Real Time Transformation of Musical Material with Fractal Algorithms

Gary Lee Nelson
Conservatory of Music
Oberlin, OH 44074
(440) 775-8223
gary.nelson@oberlin.edu

Introduction

This paper is a chronicle of the composition of four pieces during the period 1988-1993:

Fractal Mountains (1988-89)
Summer Song (1991)
Mountain Song (1992)
Goss (1993)

All of these pieces are part of my ongoing research and interest in the application of mathematical models and techniques to the composition of musical form and structure.

In "Fractal Mountains," I used recursive subdivision of time, pitch, and amplitude in a fractal algorithm that generates an accompaniment from material played freely on a MIDI wind controller (the "MIDI Horn"). The piece was realized entirely in electronic sound. "Fractal Mountains" won first prize in the international competition for micro tonal music at the 1988 Third Coast New Music Festival in San Antonio and has been recorded on CD by Wergo [1].

In "Summer Song," I used a symbolic replacement grammar (Lindenmayer Systems) to create the formal structure in a work for solo flute.

The first and second pieces were written separately without any relationship to each other. The third piece is a combination of the first two wherein the solo line from "Summer Song" is transformed in real time by a modified and enhanced version of the fractal algorithm from "Fractal Mountains." This work is performed with a Macintosh computer, digital synthesizers, and the MIDI Horn.

MIDI Horn

The MIDI Horn is as an alternative to the piano keyboard for controlling digital synthesizers. John Talbert, Oberlin's music engineer, designed and constructed the instrument. The MIDI Horn makes no sound by itself but rather reflects the character of the synthesizer that it controls.

The instrument consists of two boxes, a control module and a microcomputer. The control module (see Figure 1) contains a breath pressure transducer that determines onset and release time of notes by comparing breath pressure to a threshold. Timbre nuance can be shaped while notes are in progress by programming synthesizers to respond to continuous changes in breath pressure (MIDI controller 2).

The seven buttons on the top of the front panel control pitch. The top four of these buttons follow the pattern of a brass instrument to play a chromatic scale from C down to Db. In order from the top, the buttons lower pitch by 2, 1, 3, and 5 semitones. Normal alternate fingerings are provided.

Three additional buttons determine register within an eight octave range. These three follow the same fingering pattern as the top three buttons but instead of lowering pitch by semitones they lower pitch by octaves. The single alternate fingering (the third button alone) provides the longest drop, seven octaves.

An eighth button for the little finger of the left hand serves a function similar to the shift key on a typewriter. By pressing this button, the other seven buttons generate a program number for the MIDI synthesizer. With all combinations of seven buttons, the full range of 128 MIDI program numbers is available.

Eight buttons and two joy sticks on the back of the instrument send signals that instruct the computer to control aspects of the accompaniment. The buttons transmit on/off messages via MIDI controllers 84-91. The joysticks send continuous data as MIDI controllers 16-19 (general purpose controllers 1-4). Since the data from the MIDI Horn passes through the Macintosh first, these controllers can be mapped onto any musical parameter.

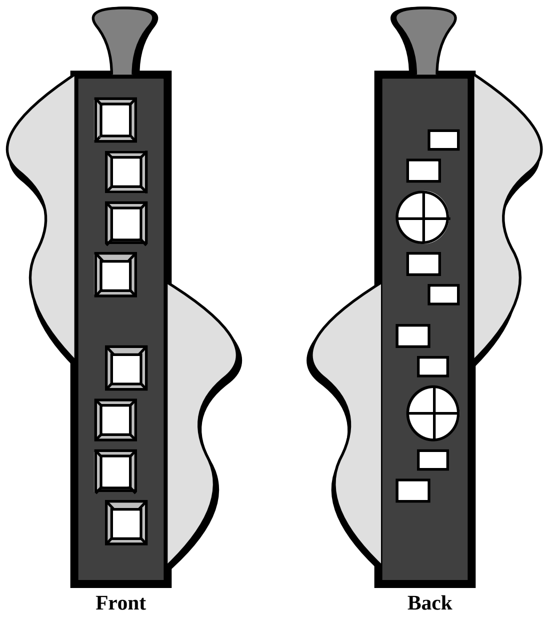

Figure 1. MIDI Horn front and back view.

