Mandelbrot

Ayda Ghalkhanbaz

Spring Term 2024

Introduction

This assignment involves calculating the Mandelbrot set and generating a visualisation of it as an image. Later in this report, we will look into the results and also discuss how the calculations can be sped up and optimised. In order to keep the report as short as possible, there won't be many code snippets included, but all the code can be found on GitHub.

Method

The Mandelbrot set is the set of complex numbers generated through iterations using a special function defined as follows: $z_{n+1} = z_n^2 + c$.
One of the interesting characteristic of this set is that the function does not diverge to infinity but rather remains within the boundaries of an absolute value of 2. The implementation of this assignment can be divided into several subtasks, which are explained below.

Complex Numbers

The module implemented for this subtask has one simple job, that is, introducing a complex number with real and imaginary parts, and further applying some basic operations on them. Coding this part was quite straightforward and the only requirement was having knowledge about complex numbers. In the code snippet below, you'll find the `Complex` module, that is able to introduce a new complex number and perform `add`, `square` and `abs` on it.

```
def new(r, i) do {r, i} end
def add({r1, i1}, {r2, i2}) do {r1+r2, i1+i2} end
def sqr({r, i}) do {(r*r) - (i*i), 2 * r * i} end
def abs({r, i}) do :math.sqrt((r*r) + (i*i)) end
```

Mandelbrot calculations

This subtask involves implementing the iteration function (discussed earlier in this section), in order to check whether the function is applicable on a complex number with the consideration to the conditions for this set, that is, the number should be less than an absolute value of 2 after the function's application.

As demonstrated in the code snippet below, the module has two functions; `mandelbrot/2` and `test/4`. The `mandelbrot/2` function calls the `test/4` function and it is the `test/4` function that has the responsibility of applying the function on the complex number until it reaches the base case, i.e. either the maximum depth is reached or the absolute value is larger than 2. It is noteworthy that the functions in this module employ the `Complex` module's functions to perform the required operations.

```
def mandelbrot(c, m) do
  z0 = Complex.new(0,0)
  i = 0
  test(i, z0, c, m)
end
def test(m, _, _, m) do 0 end
def test(i, zi, c, m) do
  z = Complex.abs(zi)
  cond do
    z > 2 -> i
    z <= 2 ->
      znext = Complex.add(Complex.sqr(zi), c)
      test(i+1, znext, c, m)
  end
end
```

Main calculations

One might say that the `Mandel` module is a connection between the calculations and the visualisations.

This module, along with the help of two previous modules, operates on the pixels of an image. Using the height and width of the image, this module operates on each pixel in every row, checks whether the corresponding complex number belongs to the Mandelbrot set and assigns a colour to the pixel.

The code for this module among other helper modules can be found on GitHub.

Result

The general visualisation result of the Mandelbrot set is illustrated in figure 1, while figure 2 and 3 give a closer look at different spots on the set.

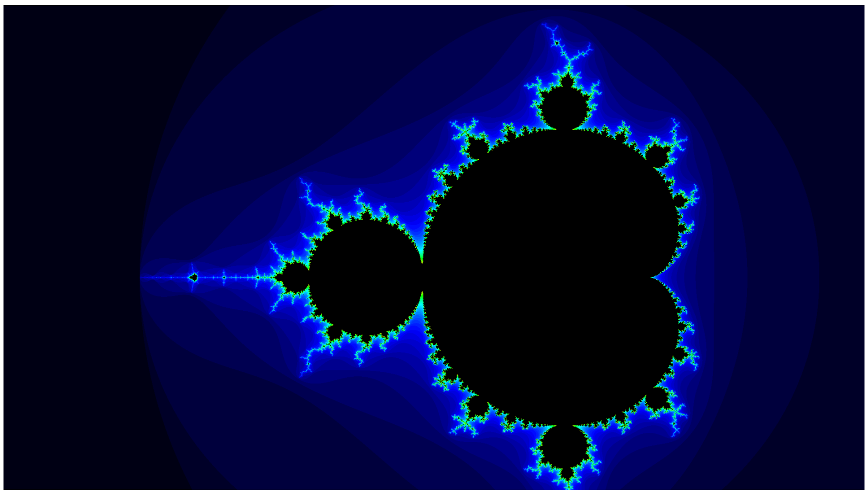

Figure 1: *The general look at the Mandelbrot set found on the coordinates* $x_0 = -2.6, y_0 = 1.2, x_n = 1.2$

