Journal of Information Systems and Communication
ISSN: 0976-8742 & E-ISSN: 0976-8750, Volume 3, Issue 1, 2012, pp.-347-352.
Available online at http://www.bioinfo.in/contents.php?id=45

FRACTINT FORMULA FOR OVERLAYING FRACTALS

SATYENDRA KUMAR PANDEY[1], MUNSHI YADAV[2] AND ARUNIMA[1]

[1]Institute of Technology & Management, Gorakhpur, U.P., India.
[2]Guru Tegh Bahadur Institute of Technology, New Delhi, India.
*Corresponding Author: Email- skp_gkp@sify.com, munshiyadav@gmail.com

Received: January 12, 2012; Accepted: February 15, 2012

Abstract- A fractal is generally a rough or fragmented geometric shape that can be split into parts, each of which is a reduced-size copy of the whole a property called self-similarity. Because they appear similar at all levels of magnification, fractals are often considered to be infinitely complex approximate fractals are easily found in nature. These objects display self-similar structure over an extended, but finite, scale range. Fractal analysis is the modelling of data by fractals. In general, fractals can be any type of infinitely scaled and repeated pattern. It is possible to combine two fractals in one image.
Keywords- Fractal, Mandelbrot Set, Julia Set, Newton Set, Self-similarity

Citation: Satyendra Kumar Pandey, Munshi Yadav and Arunima (2012) Fractint Formula for Overlaying Fractals. Journal of Information Systems and Communication, ISSN: 0976-8742 & E-ISSN: 0976-8750, Volume 3, Issue 1, pp.-347-352.

Introduction

A fractal is generally "a rough or fragmented geometric shape that can be split into parts, each of which is (at least approximately) a reduced-size copy of the whole"[1] a property called self-similarity. The term was coined by Benoit Mandelbrot in 1975 and was derived from the Latin fractus meaning "broken" or "fractured."
A fractal often has the following features[2]:

- It has a fine structure at arbitrarily small scales.
- It is too irregular to be easily described in traditional Euclidean geometric language.
- It is self-similar (at least approximately or stochastically).
- It has a Hausdorff dimension which is greater than its topological dimension (although this requirement is not met by space-filling curves such as the Hilbert curve).
- It has a simple and recursive definition.

Because they appear similar at all levels of magnification, fractals are often considered to be infinitely complex (in informal terms). Natural objects that approximate fractals to a degree include clouds, mountain ranges, lightning bolts, coastlines, and snow flakes. However, not all self-similar objects are fractals- for example; the real line (a straight Euclidean line) is formally self-similar but fails to have other fractal characteristics.
To create a Koch snowflake, one begins with an equilateral triangle and then replaces the middle third of every line segment with a pair of line segments that form an equilateral "bump." One then performs the same replacement on every line segment of the resulting shape, ad infinitum. With every iteration, the perimeter of this shape increases by one third of the previous length. The Koch snowflake is the result of an infinite number of these iterations, and has an infinite length, while its area remains finite. For this reason, the Koch snowflake and similar constructions were sometimes called "monster curves."
The mathematics behind fractals began to take shape in the 17th century when mathematician and philosopher Leibniz considered recursive self-similarity (although he made the mistake of thinking that only the straight line was self-similar in this sense). It took until 1872 before a function appeared whose graph would today be considered fractal, when Karl Weierstrass gave an example of

a function with the non-intuitive property of being everywhere continuous but nowhere differentiable. In 1904, Helge von Koch, dissatisfied with Weierstrass's very abstract and analytic definition, gave a more geometric definition of a similar function, which is now called the Koch snowflake. In 1915, Waclaw Sierpinski constructed his triangle and, one year later, his carpet. Originally these geometric fractals were described as curves rather than the 2D shapes that they are known as in their modern constructions. The idea of self-similar curves was taken further by Paul Pierre Lévy. Georg Cantor also gave examples of subsets of the real line with unusual properties- these Cantor sets are also now recognized as fractals.

Iterated functions in the complex plane were investigated in the late 19th and early 20th centuries by Henri Poincare, Felix Klein, Pierre Fatou and Gaston Julia. However, without the aid of modern computer graphics, they lacked the means to visualize the beauty of many of the objects that they had discovered.

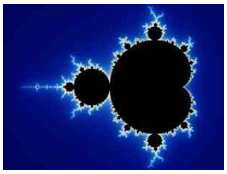

Fig. 1- The Mandelbrot set is a famous example of a fractal.

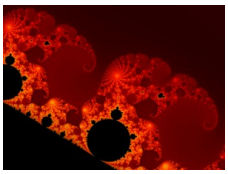

Fig. 2- A closer view of the Mandelbrot set.

