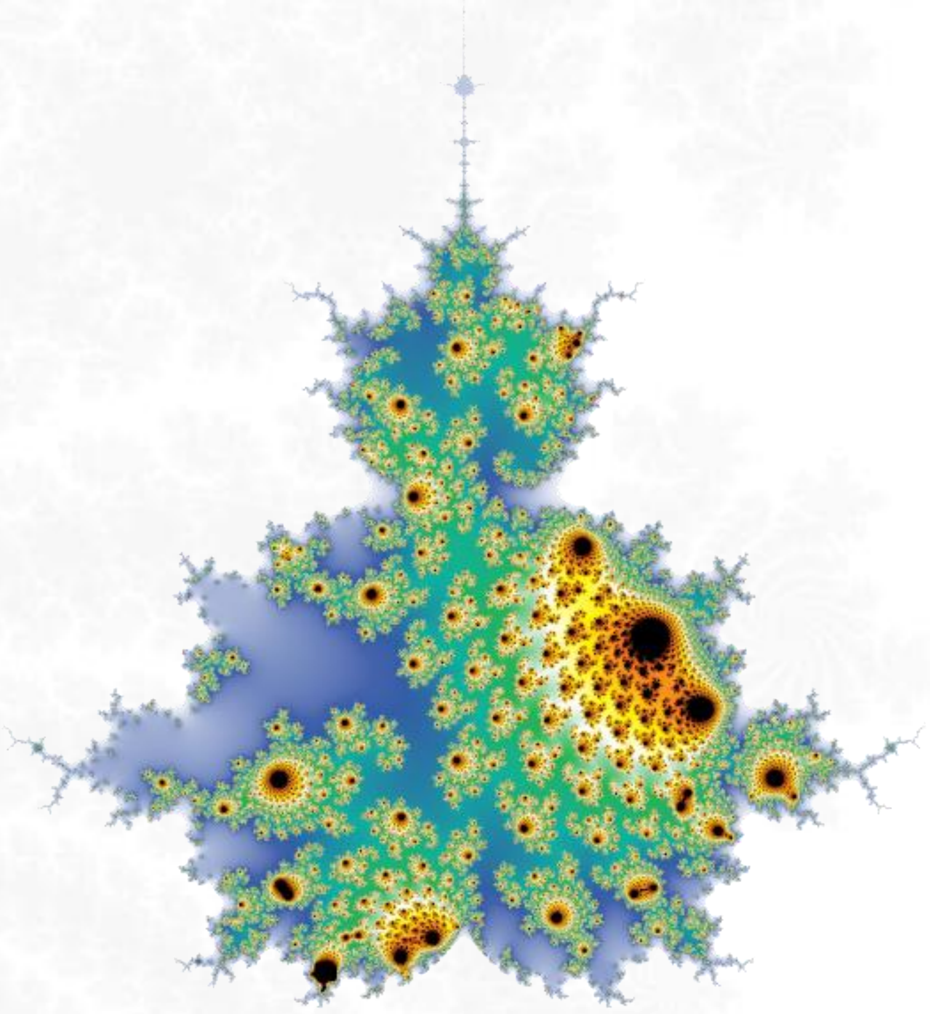
Juliter Transformation

Sergio CT



«Being a language, mathematics may be used not only to inform but also, among other things, to seduce...»

Benoît Mandelbrot, creator of the Theory of Fractals

https://fractalfun.es/en

Content

Broadly speaking, the **Juliter Transformation** consists of altering the behaviour of the Mandelbrot algorithm, in such a way that $\boldsymbol{C}$ acquires the value of a constant $\boldsymbol{J}$ in a given iteration $\boldsymbol{I}$. At that moment, the algorithm will stop calculating the Mandelbrot Set and will use the rest of the iterations to calculate the Julia Set, which will be embedded within the Mandelbrot calculated so far and influenced by the value of $\boldsymbol{Z}$ after those first iterations.

