Fractal art using variations on escape time algorithms in the complex plane

P. D. SISSON

Louisiana State University in Shreveport, USA

One University Place
Shreveport LA 71115 USA
Email: paul.sisson@lsus.edu

The creation of fractal art is easily accomplished through the use of widely-available software, and little mathematical knowledge is required for its use. But such an approach is inherently limiting and obscures many possible avenues of investigation. This paper explores the use of orbit traps together with a simple escape time algorithm for creating fractal art based on nontrivial complex-valued functions. The purpose is to promote a more direct hands-on approach to exploring fractal artistic expressions. The article includes a gallery of images illustrating the visual effects of function, colour palette, and orbit trap choices, and concludes with suggestions for further experimentation.

Keywords: Escape time algorithm; fractal; fractal art; dynamical system; orbit trap

AMS Subject Classification: 28A80; 37F10; 37F45

1 Introduction

The variety of work currently being created and classified as fractal art is pleasantly immense, and the mathematical knowledge required by artists to participate in the creative process is now essentially nil. Many software packages have been written that allow artists to explore fractal creations on purely aesthetic terms, and the beauty of much of the resulting work is undeniable (see http://en.wikipedia.org/wiki/Fractal for an up-to-date collection of fractal software, and http://www.fractalartcontests.com/2006/ for a nice collection of recent fractal art). But a reliance on existing software and a reluctance to delve into the mathematics of fractal art is inherently limiting and holds the threat of stagnation. The purpose of this article is to "step back" a bit from the current scene and try to recapture a sense of the bigger picture.

The advantages of reverting to a simpler and more primitive exploration of fractal art include a rediscovery of the breadth and freshness of possibilities. While it is impossible not to sense the infinite variety inherent in even a single example of fractal art, it *is* possible to lose sight of the many different directions variation can take. The art reproduced in section 3 of this article is a collection of images based on variations of an escape time algorithm, one of the earliest and simplest methods of generating fractal art. The algorithm is straightforward and amenable to coding in essentially any programming language; I have used the symbolic math package *Mathematica*® for its implementation.

The purpose of the escape time algorithm is to assign a colour to, in this case, each initial point z_0 of a bounded region of the complex plane $\mathbb{C}$. This entails iterating a complex-valued function *f* to generate the sequence

$$z_0, f(z_0), f(f(z_0)), \ldots, f^n(z_0), \ldots \tag{1}$$

The colour of z_0 is then assigned on the basis of the first iteration *n* for which $f^n(z_0)$ meets a pre-defined condition or reaches the maximum value permitted (denoted *N* hereafter). The use of a function in this fashion is sometimes described as a *dynamical system*, and the phrase *escape time algorithm* derives from the fact that in many famous early uses the pre-defined goal was that $|f^n(z_0)|$ should grow without bound (escape to infinity). Many references such as Falconer's [6] and the classic text by Barnsley [3] are available to provide a good overview of these topics.

The details of the implementation of the algorithm depend of course on the choice of programming environment, but its essence is an iterative loop inside a region-covering routine. The task of the region-covering routine is to discretize a portion of the complex plane (almost certainly a rectangle) into a large number of points, each of which represents both a complex number and a pixel in the image ultimately produced. Each complex number z_0 is then fed systematically into the iterative loop, which functions as follows:

1. Set $n = 1$ and $z = z_0$.
2. While *z* lies outside the orbit trap *R* and *n* is less than the pre-set iteration limit *N*, increment *n* by one and set $z = f(z)$.
3. Return *n*.

It is at this point that many artists and programmers begin to specialize and restrict themselves in their explorations. Common restrictions are to consider only one such function *f* (or a class of closely-related functions) and to adhere to the requirement that $f^n(z_0)$ escape to infinity, a natural enough reaction to the fact that the fractal images generated still contain a wealth of variety. But much more variety can be had through imaginative sampling of functions and through setting different pre-defined goals for the sequence (1). If the pre -defined goal is to determine the least *n* for which $f^n(z_0)$ falls within a specific subset *R* of $\mathbb{C}$, that subset *R* is referred to as an *orbit trap* – the terminology reflects the fact that (1) is often called the *orbit* of z_0. As we will see in section 2, orbit traps also provide a natural means of alleviating an inherent flaw of images produced by escape time algorithms.

