Generation of Fractals using Polynomial Iterations in Polar Coordinates

S. Charonov, Horiba France SAS, 455 Avenue Eugène Avinée, 59120 Loos, France

E-mail : serguei.charonov@horiba.com

Abstract.

This article describes an algorithm for generating fractals using polar coordinates. The classic Julie and Mandelbrot polynomial iteration applied to a complex number is replaced by a separate iteration for distance and angle. By varying the polynomial parameters of the iteration functions, a wide range of attractive images can be created. Distance and angle values are used to colorize fractal images.

Keywords.

Fractals, Julia set, Mandelbrot set.

Introduction.

Beginning with the classical studies of Benoit Mandelbrot (1), who expanded on the work of Gaston Julia (2), many researchers have studied Julia sets and Mandelbrot sets from different angles (3-8). Various generalizations of these sets have been proposed. Many iterative processes (such as Ishikawa, Noor, SP, MM iterations) have been successfully used to generate fractals.

In this article, following the author's previous work (9), the classical iteration $z \to z^2 + c$ has been replaced, using polar coordinates, by an iteration with separate functions for distance and angle. A partial escape criterion for iteration is proposed along with a graphical presentation of the result sets.

Algorithms and Results.

The Mandelbrot set M for the function $Q_c(z)$ can be defined in the following way (4):

$$M = \{c \in C : \{Q^n_c(0)\} \text{ does not tend to } \infty \text{ as } n \to \infty\}, \qquad (1)$$

For classical Mandelbrot set $Q_c(z) = z^2 + c$.

The proposed iteration scheme uses the polar representation of the complex number $z = re^{i\varphi}$. $Q_c(z)$ can be defined as

$$Q_c(z) = T(r, \varphi) + c \tag{2}$$

where

$$T(r, \varphi) = r^{s(r)} e^{ip(\varphi)} \tag{3}$$

and s and p are polynomials $s(r) = A_K r^K + \ldots + A_1 r + A_0$, and $p(\varphi) = B_N \varphi^N + \ldots. + B_1 \varphi + B_0$ To obtain the classical Mandelbrot set, the parameter values must equal 0, except $A_0 = B_1 = 2$.

The escape criterion can be obtained as follows: if there exists $k \geq 0$ such that for the sequence $z_k, z_{k+1}, \ldots$

$$|z_n| < |z_{n+1}| \tag{4}$$

and the sequence is not limited, then $Q^n{}_c(z) \to \infty$ as $n \to \infty$.

$$|z_n| = r_n < |z_{n+1}| = r_{n+1} = |T(r_n, \varphi_n) + c| \tag{5}$$

If $r^{s(r)} \geq |c|$ then

$$r_n < r_n^{s(r)} - |c| \leq |T(r_n, \varphi_n) + c| \tag{6}$$

and in the condition $r > 1$ and $s(r) > 1$, the value of the escape threshold r can be calculated as a solution to the equation

$$r^{s(r)} - r > |c| \tag{7}$$

$$s(r) > A_K r^K - \max(A)\, r^{K-1} = v - vw/r \tag{8}$$

where $v = A_K r^K$ and $w = \max(A)/ A_K$. If $K = 0$ then $w = 0$.

$$v - vw/r \geq v - vw = v(1-w) \tag{9}$$

$$r^{v(1-w)} - r > |c| \tag{10}$$

Considering that the sequence $|z_k|, |z_{k+1}|, \ldots$ is bounded by sequence $u_n = |z_n|, u_{n+1}, \ldots, |z_n| \geq u_n$, where

$$u_{n+1} = u_n^{s(r)} - |c| \tag{11}$$

and that

$$u_{n+2} - u_{n+1} \geq u_{n+1} - u_n > 0 \tag{12}$$

then $Q^n{}_c(z) \to \infty$ as $n \to \infty$.

Figures 1 shows the escape threshold for some sets of **A**.

