Understanding the Mandelbrot and Julia Set

Jake Zyons Wood

August 31, 2015

Introduction

Fractals infiltrate the disciplinary spectra of set theory, complex algebra, generative art, computer science, chaos theory, and more. Fractals visually embody recursive structures endowing them with the ability of nigh infinite complexity. The Sierpinski Triangle, Koch Snowflake, and Dragon Curve comprise a few of the more widely recognized iterated function fractals. These recursive structures possess an intuitive geometric simplicity which makes their creation, at least at a shallow recursive depth, easy to do by hand with pencil and paper. The Mandelbrot and Julia set, on the other hand, allow no such convenience. These fractals are part of the class: *escape-time fractals*, and have only really entered mathematicians' consciousness in the late 1970's[1]. The purpose of this paper is to clearly explain the logical procedures of creating *escape-time fractals*. This will include reviewing the necessary math for this type of fractal, then specifically explaining the algorithms commonly used in the Mandelbrot Set as well as its variations. By the end, the careful reader should, without too much effort, feel totally at ease with the underlying principles of these fractals.

What Makes The Mandelbrot Set a set?

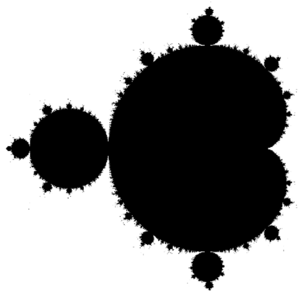


Figure 1: Black and white Mandelbrot visualization

The Mandelbrot Set truly is a set in the mathematica sense of the word. A **set** is a collection of anything with a specific property, the Mandelbrot Set, for instance, is a collection of complex numbers which all share a common property (explained in Part II). All complex numbers can in fact be labeled as either a member of the Mandelbrot set, or not. Below, painted on top of a complex plane, we can see the Mandelbrot Set in its most basic form: the points that are members of the set are painted black to differentiate them from their white non-member neighbors.

Part I

Complex Numbers (overview)

Complex numbers are of the form $a + bi$, where a is the real component and b is the imaginary component. By definition, the imaginary unit i satisfies the equation: $i^2 = -1$. In practice this convention allows simplifying complex values such as $\sqrt{-8}$,

