Mandelbrot Set

Vsevolod Zozin

4/November/2015

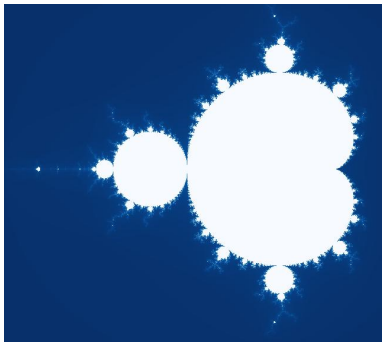


Figure 1: Mandelbrot Set

1 Introduction

1.1 Brief history

The set was first discovered some time during the beginning of the 20^{th} century by Gaston Julia. First image of the set was drawn in 1978 using computers by a team at IBM Thomas J. Watson Research Center[3]. Later the set was named in honour of Bernoit B.Mandelbrot, a Polish-born mathematician who developed new mathematical ideas as well as creating computer programs to display the graphics. As computers became faster, and more affordable, the Mandelbrot Set became one of the most recognisable shapes in abstract mathematics[7].

1.2 What is the Mandelbrot Set?

Mathematical definition of the Mandelbrot Set M is as follows.
If $P_c^n(z)$ is n^{th} iterate of $P_c(z)$ then subset of the complex plane is given by[8] $M = \left\{c \in \mathbb{C} : \exists s \in \mathbb{R}, \forall n \in \mathbb{N}, |P_c^n(0)| \leqslant s\right\}$
we can make $s = 2$ $P_c : \mathbb{C} \to \mathbb{C}$ given by $P_c : z \mapsto z^2 + c$
Informal definition.
The Mandelbrot Set, is a set of complex numbers c such that after n iterations of $z_{n+1} = z_n^2 + c$, the modulus of z does not exceed 2 (starting with $z = 0 + 0i$).
Example: Let's see if $c = 0.5 + 0.5i$ is in the Mandelbrot Set!

$$z_1 = (0 + 0i)^2 + (0.5 + 0.5i) = 0.5 + 0.5i \text{ which implies } |z_1| \leqslant 2$$

$$z_2 = (0.5 + 0.5i)^2 + (0.5 + 0.5i) = -0.25 + 1.5i \text{ which implies } |z_2| \leqslant 2$$

$$z_3 = ((0.5 + 0.5i)^2 + (0.5 + 0.5i))^2 + (0.5 + 0.5i) = -1.6875 - 0.25i \text{ which implies } |z_3| \leqslant 2$$

$$z_4 = (((0.5 + 0.5i)^2 + (0.5 + 0.5i))^2 + (0.5 + 0.5i))^2 + (0.5 + 0.5i) = 3.285 + 1.345i \text{ which implies } |z_4| > 2$$

we can see that for $n \geqslant 4$, $c = 0.5 + 0.5i$ is NOT in the Mandelbrot Set. You can see what it looks like in Figure 2.

2 Plotting and Colouring

We use real and imaginary parts of the complex number to plot each number in the set onto the Argand diagram. In my case, white colour represents points that did not escape after n iterations. Points which escaped in fewer iterations are closer to dark blue. This is the simplest form of the Mandelbrot Set colouring, it's called "Escape Time Algorithm"[5], but it results in discrete values, and bands of colour, for continuous values there are many other algorithms such as "Normalised Iteration Count"[6], which provides smooth colour transition between iterations, but they're much harder to utilise.

2.1 How I plotted the set

I used C++ to go through all the points inside the square with coordinates $(-2, 2i), (-2, -2i), (2, -2i), (2, 2i)$ one after the other, with step size of 0.001 and n value of 100.

1. Set parameters in C++ file
2. Compile C++
3. Run executable and wait for output.csv file to be generated $\approx$ 35MB
4. Use SAGE worksheet to plot the matrix from output.csv file

If you're interested in what my code looks like, you can visit my published SAGE worksheet.
https://cloud.sagemath.com/projects/3806d5dc-6178-4032-a8a8-32b86fbb7b63/files/mandelbrot.sagews

2.2 Possible optimisations

There are many things that could be done to reduce the time taken for the points to be calculated. I came up with an algorithm that in theory should reduce the number of steps required for outside tracing (it is rather complicated to implement). Here is the summary:

1. Mandelbrot points are reflected on real-axes. You only need to take complex conjugate for the ones below
2. Although Mandelbrot points are defined within a complex circle of radius 2, it is known that the minimum real value is -2.0 and maximum is 1.0; and the maximum imaginary value is 1.5 and minimum is -1.5 respectively[1]. Yet again you can save quite a few steps by only calculating within regions where you're sure that there're Mandelbrot points

To further reduce the number of steps, you can use this recursive algorithm: split the field into 3 rectangles, as shown in Figure 4 (Co-ordinates of the squares could be found manually).

1. Mark all the points inside the first rectangle as false (i.e. not Mandelbrot points)
2. (This is the point where you can choose to create different threads for the other two rectangles.) Go around the outside of the rectangle checking for Mandelbrot points
3. If all points around the outside of the rectangle have been marked as true, then fill the rest of the points inside the rectangle as Mandelbrot points. Else divide the big square into four smaller ones, and repeat step 2 (without the threading part), until required precision has been reached. You can see the diagram of the algorithm in Figure 3

It is very important that your application will be able to check points where different rectangles touch, to avoid duplication, as well as implementing algorithms that would decide how likely that some given square will contain a Mandelbrot point, thus getting rid of all the likely candidates first (You can find my partial implementation of this algorithm in C++ in my published project in SAGE, same link as before).