Fractal Mountains (1988-89)

With his discovery of fractals, Benoit Mandelbrot introduced a new class of mathematical models and a new branch of mathematical science. He may have stimulated a new branch of music as well. A growing number of composers have shown interest in the musical application of fractals. "Fractal Mountains" represents my own first attempt to integrate fractal techniques into interactive performance and composition.

A significant feature of fractals is their self-similarity. Consider the coastline of an island. On a map we see seemingly random curves that have been shaped by the elements of nature. If we look closer we see that, with magnification, smaller curves are similar in shape to the larger curves. Continuing the process of magnification to ever smaller snapshots reveals a unity that obtains to the very grains of sand on the island's beaches. The ramifications of such a process for the organization of musical form and structure seem almost overpowering.

Most of the practical literature on fractals is in the field of computer graphics with focus on the generation of realistic images for still pictures and animation. One of the simplest fractal models is the two dimensional outline of mountain ranges. Initially, we draw lines to represent the major peaks and valleys. These lines are subdivided by a recursive process to produce the next level of detail. When we continue subdivision through generations of ever shortening lines, an image emerges that reminds us of the patterns found in natural landscapes. Figure 2 illustrates the several layers of subdivision used in "Fractal Mountains" to arrive at pleasing melodic contours.

Musical sequences can be generated by mapping the meeting points of each pair of lines onto time and pitch. Each vertex represents the attack point of a note. A second fractal function is used to translate dynamics into MIDI key velocity. The durations of notes are set by another fractal function that shapes texture by controlling the number of notes that are sounding at once. To return to our island analogy, we are determining whether the beach is covered with rocks, pebbles, or fine sand.

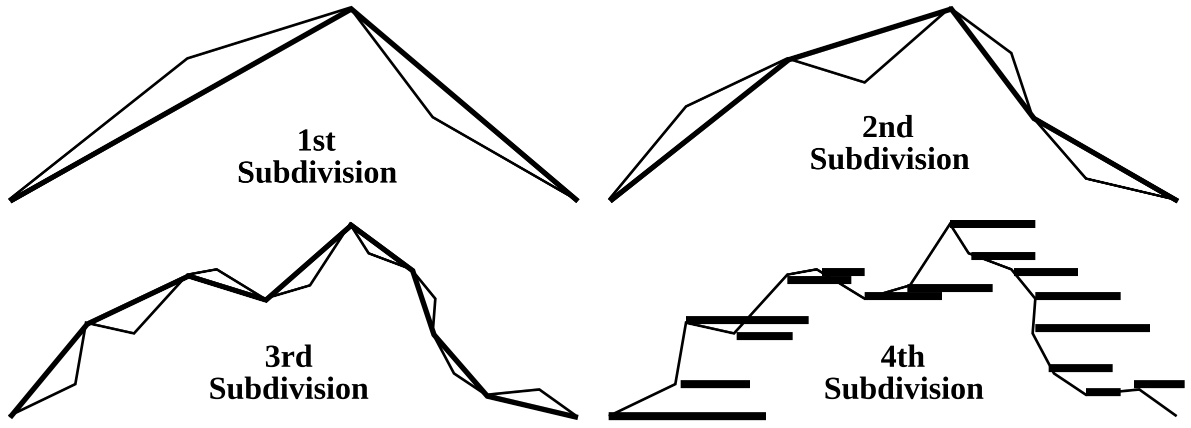

Figure 2. Subdivision of lines to simulate mountain horizon.