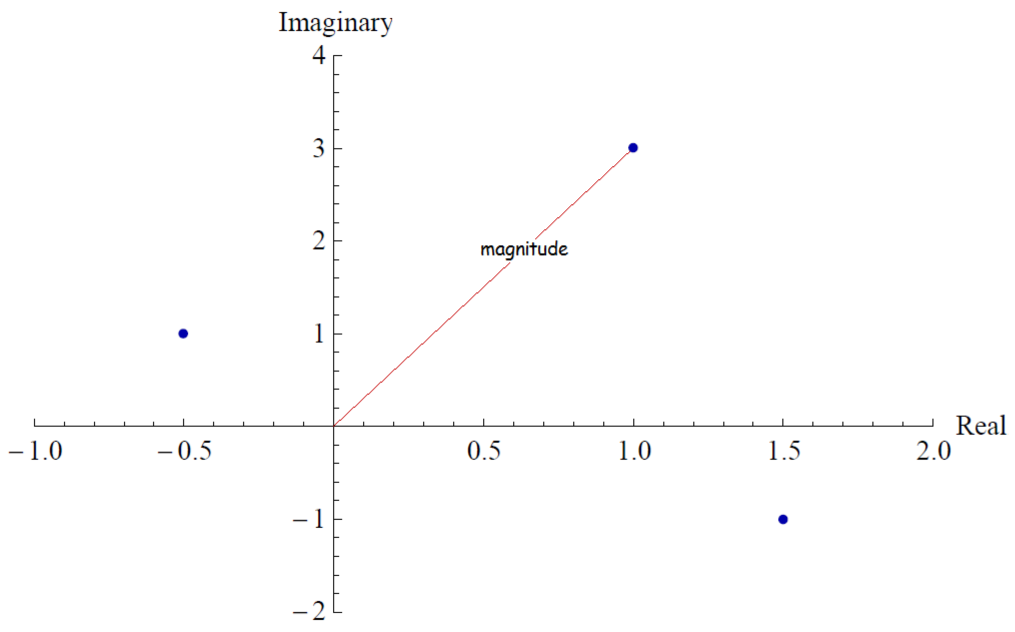


Figure 2: complex magnitude example

$$\sqrt{-8} \tag{1}$$

$$= \sqrt{4 \cdot -2} \tag{2}$$

$$= 2\sqrt{-2} \tag{3}$$

$$= 4\sqrt{2} \cdot \sqrt{-1} \tag{4}$$

$$= 4\sqrt{2} \cdot i \tag{5}$$

Complex numbers can be plotted on the Complex Plane in exactly the same way one can plot (x, y) pairs on a Cartesian Plane. The real component, a, corresponds to the x coordinate while the imaginary component, b, corresponds to the y coordinate. See points plotted on the complex plane below.

The numbers plotted above are $(-0.5 + i)$, $(1 + 3i)$, and $(1.5 - i)$. The magnitude of a complex number is the distance from the origin to the point in question. We accomplish this with the

Pythagorean Theorem, $a^2 + b^2 = c^2$. The magnitude of that complex number c

$$
\begin{aligned}
&= |c| && \text{magnitude} && (6)\\
&= \sqrt{c^* c} && \text{take root of complex conjugate} && (7)\\
&= \sqrt{(a - bi) \cdot (a + bi)} && \text{write c in component form} && (8)\\
&= \sqrt{(a^2 + b^2)} && \text{distribute} && (9)
\end{aligned}
$$

Applying this process for the complex points plotted above the respective magnitudes are: $\sqrt{(-0.5)^2 + (1)^2} \approx 1.12$, $\sqrt{(1)^2 + 3^2} \approx 3.16$, and $\sqrt{(1.5)^2 + (-1)^2} \approx 1.80$. It is important to note that non-zero magnitudes will always be real and positive. In complex arithmetic, the real and imaginary components add separately: $(4 - 8i) + (3 + 5i) = (7 - 3i)$. Multiplication acts just like binomial multiplication: $(2 + 4i) \cdot (0 + i) = (2i + 4i^2)$. Since $i^2 \equiv -1$, the previous expression simplifies to $(-4 + 2i)$.

Complex numbers in mathematics

Where do we use complex numbers in regular math? If asked to find the roots of $y = 4 - (x + 2)^2$, (ie solve for x when $y = 0$) one finds that $x = 0$ or -4 (see Figure 3 the red dots represent these roots). However, if confronted with finding the roots of $y = (x - 2)^2 + 3$, one finds that $x = 2 + 3i$. The purple parabola from Figure 3 doesn't cross the x-axis. The solution to the roots instead gives the complex coordinates of the parabola's minima (orange dot). Thus, one could deduce the second equation never crosses the x-axis and has a minima or maxima at the complex root, $2 + 3i$, without plotting the function.

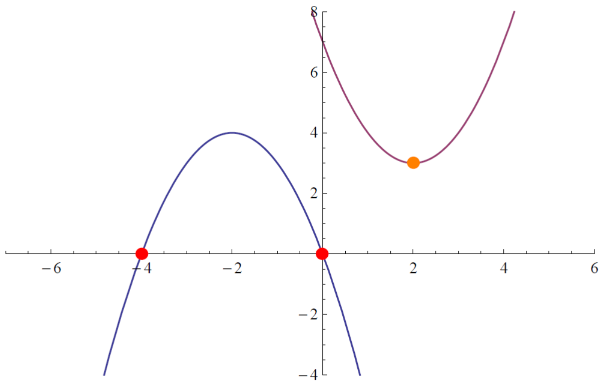


Figure 3: real and complex root example

Part II

Iteration

We know that the Mandelbrot Set is a collection of complex numbers so how does one know if a specific number/point belongs to the set or not? The following algorithm tests some complex point c for membership in the Mandelbrot Set.

$$Z_0 = c \qquad \text{initial condition} \tag{10}$$

$$Z_{n+1} = Z_n^2 + c \qquad \text{recursive function} \tag{11}$$

$$\lim_{n\to\infty} |Z_n| < 2 \qquad \text{condition for set memberships} \tag{12}$$

In truth, all we need to confirm c's membership is to show that $\lim_{n\to\infty} |Z_n|$ doesn't diverge to infinity (ie is bounded). Any point with a magnitude greater than 2 can't be part of the set so a common test for membership is checking a large Z_n against 2 for some large n.

