Fractals

Lucas Larsson

Spring Term 2022

Introduction

This is assignment will be to implement a fractal generator, the specific fractal that we will implement here is the most famous of them, the Mandelbrot Set, but the same method would work for creating any other fractal.

Furthermore we will create an image of the result set, this will be with the help of the provided `ppm.ex` file.

Mandelbrot

This is a fun not so complected assignment, so what i did was to just jump right in and start coding, but i got overwhelmed by the amount of information that there is to read about it, both the information and the actual set are truly fascinating and infinite.

So before we start digging the rabbit hole what do we need to know? Well, it is quite simple, we need to determine if a given complex number belongs to the set or not. The method used for that is also mentioned in the instructions. And briefly explained below.

Complex module

Since we will be handling complex numbers first thing to do is to implement arithmetic methods to handle them.

As usual I always start by looking for built in libraries, since complex numbers are a fundamental part of math it is almost guaranteed that there will be built in methods in any programming language.

I don't consider it cheating on the assignment because it is fairly easy arithmetic, and built in methods are usually versatile and can handle most errors and edge cases.

In this case I ended up writing the methods myself, since the the built in `Complex` module used the format `%Complex{x,y}`, and i just wanted to use a simple tuple.

The methods needed are created in a module `Cmplx` are `new/2, add/2, sqr/1` and `abs/1`

```
def sqr({r, i}) do
    real = (r * r) + (i*i) * (-1)
    im = 2*(r*i)
    {real, im}
end
def abs({r, i}) do
    :math.sqrt((r*r)+(i*i))
end
```

I don't believe I need to explain the methods, but maybe why do we need them.

As motioned earlier, we need a way to determine weather a given complex number do belong to the set. If a number start to approach infinite given the competition of the formula below, then it does not belong to the set, a quick way to determine if a number will approach infinity without actually computing to infinity is the observation that for : $z_n : if |z_n| > 2$ in other words if a numbers Absolute value is grater than 2, it will definitely continue to grow for ever.

$$z_0 = 0$$

$$z_n + 1 = z_n^2 + c$$

Brot nnodule

```
 def mandelbrot(c, m) do
    z0 = Cmplx.new(0, 0)
    i = 0
    test(i, z0, c, m)
end

def test(i, z, c, m) do
    next = Cmplx.add(Cmplx.sqr(z), c)
    case Cmplx.abs(next) <= 2 do
      true ->
        case m > 0 do
          true -> test(i + 1, next, c, m - 1)
          _    -> 0
        end
      _    -> i
    end
end
```

The above implementation is far from an effective, but as a first draft, i think it is fine. The methods above do exactly what is required, given a Complex number c and max number of iterations m, the clauses return the value i for which $|z_i| > 2$ or 0 if it does not for any $i < m$. This explanation is taken from the instructions since it is short and informing, to put it in other words, if $|z_i|$

is bigger than 2 it will continue growing for ever and it doesn't belong to the set. If a number z_i does belong to the set, it will always continue in a loop for ever, that is it will shrink for ever, this is the reason you can zoom in into Mandelbrot sets for ever without reaching a limit.and for that reason we limit it to m depths or iterations.

Printer

This module was provided to us by the teacher, a short explanation of the `PPM` module is that it takes in a name of the file `ppm` file to be created and the image which is a list of tuples representing the colors.

Colour

The `Colour` module is where I least spent time, instead I invested time in more experiments that i will show and talk about below. As I used the version from the instructions i will not include any code from the module here.

Computing

Our teacher is always amazed by how we never seem to be impressed by the programs he run in class, that is the few of us that actually attend. But when I generated my first image I was really impressed. I will not include a reflection as I think I have expressed my self throughout the report. The most time I used for this assignment was kind of waste of time, I generated a `gif` using multiple images because I thought it will be cool include in the report as it is not possible to include a large image where one can zoom in multiple times, after that I created a graph in Mathematica that i couldn't get to work in overleaf either so I'll include a image of it below.