The piece is performed on a Yamaha TX816 and a pair of EMU Proteus/1 synthesizers controlled by a Macintosh computer and a MIDI wind controller. The micro tonal pitch system in "Fractal Mountains" is achieved by dividing the octave into 96 equally tempered steps. MIDI channels 1-8 in the TX816 and Proteii are tuned in intervals of 12.5 cents by freezing the pitch bend wheel for each module at the beginning of the piece.

The numbers on the vertical axis of the graph are scaled to a range of six octaves and then rounded to the nearest eighth of a semitone. The whole number part determines what MIDI key will be pressed and the fractional part determines which of the eight channels will receive the note.

The harmonic shape in "Fractal Mountains" comes from the tendency of the fractal model to be attracted to particular numbers and thus to particular pitches and MIDI channels. The pull of these "strange attractors" is one of the most interesting feature of fractals. In music they produce a sense of tension and release that is appealing physically as well as esthetically. Phrases begin with most notes in a single channel. As the phrase progresses, the notes fan out among the eight channels and the richness of the micro tonal palette is revealed. Near the end of each phrase the notes move toward a different channel and cadence in relative consonance. The center of gravity is constantly changing.

Like the tuning system, the timbres are designed with a rich inner life. Voices vibrate and beat against other voices. Each tone color consists of a sustained sound that rises slowly from the beginnings of notes. At higher key velocities the onset of notes are punctuated by bell sounds. The timbres were constructed by interpolation between archetypes that were designed empirically. The resulting orchestra creates a "scale" of timbres with subtle variation from one end of a limited spectrum to the other.

The major features of this tonal landscape are controlled by notes played on a MIDI wind controller and by the time delay between those notes. The single notes from the soloist are passed through the Macintosh and translated into a multi-voice texture. Imagine that we are touching peaks and valleys on a blank canvas and that the mountains appear automatically.

More specifically, each note from the soloist is recorded by arrival time, pitch, and key velocity. Adjacent notes provide the endpoints of lines that represent the slopes of the mountain. The fractal algorithm constructs an accompaniment gesture that traces the space between a pair of points. A three-dimensional interpolation takes place in time, pitch, and loudness.

Some special rules of interaction between the soloist and accompaniment algorithm were found useful. Short time periods (< 400 milliseconds) between notes in the solo part overburdened the accompaniment algorithm and synthesizers and produced objectionably thick textures. Long time intervals (> 20 seconds) caused correspondingly long accompaniment gestures during which the soloist lost control over the evolution of the piece. These extreme time periods are ignored by the fractal algorithm.

In one version of "Fractal Mountains" I superimposed two ranges of mountains as illustrated in Figure 3. The ranges become lines in a monumental contrapuntal structure.

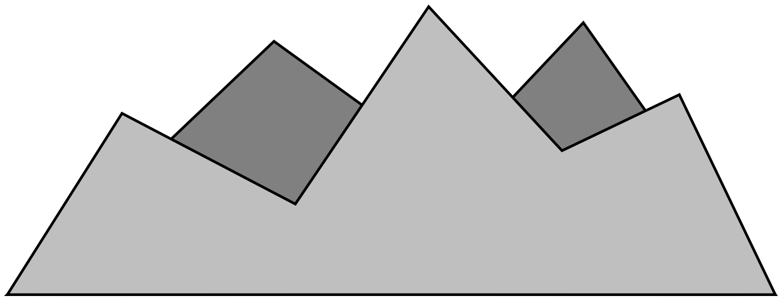

Figure 3. Mountains in counterpoint

The computer program for "Fractal Mountains" was written in C. These programs communicate with MOXC, a system for connecting the actions of the soloist to reactions by the accompanying synthesizers. MOXC is part of Roger Dannenberg's "MIDI Toolkit," public domain software available from the Center for Art and Technology at Carnegie Mellon University [2].