If

$$Z_{100} \overset{?}{<} 2 \tag{13}$$

c is probably a member of the Mandelbrot. Z_n represents a sequence that can do one of 3 things:

it can diverge to infinity, converge on a value, or oscillate indefinitely near the origin. If Z_n diverges, we can classify c as a non-member, if it converges we can classify it as a member, but for c values that result in oscillating Z_n values, it is impossible to verify membership. The larger the n, the higher one's certainty becomes that c is a member; however, it is possible for a specific c to result in a $Z_{1000} < 2$ but $Z_{1001} > 2$.

math behind formula How does the formula $Z_n^2 + c$ work? In practice one decomposes c into its real and imaginary components, $c = a + bi$, where a is the real component, $re(a + bi) = a$, and b is the imaginary component, $im(a + bi) = b$. This means we can write any complex number in the following form, $z = re(z) + im(z) \cdot i$;

$$\begin{aligned} z^2 &= (a + bi) \cdot (a + bi) && (14) \\ &= (a^2 + abi + bia + (bi)^2) && (15) \\ &= (a^2 + 2abi + b^2 i^2) && (16) \\ &= (a^2 - b^2 + 2abi). && (17) \end{aligned}$$

Now it is clear that

$$\begin{aligned} re(z^2) &= (a^2 - b^2) \qquad \text{and} && (18) \\ im(z^2) &= 2ab && (19) \end{aligned}$$

which we can use to show that

$$\begin{aligned} Z_{n+1} &= Z_n^2 + c && (20) \\ &= (a^2 - b^2 + re(c)) + i \cdot (2ab + im(c)) && (21) \end{aligned}$$

This process only takes us from the nth Z to the nth+1 Z. Starting from $Z_0 = c$, we use the equation to find Z_1, then we plug Z_1 into the equation to find Z_2, and so on and so forth.

example Consider point $c = -1.64 - 0.438 \cdot i$. Using initial conditions,

$$Z_0 = -1.64 - 0.438i \tag{22}$$

Since the magnitude is less than two (ie $\sqrt{(-1.64)^2 + (-0.438)^2} \approx 1.7 < 2$) the sequence hasn't diverged yet and we need to calculate next value in the series, Z_1. Using this Z_0 value and equations 21 we find

$$Z_1 = (-1.64 - 0.438 \cdot i)^2 + (-1.64 - 0.438 \cdot i) \tag{23}$$
$$= 0.8577559999999995 + 0.99864 \cdot i \tag{24}$$

Continuing this pattern,

$$Z_2 = -1.9015364940640007 + 1.275178903679999 \cdot i \tag{25}$$
$$Z_3 = 0.3497598018666872 - 5.2875984436160826 \cdot i \tag{26}$$
$$Z_4 = -29.476365381929398 - 4.136778767979529 \cdot i \tag{27}$$
$$Z_5 = 850.1031775537996 + 243.4364049383447 \cdot i \tag{28}$$

$|Z_5| \approx 884.3 \gg 2$, so c can't be a member of the Mandelbrot Set. The c we just examined spiraled away from zero after only a few Z values. Does that always happen? The Mandelbrot Set is a case study in strange attractors and near the set's boundary the sequence Z_n behaves erratically. Moving away from the set's border on the outside and the sequence will diverge without surprise. Furthermore, points far enough from the boundary on the inside will languidly circle around the origin. But the closer to the perimeter the Z_n sequence goes, the larger n is

Outside Set
Inside Set

needed to determine if c is a member or not. Like the Halting Problem, if the sequence neither converges nor diverges to infinity it is impossible to conclusively resolve c's membership status. To get around this obstacle, mathematicians choose an arbitrary number of iterations as a threshold number. If the sequence does not diverge after the threshold number of iterations, the initial point is included as part of the Mandelbrot set. To get higher border detail we simply make this number larger. A common threshold, or escape number, is 100.

