Fractal Art: Fractal Image and Music Generator

RAZVAN TANASIE[1] IEEE Member, MIHAI POPESCU[2], DANA BOGHEANU[2], GABRIELA CIOCOIU[2], DORIAN COJOCARU[3] IEEE Member
Software Engineering Department[1,2], Mechatronics Department[3]
Faculty of Automation, Computers and Electronics, University of Craiova
Bvd. Decebal, Nr. 107
ROMANIA

Abstract: - This paper presents a software that creates a fractal screensaver. The application generates both fractal images and music based on the Julia and Mandelbrot fractals. The four coloring algorithms implemented are founded on: escape-time, distance estimators, escape angle and curvature estimation algorithm. The music is created for different instruments on 32 voices. The horizontal scan is an 1/32 notes at a tempo of quarter note with the value 60. The iteration data from the Julia calculations is mapped to key velocity. The higher the iteration result, the louder the pitch presence. The application is configurable and permits the user to select the color palette, a region in the fractal and even the instruments to be played.

Key-Words: - fractals, Julia, Mandelbrot, fractal music, fractal images, fractal screensaver

1 Introduction

In general, a **fractal** is referred to as "a rough or fragmented geometric shape that can be subdivided in parts, each of which is (at least approximately) a reduced-size copy of the whole". The term was created by Benoît Mandelbrot in 1975 and is derived from the Latin *fractus* meaning "broken" or "fractured" [1,2,3].

A fractal as a geometric object generally has the following features:

- It is simple and recursive in its definition.
- It has a fine structure at arbitrarily small scales.
- It is self-similar (at least approximately or stochastically).
- It is too irregular to be easily described in traditional Euclidean geometric language.
- It has a Hausdorff dimension that is greater than its topological dimension.

Fractals can be considered to be infinitely complex because of the similitude at all their levels. Not all self similar objects are fractals, because they don't meet all fractals characteristics. Some objects that approximate fractals are: lightings, clouds, mountains.

2 Fundamentals

2.1 Fractal Types

Coloring algorithms do not cover some types of fractals because they are not common and are not used in most of the fractal software packages.

Basically, the fractal types can be classsified in six main groups, but the software is based on the following fractals: Julia and Mandelbrot [1,4].

These are the most widely used and known fractals and are generated by the iteration of complex polynomials. Due to their fame, they were experimented with different coloring algorithms by the fractal artists.

Many fractals sets can be considered subgroups from these types, for example fractal terrains are a three dimensional representation of a plasma fractal.

2.2 Fractal Mathematics

2.2.1 The Julia Sets

The Julia set is composed of all starting points z which have unusual orbits, that is, those points whose long-time behavior under repeated iterations can change drastically under arbitrarily small perturbations.

Definition Orbit of z: Let *f* be a function. Then the list of successive iterations of a point or number is called the orbit of that point.

For example consider $f = z^2$: If starting with $z=1$, then the orbit is (1, 1, 1,...), so the orbit converges to the number 1, so 1 is an attracting point. Starting with $z=0.5$, then the orbit is (0.5, 0.25, 0.0625, 0.0039, ...), which converges to 0, so 0 is an attracting point, too. Stating with $z=2$, the orbit is (2, 4, 16, 256, 65536, ...), which converges (diverges) to infinity, so infinity is an attracting point, too. In this case it can be seen that the orbits of all initialization values lead to different attracting point [8].

The explanations are limited mainly to quadratic systems, i.e. systems where $f(z)$ is a polynomial of the second degree:

$$f(z) = a * z^2 + b * z + c \tag{1}$$

Some things will be defined as:

Definition Critical Point of function f: Let *f* be a function. Then every point *z* with $f'(z)=0$ (first derivation of *f* at point *z* is null) is called Critical point of *f*.

Definition fixed point of f: Let *f* be any function. Then every point *z* with $z=f(z)$ (*f* lets *z* invariable) is called fixed point. Additionally, if infinity is invariable, i.e. $f(\infty)=\infty$, then infinity is a fixed point, too.

