Using Self-Similarity for Sound/Music Synthesis

Shahrokh David Yadegari
The Media Laboratory, MIT E15-484b
Cambridge, MA 02139
E-mail: sdy@media.mit.edu

Abstract

This paper is a description of work in progress exploring the use of self-similarity for sound synthesis. Motivating the work is a desire to look into the possibilities for using self-similar structures in the auditory domain (e.g., finding relationships between sound and music.) This work is a preliminary step toward a design for an alternative music notation system. The system uses hierarchy and recursion principles to organize structures and musical ideas. In any creative work, establishing the relationship between the whole and the parts is done through a constant movement between different domains (e.g., foreground and background, micro and macro levels, form and content.) Currently we are experimenting with the idea of using fractals as a tool for capturing some of the systematic natures of the creative process in music, as well as finding a new language for timbre synthesis.

In this paper, we will give a brief summary of manifestations of self-similarity in music. Further, we describe the design and implementation of a program we have developed for sound synthesis and music score generation. The program uses a paradigm similar to Lindenmayer's L-system[4] to develop the synthesis parameters. These parameters are coded in multi-layered structures, each level specifying the re-writing rules (e.g., time segmentation, frequency and amplitude progressions.) Different node re-writing algorithms are used to create self-similar or self-affine structures. The vertical harmonics are created according to horizontal frequency development.

1 Introduction

Do you sometimes feel that there are never ending amounts of structure in a piece of music? Eventhough one may "know" the piece by heart, one still can be captured by the balance between the expected and the unexpected parts. There should exist a certain amount of integrity and coherency among the different parts of a whole. Not enough integrity will cause the whole to break to pieces and eventually become noise. Too much coherency will cause it to be part of others and not be able to stand by itself. The question we ask in this paper is whether it is possible to find formal constructs in such qualities.

2 A Programmable Score Editor

As more and more parameters (e.g., scale, rhythm, form) are freed in music, computers become more useful in the compositional process. We imagined a programmable score editor. We started off with the traditional notation, and viewed the notes as icons for events in time. Then, in order to introduce the concept of hierarchy we viewed the notes as collections of notes as well (e.g., a double click on a tree note opens up a new staff.) In the definition stage, icons are assigned to different structures. They also can be ornamented at execution time to define operations from higher levels.

The decision to represent structures of different perceptual layers in the same way was a programming decision; however, it seems to have interesting musical connotations. If we define an icon using its own definition, we will be segmenting time in smaller and smaller pieces and eventually reach the timbre level. Stockhausen[9] studied and experimented with the relationship between sound and music. He has designed a rather elaborate method of timbre composition using rhythms by what he calls "phase duration" (which is supposed to act as a parallel to pitch.) It is not far fetched to think that the higher level musical understanding of our mind has evolved according to the lower level structures of the sound. Following this path a subjective question arises, "Could an entity such as music exist without sound?"

We can also view this issue from the angle of form and content. Schroeder[7] points out that the cantor set[1] can be a resolution to the seeming paradox of infinite divisibility of matter. Although it is naive to simply look at sound as material and music as form, it may be a good starting point for a model. Koblyakov[3] predicts that in new music, material and organization are going to be inseparable and new parameters (e.g., sound quality) are going to emerge. It is already rather difficult to separate many of the traditional parameters in computer music.

3 Some Instances of Self-Similarity in Music

Schroeder[7] shows that the auditory paradox created by the Shepard Tone[8] has become possible due to the self-similarity of the signal. Generally, a Shepard Tone is created according to the following Weierstrass function:

$$w(t) = \sum_{k=0}^{M} \cos(\beta^k t)$$

Although Shepard applies a formant-like envelope to the frequency domain representation of the signal, it is done for smoothing the perceptual transition and sustaining the paradox effect. The paradox is created from the fact that we try to extract a one-dimensional[2] variable (pitch) out of a multidimensional signal (timbre.) We can think of pitch as a value which identifies the relationship between the partials of a signal in a one-dimensional way. Different frequency components of the Shepard Tone are in geometrical relationships with each other. If we view the frequency domain representation of the signal, time scaling according to the same geometrical relationship β does not change the "body" of the signal but only its boundary conditions, therefore we hear the same pitch and not a pitch scaled according to the scale factor.

$$w(\beta t) = \sum_{k=0}^{M} \cos(\beta^{k+1} t) = \sum_{k=1}^{M+1} \cos(\beta^k t)$$

The self-similarity of such signals is quite obvious in the time domain. We can explain the paradox phenomenon by stating that we hear the same pitch when the signal is time-scaled by the appropriate amount because the signal is scale invariant (with limits).

Voss and Clarke[11] have found some applications of $1/f$ noise in music. The $1/f$ noise, which is characterized by the slope of its power spectrum (on a log-log graph,) has been found in many natural phenomena, from electric components to flood level of the river Nile[12]. Dodge[1] finds fractals and $1/f$ noise to be an interesting paradigm for computer-aided composition. He also suggests that the "memory"of $1/f$ noise can account for its success. This "memory" can be explained by its

[1] The cantor set is created by a recursive percolation of a line segment[5].

