Final Coding Write-Up: The Mandelbrot Set

Micah Gruenwald

December 2024

1 Introduction : What is the Mandelbrot Set

You may be asking, what is the Mandelbrot set. Well, its a set of complex numbers. A complex is a number expressed as $a + bi$, and lies on a two dimensional plane, the complex plane, where cordinates are determined by (a, b)
The Mandelbrot set is in turn the set of complex numbers z=$(a + bi)$, such that $\lim_{n\to\infty} z^n \neq \infty$. This is hard to physically conceptualize, but in essense we are asking, if I indefinetly multiply this complex number by itself, will it get really big or will it stay small?

Figure 1: Complex Plane Example

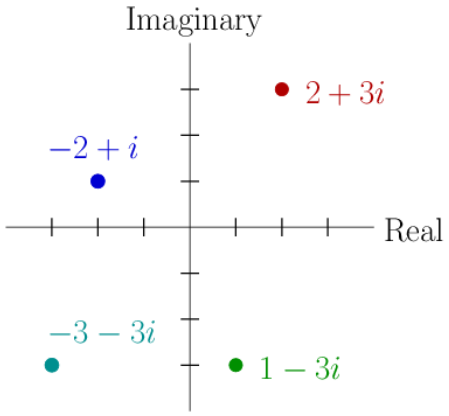
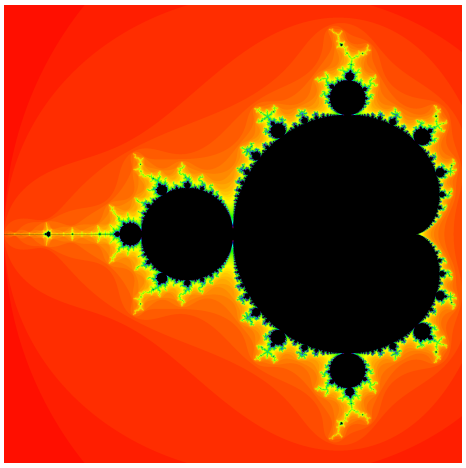


Image courtesy of galileospendulum.org

1.1 How is it Rendered?

We obviously cannot raise each and every number to the ∞, so our code chooses an iteration count, and raises each complex value to that count. It determines if the number diverges by checking is $\sqrt{a^2 + b^2} > 2$, becuase all complex numbers with that property do diverge. If we determine our value diverges, the loop breaks. The function returns the amount of iterations went through divided by the total itterations to show how close the number is to being in the mandelbrot set. This value says that it is rapidly divergent if we only get a value around 0 and is in the Mandelbrot srt if we get 1. Everything else is in that range. Each point in our image is colored in based on that value.

Figure 2: Mandelbrot Set Example



2 Functions

The code has a few basic fucntions:

- mandelbrotValue: This takes in the pixel coordiante of a spot in our image, and returns the mandelbrot value, between 0 and 1.
- pixelToPoint: This takes in a pixel in our image, and returns it's mapped coordinate on the complex plane.
- color: This takes in a mandelbrotValue and returns a rgb color
- renderMandelbrot: This lists through the pixels of our image, and sets the color of each pixel to the color of the mandelbrotValue at that pixel. Finally, it writes an image file called render.png.

3 How the Code Works

3.1 mandelbrotValue

We start by defining the function, taking in a point px and py. We then define variables. This code is a bit optimized but in essense the cordinate is the complex $x + yi$, and becuase compelx numbers multiply as they do, we can find our next value with the forumlae in the code. The function loops through the set, counting up iterations, and breaks when it finds the number diverges, finally returning a max iteration.

```
1  def mandlebrotValue(px,py):
2      point = pixelToPoint(px,py)
3      x2 = 0
4      y2 = 0
5      w = 0
6      iteration = 0
7      while(x2+y2 <= 4 and iteration<
           maxIterations):
8          x = x2 - y2 + point[0]
9          y = w -x2 - y2 + point[1]
10         x2 = x*x
11         y2=y*y
12         w = (x+y)*(x+y)
13         iteration += 1
14     return iteration/maxIterations
```

3.2 pixelToPoint

This is analagous to taking in
$\begin{bmatrix} px \\ py \\ 1 \end{bmatrix}$
and returning
$\begin{bmatrix} \frac{domainLength}{width} & 0 & startX \\ 0 & \frac{rangeLength}{height} & startY \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} px \\ py \\ 1 \end{bmatrix}$

```
15 def pixelToPoint(px,py):
16     point=[500,500]
17     point[0]= startX + px/width*
           domainLength
18     point[1]= startY+ py/height*
           rangeLength
19     return point
```

3.3 color

Takes in the madnelbrot lightness value. If it is 1 it returns black, otherwise it converts the lightness to a angle on the color wheel, and retruns the color at max saturation and brightness.

```
20 def color(lightness)
21     if(lightness ==1):
22       return (0,0,0)
23     return tuple([255*x for x in
           hsv_to_rgb(lightness+indent,1,1)
           ])
```