0.1 Coloring Schemes

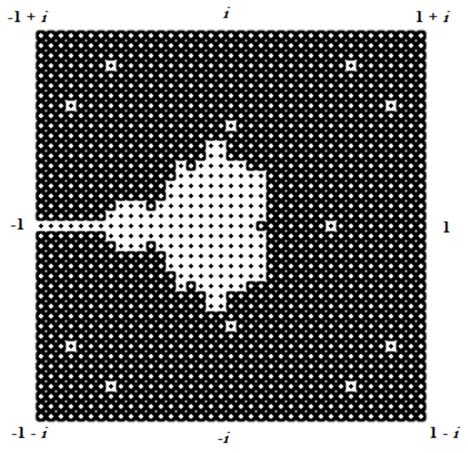


Figure 4: 2 by 2 grid divided into 40×40

To visualize the Mandelbrot set one only needs two colors, such as in figure 1 where Mandelbrot members were colored black and non-members white. It is impossible to examine every number in the complex plane but one could imagine sampling a grid on the complex plane from -1 to 1 vertically and horizontally. Like rows of corn, one could divide this grid into a 100 rows and columns. Now we have 100×100 complex points (a manageable number) to deal with. Use the iteration formula on each of these 10^4 points to classify members and non-members. Furthermore, during this process one can paint each point one of two colors based on membership (see figure 4). Increasing the number of sampled points in our grid improves the resolution. Once the subtleties of the two color problem are understood, it is an easy leap of imagination to add more colors. The most common technique for producing beautiful multi-colored visualizations such as figure 5, is *the escape time* algorithm.

0.1.1 The Escape Time Algorithm

The Escape Time Algorithm counts how many iterations it takes before $Z_n > 2$. For non-members the escape number will be $0 < n <$ *threshold limit*. Members will never escape so

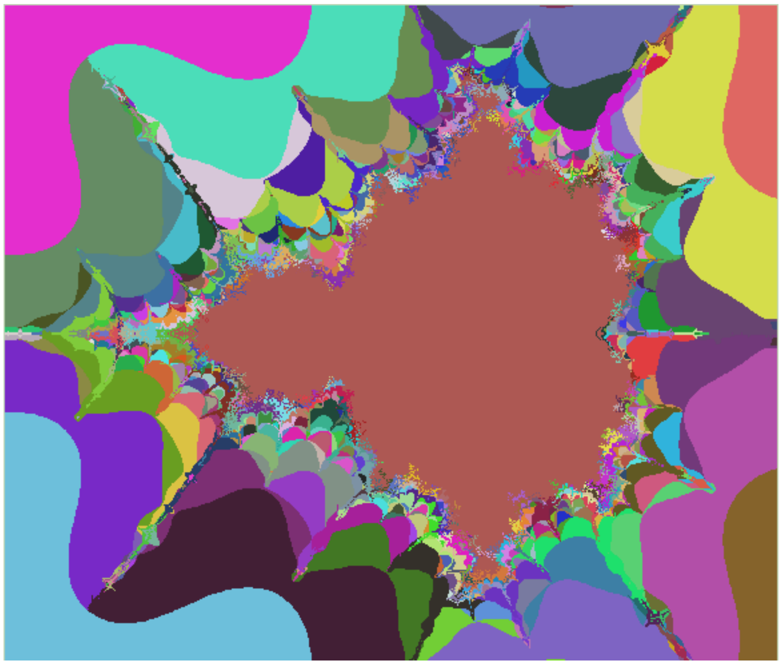


Figure 5: Smudged escape time coloring

they will not have an escape number. Now instead of having members and non-members divided into two groups we have members in one group and non-members grouped by their escape number. Each point can be colored in a unique way based on these values (such as exploiting RGB values).

