898F3: Programming in C++
Visualising Mandelbrot Fractals

Candidate Number: 229538

January 16, 2021

```
// To compile
g++ -o fractal main.cpp mandelbrotSet.cpp mandelTransform.cpp
```

1 Creating the Mandelbrot Set

Whilst neither of my X and Y coordinate vectors are very large (160 and 120 elements respectively), I still decided it would be good to reserve space within my vectors thus saving the unnecessary expense of memory reallocation. With this, I can improve the granularity of the fractal by simply reducing the spacing.

2 Applying the Transform

I tried a few methods for applying the transform but I found the simplest method was just to use a ‘vector‘ to hold all previous X and Y values. I am confident you could improve the space complexity because to calculate x_{n+1} and y_{n+1} you only need x_n and y_n respectively. With the maximum number of iterations at 100, however, I am effectively storing at most an 800 byte vector which is inconsequential.

I also chose to use a ‘for‘ loop rather than a ‘while‘ loop just because I prefer the control a ‘for‘ loop provides and the certainty that you can’t accidentally enter an infinite loop. I did also try running every point for 100 iterations and then using logic checks for whether $x^2 + y^2 < r^2$ rather

than checking if N had been reached - this worked the same but just required considerably more computation.

My biggest problem here was visualising my fractal; the default Windows Notepad has no 'line wrapping' as default and as such, I was looking at something that did not resemble a fractal. After using a different text editor that allowed me to limit the length of each line to 160 characters, that is when I saw that my code produced the desired result. This led me to add a new line after the outer loop as I have highlighted below.

```
    for (int i = 0; i <= Y.size(); ++i) {
        for (int j = 0; j <= X.size(); ++j) {
            if (stableSolver(X[j], Y[i])) {
                fileOut << 'o';
            } else {
                fileOut << '-';
            }
        }
        // This line made it incredibly easy to view my fractals
        fileOut << "\n";
    }
```

By experimenting with the spacing value, I have produced the following Mandelbrot fractals.

Figure 1: The default fractal we were asked to produce. Spacing = 0.02. You can see the fractal is quite coarse but the overall shape is distinguishable.

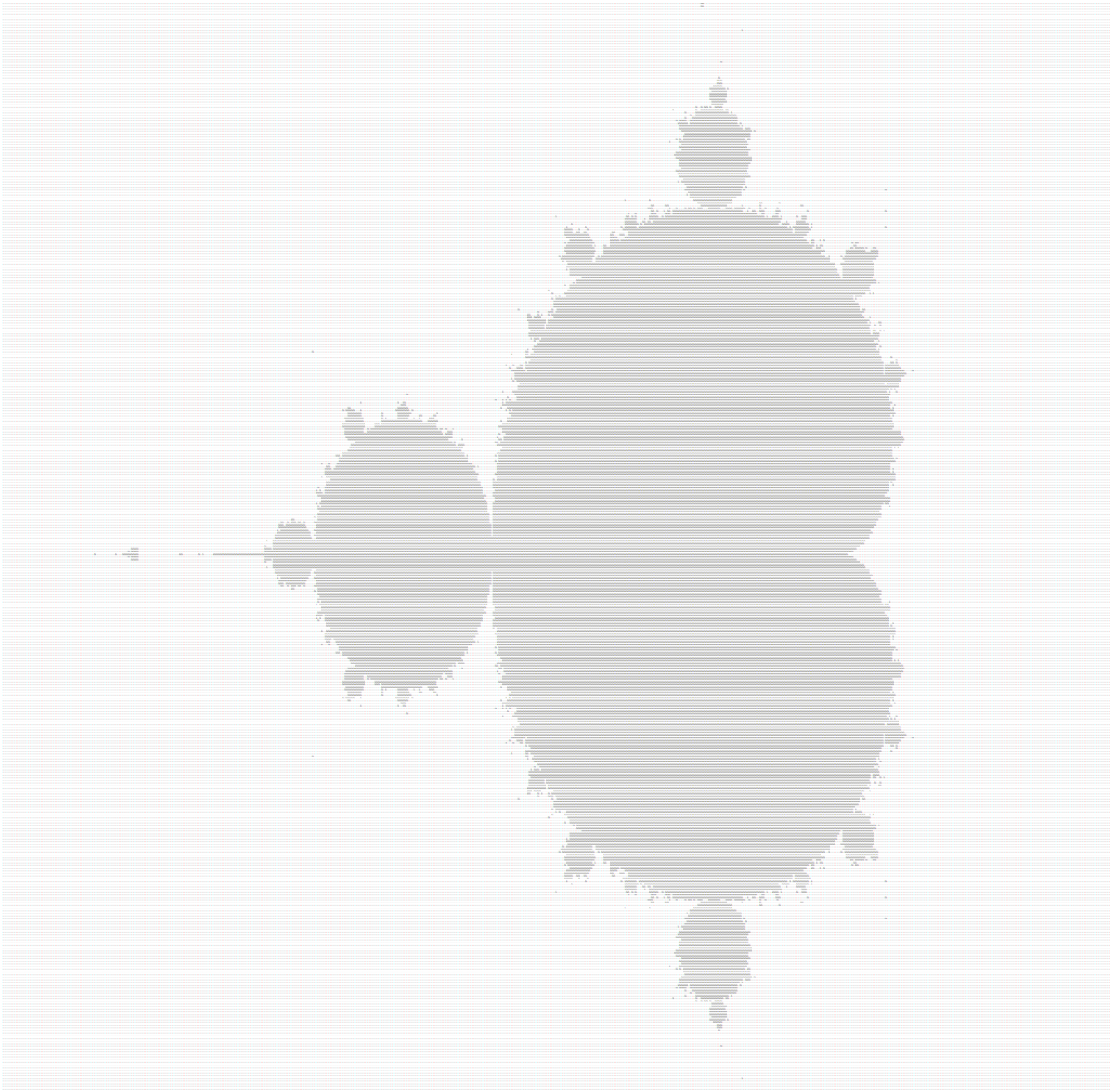

Figure 2: As we reduce the spacing, we see that the fractal becomes more detailed. This has spacing = 0.005, a quarter of the original.

Figure 3: This fractal has spacing = 0.001. At this resolution and zoom, we can now begin to see the remarkable qualities of the Mandelbrot fractal.