3.4 renderMandelbrot

Lists through all of the pixels and assigns the tuple color value to each point as an elemnt fo a 2d array. Then it mpas that array onto an 8 int hex value, and finally it assigns that array to a png file.

```
24 def renderMandelbrot(output):
25    pixels = [[color(mandlebrotValue(px,
          py)) for px in range(width)] for
          py in range(height)]
26    array = np.array(pixels, dtype=np.
          uint8)
27    img = Image.fromarray(array)
28
29    img.save("/Users/Period2/Desktop/
          VSCode/Mandelbrot-py/movie/"+
          output+".png","JPEG")
```

4 Renders

Figure 3: Inside

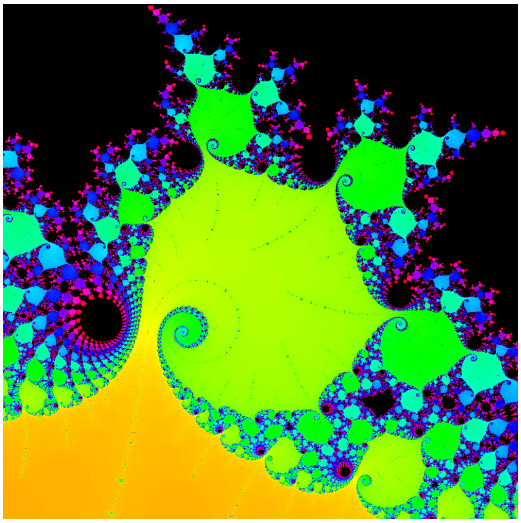


Figure 4: Sun Bay

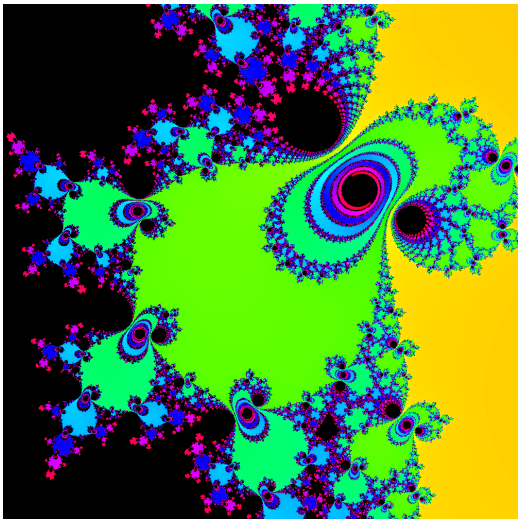


Figure 5: Micahbrot

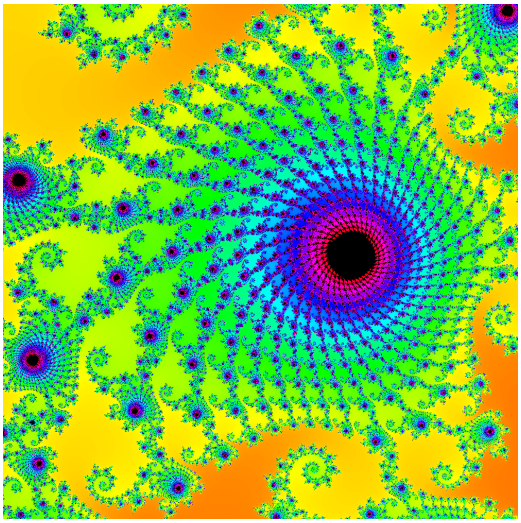


Figure 6: Fantasy

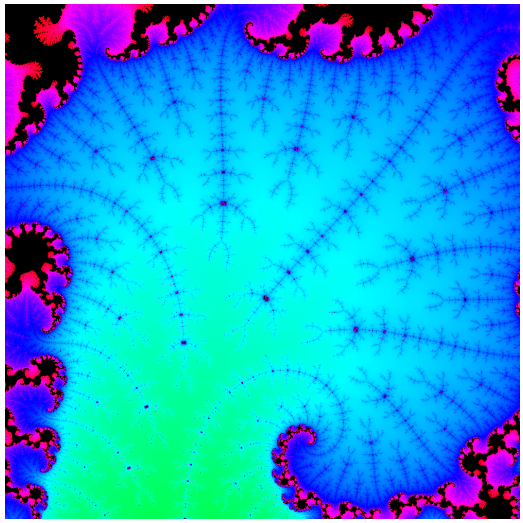


Figure 7: Hurricanes

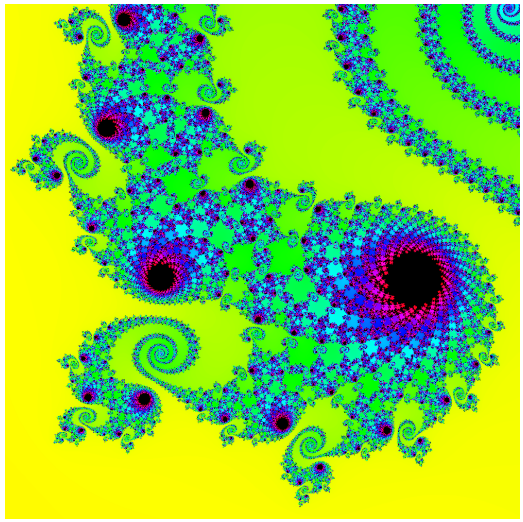


Figure 8: Strange