The function $f(z) = a * z^2 + b * z + c$ can be simplified by applying a transformation which gives special values to some points. Two forms are widely used:

$$f(z) = z * z + c \tag{2}$$

and

$$f(z) = m * z + (1-m) * z * z \tag{3}$$

In the first case the critical point of *f* is at 0, and in some situations this can be very useful for theoretical and practical purposes. The fixed points of *f* are: $0.5+\sqrt{0.25-c}$ and $0.5-\sqrt{0.25-c}$.

In the second case the fixed points of *f* are at 0 and 1.

Most of the time fractal programs use standard formula (2).

The attracting fixed points of a Julia set determine its property.

Consider p is a fixed point of f, i.e. $f(p)=p$
Zooming in at p very deep and taking z_0 close to p, yields $z_1=f(z_0)$, similarly with:

$$z_1 - p = m*(z_0 - p), \tag{4}$$

with $m=f'(p)$

Writing it in another way will result:

$$f'(p) = \lim_{z \to p} \frac{f(z) - f(p)}{z - p}, \tag{5}$$

the first derivation of *f* at point *p*.

Choosing z_0 close to p, then $m=f'(p)$ is approximated:

$$m = \frac{f(z_0) - f(p)}{z_0 - p} = \frac{f(z_1) - f(p)}{z_1 - p} \tag{6}$$

The so called eigen value *m* of a fixed point *p* of *f* is $m=f'(p)$. As seen earlier, if $|m|$ is less than 1, then z_1 is closer to p than z_0. Thus an attracting fixed point is obtained. If $|m|$ is greater than 1, then z_1 is more far away than z_0, so a repulsing fixed point is computed. If $|m|=1$, then the fixed point is neutral or indifferent. Then numerous interesting effects can occur (at least theoretically interesting) [6].

2.2.2 The Mandelbrot Set

Consider $J(f)$ any Julia set using function *f*.

If a complex number *c* is taken and the Julia set $J(f)$ is calculated using that number, i.e. $J(z^2+c)$, then what it must become known if the Julia set $J(z^2+c)$ is connected or like “dust” [7].

$$G(c) = \begin{cases} 0 \quad if \quad J(z^2+c) \quad is \quad connected \\ 1 \quad if \quad J(z^2+c) \quad is \quad dust \quad like \end{cases} \tag{7}$$

The type of the Julia set is shown by every point of the Mandelbrot set.

An arbitrary point *c* can be taken and lied “inside” the Mandelbrot set (i.e. normally the black region), therefore the Julia set $J(z^2+c)$ is connected.

Taking another point *d* from “outside” the Mandelbrot set, therefore the Julia set $J(z^2+d)$ is dust like.

It is not necessary to calculate the whole Julia set, however the orbits of specific points must be examined.

The specific points mentioned above are the critical points, i.e. all points where the first derivation vanishes (*$f'(z)=0$*). The type of the Julia set is defined by the orbits of the critical points [8].

When all orbits remain limited, the Julia set is connected. (If at least one orbit tends to converge to infinity, then the Julia set is dust like).

Consider the following example:
In the case of examining z^2+c there is only one critical point: $f'(z)=2*z=0$, i.e. $z=0$ is the only critical point. Thus in this case only the orbit of 0 must be examined.

The most interesting Julia sets related to the Mandelbrot set are those that lie near its border.

In order to calculate an interesting Julia set the parameter *c* must be set to a value lying near the border of the Mandelbrot set [9].

The Mandelbrot set appears as a fringed horizontal eight that is symmetrical on the real axis, as in the following figure:

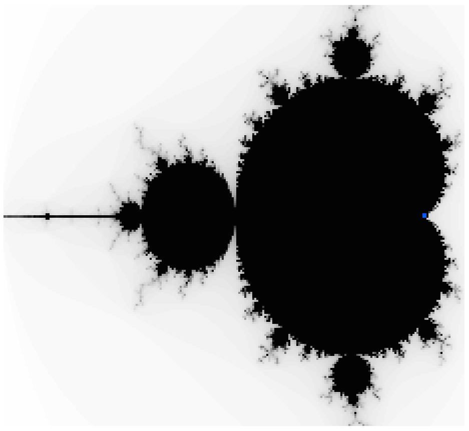


Fig. 1 A Mandelbrot set

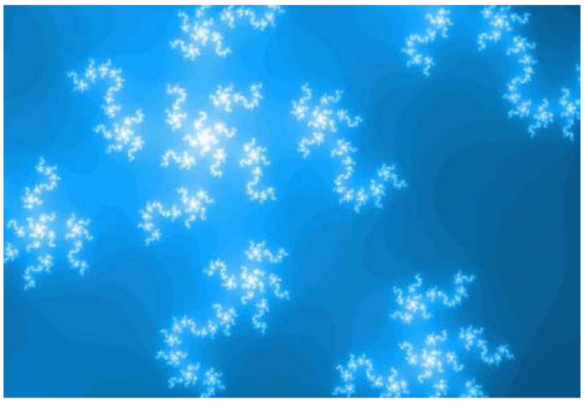


Fig. 2 A Julia set

Note: The difference between Mandelbrot and Julia sets consists of the fact that the complex coordinates of the point are substituted to c and not to *z* (established at first with a 0 value). The complex number *c* should not be a constant defined when starting the elaboration. An example of a Julia set obtained from the application is shown in Fig. 2.

2.3 The software

The software was developed in order to follow two major lines: visual and acoustic, merged into a single final product - a screen saver.

The application is based upon the previous described sets, making a visual link between them, generating the Julia set by using a constant selected from the Mandelbrot set. It can be observed that if the constant is outside the Mandelbrot set, the Julia set is represented as a fractal image with little complexity. Also, if the constant is inside the Mandelbrot set, the image is not interesting. Infinite complexity can be achieved by choosing constants merely on the “coast line” of the Mandelbrot set [10].

The music can be generated upon the fractal image created and is strongly depending on the image. For example, a zoom could be made in or out on the same spot in the complex plane and the result is that the music changes accordingly [11].

3 Fractal Generation Algorithms

The new fractal techniques tend to lean from fractal formulas towards algorithms. The mathematical equations are substituted by coloring algorithms, and, more interesting, music generation algorithms [12].

3.1 Fractals and Dynamical Systems

Dynamical systems are a well-known branch of mathematics, but until the arrival of computers the sheer number of calculations involved made them impractical for real use. Benoit Mandelbrot was the first one to use computer’s ability to perform rapid calculations to produce graphical representations of dynamical systems in the complex plane. This representation was based on the quadratic formulas described by the French mathematician Gaston Julia at the beginning of the 20th century.

The 1980s was the period when fractals started to be explored not only for their mathematical significance, but also for their artistic nature. The mathematical apparatus is still the base, but the purpose changed to art representation. Fractal artists tested hundreds of different fractal equations which lead to the discovery of many new different fractal types. The choosing of the parameters was very important to refine the fractal form, aspect and even sound.

After 1995, few new major fractal types have been introduced. This is because the newest innovations in fractal art do not come from changing the fractal equation, but from new ways of coloring the results of those equations. The increasing complexity of the coloring algorithms made it possible to use simpler equations. This flexibility made more space for personal artistic expression in the crisp fractal world.

3.2 Coloring Algorithms

Every dynamical system produces a sequence of values z_0, z_1, z_2… z_n (the orbit of z – start point). Fractal images are created by producing one of these sequences for each pixel in the image and the coloring algorithm is what interprets this sequence to produce a final color. An example is presented in Fig. 3.