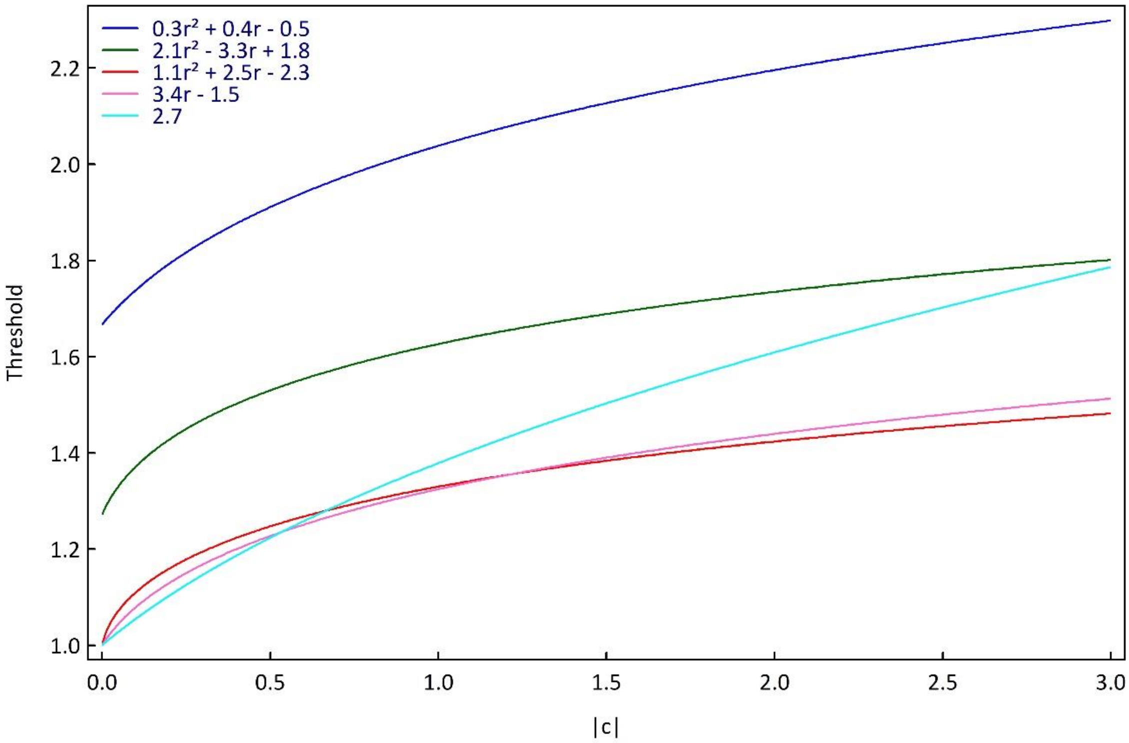


Fig 1. The escape threshold as function |c| for different parameters **A**.

To create a color image the distance |z| and angle φ is mapped in the palette lookup table. Averaging these values provides a smoothing effect. Figure 2 shows examples of fractals with some parameter sets **A**, **B** and different color schemes. All fractal images in this work are y-mirrored and stretched in x by factor 1.6. The maximum order of polynomials is 2.

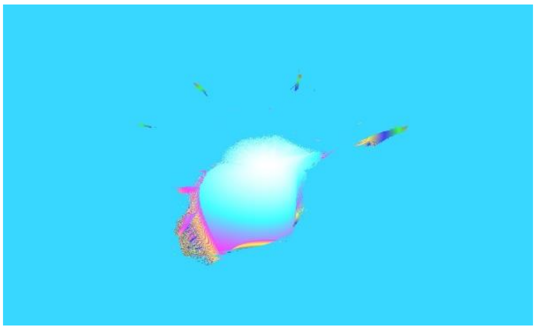
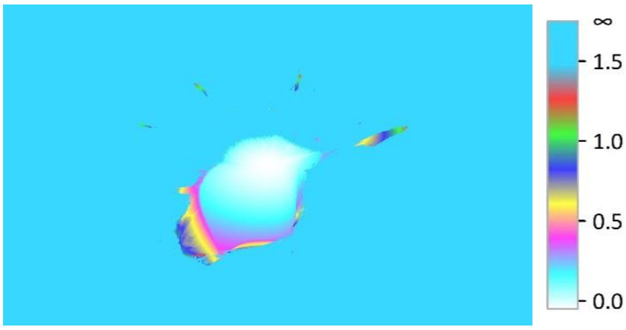



Figure 2. Fractal "Bird". **A** = 1.3, -3.45, 45.0. **B** = 0.0, 0.0, 2.0. Coloring by |z|.

Figure 3. Fractal "Bird". **A** = 1.3, -3.45, 45.0. **B** = 0.0, 0.0, 2.0. Coloring by Mean(|z|).

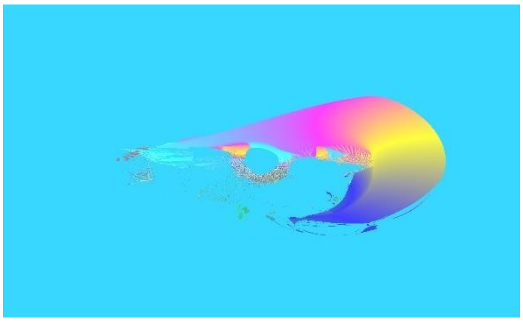


Figure 4. Fractal "Shrimp". **A** =0.73, 1.26, -2.41. **B** = 2.51, 2.87, -0.35. Coloring by φ.