Introduction

This document describes how to apply the "Juliter Transformation" to fractals represented by the Mandelbrot Method. This method consists of analysing the behaviour of the iterative formula $\boldsymbol{Z_{n+1}} = \boldsymbol{f}(\boldsymbol{z_n}) + \boldsymbol{C}$ for each point $\boldsymbol{C}$ of the complex plane, where $\boldsymbol{f}$ is a previously chosen function and the initial value of $\boldsymbol{Z}$ is the complex number (0, 0i).

When the sequence of values of $\boldsymbol{Z}$ is bounded, $\boldsymbol{C}$ is considered to belong to the Mandelbrot Set and, in general, it is assigned the colour black. On the contrary, if the sequence of values of $\boldsymbol{Z}$ exceeds a previously established escape radius, a colour is assigned to $\boldsymbol{C}$ that depends on the iteration number in which the orbit of $\boldsymbol{Z}$ exceeds said limit.

Also note that the Mandelbrot Set represents all related Julia Sets. Which are calculated with the same iterative formula, but in this case $\boldsymbol{Z}$ would be the variable that runs through the complex plane, and $\boldsymbol{C}$ would have a previously established constant value.

Algorithm

The “Juliter Transformation” arises during the development of new functionalities for the FFExplorer application, belonging to the FractalFun Project, and as a test of the hypothesis that fractals can be represented with an adjustable mixture between the Mandelbrot and Julia methods.

The transformation is extremely easy to understand and implement. For example, given the function $\boldsymbol{f}(\boldsymbol{z}) = \boldsymbol{Z}^2$, we have the Mandelbrot Set formula $\boldsymbol{Z_{n+1}} = \boldsymbol{Z_n^2} + \boldsymbol{C}$, for which we can implement the following algorithm that represents the fractal:

pseudo-code:

```
CONST Palette(Lenght) AS COLOR = [custom colour list]
CONST MaxIterations AS INTEGER = 256
FOR EACH Pixel{
        VAR Iteration AS INTEGER = 0
        VAR PixelX AS INTEGER = [X coordinate of the pixel]
        VAR PixelY AS INTEGER = [Y coordinate of the pixel]
        VAR Px AS DOUBLE = [X coordinate of the pixel in the complex plane]
        VAR Py AS DOUBLE = [Y coordinate of the pixel in the complex plane]
        VAR Z as Complex = (0, 0i)
        VAR C as Complex = (Px, Py)
        VAR Bailout AS DOUBLE = 2
        VAR Colour AS COLOR = NULL
        DO WHILE Iteration < MaxIterations AND Z.Magnitude ≤ Bailout {
                Z = Z^2 + C
                Iteration += 1
        }
        IF Iteration = MaxIterations { Colour = Black }
        ELSE { Colour = Palette(Iteration MOD Palette.Lenght) }
        PLOT(PixelX, PixelY, Colour)
}
```

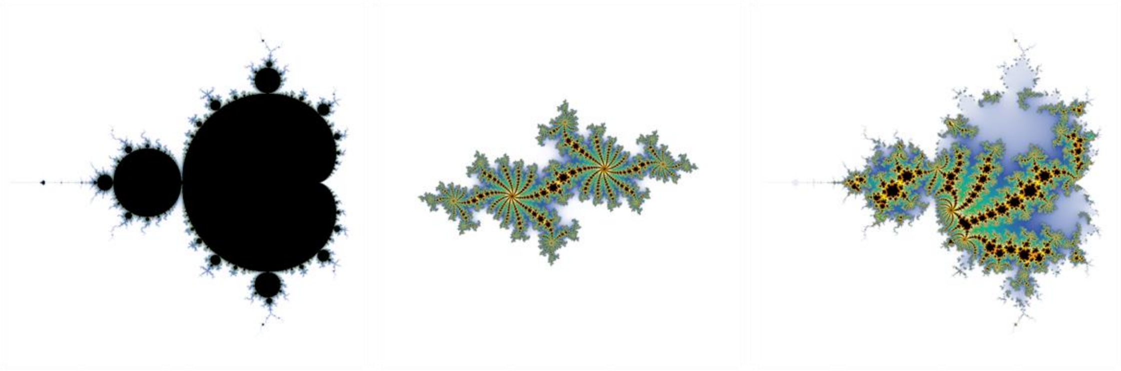
To add the transformation to the algorithm described above we need two new parameters, which are what give Juliter its name: $\boldsymbol{J}$, which will be the complex constant that represents the **Jul**ia Set that we want to use in the mix; and $\boldsymbol{I}$, which will be the value of the **iter**ation from which you want to use the constant $\boldsymbol{J}$. To do this, you just have to add the following condition after increasing the number of iteration in the loop:

