# Project: Zooming into Mandelbrot Set

The aim of the project is to visualize the Mandelbrot Set.

## 1 Mandelbrot Set

The Mandelbrot set is a set of complex numbers defined as $M = \{c \in \mathbb{C} \mid \lim_{n\to\infty} Z_n \neq \infty\}$ where $Z_0 = c, Z_{n+1} = Z_n^2 + c$. In other words, the Mandelbrot set is the set of complex numbers for which the value of the function $Z_n$ is not infinite (the set is bounded) as $n$ approaches infinity.

### 1.1 Implementation Details

It has been shown that if the absolute value of Z (its distance from 0+0i) is larger than 2, it will quickly reach to infinity. So do not check it for infinity, just for 2. Set a threshold for the maximum number of iterations while calculating $Z_n$ to see if it goes farther away than 2.

Complex numbers consist of the real and imaginary parts and are drawn in a cartesian coordinate system with the x-axis representing the real part and the y-axis representing the imaginary part. When we draw the Mandelbrot set we draw a point (in black if it belongs to the set, and another color if it does not) at the corresponding image location.

### 1.2 How to Move from Complex Numbers to Image Pixels

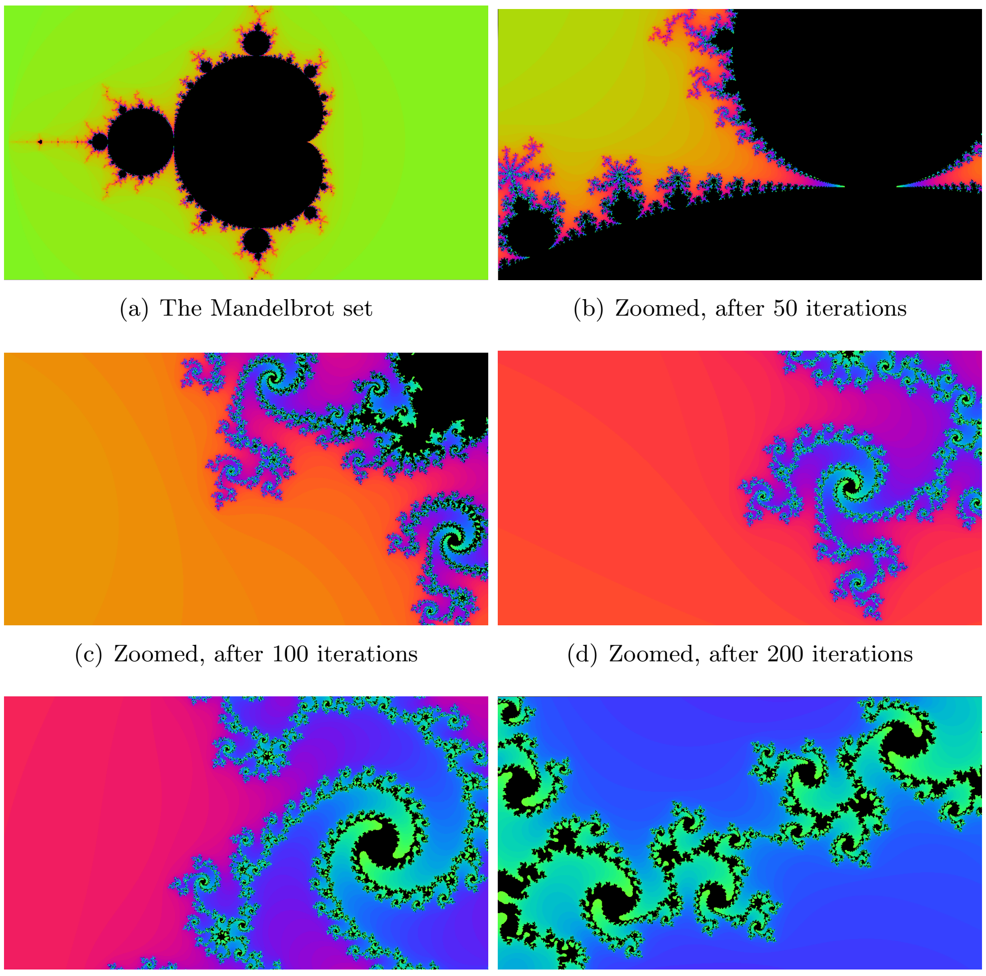
Each pixel in the image corresponds to a complex number. The color represents if the complex number belongs to the set or not. One way to establish the mapping between the two is to define which complex numbers correspond to the corner pixels, then interpolate for the pixels in between.

### 1.3 Colormap

For the points that belong to the set you can use black color. For the ones that do not belong to the set, you can use the number of iterations $n$ and the maximum number of iterations to choose a color. For instance you can use the following to calculate red, green and blue components:
double i = (n * 255 / maxIteration);
r = round(sin(0.024 * i + 0) * 127 + 128);
g = round(sin(0.024 * i + 2) * 127 + 128);
b = round(sin(0.024 * i + 4) * 127 + 128);

### 1.4 How to Zoom into the Set

At each iteration where you zoom into the image, adjust the minimum and maximum complex numbers that you used for the image corners.


(a) The Mandelbrot set

(b) Zoomed, after 50 iterations

(c) Zoomed, after 100 iterations

(d) Zoomed, after 200 iterations

(e) Zoomed, after 300 iterations

(f) Zoomed, after 400 iterations

Figure 1: Visualizing the Mandelbrot set using a colormap. The set is represented by the black region in the image.

## 2 SDL library

You can download a code skeleton from `www.csc.kth.se/~yaseminb/skeleton.c`. Take a look at how to color a pixel with the example function putpixel(). Also, take a look at the example Makefile in `www.csc.kth.se/~yaseminb/Makefile` and add your source files to that to compile your program.

## 3 Testing Your Program

First ask the user to set the width and the height of the image. Then have two modes in your program. You can ask the user to choose one mode in the begining of your program e.g., by prompting: ”please press 1 to see one set or 2 to to zoom in continuosly.”.

- Show only one set as in Figure 1a.
- Continuosly zoom into the set by iterating a fixed number of times. See an example here: `www.csc.kth.se/~yaseminb/MandelbrotExampleFinal.mp4`

## 4 Speedup the Demo

Try to optimize your code so that it completes in less time, e.g., by reducing the number of multiplications. Measure the time spent and report how your changes affect the spent time.

## 5 Practical Points

While writing your code: add comments, use intuitive names for variables and functions, have function declarations and definitions in a separate file (as in the example we had in example source codes in /21Sep/header2/), indent your code so that it is readable. If there are statements that are repeated, use them in functions.

## 6 Examination

Prepare a brief report explaining your implementation and submit all the source files (preferably as .rar or .tar) needed along with a short README file where you explain how to run your code.

## 7 Links

Before you start implementing your project, first test your VM to see if you can run the sample code from `www.csc.kth.se/~yaseminb/trySDL.c` and produce the output seen in the image here: `www.csc.kth.se/~yaseminb/sample.png`. Use the same example Makefile replacing the source file with trySDL.c.