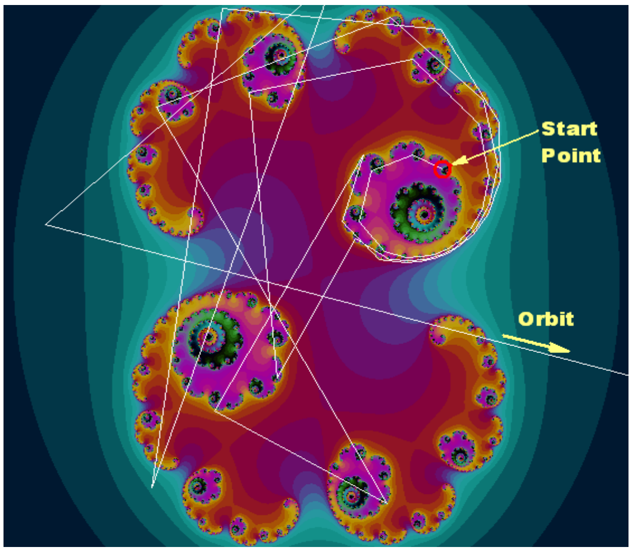


Fig. 3 Example of an orbit

Typically, the coloring algorithm produces a single value for each pixel. Since color is a three-dimensional space, this one-dimensional value must be expanded to produce a color image. A spread method is to create a linear color palette as a sequence of 3D color values. The value computed by the coloring algorithm is used as a gradient along this line.

If the last palette color is connected to the first, a closed, segmented loop is formed and any real value from the coloring algorithm can be mapped to a defined color in the gradient.

Generally, gradients are linearly interpolated in RGB space, but they can be interpolated in other spaces like HSL and interpolated with spline curves instead of straight line segments.

The gradient selection is essential in creating a fractal image. This value determines which parts of the image are stressed out. In extreme cases, two images with the same fractal parameters, but different color schemes may be totally different.

3.2.1 The Escape-Time Algorithm

The *escape-time algorithm* is one of the earliest coloring algorithms, and in many programs it is still the only option available. Due to its simplicity, it is recommended for those that are rookies in fractal software. Unfortunately it is not very interesting from the artistic point of view because it produces discrete values.

The algorithm is based on the number of iterations used to compute if the orbit sequence converges to infinity or not. It can be demonstrated that when the orbit of any value of z_0, z_1, z_2… z_n exceeds a border region R, it always diverges towards infinity. For each fractal type a different shape R, with a minimum size is defined. If the orbit is stopped when z_n is outside of R, the escape-time algorithm uses as coloring value the length n. In software implementations it is necessary to set a limit to the number of iterations to prevent infinite loops.

Generally, R is set as a circle, centered in the origin, with radius 2. The reason is that for the Mandelbrot set, it can be proven that as soon as $|z| > 2$, the orbit will diverge too. Different types of image were created by using other shapes and positions for R: stars, triangles, ellipses and others.

Although these shapes must, mathematically speaking, include the circle of radius 2 in order to correctly compute the divergence, some tests were made on other shapes (smaller radius).

Other complex coloring algorithms are implemented but this article is focused on how a simple formula and simple algorithms can generate interesting images and music.

3.3 Fractal Music

The data can be mapped into a single tonality which could help illustrating the pitch data in relation to itself, but no tonal motion is allowed in the music. This does not represent a musically creative decision, but mono-tonality allows a clearer illustration of the set data.

3.3.1 Music Algorithm

For each "image" with the borders: 312 pixels wide by 256 pixels vertical, the distance between each pixel is computed according to the Julia world window. By scanning 8 times from left to the right the "image" to cover the 256 pixel vertical height the 32 voices are formed. The pitch remains the same for each voice throughout, voice 1 the highest pitch and voice 32 the lowest pitch. The horizontal scan is an 1/32 notes at a tempo of quarter note with the value 60. The iteration data from the Julia calculations is mapped to key velocity. The higher the iteration result, the louder the pitch presence.

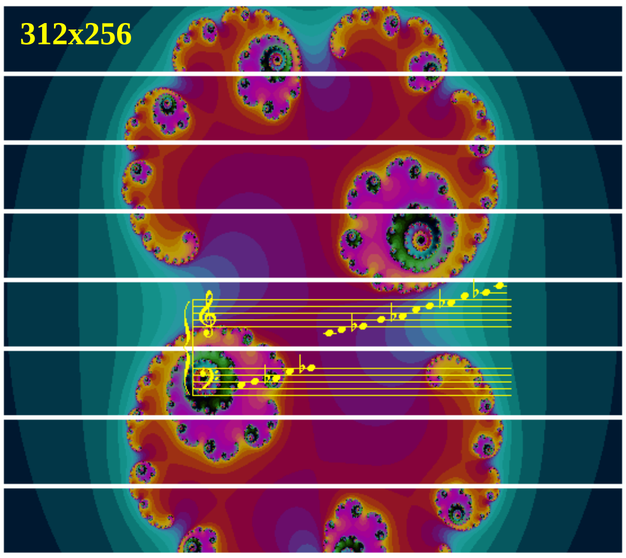


Fig. 4 Fractal image generation sample

The velocity data is stratified to 26 values (dwell bands), the lowest being suppressed (i.e. silent background).