MOXC consists of a parser, an interpreter, and a scheduler. The parser receives signals from the MIDI Horn and informs the interpreter that musical events have occurred. In the interpreter, the composer writes programs that decide what these events mean and what reaction to trigger in the accompaniment. The reactions may be immediate or delayed for a period by passing them to the scheduler. The scheduler keeps track of pending events and executes each event when its delay period expires.

Summer Song for solo flute (1991)

The methods I used in composing Summer Song are extensions of an idea presented in a paper by Przemyslaw Prusinkiewicz at the 1986 International Computer Music Conference[3]. This work was realized with the aid of Lindenmayer systems (L-systems) . L-systems are formal grammars for growing a structured sequence of symbols. The process begins with a "seed" and progresses through a series of generations where each symbol is replaced by a string of symbols. For example, the grammar:

```
X ’ -YF+XFX+FY-
Y ’ +XF-YFY-FX+
```

means that the symbol 'X' will be replaced by the string '-YF+XFX+FY-' in the next generation and the symbol 'Y' will be replaced by '+XF-YFY-FX+'. Since the replacement string contains 'X' and 'Y' the process is recursive and fractal. This particular grammar describes the Hilbert curve.

I implemented L-systems in APL (A Programming Language). APL is a general purpose interactive programming environment that is particularly hospitable to musical experiment. In 1974, I used APL to implement my Musical Program Library [4].

If we execute an APL function for 4 generations with the seed 'X'.

```
4 HILBERT 'X'
```

The result is the string show in Figure 4.

```
-+-+XF-YFY-FX+F+-YF+XFX+FY-F-YF+XFX+FY-+F+XF-YFY-FX+-F-+-YF+
XFX+FY-F-+XF-YFY-FX+F+XF-YFY-FX+-F-YF+XFX+FY-+F+-YF+XFX+FY-F
-+XF-YFY-FX+F+XF-YFY-FX+-F-YF+XFX+FY-+-F-+XF-YFY-FX+F+-YF+XF
X+FY-F-YF+XFX+FY-+F+XF-YFY-FX+-+F+-+-YF+XFX+FY-F-+XF-YFY-FX+
F+XF-YFY-FX+-F-YF+XFX+FY-+F+-+XF-YFY-FX+F+-YF+XFX+FY-F-YF+XF
X+FY-+F+XF-YFY-FX+-F-+XF-YFY-FX+F+-YF+XFX+FY-F-YF+XFX+FY-+F+
XF-YFY-FX+-+F+-YF+XFX+FY-F-+XF-YFY-FX+F+XF-YFY-FX+-F-YF+XFX+
FY-+-F-+-YF+XFX+FY-F-+XF-YFY-FX+F+XF-YFY-FX+-F-YF+XFX+FY-+F+
-+XF-YFY-FX+F+-YF+XFX+FY-F-YF+XFX+FY-+F+XF-YFY-FX+-F-+XF-YFY
-FX+F+-YF+XFX+FY-F-YF+XFX+FY-+F+XF-YFY-FX+-+F+-YF+XFX+FY-F-+
XF-YFY-FX+F+XF-YFY-FX+-F-YF+XFX+FY-+-+F+-+XF-YFY-FX+F+-YF+XF
X+FY-F-YF+XFX+FY-+F+XF-YFY-FX+-F-+-YF+XFX+FY-F-+XF-YFY-FX+F+
XF-YFY-FX+-F-YF+XFX+FY-+F+-YF+XFX+FY-F-+XF-YFY-FX+F+XF-YFY-F
X+-F-YF+XFX+FY-+-F-+XF-YFY-FX+F+-YF+XFX+FY-F-YF+XFX+FY-+F+XF
-YFY-FX+-+-
```

Figure 4. String representation of Hilbert curve.

If we interpret the string in Figure 4 as a series of commands for drawing lines we can make the image shown in Figure 5. In this language "F" means move forward and draw a line of a fixed length, "+" means turn left, and "-" means turn right by a specified angle. The angle of turn in Figure 5 is 90 degrees. The "**X**" and "**Y**" are tokens for parsing the grammar. They are ignored when the image is rendered.