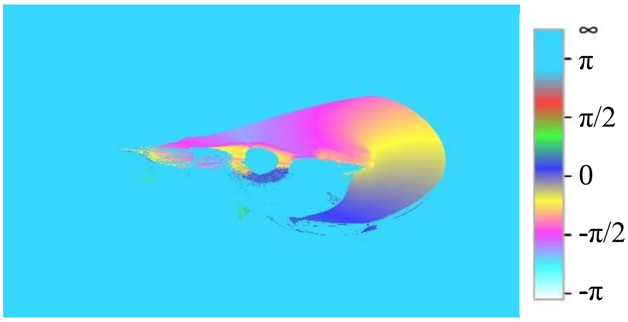


Figure 5. Fractal "Shrimp". **A** = 0.73, 1.26, -2.41. **B** = 2.51, 2.87, -0.35. Coloring by Mean(φ).

The generated images enable the identification of various types of fractal types. The first type corresponds to the classical one with a fractal border between the sets of escaped and bounded points. Figure 6 shows a sequential zoom of the "head" of the "Bird" fractal.

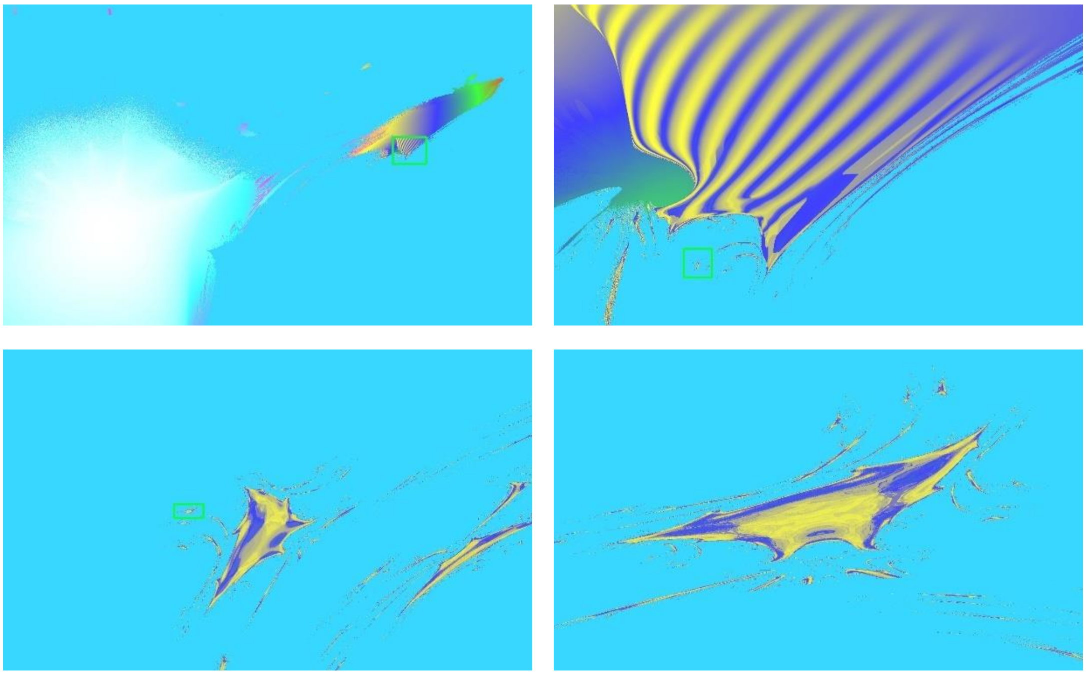


Fig 6. A sequential zoom of the "head" of the "Bird" fractal allows to see a self-similar pattern generated by the iterative process. The fractal border separates the escaped and non-escaped points. **A** = 1.3, -3.45, 45.0. **B** = 0.0, 0.0, 2.0. Coloring by |z|.

The second type can be considered as inversed to first. In comparison with the classical situation, the set M can be defined as

$$M = \{c \in C : \{Q^n{}_c(0) < Thr\} \ \text{ as } n \to \infty\} \tag{13}$$

The escaped set is split into an infinite number of unconnected subsets, bounded inside non-escaped region. The Figure 7 shows the sequential zoom of the "antenna" of the "Bracelet" fractal.