Though the Julia set can be mapped to sound in many ways, the purpose here is two-fold:

1. Demonstrating the feasibility of sonifying (as opposed to visualizing) scientific data, and
2. Demonstrating the possibility of finding musical structures in the Julia set.

When composing music, creative manipulation must be involved. The "composition" part was kept to a minimum in the algorithm; it might lessen the musical qualities but having as a purpose to preserve the objectivity toward the scientific data.

For any dwell band, the initial "attack" is the only one sounded in any singular voice and the reiteration of attacks on a repeated note in a given voice is suppressed. A note is sounded when for a given voice there is a change of value. Therefore, the ongoing background noise is cleaned up in order to clearly hear the leading edge of the shape.

4 Implementation

Every fractal rendering software requires massive computational efforts and any optimization of the code is more than welcomed.

First, one could observe that the bail out condition could be optimized for speed. $|z| > 2$ is equivalent with $|z|^2 > 4$, but the latter is much faster because we don't compute the square root.

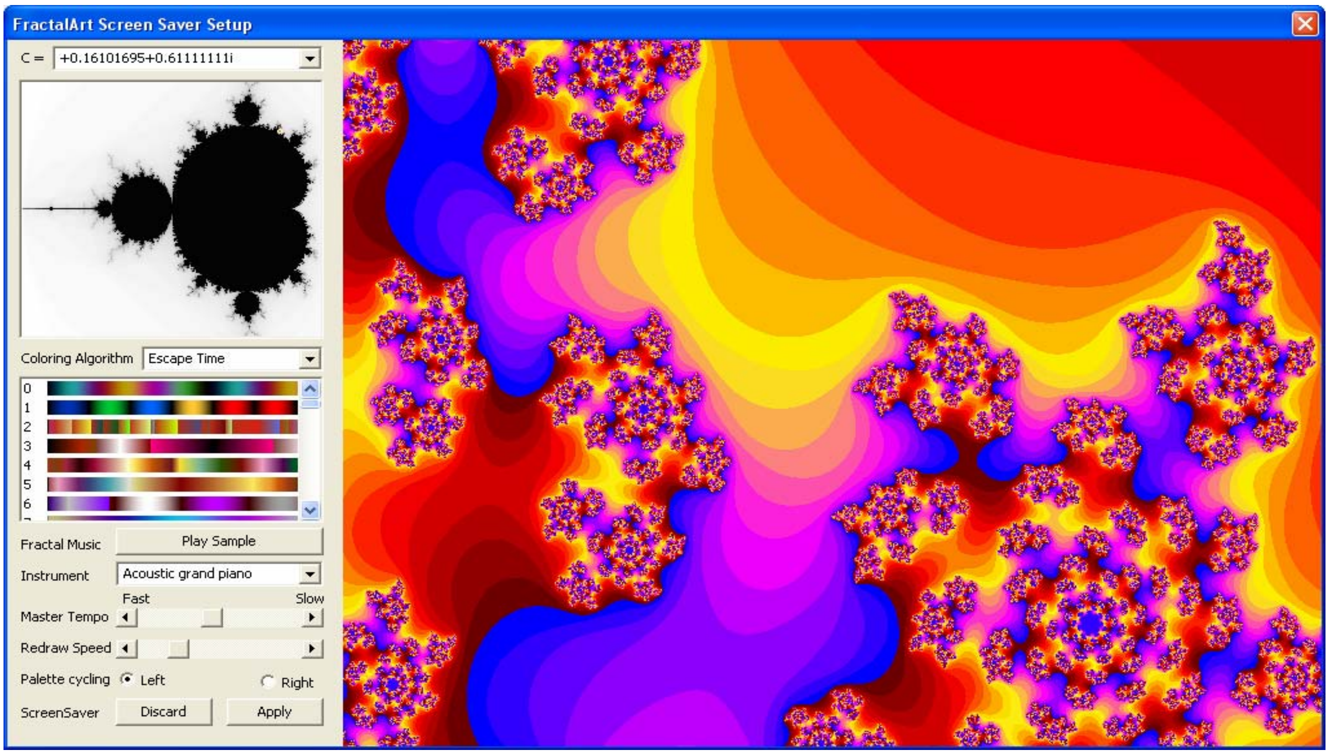
```
complex P, Z;
for each point P on the complex plane {
  Z = 0;
  for i = 0 to ITER {
    if abs(Z)^2 > 4.0
      set point color to palette[i];
      break the inner loop and go to the next
point;
    Z = Z * Z + P;
  }
  set point color to black;
}
```