2 Algorithm specifics

The algorithm described above provides for the choice of three critical components on purely aesthetic grounds. They are: (1) the complex-valued function *f*, (2) the orbit trap *R*, and (3) the colour palette. There is a great deal of latitude in all three components, and considerable time

can be spent fine-tuning choices in pursuit of an artistic vision. The following paragraphs will serve simply as a brief introductory guide.

2.1 Choice of function

Any nonlinear function *f* mapping the complex plane $\mathbb{C}$ back to itself is worth exploring – many classes will quickly reveal their possibilities and limitations and guide further exploration. Rational functions (ratios of two polynomials in the complex variable *z*) and logarithms, exponentials, and roots of rational functions are notably rich in variety. The poles and/or branch cuts of such complex-valued functions are of particular interest, as their contributions to the resulting images are a marked departure from such venerable fractals as the Mandelbrot set (see Devaney and Keen [5] for a comprehensive exploration of the Mandelbrot set). The mathematical sophistication of the programmer and the language of choice are practical considerations, however. It is very likely, for example, that computations of rational functions acting on complex numbers will pose no difficulty, but not every language will have a built-in understanding of logarithmic, exponential, or trigonometric functions extended to the complex plane. A symbolic math package that provides programming and graphing capability offers an easy solution to this problem (*Mathematica*®, *Maple*®, and *Matlab*® are three examples).

2.2 Choice of orbit trap

As mentioned, historically the "orbit trap" used by escape-time algorithms was the point at infinity; in practice, the algorithm returned the first *n* for which $\left| f^n(z_0) \right|$ exceeded a fixed pre-defined value (for the Mandelbrot set, that value was 2). But a stunning display of images is available through the choice of different traps, especially bounded traps (those contained inside some circle of fixed radius centred at the origin). Further, certain artistic goals can be pursued through the modification and refinement of the trap. As a starting point for exploration, disks, squares, annuli, and various combinations of disks and annuli are usually inspirational. Annular regions between non-circular concentric shapes offer a multitude of additional possibilities.

2.3 Choice of colour palette

A *colour palette* is, in this usage, a one-dimensional representation of the colours to be used in the image. In the common *Hue, Saturation, Brightness* colour model (HSB), in which H, S, and B typically can be assigned any value between 0 and 1, Hue represents a built-in linear spectrum based on mixtures of Red, Green, and Blue (RGB) intensities. (Note, though, that it is possible to define one's own palette by choosing any one-parameter path through RGB space.) The reason for working with a one-dimensional palette is that it is very easy to correlate the output of the escape-time algorithm with a point on the palette; for the purposes of exposition, we will assume here that the palette is the Hue colour spectrum. The colour assignment can then be achieved as follows: given *N* as the fixed iteration limit, and *n* as the output of the iterative loop for the point z_0, assign z_0 the Hue value n/N. As a minor variation, it may be desirable to assign z_0 a pre-selected background colour if $n = N$ (black was used as the background colour for each of the images in section 3 of this paper).

The interplay between the orbit trap and variation of the Saturation or Brightness parameter warrants elaboration. In general, an orbit trap *R* with a nonempty interior is called for; such traps are more likely to generate rich and densely-colored images. A trap with a nonempty interior also provides the natural opportunity to use the Euclidean distance between $f^n(z_0)$ and the boundary of *R* (where *n* is the output of the iterative loop) to shade the color of z_0 and eliminate the "banding" effect commonly seen in early fractal images. In fact, the desire to alleviate this banding effect has been the motivation behind the development of many of the features of fractal software, and there are many creative solutions. Some of these solutions are known as *distance estimation algorithms*, *continuous potential algorithms*, *normalized count algorithms*, and *exponential smoothing* (Barallo and Jones [1]). In the HSB colour model, the distance *d* between $f^n(z_0)$ and the boundary of *R* can be used as a Saturation or Brightness parameter, either lightening or darkening the default colour of z_0.

