Fractals – mini Intro

Mandelbrot and Julia Sets

Let's talk numbers

- natural numbers - counting
- zero - nothing
- fractions – dividing pie
- negative numbers – debt
- real numbers – measure/draw
- *imaginary numbers* $i = \sqrt{-1}$ or $i^2 = -1$

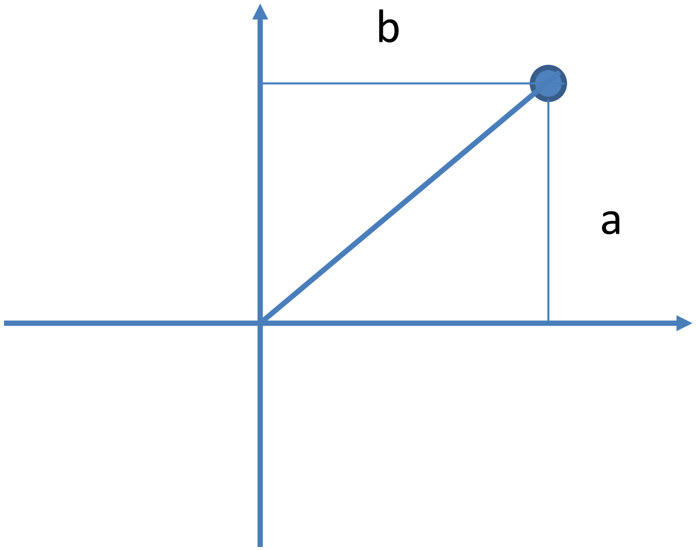
Ta Da – complex numbers! A convenient invention by Euler to solve polynomial equations

ie. $5x^2 = -5$ so how do we solve this? In comes i (which doesn't exist but makes things easier to deal with)

Complex number

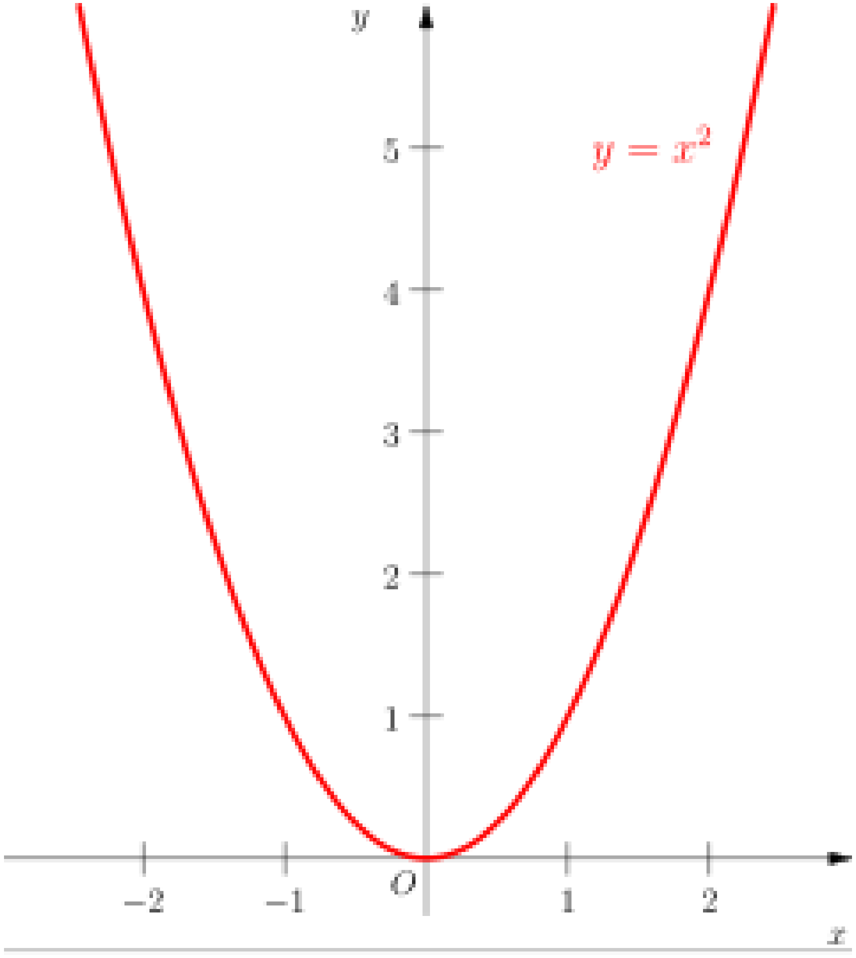
- Expressed in terms of two real numbers and i