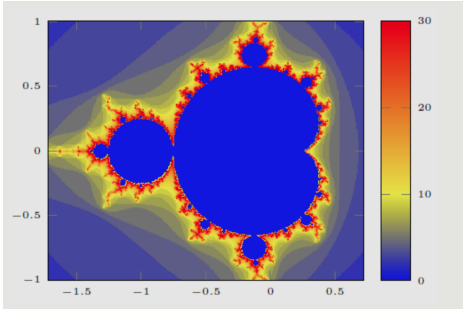

Figure 1: A graph from mathimatica showing the Mandelbrot set.

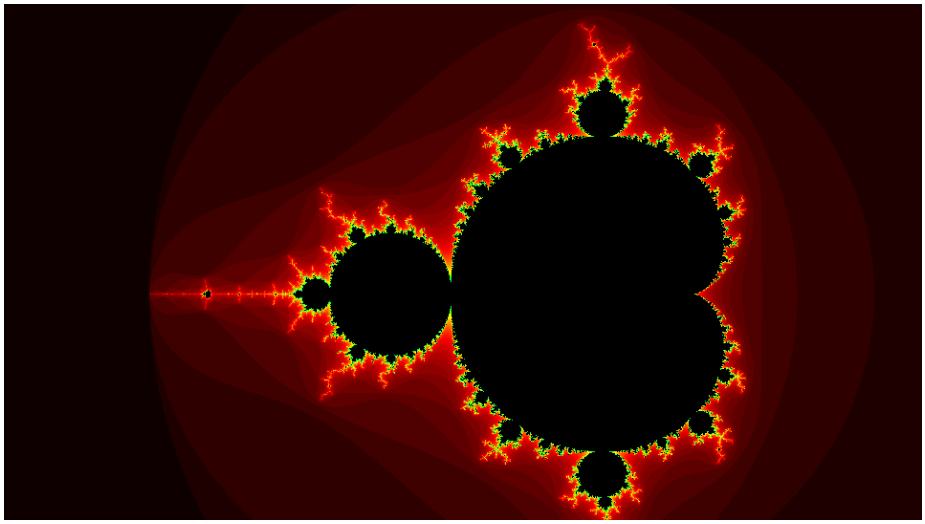

Figure 2: Image showing the Mandelbrot set

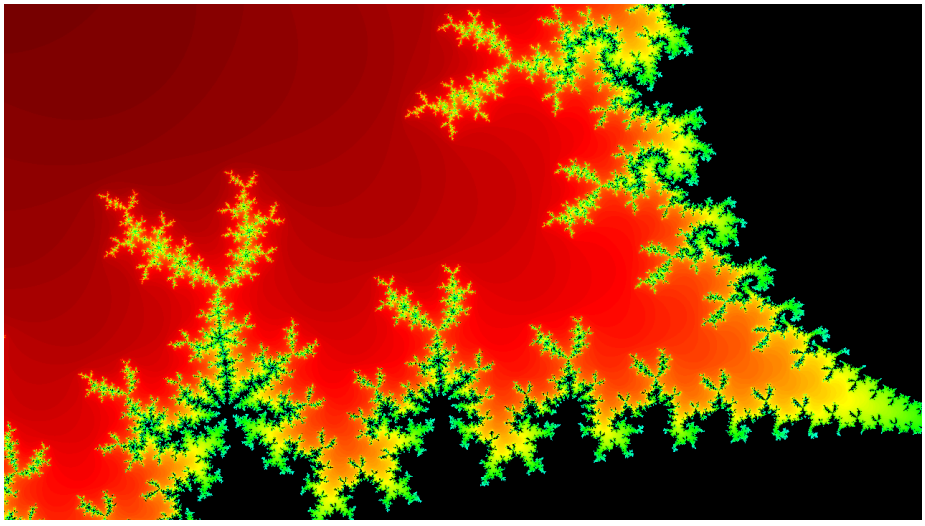

Figure 3: Image shoeing the Mandelbrot set, zoomed in.

After multiple trail and error i zoomed in on the exact spot i want to show where figure 3 is a zoomed in version of figure 2 showing the beautiful set in its infinite size.