3 Image Gallery

With the algorithm and terms now defined, all that remains is to illustrate the results of some combinations of choices. Each of the images in this section is accompanied with information on its associated function, palette, and trap. The resolution (that is, the fixed horizontal and vertical distance between pixels in the discretization routine) is either 0.004 or 0.005 units in each case. The intent of the collection is to indicate some of the effects of varying, individually, the components that make up each image.

The first, *i of Medusa* (Figure 1), is based on the function

$$f(z) = \ln\left[\frac{(1-i)z^6 + (7+i)z}{2z^5 + 6}\right].$$

The orbit trap *R* is an annulus centred at the origin with an inner radius of 0.9 and an outer radius of 1.1. The iteration limit *N* is 12, and the colour palette is the built-in Hue spectrum of *Mathematica*® with shading provided by varying the Brightness parameter as outlined in section 2.3. Saturation is held fixed at its maximum value.

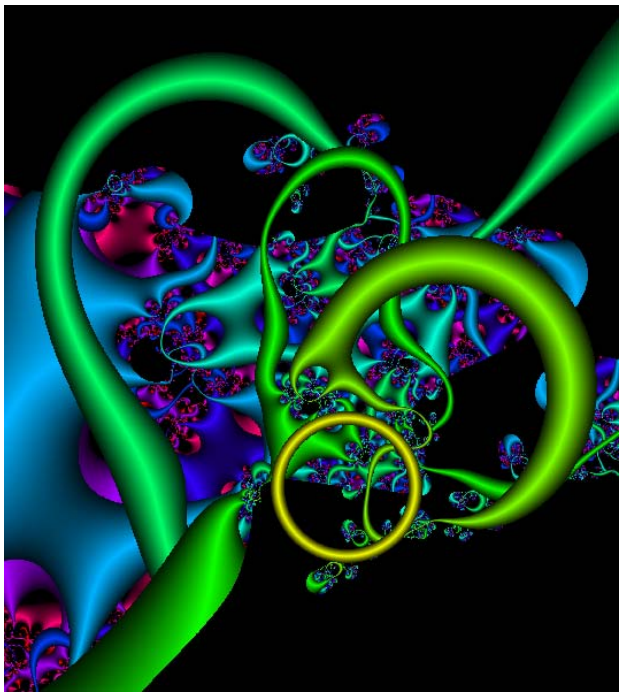

Figure 1: *i of Medusa*

At the C-shore (Figure 2) comes from the function

$$f(z) = \sqrt{(1+i)z - 1}$$

with orbit trap and palette identical to Figure 1.

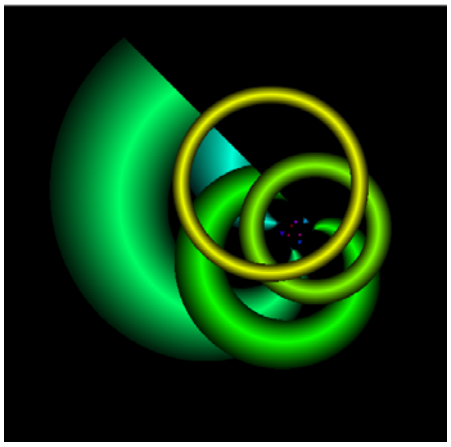

Figure 2: *At the C-shore*

Figures 3, 4, and 5 are all based on the function

$$f(z) = \frac{(1-i)z^4 + (7+i)z}{2z^5 + 6}.$$

The image *i of the Storm* (Figure 3) uses the same iteration limit *N* and orbit trap *R* as Figures 1 and 2, but illustrates the effect of holding Hue constant and instead varying Saturation and Brightness. Figure 4, *i of the Storm Two*, replaces the circular annulus with an annular region between two concentric squares, one of edge length 1.8 units and the larger with edge length 2.2 units. The iterated echoes of the corners of the trap are hallmarks of the image. The palette is the full Hue spectrum again, but in this image the trap itself is not made visible. Figure 5, *Wisdom*, uses an orbit trap of five disks of radius 0.5 clustered around the origin.

