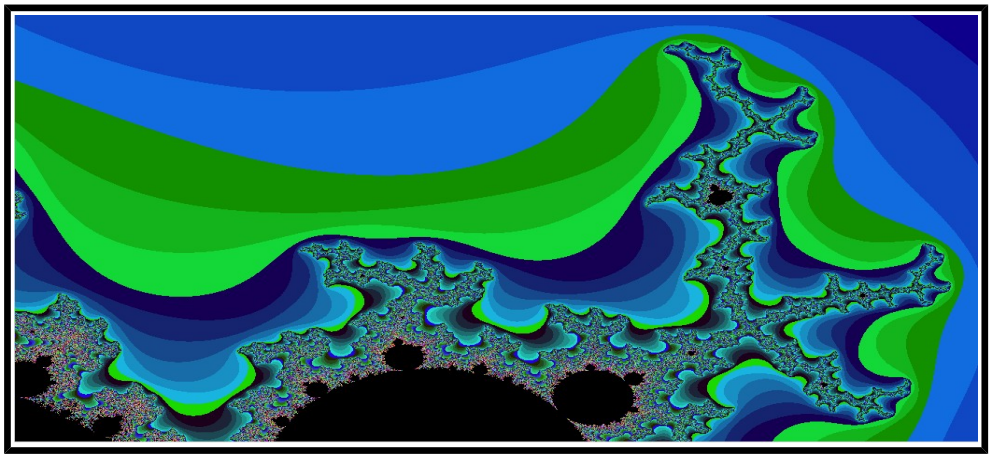
CO225: Software Construction
Fractals: Natures building blocks

Deadline: 16th November

This is a group project with two members in a group. You can form groups and inform the ICC.

Aim: The aim of this lab is to see how good you are at designing somewhat complicated open ended software.

Introduction: You are required to design and implement a software that would plot two of the most popular fractals: Mandelbrot set and Julia set. Fractals are infinity precise, self-similar shapes formed by some simple mathematical computations. The computation varies from fractal set to set but are



generally based on complex numbers.

The Mandelbrot set: In mathematics Mandelbrot set is defined as the set of complex numbers C such that: $\mathbf{Z_{n+1} = Z_n^2 + C}$, starting with $Z_0 = 0$ remains bounded when *n* reach infinity. In other words if for some C the above equation remains bounded for any number of iterations then that C is in the Mandelbrot set. There is a mathematical proof which shows that if $ABS(Z_n) > 2$ the above equation will diverge hence C is not in the Mandelbrot set. Some complex numbers will be in the Mandelbrot set and some are not. In the basic case one can assign one color (say black) to the complex numbers in the set and another colour (say white) to the ones which are not.

In any case all Mandelbrot numbers are within a region of the complex plane; -1 < real part < 1 and -1< complex part<1. We call this the **region of interest** which you should be able to set.

When you take a point on the canvas (the area where the image needs to be drawn) you need to first map into a within the region of interest. Once that is done use that value as C and perform the above computation $\mathbf{Z_{n+1} = Z_n^2 + C}$. Perform 1000 iterations and then see if $ABS(Z_n) > 2$ for any n < 1000. If so C is not a Mandelbrot number so assign some colour to it based on value of n when $ABS(Z_n) > 2$. Else assign black. Repeat this process for all points on the canvas. You should be able to set the iterations and the region of interest from the command-line by passing arguments; *java Fractal* ***Mandelbrot -0.5 0.5 -0.1 1 1000*** means the region of interest for the image should be from -0.5<real<0.5 and -0.1<complex<1 and for each point you need to do 1000 iterations before deciding that it is in the set. Note that *Fractal* is the name of the application.

The Julia set: The Julia set is similar to the Mandelbrot set in that it uses the same equation $\mathbf{Z_{n+1} = Z_n^2 + C}$ but Z_0 is the point in the complex plane corresponding to the pixel and C is a constant. The rest of the computation is the same. If one types; *java Fractal* ***Julia -0.5 0.156 1000*** then you should plot the Julia set for C = -0.5 + 0.156i with 1000 iterations for each point. You may take the region of interest in the complex plan as 1 < real part < 1 and -1< complex part<1 which cannot be modified.

Generating the image: We have a area (with some dimension) to plot the pattern. This area will contain pixels and will be fixed for 800x800 pixels. For each point you need to see if the point is in the set specified by the user and assign colour accordingly. See the sample code for hints.

Program: Your program should accept arguments; at least one which would specify what set to plot. If the set selected is Mandelbrot the user should give either 0, 4 or 5 arguments. If there are 0 arguments then use the default values as specified in the table below. 4 arguments will be the region of interest in the complex plane and the 5th one is the number of iterations to do for a point.

For the Julia set the user should give 0 or 2 arguments. If there are no arguments one should use the default arguments and 2 arguments will be the real and complex part for C.

Item	Default value	Note
Region of interest	-1<real<1 -1<complex<1	Always use default for Julia
Number of iterations	1000	
C	-0.4+0.6i	Only for Julia set

Programming: You are **not given** a skeleton code for this project. You may use fragments of code and/or classes from lectures. You need to design the classes etc. before you start coding. You can think of interfaces, abstract classes etc. You are free to make your own choice but ***might be*** asked to justify the choice in front of the class. (3 groups with the least marks and 2 other groups randomly selected will be asked to discuss their design and code with the rest of the class).

Attention: You might want to consider the following:

1. Precision of the complex numbers and the data type for the complex and real parts
2. Execution time. Since there are lots of computation one might think of using techniques to reduce the computation time. For example instead of testing $ABS(Z_n) > 2$ one might use $ABS(Z^2_n) > 4$. This make sense since the equation gives you Z^2_n.
3. **As an advanced part one might consider using Java threads. This is not expect from you but you may try that for additional 10marks.**

Submission: You should submit all the *.java files as a **single zip** file via Moodle before the deadline. As always you will be given marks for coding logic, correctness and neatness.