

# Fractal Generated Music

Patrick Chavez and Darren Tully  
Math 45 — College of the Redwoods

December 19, 1997

## Abstract

The reason for the writing of this paper is to explain the basics behind fractal generated music and its relationship to linear algebra.

## 1 Fractal Music

Before showing any mathematics; some background of fractals and fractal music are to be addressed. Music and math have always had a close relationship, Pythagoras even had a hand in showing that tonal harmony is closely related to the numerical values of frequencies. Fractals, came into the public eye in the late seventies with the Mandelbrot set being graphed from a computer. Soon after the boom of personal computers, musicians were given a new tool to compose music, with the aid of MIDI(Musical Instrument Data Interface), and PC programs. It only took a few years for people to see the originality of fractals and for new way for musicians to express a fractals' beauty.

In the late eighties Music From the Fringe was being created. Music From the Fringe was a series of programs that exploited the potential of fractals to create music. The result from the program mapping calculated values of fractals to audio frequencies, was a bizarre bunch of screeches, howls, and beeps; nothing really musical. Now in the late nineties, there are quite a few really good fractal music programs that can be downloaded from the Internet. Most programs are based on the original concepts of Music From the Fringe, but use the sound capabilities of MIDI, and/or other new sound programs.

### 1.1 Fractals

Shapes that we think of as random, are really the products of complex shifting webs of numbers obeying simple mathematical rules. The word “natural”, that we have often thought to mean “unstructure” is, in fact, describing shapes and processes that appear so intensely complex that we cannot understand the simple natural laws at work. Mathematical analysis and computer modeling have revealed that the shapes and processes we encounter in nature— the way that mountains erode or rivers flow, the way that snowflakes or islands achieve

their shapes, the way that light plays on a surface, the way milk folds and spins into coffee as it is stirred—all these things can be described by the interaction of a mathematical processes.

Consider the coastline of an island. On a map we can see random curves that have been shaped by the elements of nature. If we look closer, we see that smaller curves are similar in shape to the larger curves. Continuing the process of magnification to continually smaller snapshots reveals a similarity that obtains to the very grains of sand on the island's beaches. Fractals have the same characteristics.

### 1.1.1 The Basic Equation

The basic linear iterated function equation is given by

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} r * \cos \theta & s * -\sin \varphi & x_n \\ r * \sin \theta & s * \cos \varphi & y_n \end{bmatrix} + \begin{bmatrix} h \\ k \end{bmatrix}, \quad n = 1, 2, \dots, k \quad (1)$$

Where the original point is given by the vector

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix}$$

The matrix of rotation is given as

$$\begin{bmatrix} r * \cos \theta & s * -\sin \varphi \\ r * \sin \theta & s * \cos \varphi \end{bmatrix}$$

The horizontal and vertical shift is the vector

$$\begin{bmatrix} h \\ k \end{bmatrix}$$

The new vector is

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix}$$

The number of transformations that the given fractal goes through is given as  $n$ : where  $r$  and  $s$  are scalars.

## 1.2 Fractal Example: Fern

In *Matlab*, the fern iterated function can be written as an M-file using the `if`, and `elseif` commands as a type of `for` loop. David Arnold was able to show us the basic structure for plotting the  $x$  and  $y$  points using the following *Matlab* commands:

```
function x=Fern(n)
x0=[1 1]';
```

```

x=zeros(2,n);
x(:,1)=x0;
for j=2:n
    p=rand;
    if ((0<=p)&(p<0.25))
        theta=0;
        phi=0;
        h=0;
        k=0;
        r=0;
        s=.16;
    elseif ((0.25<=p)&(p<.50))
        theta=-.0436;
        phi=-.0436;
        h=0;
        k=1.6;
        r=.85;
        s=.85;
    elseif ((0.50<=p)&(p<.75))
        theta=.8552;
        phi=.8552;
        h=0;
        k=1.6;
        r=.3;
        s=.3;
    else
        theta=2.094;
        phi=-.8726;
        h=0;
        k=.44;
        r=.3;
        s=.37;
    end
    R=[r*cos(theta),-s*sin(phi);r*sin(theta),s*cos(phi)];
    x(:,j)=R*x(:,k-1)+[h;k];
end
plot(x(1,50:n),x(2,50:n),'.','markersize',1)

```

Where  $p$  is the probability of where the next transformation will occur. If the probabilities are not evenly distributed, sections of the graph will become more defined than other sections. The graph shown below was created using this program:

The values for each of the transformations (maps) are given in the table below:



Figure 1:

Maps	Translations		Rotation in degrees		Scalars	
	$h$	$k$	$\theta$	$\varphi$	$r$	$s$
1	0.0	0.0	0	0	0.0	0.16
2	0.0	1.6	-2.5	-2.5	0.85	0.85
3	0.0	1.6	49	49	0.3	0.3
4	0.0	.44	120	-50	0.3	0.37

Each transformation follows the basic iteration shown in equation (1).