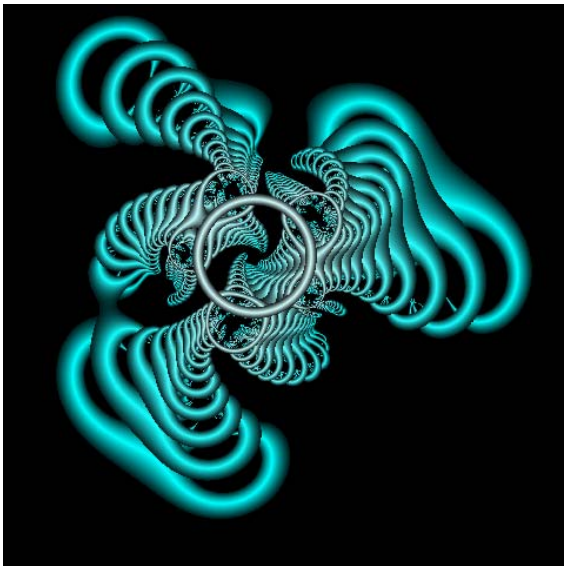

Figure 3: *i of the Storm*

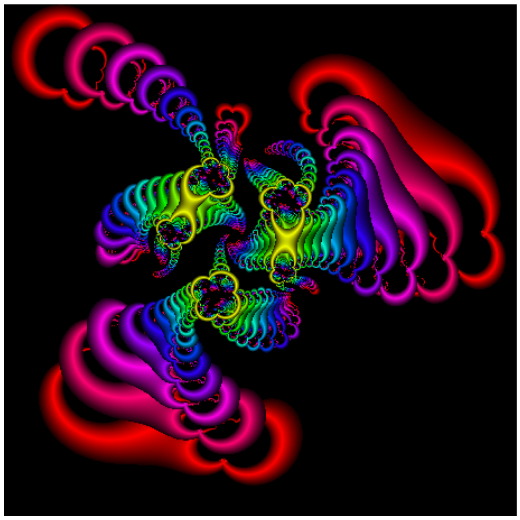

Figure 4: *i of the Storm Two*

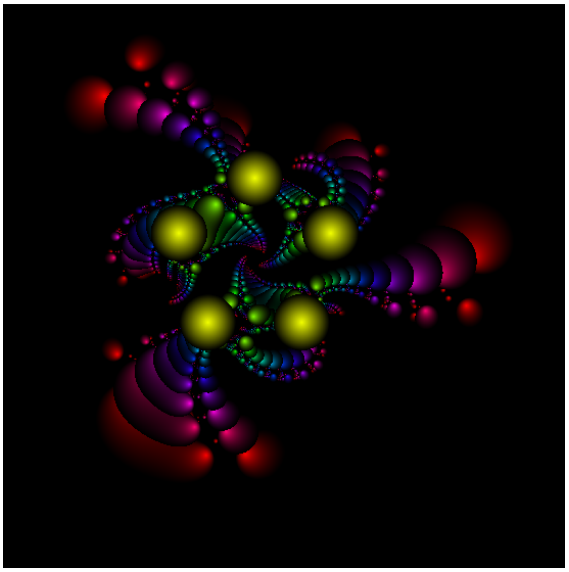

Figure 5: *Wisdom*

The last two images, *i belong to U* (Figure 6) and *Corona* (Figure 7), are both based on the function