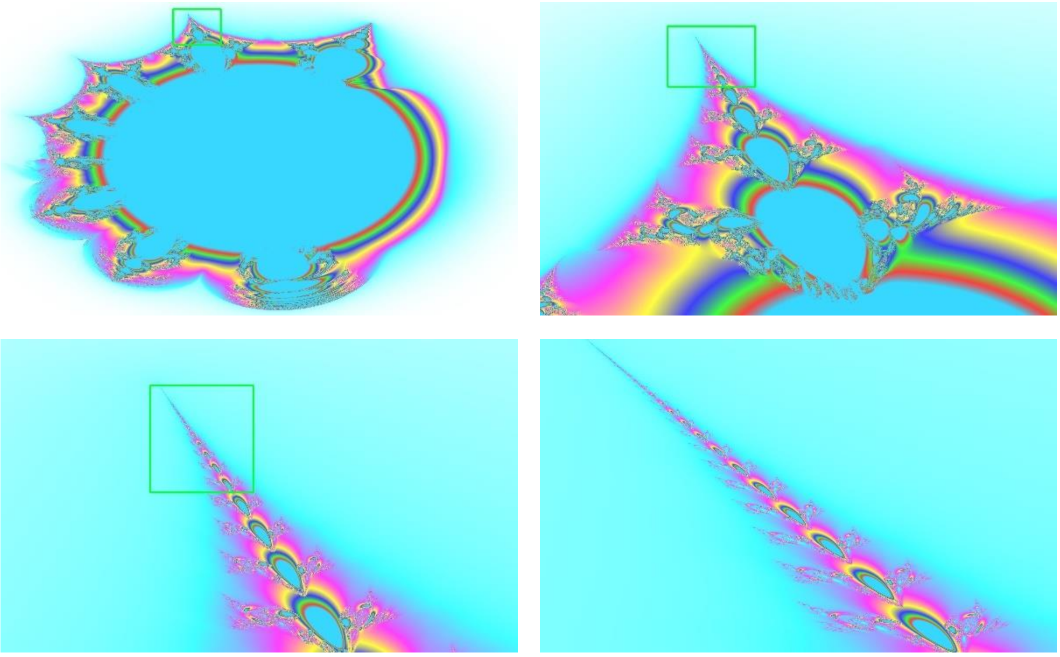
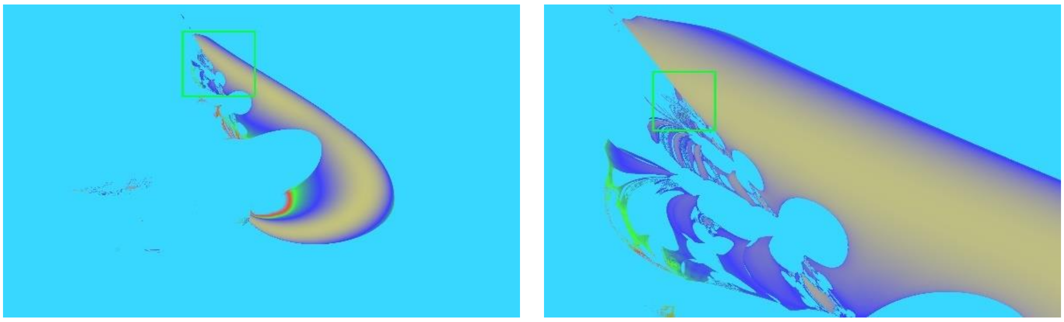


Fig 7. A sequential zoom of the "Bracelet" fractal allows to see a self-similar pattern generated by the iterative process. **A** = 2.52, 0.15, -0.88. **B** = -2.43, 0.45, -2.81. Coloring by Mean(|z|).

The third type is characterized by smooth borders. The Figure 8 shows the sequential zoom of the" Perforated Nose" fractal.



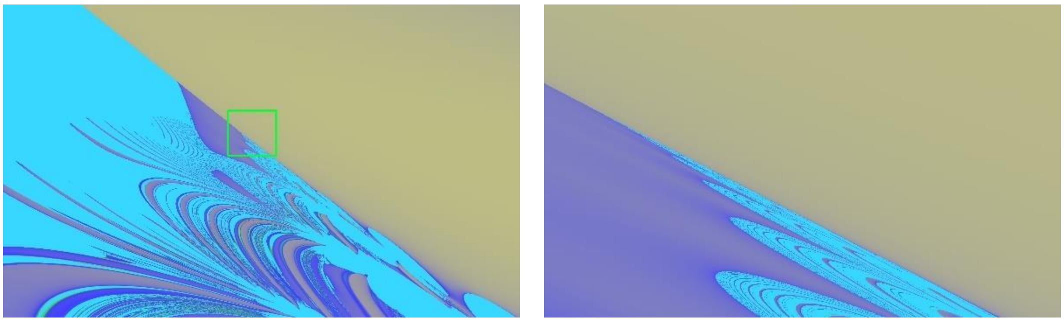


Fig 8. A sequential zoom of the "Perforated Nose" fractal shows smoothed a self-similar pattern generated by the iteration process. **A** = 0.73, 0.26, -0.85. **B** = 0.83, -1.30, 0.77. Coloring by Mean(|z|).