Python Example

```
from numpy import complex
#returns 0 if point is member, escape number otherwise
def test_point(real,imaginary):
    z = complex(0,0)
    c = complex(real,imaginary)
    escape_number = 0
    max_iter = 100
    while(escape_number < max_iter):
        escape_number += 1
        if abs(z) < 2:  #abs(z) is magnitude of z
            z = z*z + c
        else:
            return escape_number
    return 0
```

Java example

```
private static final int MAXITER = 100;
private static final int ESCAPEBOUNDRY = 4; //2^2
public int testPoint(double real, double imaginary){
    int escapeNumber = 0;
    double zr = 0; double zi = 0;
    double r, i;
    while (escapeNumber < MAXITER){
        escapeNumber++;
        r = zr; i = zi;
        if ((r*r + i*i) < ESCAPEBOUNDRY){
            zr = r*r - i*i + real;
            zi = 2*r*i + imaginary;
        }else
            return escapeNumber;//point outside set
    }return 0;//point inside set
}
```

0.1.2 DEM Algorithm

The *Distance Estimation Method* (DEM) is a procedure that approximates the shortest distance from a given complex point to the Mandelbrot perimeter. The escape-time algorithm's resolution depends proportionally upon the number of sampled points; every detail is colored independently of its neighbors. The DEM can reveal fine structures with fewer sampled points than the escape time algorithm because with the DEM one simply needs to probe points in the proximity of fine structure rather than the exacts points that comprise the fine structure. The DEM is contingent upon the Hubbard-Douady Potential, which is a function developed by Hubbard and Douady during their study of external rays in the Mandelbrot and Julia Set.
was invented by the Mathematician of the same name.

Distance estimation = d = $\frac{G'(n)}{|G(n)|}$

The Hubbary-Douady potential:

$$G(c) = \lim_{n\to\infty} \frac{1}{2^n} \cdot ln|z_n|$$

First derivative of $G(c) =$

$$G'(c) = \lim_{n\to\infty} \frac{1}{2^n} \frac{|z_n|}{|z'_n|}$$

Putting in the values for distance d gives us:

$$d = \lim_{n\to\infty} \frac{|z_n| \cdot ln|z_n|}{|z'_n|}$$

Consider the Hubbard-Douady Potential a given. If you would like to know more about where it comes from you can read about it online.

So now we have an equation for approximate distance from Mandelbrot perimeter–how does it work? Well first we note that d already uses z_n which we know how to calculate. Unfortunately it also requires we know dz_n (ie z'_n). What is the first derivative of z? If $z_{n+1} = z_n^2 + c$ then $d(z_{n+1}) = 2z_n \cdot dz + 1$. WHY? The result makes sense because we are differentiating z_{n+1} with respect to C. Thus, $d(z_n^2)$ is $2z_n \cdot dz$ from product rule and dC with respect to C is 1. Therefore, $d(z_{n+1}) = 2z_n \cdot dz + 1$. To modify the test point functions above to calculate the point-to-perimeter distance, we would only change a few things. In addition to updating z_n we will also be updating dz_n. dz can be initialized as 0 because $z_0 = 0$ so $d(z_0) = 0$. If you choose to make $z_0 = 0 + C$ then dz starts as 1. Then at the end, instead of returning the escape number, one returns the distance estimation value described above:

$$d \approx \frac{|z_n| \cdot ln|z_n|}{|z'_n|}$$

by inserting the calculated values for z and dz. It is also important to note that we no longer care about escape conditions because we need to calculate large n for each point to get a more accurate distance estimation [2]. The python example would be modified as follows:

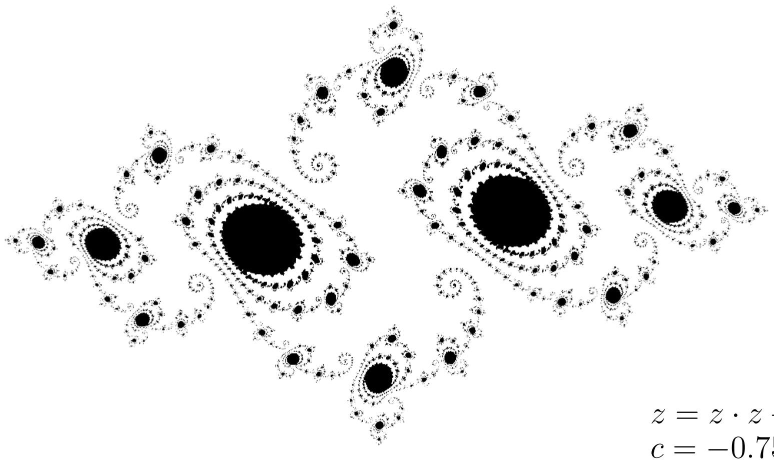
Python DEM Example

```
from numpy import complex,log
def test_point(real,imaginary):
    z,c = complex(0,0),complex(real,imaginary)
    dz,count,max_iter = 0,0,100
    while(count < max_iter):
        count += 1
        dz = 2*z*dz + 1
        z = z*z + c

    return abs(z)*log(abs(z))/abs(dz)
```

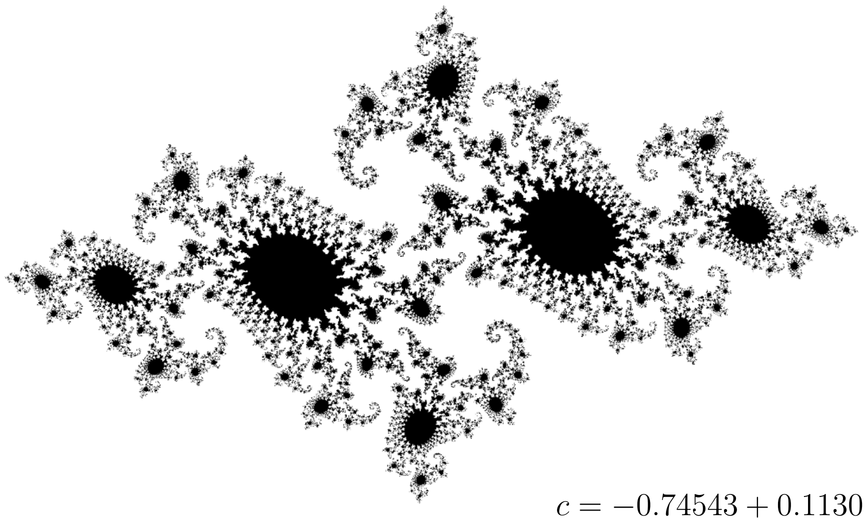
Part III

The Julia Set



$z = z \cdot z + c$
$c = -0.75 + 0.11i$

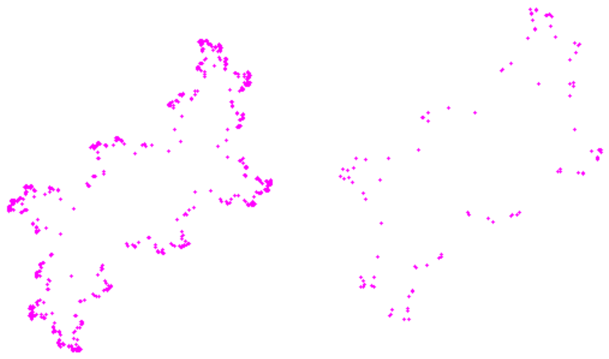
The Julia Set is another fractal that arises from an iterative formula that differs only slightly from the Mandelbrot Set. In fact, to make a Julia Set Fractal we use the same formula with $z_{n+1} = z_n^2 + c$. The only difference is that the c we use is the same for every sampled point on the complex plane instead the unique c at the point being sampled. That unique point is set to be z_0. Then for every successive iteration you use the predetermined c value. It is hard to imagine that this slight variation would give such drastic results, but it does.