[2] Here the word "dimension" is used as a perceptual quality and not a technical term, although they may be close. Especially in the context of dynamical systems, "dimension" has a very distinct technical meaning.

scale invariancy and its long-term autocorrelation. Voss and Clarke have shown that most music, regardless of culture, behaves very close to $1/f$ noise. They have also stated that listeners have found $1/f$ noise to be most "music-like". Listeners found white noise to be too random and $1/f^2$ noise to be too correlated. One can think of $1/f$ noise as a border between randomness and predictability. Short utterances of $1/f$ noise can masquerade themselves as music; however, a longer listening leaves the listener unsatisfied, since naturally, there is no thought or culture behind the signal. Mandelbrot[5, page 375] believes that this scheme does not extend below the note level, since the high frequency energy in instruments (e.g., fiddle body, wood-wind pipes, and the resonance of the lungs) are governed by a different mechanism; therefore the high energy spectrum is more like $1/f^2$ than $1/f$.

Vaughn[10] has studied the emotion in some recorded Karelian Laments. In her study, she views the pitch contour as an analog of musical behavior, and she treats the pitch contour as a set of shapes (a signal), rather than a series of notes. She has investigated the boundary properties of repetitive melodies in the Laments and she has found self-similar structures in the pitch contour at the point of transition of the singer to a trance-like state near the ritual moment.

4 A Synthesis Program

In order to test some of these ideas, we developed a program which embodies a language[3] to define the synthesis parameters in recursive and hierarchical manner. The paradigm is similar to Lindenmeyer's string-rewriting L-System[6], which was originally developed for modeling growth and interaction of cells in a multicell organisms[4]. In short, the L-system is a system in which one defines a state table, a next-state matrix according to the current state and the input, and an output matrix (which usually is the current state.) Lindenmayer used this system to model the growth of plant filaments, and he used two-sided input cells to model filaments with branching.

Originally we developed the software without any knowledge of the L-system. The programming decisions were made according to the design of the previously described score editor. The paradigm itself was based on our perception of the physical matter, which is perhaps one of the most basic manifestation of structure in our consciousness. The initial intent for design of the language was so that it would be possible to keep a library of structures, make structures be context sensentive, and be able to connect different structures in hierarchical or recursive connections. This language would then be used for storing what the user would specify as the score in a graphical score editor. Currently what is implemented in the language is only recursive and hierarchical definition of the parameters.

5 The Synthesis Language

For every layer of the parameter definition, one defines a *seed*[4] which itself is a collection of *structures* and pointers to objects for production of the end result. These latter objects are responsible for mapping the developed parameters to the desired output (e.g., soundfiles, scores for other systems, or graphical pictures.) *Structures* are a collection of *points*. *Points* are a collection of factors and a pointer to a *seed*, which defines their lower content. Currently we are using "time", "frequency", and "amplitude" as different factors in *points*. The program first starts with the *seed* called "main", which has a *point* as its initial starting value. Then according to the factors found in the *points* in the *structure* of "main", it re-writes the initial "main" as a series of *seeds*. This procedure is repeated recursively until the time segment in a point is smaller than the "stop recursion" time. At every level for every *seed* an output production service routine is called, with the *seed* value (which is represented as a *point*) as its argument.

[3] Mammad Zadeh developed the initial parser.

[4] The name, seed, was suggested by Gerhard Eckel during discussions on the subject in summer of 1989.

Imagine the following score (which is the input to the program):

```
/* a sample score */
point i1 { time: 40; freq: 8000; amp: .01; seed: main; }

point p1 { time: 0.05; freq: .4; amp: 1.5; seed: main; }

point p2 { time: 0.95; freq: .9; amp: 1.01; seed:main; }

struct s1 { p1; p2; }

seed main { value: i1; struct: s1; seedobj: snd; }

sound snd {
        srate:    22050;             /* sampling rate */
        file:     "sound_file";      /* out file */
        stop_rec: .01;               /* stop recursion time */
}
```

The time and frequency development is illustrated for the first two levels of recursion in Figure 1. With a "stop recursion" value of 0.01 seconds, some parts are developed to as many as 57 levels.

We experimented with a few different output production and parameter development schemes. Currently the parameter developments can be according to geometric or arithmetic relationships. The arithmetic development is done according to a constant value (which is found in the *main seed.*) Once we understand the behavior of the system better, we can make this relationship programmable as well.

For output production we first used the parameters as instantaneous frequencies and kept a linear phase, which means that we only used the last level parameters. Because of the Devil's Staircase (Cantor function) effect[5, page 82] we had a difficult time with noise created from sudden changes in frequency or amplitude. Then we experimented with "stop recursion" values small enough to be less than the period of instantaneous frequencies. In this way we were shaping the wave according to the lower-level parameters. This approach may technically seem simple minded or naive. The motivation behind the experiment was to see if we could establish a structural relationship between the normal level of music perception[3] and the shape of the auditory signal.