Figure 9 shows examples of the Mean(|z|) value as a function of the iteration step for the escaped and bounded points of these 3 fractal types.

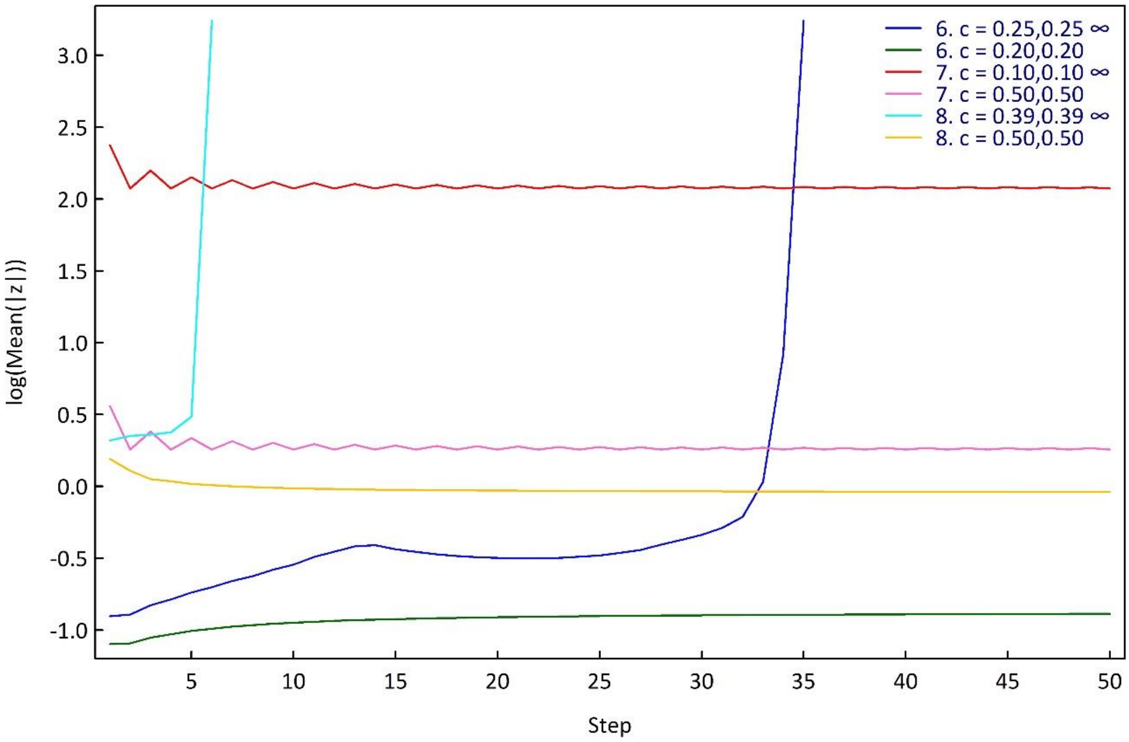


Fig 9. The logarithm of the average distance as a function of the iteration step number for the fractals shown in Fig. 6,7,8. The ∞ symbol indicates escaped tracks.

There are many other types of iteration results and self-similar patterns generated by changing parameters **A** and **B**. Figures 10-22 show an image gallery with a wide variety of fractal shapes and structures.

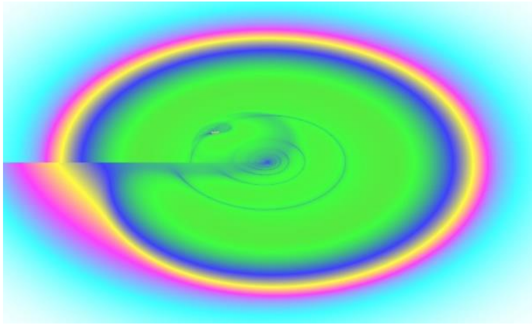


Figure 7. Fractal "Spiral". **A** = -0.12, 0.71, 0.38. **B** = -0.99, 0.89, 0.94.