In the 1960s, Benoit Mandelbrot started investigating self-similarity in papers such as How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension, which built on earlier work by Lewis Fry Richardson. Finally, in 1975 Mandelbrot coined the word "fractal" to denote an object whose Hausdorff-Besicovitch dimension is greater than its topological dimension. He illustrated this mathematical definition with striking computer-constructed visualizations. These images captured the popular imagination; many of them were based on recursion, leading to the popular meaning of the term "fractal". A Julia set, a fractal related to the Mandelbrot set

A relatively simple class of examples is given by the Cantor sets, Sierpinski triangle and carpet, Menger sponge, dragon curve, space-filling curve, and Koch curve. Additional examples of fractals include the Lyapunov fractal and the limit sets of Kleinian groups. Fractals can be deterministic (all the above) or stochastic (that is, non-deterministic). For example, the trajectories of the Brownian motion[3] in the plane have a Hausdorff dimension of 2.

Chaotic dynamical systems are sometimes associated with fractals. Objects in the phase space of a dynamical system can be fractals. Objects in the parameter space for a family of systems may be fractal as well. An interesting example is the Mandelbrot set. This set contains whole discs, so it has a Hausdorff dimension equal to its topological dimension of 2-but what is truly surprising is that the boundary of the Mandelbrot set also has a Hausdorff dimension of 2 (while the topological dimension of 1), a result proved by Mitsuhiro Shishikura in 1991. A closely related fractal is the Julia set.

Even simple smooth curves can exhibit the fractal property of self-similarity. For example the power-law curve (also known as a Pareto distribution) produces similar shapes at various magnifications.

Generating fractals

Four common techniques for generating fractals are:

1.**Escape-time fractals-** (also known as "orbits" fractals) These are defined by a recurrence relation at each point in a space (such as the complex plane). Examples of this type are the Mandelbrot set, Julia set, the Burning Ship fractal, the Nova fractal and the Lyapunov fractal. The 2d vector fields that are generated by one or two iterations of escape-time formulae also give rise to a fractal form when points (or pixel data) are passed through this field repeatedly.

2.**Iterated function systems-** These have a fixed geometric replacement rule. Cantor set, Sierpinski carpet, Sierpinski gasket, Peano curve, Koch snowflake, Harter-Heighway dragon curve, T-Square, Menger sponge, are some examples of such fractals.

3.**Random fractals-** These are generated by stochastic rather than deterministic processes, for example, trajectories of the Brownian motion, Levy flight, fractal landscapes and the Brownian tree. The latter yields so-called mass- or dendritic fractals, for example, diffusion-limited aggregation or reaction-limited aggregation clusters.

4.**Strange Attractors-** These are generated by iteration of a map or the solution of a system of initial-value differential equations that exhibit chaos.

Classification of fractals

Fractals can also be classified according to their self-similarity. There are three types of self-similarity found in fractals:

1.**Exact self-similarity-** This is the strongest type of self-similarity; the fractal appears identical at different scales.

2.**Quasi-self-similarity-** This is a loose form of self-similarity; the fractal appears approximately (but not exactly) identical at different scales. Quasi-self-similar fractals contain small copies of the entire fractal in distorted and degenerate forms.

3.**Statistical self-similarity-** This is the weakest type of self-similarity; the fractal has numerical or statistical measures which are preserved across scales. Most reasonable definitions of "fractal" trivially imply some form of statistical self-similarity.

Fractals in nature

Approximate fractals are easily found in nature. These objects display self-similar structure over an extended, but finite, scale range. Examples include clouds, snow flakes, crystals, mountain ranges, lightning, river networks, cauliflower or broccoli, and systems of blood vessels and pulmonary vessels. Coastlines may be loosely considered fractal in nature.

Trees and ferns are fractal in nature and can be modeled on a computer by using a recursive algorithm. This recursive nature is obvious in these examples - a branch from a tree or a frond from a fern is a miniature replica of the whole: not identical, but similar in nature.

In 1999, certain self similar fractal shapes were shown to have a property of "frequency invariance" - the same electromagnetic

properties no matter what the frequency - from Maxwell's equations.
Fractal patterns have been found in the paintings of American artist Jackson Pollock. While Pollock's paintings appear to be composed of chaotic dripping and splattering, computer analysis has found fractal patterns in his work. Decalcomania, a technique used by artists such as Max Ernst, can produce fractal-like patterns. It involves pressing paint between two surfaces and pulling them apart.
Fractals are also prevalent in African art and architecture. Circular houses appear in circles of circles, rectangular houses in rectangles of rectangles, and so on. Such scaling patterns can also be found in African textiles, sculpture, and even cornrow hairstyles.