pseudo-code:

```
IF Iteration = I { C = J }
```

Important: if the algorithm needs to modify $\boldsymbol{C}$ after calculating $\boldsymbol{Z}$, this modification must be made after the transformation, so that it is applied correctly in the next iteration.

In the following image you can see the result of applying the “Juliter Transformation” to the Mandelbrot Set with values of $\boldsymbol{J} = (-\mathbf{0,7042}, -\mathbf{0.2865}\boldsymbol{i})$, $\boldsymbol{I} = \mathbf{16}$.



Mandelbrot Set
Iterations = 256

Julia Set
C = (-0.7042, -0.2865i)

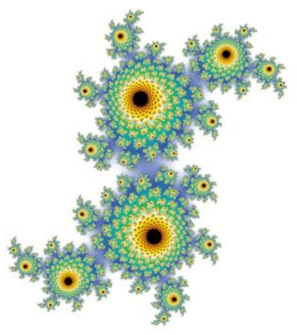
Juliter
J = (-0.7042, -0.2865i), I = 16

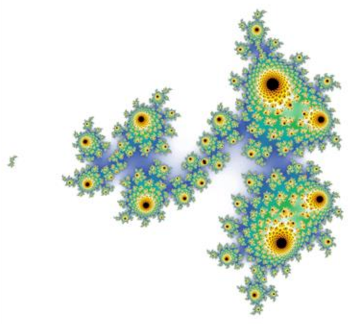
Features

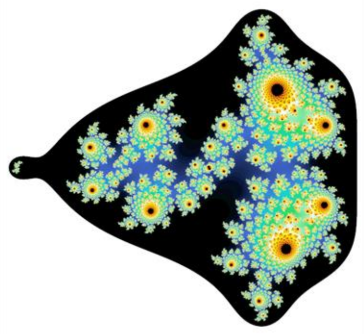
1. If $\boldsymbol{I}$ is equal to 1, the transformed Julia Set* whose identity is $\boldsymbol{J}$ will be obtained, and if $\boldsymbol{I}$ is equal to the maximum number of iterations, the Mandelbrot Set will be obtained, leaving all intermediate values of $\boldsymbol{I}$ to mix both sets.

   * Note that the function is processed before the transformation is applied for the first time, so the resulting Julia Set for $\boldsymbol{I} = \mathbf{1}$ might not match the standard Julia Set for $\boldsymbol{C} = \boldsymbol{J}$. Its appearance will depend on the function and / or the algorithm.

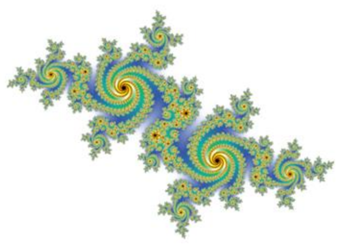
2. By assigning the constant value $\boldsymbol{J}$ to $\boldsymbol{C}$ in iteration $\boldsymbol{I}$ of the iteration process, the Transformed Julia Set will be bounded within the representation with $\boldsymbol{I}$ iterations of the Mandelbrot Set, using the rest of the iterations in the definition of the Julia Set.