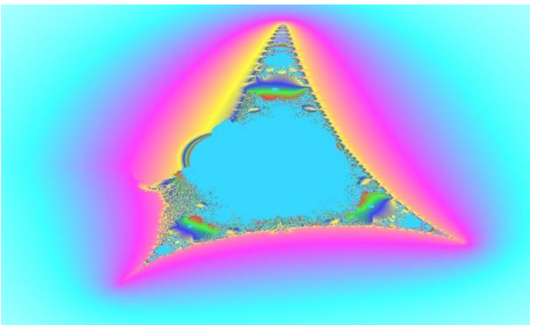


Figure 8. Fractal "Triangle". **A** = -0.21, -2.24, -1.23, **B** = 0.0, 0.0, -2.09.

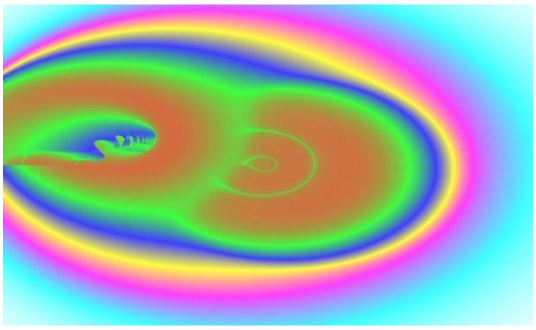


Figure 9. Fractal "Infinity". **A** = 0.01, 0.16, 0.26. **B** = -0.43, -1.24, 0.49.

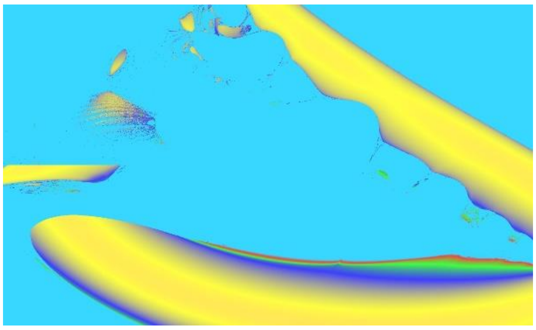


Figure 10. Fractal "Crocodile breakfast". **A** = 0.40, -1.33, -2.06. **B** = 0.87, -1.17, -1.39.

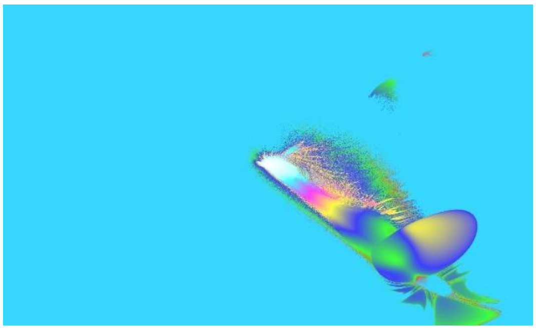


Figure 11. Fractal "Fish". **A** = -0.61, 1.01, -2.67. **B** = 0.93, -1.48, 1.25.

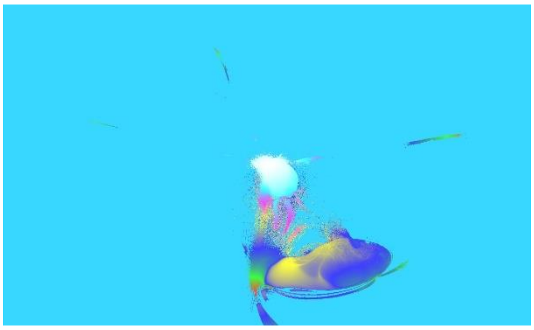


Figure 12. Fractal "Duckling". **A** = -1.15, 2.58, -2.66. **B** = 2.07, -2.73, 1.57.

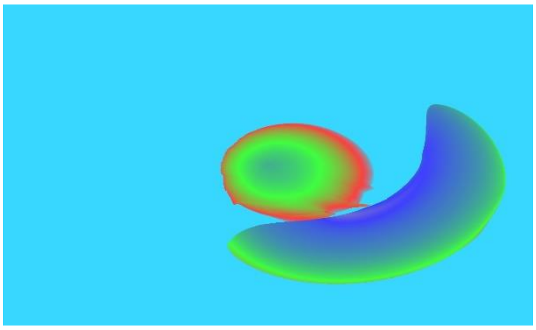


Figure 13. Fractal "Banana Snail", **A** = -0.10, 0.96, 1.78. **B** = 0.74, -0.58, 0.28.