Fig. 3- A fractal is formed when pulling apart two glue-covered acrylic sheets

Fig. 4- A DLA cluster grown from a copper (II) sulfate solution in an electrodeposition cell

Fig. 5- A fractal flame created with the program Apophysis

Fig. 6- High voltage breakdown within a 4" block of acrylic creates a fractal Lichtenberg figure.

Fig. 7- A "woodburn" fractal

Fig. 8- Pascal generated fractal fractals

Fractal analysis
Fractal analysis is the modelling of data by fractals. It consists of methods to assign a fractal dimension and other fractal characteristics to a signal, dataset or object which may be sound, images, molecules, networks or other data. Fractal analysis is now widely used in all areas of science.
Fractal analysis is a contemporary method of applying nontraditional mathematics to patterns that defy understanding with traditional Euclidean concepts.
In essence, it measures complexity using the fractal dimension.
The field was developed to describe computer-generated fractals such as the diffusion limited aggregates shown on this page, but fractals are not necessarily computer-generated images. Rather, whereas the Euclidean geometry in familiar shapes like oranges and watermelons. Fractal geometry in familiar forms like meandering coastlines, growing crystals, and swirling galaxies are seen.
Fractals are not necessarily physical forms - they can be spatial or temporal patterns, as well. In general, fractals can be any type of infinitely scaled and repeated pattern. In this regard, it is important to be aware that theoretical fractals are abstractions, but the subjects of fractal analysis, such as digital images limited by screen resolution, are generally not true fractals in the strictest sense. Similarly, the so-called fractals typically found in nature are not infinitely scaled, thus, like finite computer generated patterns, are generally only approximations to fractals in the strictest sense.
We believe fractal geometry makes fundamental changes in science and far beyond, but are waiting for the evidence to accumulate. One of the roles of this Panorama is to share some evidence we have already seen. Another role, more important in our eyes, is to invite us to share our examples with us.

Pseudo HiColor formula for Overlaying two fractals
It is possible to combine two fractals (say fractal_0 and fractal_1) in one image. Let us imagine that alternate screen pixels form a checkerboard pattern (represented by 0's and 1's) as follows:

```
0 1 0 1 0 . .
1 0 1 0 1 . .
0 1 0 1 0 . .
1 0 1 0 1 . .
. . . . . .
```

If one fractal is drawn on the "white squares" (the 1's) and the other on the black squares (the 0's), the separate fractals is visible, and at higher screen resolutions we are not be able to see the way the individual pixels intermesh with the others. The effect is as if the two fractals were drawn on separate transparent sheets and overlaid.
Fractint v. 19.5 provides a predefined variable "whitesq", which is automatically set to 1 prior to the calculation of a white square pixel, and to 0 prior to calculation of a black square pixel. Use of this variable in a formula is described as under.

Assignment statements
Let us suppose that fractal_0 and fractal_1 have the following assignment statements:
fractal_0
var = something
fractal_1
var = somethingelse
To overlay the two fractals in PHC (Pseudo HiColor) fashion, we

have used the following IF..ELSE instruction in a formula:

```
IF (whitesq == 0)       ; "whitesq == 0" is TRUE
var = something
ELSE                    ; "whitesq == 0" is FALSE
var = somethingelse
ENDIF
```

or, even more simple:

```
IF (whitesq)            ; "whitesq == 1" is TRUE
var = somethingelse
ELSE                    ; "whitesq == 1" is FALSE
var = something
ENDIF
```

To assign the appropriate value to var, PHC formula have been simulated to an IF...THEN...ELSE... construct:

```
if (whitesq == 0)  /* if "whitesq == 0" is TRUE */
var = something ;
else          /* if "whitesq == 0" is FALSE */
/* or "whitesq == 1" is TRUE */
var = somethingelse ;
```

This have been done with the following line:

var = something * (whitesq == 0) + somethingelse * (whitesq == 1)

which results in "something" being assigned to var for the black squares, and "somethingelse" being assigned to var for the white squares.