We also experimented with using every level of the development parameters as values for partials of the sound, which means that at every level every *point* adds a partial for its period of time. Again because of the Devil's Staircase effect, there was some noise which we could not avoid. We solved this problem by applying a window function to every partial. With this configuration we were able to obtain very interesting auditory results. Figure 2 is the spectrogram of the sound created by the sample score.[5]

As we can see, time is fractally segmented according to the given structure. This is true for any length of time (the boundaries have to be picked correctly, and the lower limit is defined by the "stop recursion" value.) The frequency content of every segment has a static part, which is created by higher level developments, and a changing part, which is created by the lower levels. The changing part of the frequency content is changing according to the same structure. If we assume that the hierarchy of time is one of the hierarchies of our perception, then we have a sound which can manifest the same structure in different layers of our perception.

It is perhaps too early to conclude anything from this work. We are currently in the process of redesigning the language to implement the ability to have libraries of structures. We are also thinking about how different structures can interact with each other in the development process.

[5]The spectrogram was created using Dan Ellis' software for digital audio[2]. This system was an indispensable tool for implementing the project.

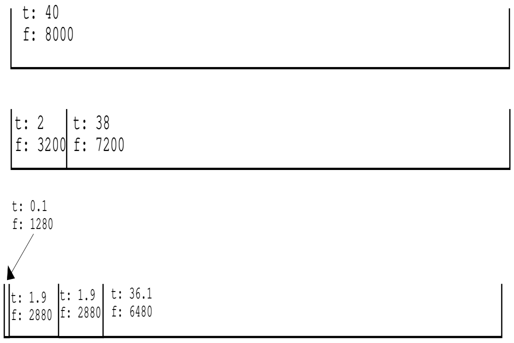

Figure 1: Time and frequency development of the sample score for two levels

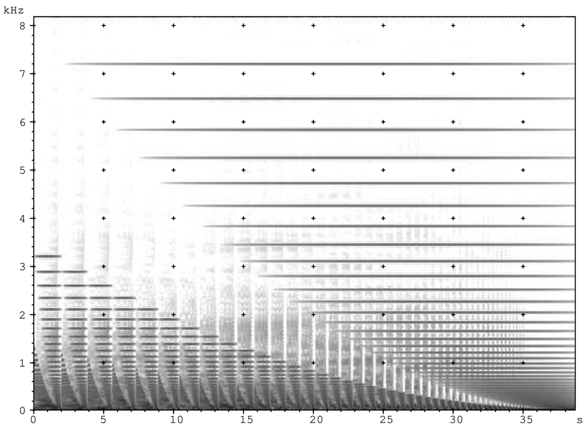

Figure 2: Spectrogram of the sound created by the sample score

6 Acknowledgment

I would like to thank Prof. Tod Machover for his support, direction, and encouragement, without which this work would not have been possible. I would also like to thank Robert Rowe, Dan Ellis and Prof. Barry Vercoe for their patience.

References

[1] C. Dodge. *Profile:* A Musical Fractal, *Computer Music Journal* Vol. 12, No. 3, 1988.

[2] D. Ellis. Some Softeware Resources for Digital Audio in UNIX, *Media Lab Technical Report,* 1991.

[3] L. Koblyakov. Notes préliminaires au sujet de la musique nouvelle, *Dissonanz,* No. 7, Zurich 1986.

[4] A. Lindenmayer. Mathematical Models for Cellular Interaction in Development, Parts I and II. *J. of Theoretical Biology,* 18:280-315, 1968.

[5] B. B. Mandelbrot. *The Frtactal Geometry of Nature,* Freeman, San Francisco, 1983.

[6] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants,* Springer-Verlag New York, 1990.

[7] M. R. Schroeder. Self-Similarity and Fractals in Science and Art, *J. Audio Eng. Soc.,* Vol. 37, No. 10:795-808, October 1989.

[8] R. N. Shepard. Circularity in Judgments of Relative Pitch, *J. Acoust. Soc. Am.,* Vol 36, No. 12, pp. 2346-2353, 1964.

[9] K. Stockhausen. .....how time passes...., *Die Reihe 3,* English Version Theodore Presser Co., Bryn Mawr, Pennsylvania, 1959.

[10] K. Vaughn. Emotion in Sub-Structural Aspects of Karelian Lament: Application of Time Series Analysis to Digitized Melody, *Yearbook for Traditional Music,* International Council for Traditional Music, 1990.

[11] R. F. Voss and J. Clarke. "1/f noise" in music: Music From 1/f noise, *J. Acoust. Soc. Am.,* Vol 63, No. 1, 1978.

[12] R. F. Voss. Random Fractal Forgeries, *Fundumental Algorithms for Computer Graphics,* NATO ASI Series, Vol. F17 Springer-Verlag, Berlin, Heidelberg, 1985.