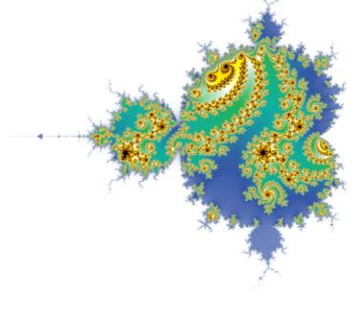

Julia Set
C = (0.3, -0.45i)

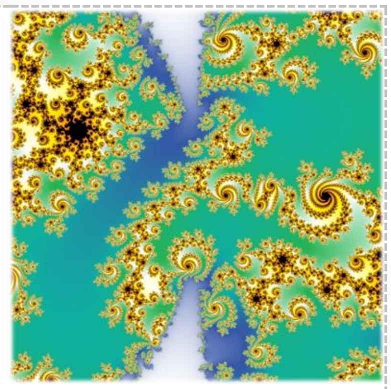

Juliter
J = (0.3, -0.45i), I = 4


Juliter with 256 iterations on a black Mandelbrot Set with 4 iterations

3. The Transformed Julia Set shows continuity as it extends between convergent zones of different periods of the Mandelbrot Set. Understanding by "continuity" that there are no cuts or jumps in the fractal image.


Julia Set
C = (-1.24967, -0,03172i)


Juliter
J = (-1.24967, -0,03172i), I = 32


Juliter transit between period 1 and period 2 bulbs of Mandelbrt Set

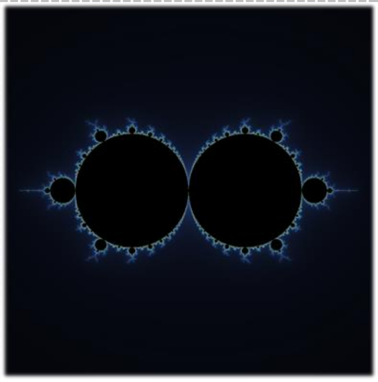
4. Depending on the values of J and I, the fractal calculation process can be sped up considerably, reducing the calculation time with respect to that of a Mandelbrot Set with the same number of iterations.

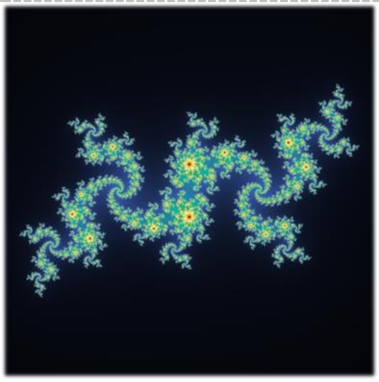
Reflection

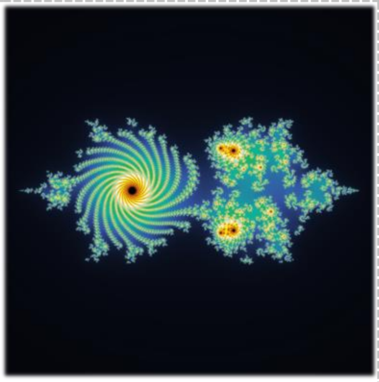
The “Juliter Transformation” is capable of creating explorable zones within the convergence zones that, with some coloring algorithms, lack aesthetic interest. It is a simple transformation to implement and therefore can be included in any fractal graphing application without much effort.

Testing with FFExplorer seems to indicate that the transformation performs well for any function handled by the Mandelbrot method. So far the following have been successfully tested: Mandelbrot, Lambda, Phoenix, Manowar, Magnets, Burning Ship, Nova and Spider.