To understand how it works, it's important to know that Fractint represents TRUE with a one, and FALSE with a zero, therefore:

* if whitesq is equal to 0:
- "whitesq == 0" evaluates to 1
- "whitesq == 1" evaluates to 0

and var = something * 1 + somethingelse * 0
= something

* if whitesq is equal to 1:
- "whitesq == 0" evaluates to 0
- "whitesq == 1" evaluates to 1

and var = something * 0 + somethingelse * 1
= somethingelse

We have noticed that the result of the test "whitesq == 1" is equal to the value of the variable "whitesq", so we can use the following statement:

var = something * (whitesq == 0) + somethingelse * whitesq

Bailout tests

Let us suppose that fractal_0 and fractal_1 use the bailout tests bailout_0 and bailout_1. If the answer of a bailout test is "TRUE" (the real portion of the complex number is nonzero), the loop must be performed again; otherwise, it is time to quit iterating. The bailout test of the PHC formula must be the translation in the parser language of the following rule:

```
PHC_bailout is TRUE only in two cases:
when (whitesq == 0) and (bailout_0 == TRUE)
or
when (whitesq == 1) and (bailout_1 == TRUE)
Under Fractint 19.6, this expression becomes:
IF (whitesq)
PHC_bailout = bailout_1
ELSE
PHC_bailout = bailout_0
ENDIF
```

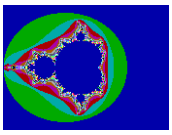

Fig. 9- Mandel Set

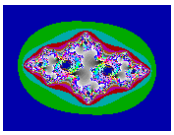

Fig. 10- Julia Set

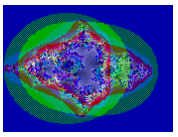

Fig. 11- Overlay of Mandel and Julia Sets

Mandel and Julia types

This is the easiest case: both fractal use the same iteration instruction and the same bailout test.

```
mandel { ; Mandel set of z^2 + c
z = c = pixel :
z = z*z + c
|z| <= 4
}
julia { ; Julia set of z^2 + (-0.75,0.1234)
z = pixel , c = (-0.75,0.1234) :
z = z*z + c
|z| <= 4}
```

Since the only difference is the initial value of c, in the PHC formula we have used whitesq only in the init section:

```
phc_mj { ; overlay the Mandel set of z^2 + c with
; the Julia set of z^2 + (-0.75,0.1234)
; Modified for if..else logic
z = pixel
IF (whitesq)
c = (-0.75,0.1234)
ELSE
c = pixel
ENDIF
:
z = z*z + c
|z| <= 4
}
```

and the Julia set is drawn on the white squares of the checkerboard, and the Mandelbrot set on the black squares.

Mandel and Newton types

Here, except "z = pixel", everything is different.

```
mandel { ; Mandel set of z^2 + c
z = c = pixel :
z = z*z + c
|z| <= 4
}
newton { ; Julia set of Newton's method
; applied to z^3 - 1 = 0
z = pixel :
n = z^3 - 1 , d = 3*z^2
z = z - n/d
|n| >= 0.000001
}
```

The resulting PHC(Pseudo HiColor) formula is:

```
phc_mn_A { ; overlay the Mandel set of z^2 + c with the Julia
; set of Newton's method applied to z^3 - 1 = 0
; Modified for if..else logic
z = pixel :
IF (whitesq)
n = z^3 - 1 , d = 3*z^2 , z = z - n/d
PHC_bailout = |n| >= 0.000001
ELSE
z = z*z + pixel , PHC_bailout = |z| <= 4
ENDIF
PHC_bailout
}
```

We have noticed that c is initialized to pixel even when whitesq == 1 but it is not used by the Newton algorithm

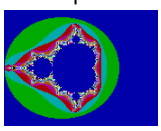

Fig.12- Mandel Set

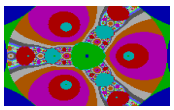

Fig. 13- Newton Set

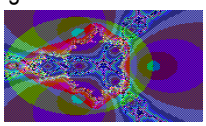

Fig.14- Overlay of Mandel Set and Newton Set

Pseudo True Color formula for overlaying three fractals ("24-bit PTC")

Overlaying three fractals can be done with the following pattern:

```
0 1 2 0 1 2 . .
1 2 0 1 2 0 . .
2 0 1 2 0 1 . .
0 1 2 0 1 2 . .
. . . . . . . .
```

Fractint v. 19.5 provides a predefined variable "scrnpix", which is set to (column, row) prior to calculation of each pixel. The upper left hand corner of the screen is (0,0); at resolution 1024x768, the lower right hand corner is therefore (1023,767). Here, we have used scrnpix to assign the value 0, 1 or 2 to a variable r. With col = real(scrnpix) and row = imag(scrnpix), the value of r should be:

```
r = (col + row) modulo 3
or, using the parser language:
cr = real(scrnpix) + imag(scrnpix)
r = cr - 3 * trunc(cr / 3)
But this instruction has not worked.
The following instruction worked:
r = cr - 3 * trunc(cr / real(3))
Let us suppose that we want to overlay the three following
fractals:
mandel { ; Mandel set of z^2 + c
z = c = pixel :
z = z*z + c
|z| <= 4
}
julia { ; Julia set of z^2 + (-0.75,0.1234)
z = pixel , c = (-0.75,0.1234) :
z = z*z + c
|z| <= 4
}
newton { ; Julia set of Newton's method
; applied to z^3 - 1 = 0
z = pixel :
n = z^3 - 1 , d = 3*z^2
z = z - n/d
|n| >= 0.000001
}
```