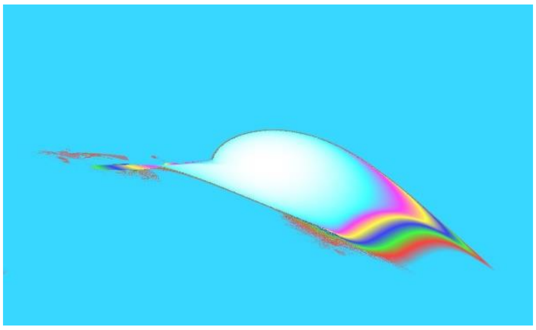


Figure 14. Fractal "Dolphin", **A** = -0.29, -0.44, -1.81. **B** = -0.20, -0.45, 2.74.

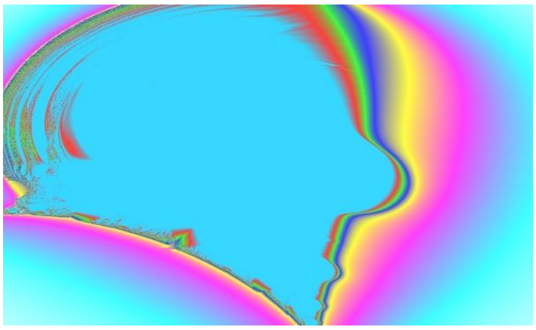


Figure 15. Fractal "Face profile", **A** = -0.56, -0.98, 0.74. **B** = 0.01, -0.82, -1.34.

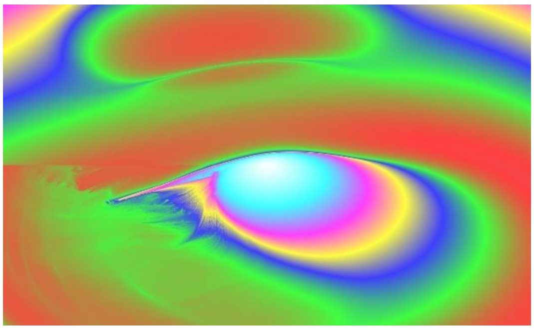


Figure 16. Fractal "Eye". **A** = 0.26, 0.56, -1.67. **B** = -0.52, 0.37, 1.24.

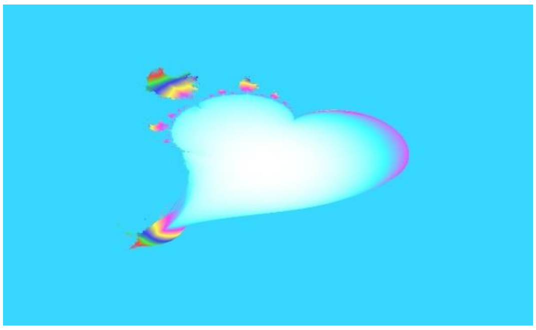


Figure 17. Fractal "Heart". **A** = 0.90, 1.12, 1.51. **B** = 0.35, 1.09, 2.88.

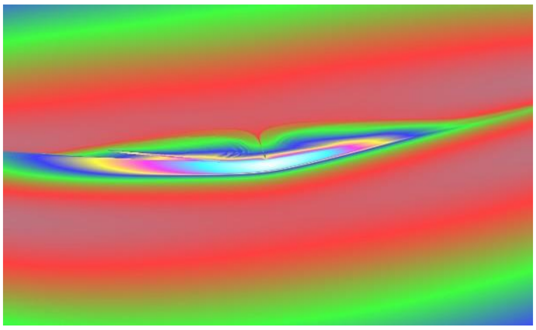


Figure 18. Fractal "Smile". **A** = 0.10, -0.97, 2.44. **B** = -0.09, -0.23, 1.22.

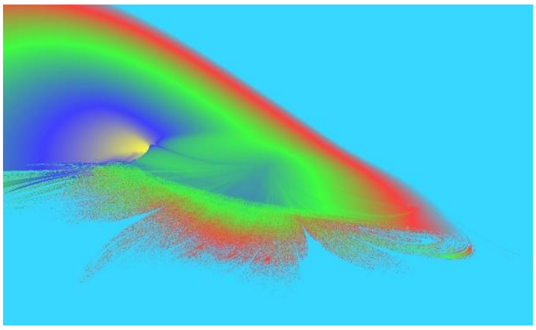


Figure 19. Fractal "Clam" **A** = -0.49, -1.94, -1.88. **B** = 0.10, 0.00, 0.24.