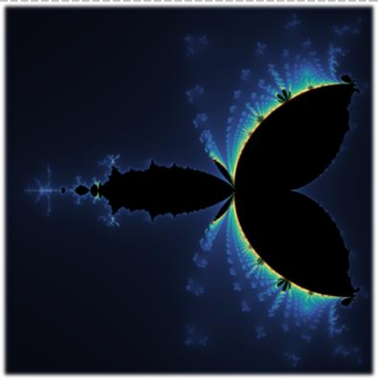
Lambda

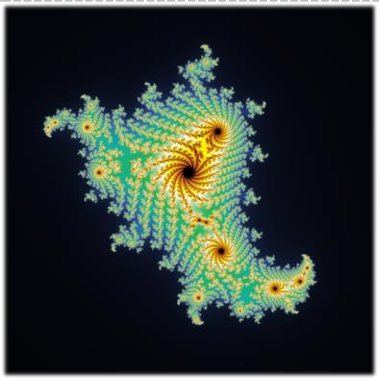

Lambda Mandelbrot

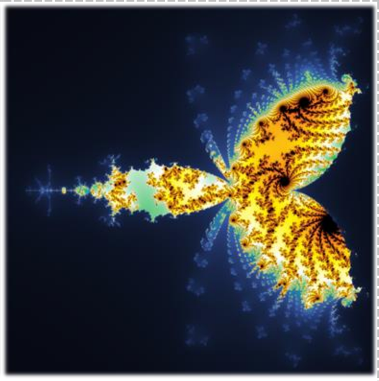

Lambda Julia


Lambda Juliter

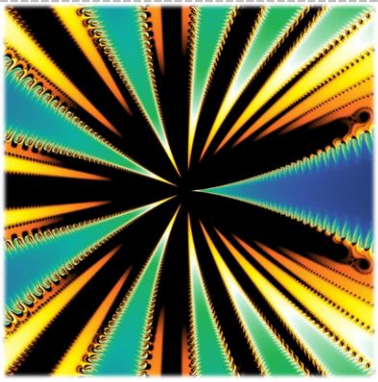
Phoenix

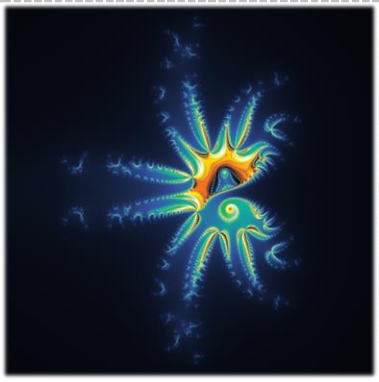

Phoenix Mandelbrot

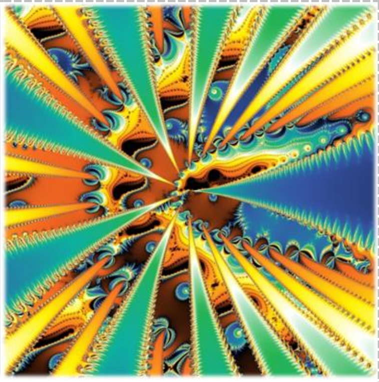

Phoenix Julia


Phoenix Juliter

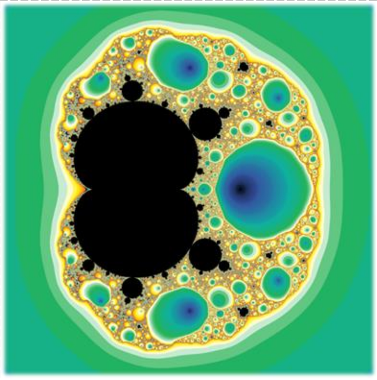
Manowar

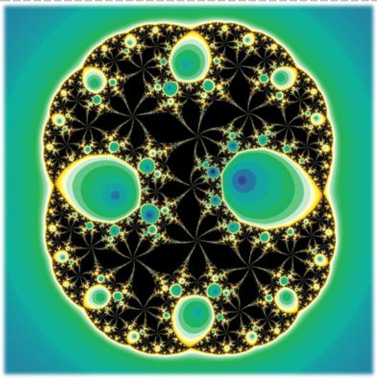

Manowar Mandelbrot (zoom)

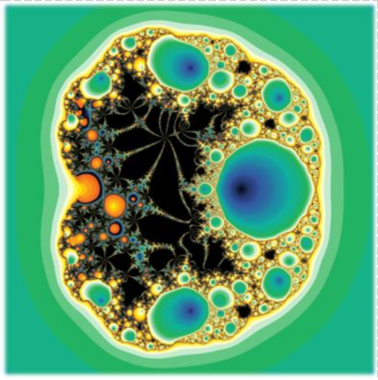

Manowar Julia


Manowar Juliter (zoom)