3 Bigger picture

Mandelbrot set it self does not have any practical applications, apart from drawing pretty pictures and/or benchmarking performance of a computer, but fractals in general do. Their self-similar patterns at different scales let us generate images of dynamic systems - pictures of chaos [2], allowing us to understand the world around us[9].

References

[1] Paul Derbyshire. Mandelbrot set frequently asked questions. `http://www.nahee.com/Derbyshire/mandlfaq.html`. [Online; accessed 04-November-2015].

[2] Fractal Foundations. What are fractals? `http://fractalfoundation.org/resources/what-are-fractals/`. [Online; accessed 04-November-2015].

[3] History.mcs.st-and.ac.uk. Mandelbrot set history bibliography of bernoit. `http://www-history.mcs.st-and.ac.uk/Biographies/Mandelbrot.html`, 2000. [Online; accessed 04-November-2015].

[4] Vince Knight. Plotting complex numbers in sage., code to plot a circle in complex plane. `http://drvinceknight.blogspot.co.uk/2013/11/plotting-complex-numbers-in-sage.html`, 2014. [Online; accessed 04-November-2015].

[5] Wikibooks. Fractals/iterations in the complex plane/mandelbrot set — wikibooks, the free textbook project. `https://en.wikibooks.org/w/index.php?title=Fractals/Iterations_in_the_complex_plane/Mandelbrot_set&oldid=3008880`, 2015. [Online; accessed 04-November-2015].

[6] Wikipedia. Benoit mandelbrot — wikipedia, the free encyclopedia – continuous (smooth) coloring. https://en.wikipedia.org/wiki/Mandelbrot_set#Continuous_.28smooth.29_coloring. [Online; accessed 04-November-2015].

[7] Wikipedia. Benoit mandelbrot — wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Benoit_Mandelbrot&oldid=690475255, 2015. [Online; accessed 04-November-2015].

[8] Wikipedia. Mandelbrot set — wikipedia, the free encyclopedia – formal definition. https://en.wikipedia.org/wiki/Mandelbrot_set#Formal_definition, 2015. [Online; accessed 04-November-2015].

[9] Fractal Zone. Fractals applications. http://www.fractalzone.be/applications.php. [Online; accessed 04-November-2015].

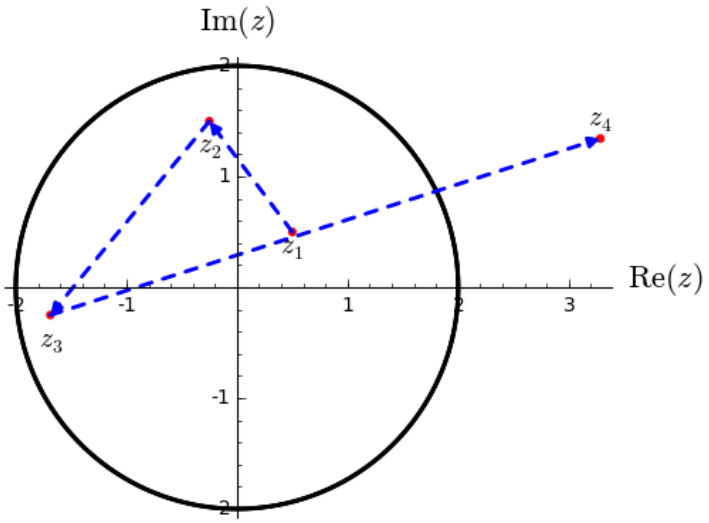


Figure 2: Visual representation[4]

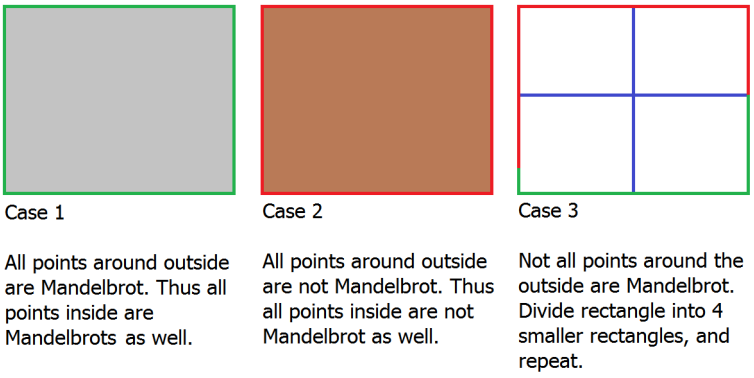
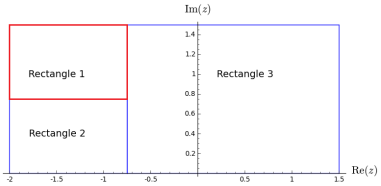
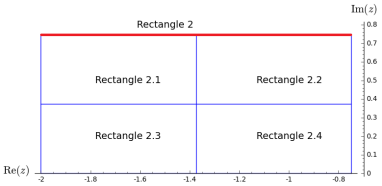


Figure 3: Condition checking in rectangle



(a) Dividing initial space into 3 rectangles



(b) Divide rectangle into smaller rectangles

Figure 4: Grid preperation