$$a + bi$$



Graphing

$f(x) = x^2$



Mandelbrot Set

$$f_c(z) = z^2 + c$$

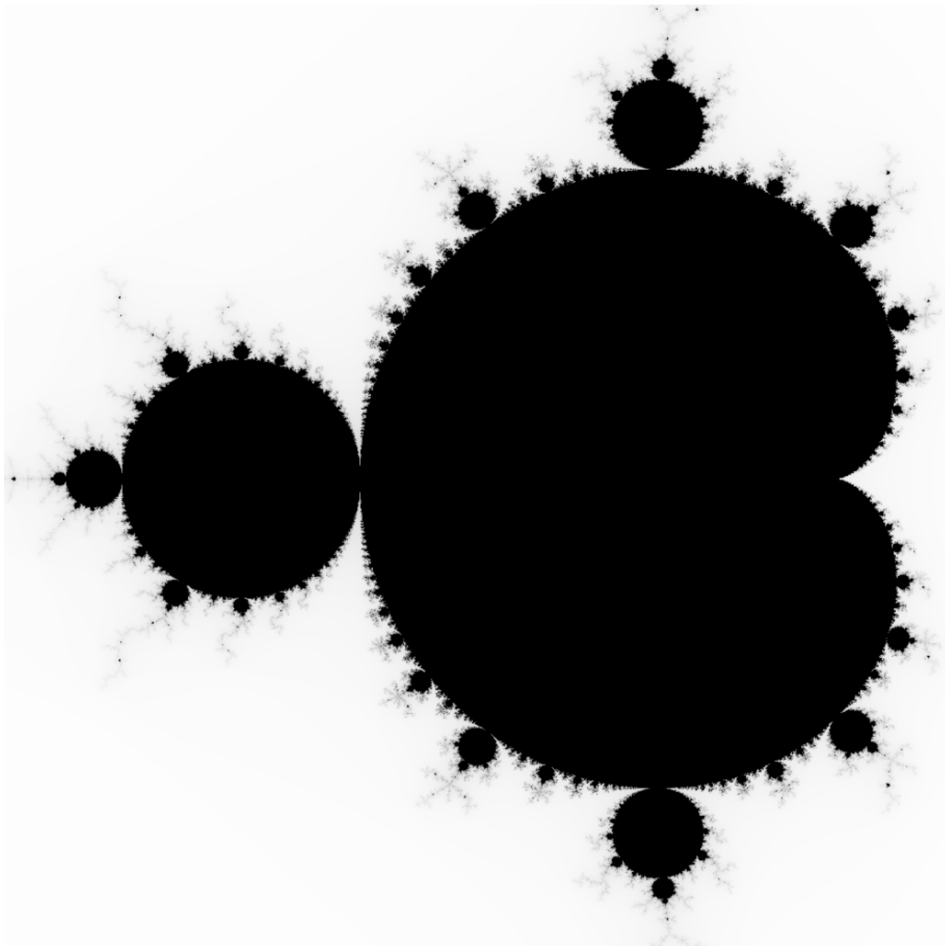
A Graph of the set of numbers as we iterate on zero

Let’s say c = 1
f(0) = 0 + 1 = 1
f(1) = 1 + 1 = 2
f(2) = 4 + 1 = 5
f(5) = 25 + 1 and so on

Let’s say c = -1
f(-1) = -1 + 1 = 0
f(0) = 0 + -1 = -1
f(-1) …

Two behaviors: goes to infinity or is bounded
The bounded numbers are the Mandelbrot set

So how does this work?



Pseudocode from Wiki

```
For each pixel (Px, Py) on the screen, do:
{
  x0 = scaled x coordinate of pixel (scaled to lie in the Mandelbrot X scale (-2.5, 1))
  y0 = scaled y coordinate of pixel (scaled to lie in the Mandelbrot Y scale (-1, 1))
  x = 0.0
  y = 0.0
  iteration = 0
  max_iteration = 1000
  while ( x*x + y*y < 2*2  AND  iteration < max_iteration )
  {
    xtemp = x*x - y*y + x0
    y = 2*x*y + y0
    x = xtemp
    iteration = iteration + 1
  }
  color = palette[iteration]
  plot(Px, Py, color)
}
```

where relating the pseudocode this is just z = x + iy and z^2 = x*x + i2xy - y*y and c = x0 + iy0
so the computation of *x* and *y* (just substituting and taking the real and imaginary parts separate creates the code expressions x = Real(z^2 + c) = x*x - y*y + x0 and then y = Imaginary(z^2 + c) = 2xy +y.

Julia sets are similar but now we are looking at a specific value of c

```
For each pixel (Px, Py) on the screen, do:
{
  x0 = scaled x coordinate of pixel (scaled to lie in the Mandelbrot X scale (-2.5, 1))
  y0 = scaled y coordinate of pixel (scaled to lie in the Mandelbrot Y scale (-1, 1))
  x = 0.0
  y = 0.0
  iteration = 0
  max_iteration = 1000
  while ( x*x + y*y < 2*2  AND  iteration < max_iteration )
  {
    xtemp = x*x - y*y + x0
    y = 2*x*y + y0
    x = xtemp
    iteration = iteration + 1
  }
  color = palette[iteration]
  plot(Px, Py, color)
}
```

where relating the pseudocode this is just z = x + iy and z^2 = x*x + i2xy - y*y and c = x0 + iy0 so the computation of *x* and *y* (just substituting and taking the real and imaginary parts separate creates the code expressions x = Real(z^2 + c) = x*x - y*y + x0 and then y = Imaginary(z^2 + c) = 2xy +y.