$$f(z) = \frac{(1-i)z^6 + (7+i)z}{2z^5 + 6}.$$

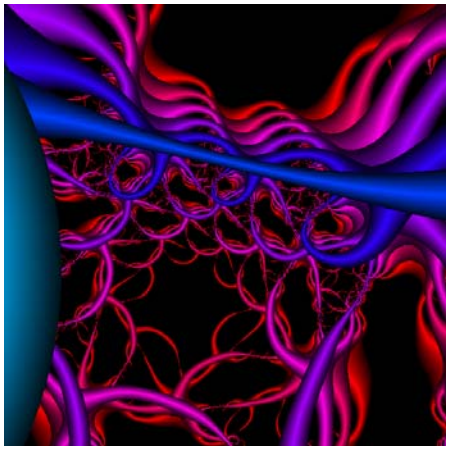

Figure 6: *i belong to U*

Figure 6 is a detail of a region of the complex plane away from the origin, but the orbit trap *R* remains an annulus centred at the origin. Figure 7 uses a disk of radius 1 centred at the origin for the trap, and heavy occlusion is achieved by extreme values of the Brightness parameter.

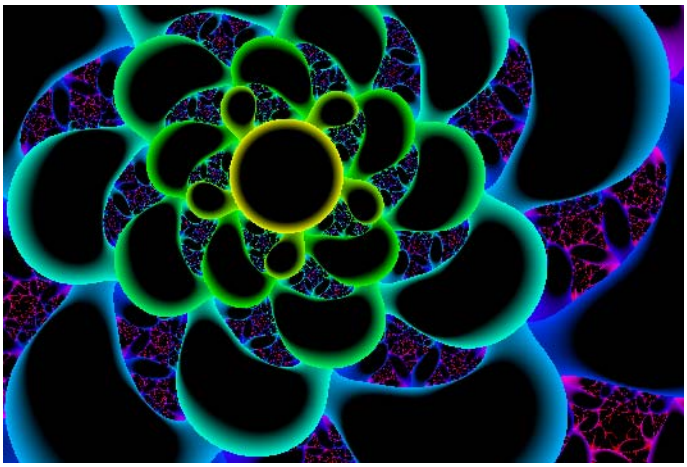

Figure 7: *Corona*

4 Future Work

Experimentation with different orbit traps, different selections of colour palette, and different applications of the distance between $f^n(z_0)$ and the boundary of *R* are obvious avenues of exploration. But other possibilities are ripe for investigation:

(1) The colour-assignment procedure is easily modified so as to use different palettes on the basis of the location of z_0. For instance, a monochromatic palette may be used to complement a full-spectrum palette, with the monochromatic palette applied in a region tailored to the specific function and orbit trap.
(2) The multi layer colouring algorithms described by Barrallo and Sanchez in [2] can be easily applied and hold rich promise.
(3) For those inclined toward dynamic art, animations based on small variations of one or two parameters (such as coefficients) in the chosen function are easily accomplished. See also the comprehensive discussion of animations of the Mandelbrot set in [4].
(4) The effect of branch cuts of *f* under iteration appears not to have been well studied at present, and is an appropriate area of research for those who are mathematically inclined.

References

[1] Barrallo, J. and Jones, D., 1999, Coloring algorithms for dynamical systems in the complex plane, *ISAMA 99 Proceedings*, 31-38.

[2] Barralo, J. and Sanchez, S., 2001, Fractals and multi layer colouring algorithms, *Bridges Conference Proceedings 2001*, 89.

[3] Barnsley, M., 1988, *Fractals Everywhere* (San Diego: Academic Press, Inc).

[4] Burns, A., 2002, Plotting the escape: an animation of parabolic bifurcations in the Mandelbrot set, *Mathematics Magazine*, **75**(2), 104-116.

[5] Devaney, R. and Keen, L. (Eds.), 1989, *Chaos and Fractals: the Mathematics Behind the Computer Graphics*, Proceedings of Symposia in Applied Mathematics vol. 39 (Providence: American Mathematical Society).

[6] Falconer, K., 2003, *Fractal Geometry: Mathematical Foundations and Applications* (West Sussex: John Wiley & Sons, Ltd).