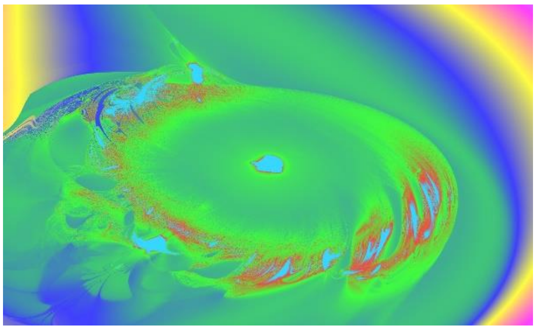


Figure 20. Fractal "Cyclone". **A** = 0.50, 0.85, 0.86. **B** = -0.71, 1.36, -0.54.

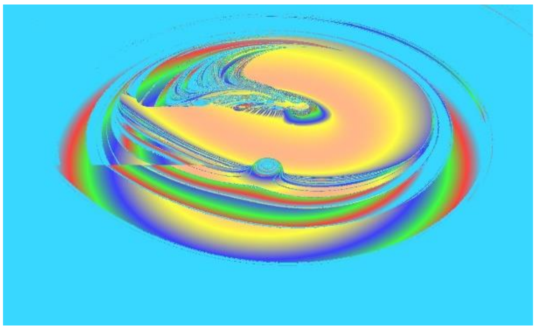


Figure 21. Fractal "UFO". **A** = 0.18, 0.50, 1.61. **B** = -0.38, 2.20, -0.19.

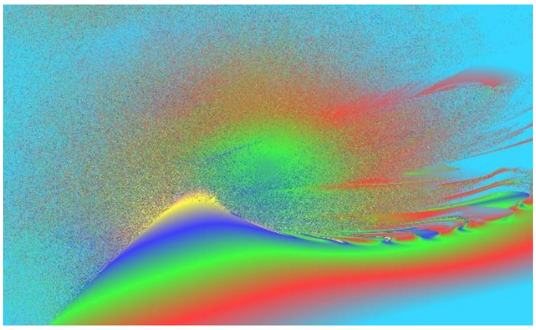


Figure 22. Fractal "Dune" **A** = -0.77, 1.68, -0.91. **B** = 0.00_0.00_0.48.

Conclusions.

The proposed method allows to create attractive-looking fractals. An escape criterion is proposed for a certain range of parameters. The algorithm is simple to implement, it can be adapted and extended.

Supplemental Material

The datasets can be downloaded from (10). The online version of the fractal generator is available at (11).

References.

[1] M. Barnsley, Fractals Everywhere, 2nd ed. Boston, MA, USA: Academic,1993.

[2] B. B. Mandelbrot, The Fractal Geometry of Nature, vol. 2. New York, NY, USA: WH Freeman, 1982

[3] M. Tanveer, I. Ahmed, A. Raza, S. Nawaz, Y-P. Lv. New escape conditions with general complex polynomial for fractals via new fixed point iteration[J]. AIMS Mathematics, 2021, 6(6): 5563-5580. doi:10.3934/math.2021329

[4] Y. C. Kwun, M. Tanveer, W. Nazeer, K. Gdawiec and S. M. Kang, "Mandelbrot and Julia Sets via Jungck–CR Iteration With s–Convexity," in IEEE Access, vol. 7, pp. 12167-12176, 2019, doi: 10.1109/ACCESS.2019.2892013.

[5] H. Qi, M. Tanveer, M. S. Saleem and Y. Chu, "Anti Mandelbrot Sets via Jungck-M Iteration," in IEEE Access, vol. 8, pp. 194663-194675, 2020, doi: 10.1109/ACCESS.2020.3033733.

[6] M. Kumari, S. Kumari, R. Chugh. Generation of Anti-Fractals in SP-Orbit. International Journal of Computer Trends and Technology. 43. 105-112, 2017, doi: 10.14445/22312803/IJCTT-V43P115.

[7] A. Y. Cherny, E. M. Anitas, A. I. Kuklin, M. Balasoiu and V. A. Osipov. Scattering from generalized Cantor fractals. J. Appl. Cryst. (2010). 43, 790-797, doi:10.1107/S0021889810014184

[8]. B. Martin. Inherent Generation of Fractals by Cellular Automata. Complex Systems 8 (1994) 347-366.

[9]. S. Charonov. Generation of Fractals using Polar Coordinates. ScienceOpen Preprints. 2021. doi: 10.14293/S2199-1006.1.SOR-.PPBKDMO.v1

[10]. Data sets. https://doi.org/10.6084/m9.figshare.16673947

[11]. The online fractal generator.

https://spectralmultiplatform.blogspot.com/p/polar-polynomial-fractals.html