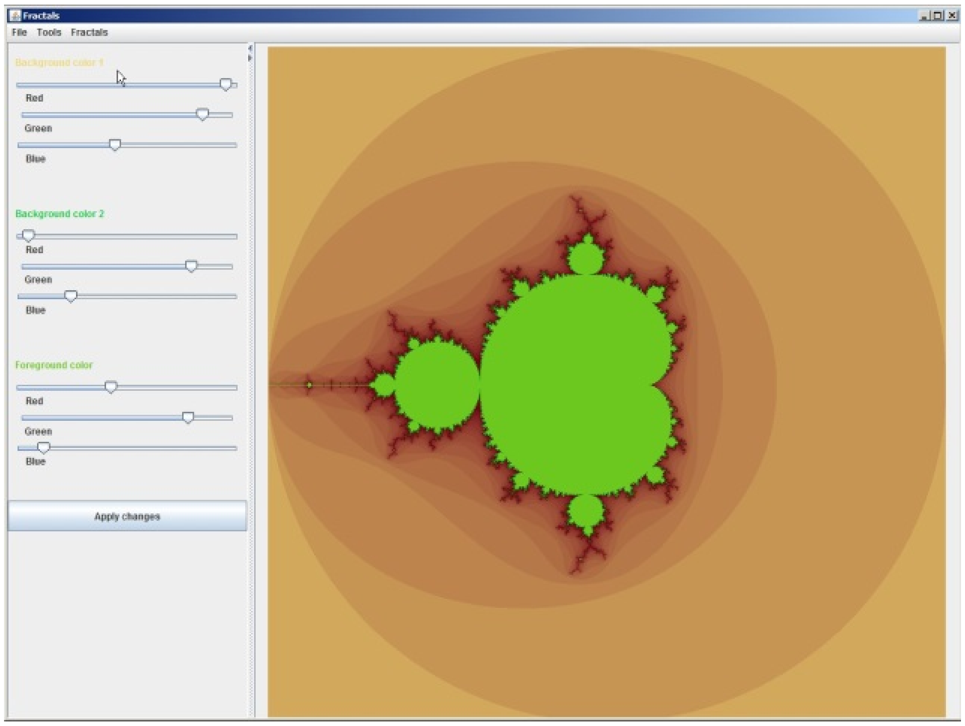
The program we are going to develop consists of multiple steps. Each step can be seen as adding a layer of complexity and takes a fair amount of time on its own.

Part 1: The framework

Create a nice framework for your program. There are 3 integral parts of your framework.

1. You need to have a menu bar, in which you can select different types of fractals, toggle the sidebar, and change the behaviour of your mouse. (See Figure 3b).
2. You need to have a (resizable) sidebar, in which you can set all parameters important for the generation of the currently selected fractal. (See Figure 3a).
3. And you need a main window, which renders the fractal, with a mouse listener for manipulation of the rendered fractal.

Additionally you are supposed to set key accelerators for most menu items, make the application quit whenever you click the exit button, and add a menu (e.g. File) where you can load saved settings, save the current settings to a .txt file, and exit the program.



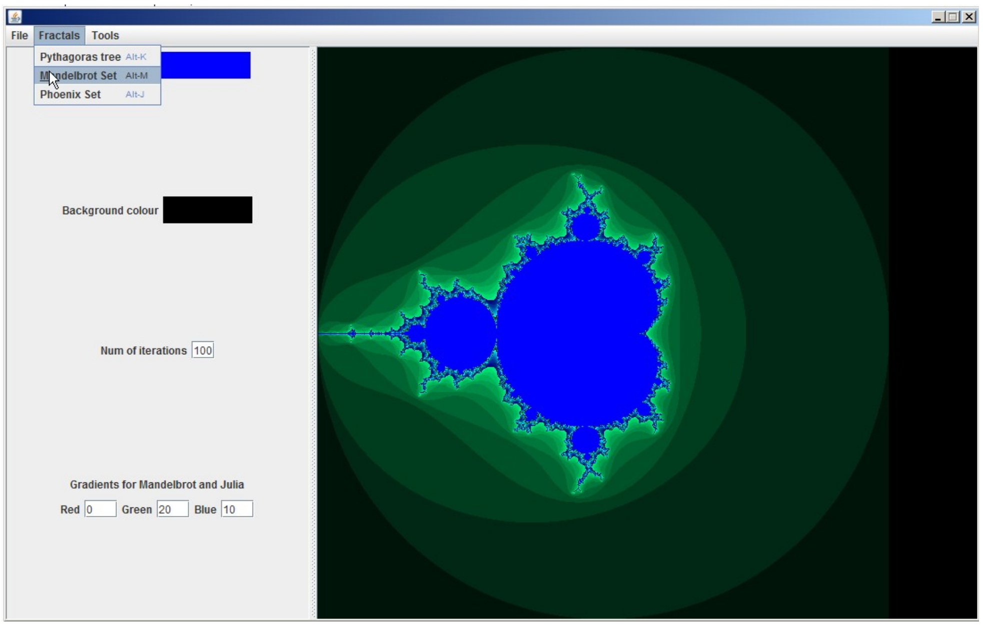


Figure 3: Two Gui versions of a program showing a main window with a Mandelbrot and a sidebar. The second version is shown with an open menu bar.

Part 2: The first fractals

After the framework is done we add the first two simple fractal: the Pythagoras tree:
Pythagoras tree: "The Pythagoras tree is a plane fractal constructed from squares. Invented by the Dutch mathematics teacher Albert E. Bosman in 1942, it is named after the ancient Greek mathematician Pythagoras because each triple of touching squares encloses a right triangle, in a configuration traditionally used to depict the Pythagorean theorem" See Figure 4.