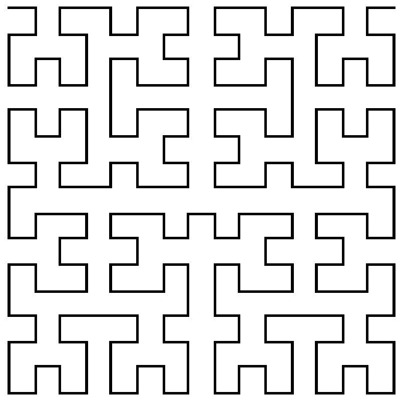
Figure 5. Graphic representation of Hilbert curve.

If we use a musical interpretation that reads the vertices as notes we can map the points into time and pitch and transcribe the result. This is what I did to compose "Summer Song," a work for solo flute.

The curve in Figure 5 proved to be too symmetrical and produced rather dull repetitive patterns with little variety in pitch or rhythm. I changed the angle from 90 to 101 degrees. Then I stretched, twisted and warped the curve a bit. I found the result shown in Figure 6 to be much better suited to make the piece I envisioned.

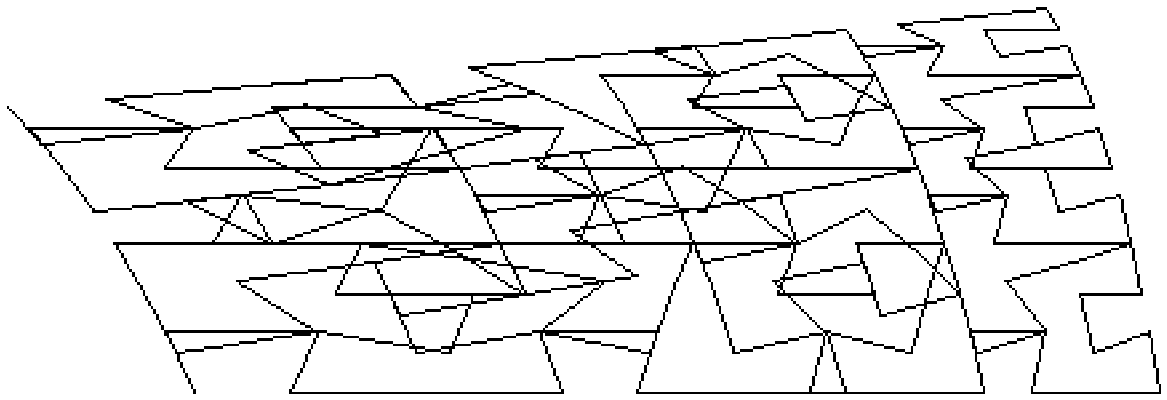
Figure 6. Hilbert curve stretched, warped, and twisted.

The image has become asymmetrical with a compression towards the upper right corner.

Mapping of Figure 6 onto time and pitch involved several steps. The vertical coordinate of each vertex wash interpreted as pitch. The horizontal distance between vertices was interpreted as duration. In drawing Figure 6 the pen moves both forward and backward in the horizontal. Since time cannot flow backward in music, I used the absolute values of the horizontal differences between vertices. The image is, in effect, unraveled into a line that moves continuously from left to right. The result of this interpretation can be seen graphically in Figure 7.