$c = -0.74543 + 0.11301i$

0.2 IIM

IIM stands for Inverse Iteration Method; the idea is to calculate points backwards from usual method. For the regular Julia Set we have a specific fractal associated with C constant and the typical equation $z_{n+1} = z_n^2 + c$ where $z_0 =$ coordinate on complex plane we are examining. Let us rearrange the above equation to find z_n with respect to z_{n+1}. Using simple algebra we find that $z_n = \pm\sqrt{z_{n+1} - c}$. The boundary of the Julia Set, instead of repelling points, now attracts them. The neat thing is we can now pick any point anywhere on the complex plane and apply this formula. If we plot every z_n value as we apply the formula we see that z never converges to the boundary but oscillates around the perimeter, tracing the outline of the Julia Set associated with the c constant being used.



From the above images it can be seen that a lot of information is lost from the fractals seen earlier in the Julia Set section. We have no concept of coloring, and most of the nuance at the border is missed. A reason that a certain amount of information is lost from the boundary is that different parts of the strange attractor perimeter ”attract” stronger than other sections. With higher precision, stunning visuals are possible; however, optimizations in the formula are required that we will not delve into here. The chief benefit of using IIM at all is that instead of surveying thousands to millions of points and running 100 to a 1000 computations on each one, we need only apply a few thousand operations on a net total of one point. The smokey outline of the Julia Set may look meager, but it can potentially be calculated 100,000 times faster than the other described methods.

Part IV

Fractal Variations

The iterative equation we have been using so far, $z^2 + c$, is not in of itself special. To a certain extent we study it purely because the equation is simple and the results profitable. With that said, we could choose any iterative equation with any escape boundary to explore. One of these explorations led to the haunting and beautiful ”Burning Ship ” fractal seen below. The ”Burning Ship” fractal gets its name from its similarity to the silhouette of the prow of a ship on fire.

Buddhabrots

Another beautiful and innovative idea shows up in the so-called ”Buddhabrot.” Buddhabrots use exactly the same scheme for producing the Mandelbrot fractal except instead of coloring each point after a certain number of iterations every z_n is plotted as a single black point from $n = 0$ to $n =$ iteration limit. The result is a beautiful ghostly outline of the Buddha.

Burning Ship

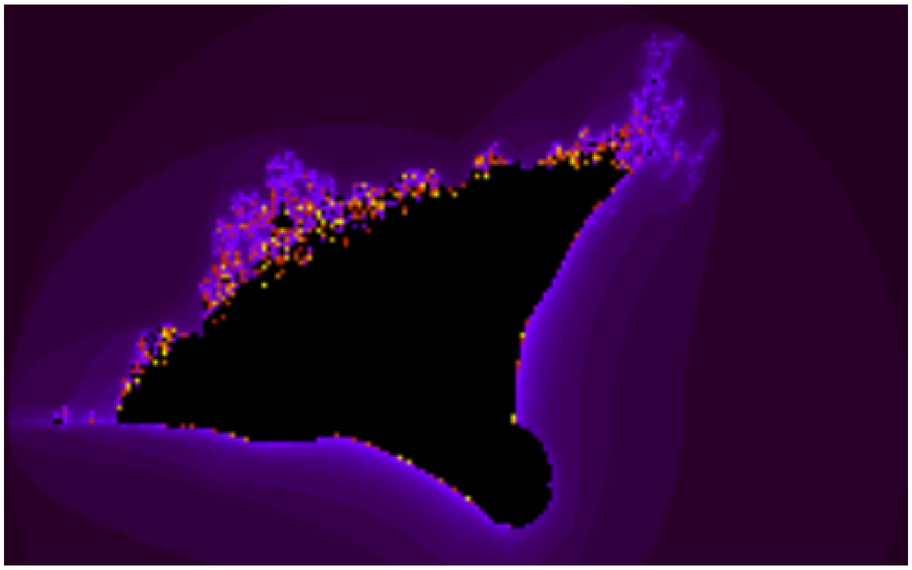


Figure 6: $z_{n+1} = (|re(z_n)| + i \cdot |im(z_n)|)^2 + c, z_0 = 0$

conclusion

The variations we can apply to the fractal generation process are limitless.