Discussion

By looking at figure 1, we can realise that the black space in the centre of the picture is the set of complex numbers greater than absolute value of 2. Another fascinating observation is illustrated in figure 2 and 3, i.e the fractional nature of the Mandelbrot. This means that any part of the set has self-similar patterns repeating at smaller scales.

One possible solution to speedup operations in this program is parallelism. We know that the calculations done in the second module is time consuming since we have to continuously build tuples, separate them to finally represent a tuple. These operations come with a cost when they have to be applied on several rows of an image. However, with the help of parallelism and concurrency, operations will be done faster.

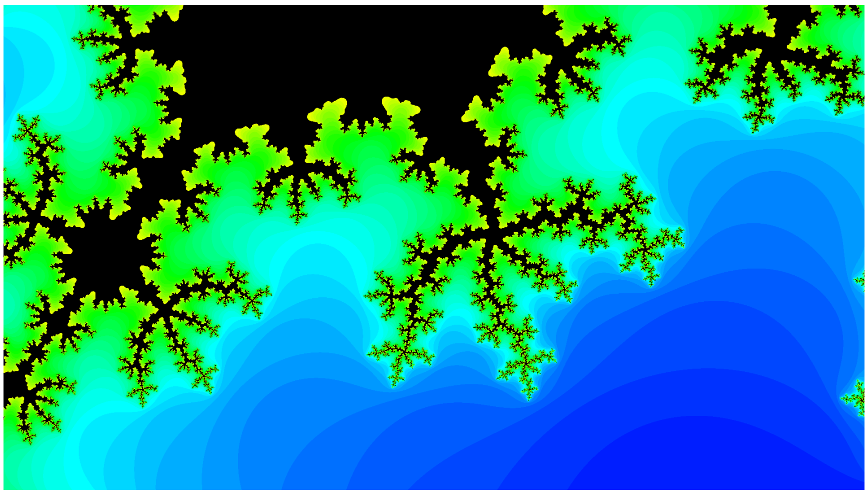

Figure 2: *A closer look at the Mandelbrot set found on the coordinates* $x_0 = -0.5577, y_0 = -0.6099, x_n = -0.5$

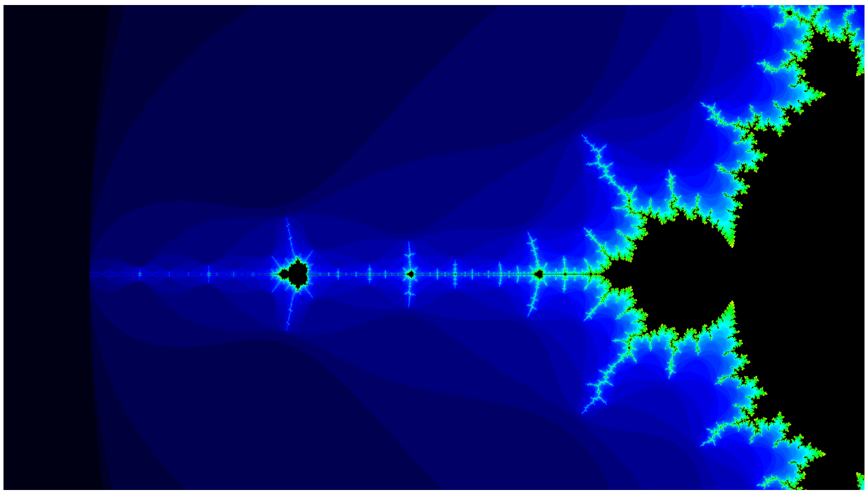

Figure 3: *A closer look at the tail of the Mandelbrot set found on the coordinates* $x_0 = -1.1, y_0 = -0.25, x_n = -2.1$