We have merged them in the following way:
ptc_mjn_A { ; overlay the Mandel set of

```
ptc_mjn_A { ; overlay the Mandel set of
;z^2 + c with the Julia set
; of z^2 + (-0.75,0.1234) and
;the Julia set of Newton's
;method applied to z^3 – 1 = 0
; Modified for if..else logic
cr= real(scrnpix) + imag(scrnpix)
r = cr - 3 * trunc(cr / real(3))
z = pixel
IF (r == 0)
c = pixel
ELSEIF (r == 1)
c = (-0.75,0.1234)
ENDIF
:
IF (r == 2)
n = z^3 - 1 , d = 3*z^2 , z = z - n/d
PTC_bailout = |n| >= 0.000001
ELSE
z = z*z + c
PTC_bailout = |z| <= 4
ENDIF
PTC_bailout
}
```

Pseudo True Color formula for overlaying four fractals ("32-bit PTC")

The best dithering is produced by the following pattern:

```
0 1 2 3 0 1 . .
2 3 0 1 2 3 . .
0 1 2 3 0 1 . .
2 3 0 1 2 3 . .
```

and r is given by the formula:
r = (col + 2*row) modulo 4
or, using the parser language:
cr = real(scrnpix) + 2 * imag(scrnpix)
r = cr - 4 * trunc(cr / 4)
and r can then be used as in the previous examples, to combine four fractals in one image.

```
mand_0 {                    mand_1 {
z = c = sin(pixel) :          z = c = pixel :
z = z*z + c                   z = z*z + c
|real(z)| <= 4                |z| <= 4
}                           }

mand_2 {                    mand_3 {
z = c = 1/pixel :             z = c = -pixel :
z = z*z + c                  z = z*z + c
|imag(z)| < 4                |real(z)+imag(z)| < 4
}                           }
```

```
ptc_4m_A { ; overlay four Mandels with different
;initializations and bailout tests
; Modified for if..else logic
cr = real(scrnpix) + 2 * imag(scrnpix)
r = cr - 4 * trunc(cr / 4)
IF (r == 0)
z = c = sin(pixel)
ELSEIF (r == 1)
z = c = pixel
ELSEIF (r == 2)
z = c = 1/pixel
ELSE
z = c = -pixel
ENDIF
:
z = z*z + c
IF (r == 0)
PTC_bailout = |real(z)| <= 4
ELSEIF (r == 1)
PTC_bailout = |z| <= 4
ELSEIF (r == 2)
PTC_bailout = |imag(z)| <= 4
ELSE
PTC_bailout = |real(z)+imag(z)| <= 4
ENDIF
PTC_bailout
}
```

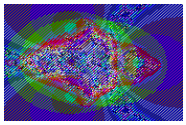

Fig. 15- Merged Set of mandel, Julia and Newton sets

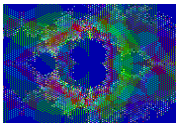

Fig.16- Overlay of four Mandels with different initializations

Conclusion

While Fractint has many different type of fractal formula built into it, the formula parser allows us to add new fractals without having to change the program. These formulas are stored in simple text files, and may be viewed and edit by the user. Fractint formula is

actually a little computer program not a set of mathematical equations.To avoid exponentiation , unnecessary calculations , use the algebraic rules are some precautions have to be taken to make formulas run a little faster

References

[1] Mandelbrot B.B. (1977) *The fractal geometry of nature W.H. Freeman and Company.*

[2] Gouyet J. (1996) *Physics and fractal structures Springer-Verlag,*

[3] Mandelbrot B.B. (1988) *Science,* 279, 783-784.

[4] "*Mandelbrot Set Explorer: Mathematical Glossary*".

[5] Lei T. (1990) *Communications in Mathematical Physics* 134, 587-617.

[6] Peitgen, Heinz-Otto, Richter Peter (1986) *The Beauty of Fractals. Heidelberg: Springer-Verlag*.

[7] John Briggs (1992) *Fractals: The Patterns of Chaos*. 80.

[8] Falconer, Kenneth (2003) *Fractal Geometry: Mathematical Foundations and Applications*. *John Wiley & Sons*, *Ltd*.

[9] John W. Milnor (2006) *Dynamics in One Complex Variable (Third Edition), Annals of Mathematics Studies* 160.