- background color;
- line color;
- square and triangle fill color;
- size of acute angles;
- number of iterations. (0 being infinitely many).

In case of infinitely deep generation of the fractal, you are allowed to either generate a reasonable depth based on the resolution of a 4K-screen with a zoom of 2000 , or come up with a more clever way to render the fractal such that you can actually "zoom to infinity".

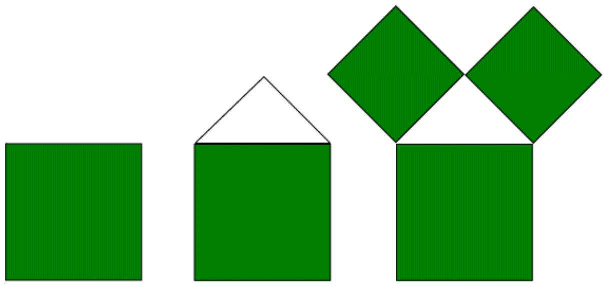


Figure 4: Pythagoras tree fractal after iteration 1.

The triangles that are attached to each hypotenuse can be any right triangle with acute angles. The images in Figure 5 shows what happens after 20 iterations if each triangle has both acute angles 45° degrees (left) and 30°angles 60° (right).

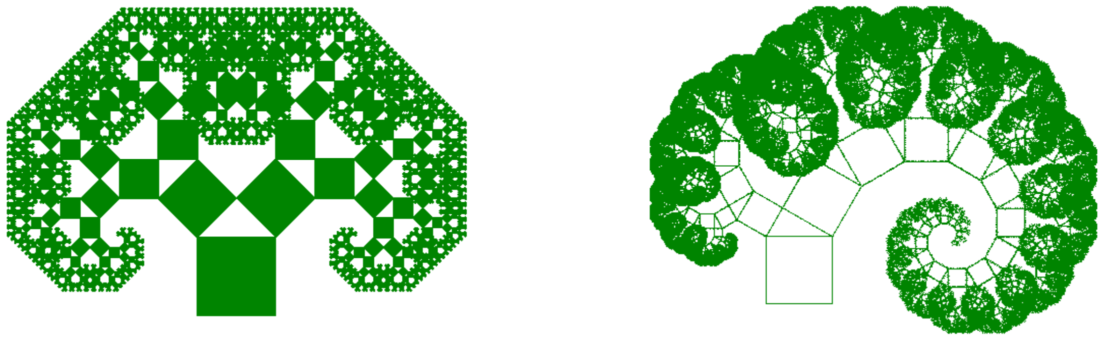
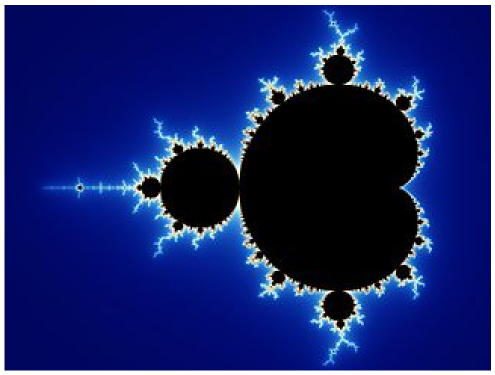


Figure 5: Pythagoras tree fractal after 20 iterations.

Part 3: More interesting fractals

When you are done implementing the more easy fractals from Part 2, you should continue development of you program by adding two more famous fractals: The Mandelbrot set, and the Phoenix set. For the mathematical definitions of these sets take a look at Wikipedia, or another reliable source.
You are required to, add the following settings for these fractals:

- a special color for the body of the fractal (the inner part);
- a color gradient consisting of a variable (minimum of 2) amount of colors;
- number of iterations. (0 being infinitely many);
- other relevant parameters for generation.



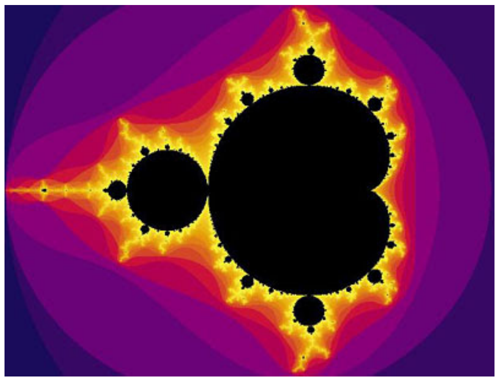


Figure 6: Some images of possible rendering of Mandelbrot set

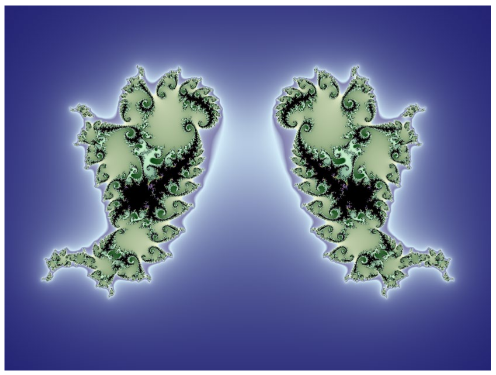


Figure 7: Some images of possible rendering of Phoenix set.

Constraints

Since this assignment is mostly about native Java GUI, **you are not allowed to use the Netbeans / IntelliJ IDEA / Eclipse GUI or their likes' generators** to create the GUI. Also, you are **not allowed to use special libraries for drawing these fractals**. Try to only use Java that is described in this course.

Surely, there are a lot of implementations for fractal generation online. You are not supposed to copy algorithms directly from the internet, however it is no problem if your algorithm is 'inspired' by some code you find online.

The grading scheme for this assignment is described on the Canvas assignment page.

Handing in the assignment

Upload into Canvas a zip file FractalGenerator.zip containing all the .java files of the project in one folder.