References

[1] John Horgan *Who Discovered the Mandelbrot Set?* Scientific America: March 13, 2009

[2] Dr. Lindsay Robert Wilson *Distance estimation method for drawing Mandelbrot and Julia sets* 20/11/12

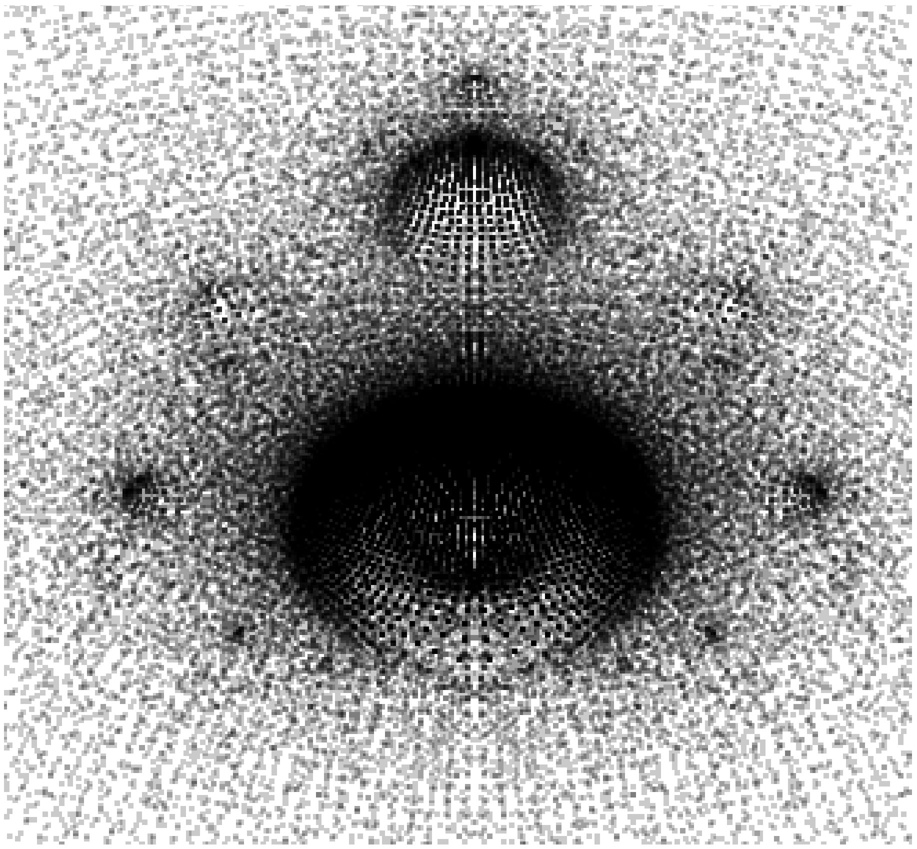


Figure 7: $Buddhabrot$

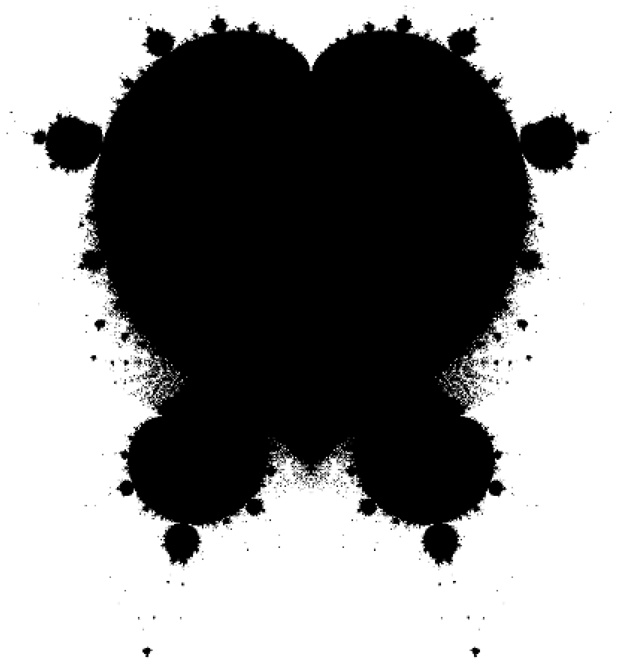


Figure 8: $\mathbf{z_{n+1} = z_n^{\pi-1} + c}$