Magnet 1

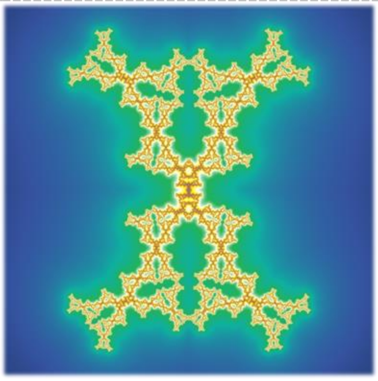

Magnet 1 Mandelbrot

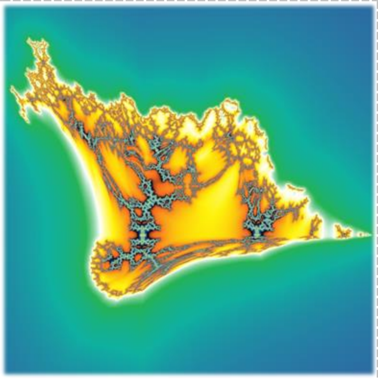

Magnet 1 Julia


Magnet 1 Juliter

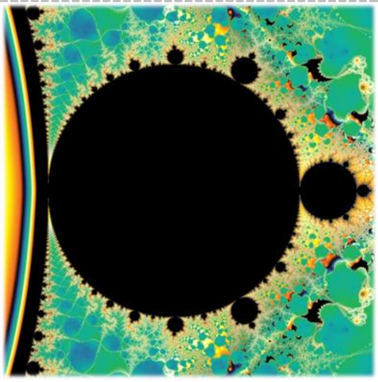
Burning Ship

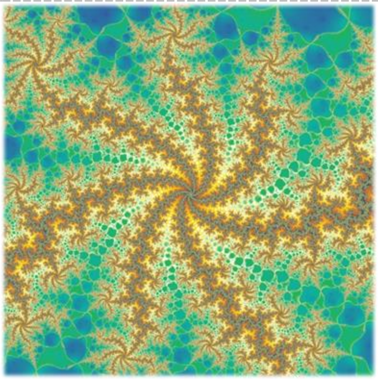
Burning Ship Mandelbrot

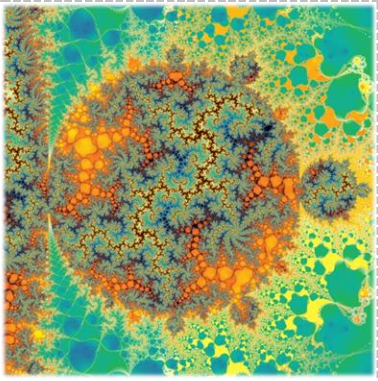

Burning Ship Julia


Burning Ship Juliter

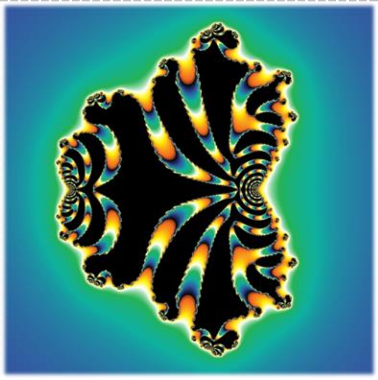
Nova

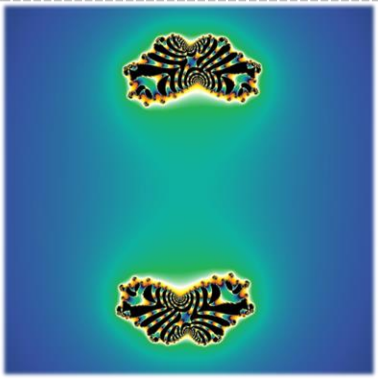

Nova Mandelbrot (zoom)

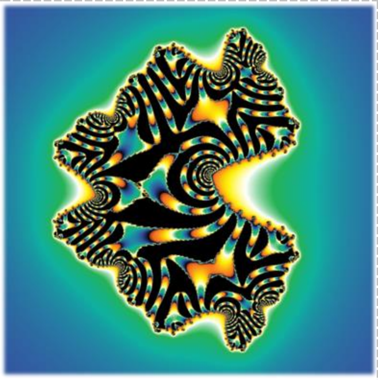

Nova Julia


Nova Juliter (zoom)

Spider


Mandelbrot Spider


Julia Spider


Juliter Spider