### 1.3 Composing the Numbers

For fractals to be able to produce music that would pass as musical, a degree of control on the mathematics is needed. There are an infinite number of ways to have the values mapped to audible frequencies or notes. The real music comes about, when more than just the notes are mapped from the values created by the fractal function. Changes in the position of plotted values are also taken into account for pitch, rest, and duration of the notes being played. All of this is easily used because there are usually more plotted points than there are notes in typical musical composition.

In *Matlab*, using the `sound` command, the computer is able to map the values generated into preset frequencies. The command runs off of an  $[n \times 2]$  matrix; therefore the output values should be transposed and normalized (values range from 0 to 1) before using the command. The values should be given a name,

i.e. Y or X. More can be done to change variables of the sound by typing `help sound` at the *Matlab* control prompt. To use the command, type: `sound(Value Name, Frequency in Hertz(Speed of reading the values), Bits(8 bits gives monosound and 16 gives stereo sound))`. *Matlab's* sound capabilities are not the greatest, but an idea of fractal generated music can be achieved fairly easily.

In other, more detailed fractal music programs, the numbers on the vertical axis of the graph can be scaled to a range of six octaves and then rounded to the nearest eighth of a semitone(vibrational frequencies). Then, with the aid of MIDI, the notes are read as analog, converted to digital data and given a type of sound i.e. tuba, guitar, bells...etc. The digital information is then reread as an analog signal for smoother changes in pitches and notes and can be fed back into the computer or played through an instrument with MIDI capabilities.

## 1.4 Criteria for Music Made from Fractals

A complete fractal music program will cover the following criteria:

1. Produce musical notes instead of precise frequencies.
2. Create music over a limited range of octaves.
3. Create music using only a fraction of the total number of plotted values.
4. Produce recognizable patterns in the music.
5. Introduce music which isn't too predictable nor too random.

An excellent program that we used to help us understand and create fractal music was MusiLab. This program can be downloaded for free for thirty days. MusiLab follows all of the criteria above with the addition of many MIDI sounds, and control of the mathematics involved to play the fractal. The net address is given in the **References** section of this paper.

## 1.5 Conclusion

To create actual music from a fractal equation is an extremely long process. Even with a great program as a guide, many hours are needed to manipulate the data into an actual musical composition. Fractals create a recursively random set of points. For music to be brought out of a fractal, human touch is needed. Most people that use fractal generated music aren't looking for an easy way to compose, they are looking for something new, something that will spark the imagination and help create a new style of music that others can appreciate. Today, most popular music seems redundant and boring, but with the aid of fractal generated music, maybe some of the traditional musical barriers, and current systems can be broken. Fractals are a new source of original musical inspiration. This is a huge step in the future of music composition and technology. Fractal music gives us a new way to express a fractals beauty orally,

and offers musicians an endless pallet of original and complex sounds to create from when painting a musical piece.

## 1.6 References

- Arnold, David, **Iterated Function Systems**,  
<http://192.168.12.1/instruct/darnold/ifs/index/html>, December 1997.
- Fractal Music**, <http://wso.williams.edu/~skaplan/fractal/chapter4.htm>
- Greenhouse, Robert, **The Well-Tempered Fractal** ,  
<http://www-ks.rus.uni-stuttgart.de...le/schulz/fmusic/wtf/docu.html>, December 1997.
- Kinderman, Lars, **MusiNum-The Music in Numbers** .  
<http://www.forwiss.de/~kinderman/musinum/musinum.html>, December 1997.
- McDuff, Richard, **Music and Fractal Landscapes**,  
<http://sdcc10.ucsd.edu/~icamacla/musicfrc.html>, December 1997.
- Nelson, Gary, **Nelson Info Page**, “Real Time Transformation of Musical Material with Fractal Algorithms”  
<http://timara.con.oberlin.edu/~gnelson/~gnelson.html>, December 1997.
- Peitgen and Saupe, **The Science of Fractal Images**. New York: Springer-Verlag, 1988.
- Strohbeen, David, **Fractal Music Lab Musiclab** , “MusiLab” ,  
<http://member.aol.com/strohbeen/fml.html>, December 1997.