Another optimization is the implementation of SSE (Streaming SIMD Extensions) Julia computation routines on Intel Pentium 4 Processors. These instructions perform 4 pixel color computation at the same time using a more precise format.

The code is written in C++ and uses Win32 API which is optimized also for speed of execution. The application plays all music data on a MIDI device, using a selected instrument from a list of 127 possible MIDI instruments.

One major problem of the application was the music as it was hard to play all data from the fractal image because of the self-similarity property was reflected in the music too. This issue was solved by playing only the edges of the fractal and the rest of the music data skipped.

One can also play a fractal music sample that is the music played during the screen saver period but is 8 times compressed in time and space.



C = +0.16101695+0.61111111i
Coloring Algorithm: Escape Time
Fractal Music
Instrument: Acoustic grand piano
Master Tempo
Redraw Speed
Palette cycling: [✓ Left] Right
ScreenSaver

Fig. 5 Fractal Art Screen Saver Setup

All these information are saved into the system registry and used when displaying and animating the fractal. The animation is performed by cycling the colors from the palette and simply updating the display.

5 Conclusions

This paper presented a software that creates a fractal screensaver. The application generates both fractal images and music based on the Julia and Mandelbrot fractals. The four coloring algorithms implemented are founded on: escape-time, distance estimators, escape angle and curvature estimation algorithm.

In this article it was emphasized only the escape-time coloring algorithm and the music generation. The music is created for different instruments on 32 voices. The horizontal scan is an 1/32 notes at a tempo of quarter note with the value 60. The iteration data from the Julia calculations is mapped to key velocity. The higher the iteration result, the louder the pitch presence.

The application creates a linear color palette as a sequence of 3D color values. The value computed by the coloring algorithm is used as a gradient along this line.

The application is configurable and permits the user to select the color palette, a region in the fractal and even the instruments to be played.

The result is a screen saver application that manages to combine beautiful fractal images and strange fractal music.

References:

[1] Benoit B. Mandelbrot, *Fractals and Chaos: The Mandelbrot Set and Beyond*, Springer, 2004

[2] Benoit B. Mandelbrot, *The Fractal Geometry of Nature*, W. H. Freeman, 1982

[3] Benoit Mandelbrot, *Gaussian Self-Affinity and Fractals*, Springer, 2001

[4] Benoit B. Mandelbrot, Michael Frame, *Fractals, Graphics, and Mathematics Education*, The Mathematical Association of America, 2002

[5] Charles Madden, *Fractals in Music: Introductory Mathematics for Musical Analysis*, High Art Press, 1999

[6] Heinz-Otto Peitgen, Hartmut Jürgens, Dietmar Saupe, *Chaos and Fractals*, Springer, 2004

[7] Kenneth Falconer, *Fractal Geometry: Mathematical Foundations and Applications*, Wiley, 2003

[8] Masahiro Nakagawa, *Chaos and Fractals in Engineering*, World Scientific Publishing Company, 1999

[9] Nigel Lesmoir-Gordon, *Introducing Fractal Geometry*, Totem Books, 2006

[10] Nigel Lesmoir-Gordon, *The Colors of Infinity: The Beauty, The Power and the Sense of Fractals*, Clear Books, 2004

[11] Roger T. Stevens, *Creating Fractals*, Charles River Media, 2005

[12] Roger T. Stevens, *Fractal Programming in C*, M & T Books, 1989