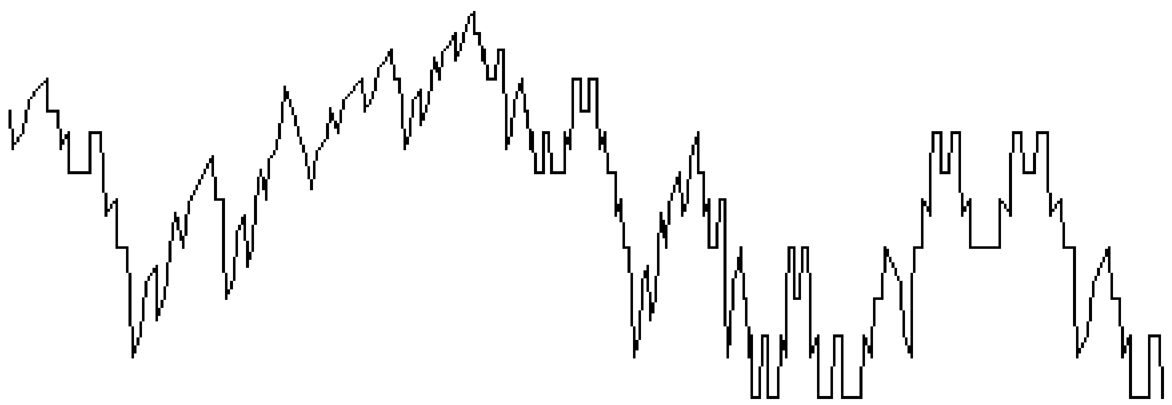

Figure 7. Hilbert curve unraveled to produce pitch and time contour.

The curve in Figure 7 was stretched to a duration of five minutes. The vertical axis was mapped onto the scale shown in Figure 8.

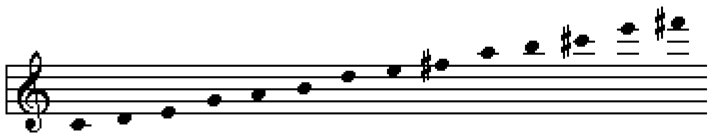

Figure 8. Scale for Summer Song

This scale is a projection of the interval series 2 2 3 over two and a half octaves. The scale is different in each octave although there is a hexatonic quality that was inspired by listening to traditional Chinese flute music in Taiwan during the summer of 1991.

When the pitch mapping was complete, I created a MIDI file and loaded it into Finale 2.6. I used floating quantization to limit rhythmic subdivision to eighth and sixteenth notes and eighth note triplets. This limit on rhythmic complexity along with the conservative pitch range reflects my intent that this piece be accessible to young players.

Mountain Song (1992)

In 1991, I abandoned MOXC in favor of MAX [5] as the vehicle for my work in interactive systems. MAX provides most of the functionality of MOXC in a programming environment that facilitates experiment and incremental development. MAX's object based architecture and graphical user interface allowed me visualize and construct algorithms more intuitively.

My first project in MAX was the translation my repertoire of interactive pieces from MOXC to the MAX environment. After implementing the recursive subdivision algorithm of "Fractal Mountains," I tested it by playing a MIDI files through it. One of the files that happened to be at hand was "Summer Song." The initial result was quite pleasing. Encouraged by this happy accident, I spent some time polishing the orchestration and tuning the algorithm.

In "Mountain Song," the pitches from "Summer Song" are read into a MAX table object. The notes are played out in sequence each time a new note is articulated with the MIDI Horn. The pitches are subjected to an affine transformation and then fed to the recursive subdivision algorithm. As in "Fractal Mountains," very short and very long periods between attacks in the solo part are ignored by the fractal algorithm. The solo player steps to the next note in the "Summer Song" sequence with each new note regardless of the fingering chosen. However, the MIDI Horn player can "take a solo" by articulating notes without changing fingerings. This maneuver locks the gate that feeds notes to the fractal algorithm and prevents the accompaniment from responding.

Goss (1993)

The composition method for the solo violin part in "Goss" is similar to that of "Summer Song." A different function was used, the Gosper Curve shown in Figure 9. In a private communication, Douglass McKenna correctly points out that he is the author of this function. I am happy to reidentify it as the McKenna E Curve. I tracked down my original attribution to a piece of software for exploring L-systems by Paul Bourke from New Zealand. Bourke identifies this function as the "Quadratic Gosper Curve [Dekking, 1982]."

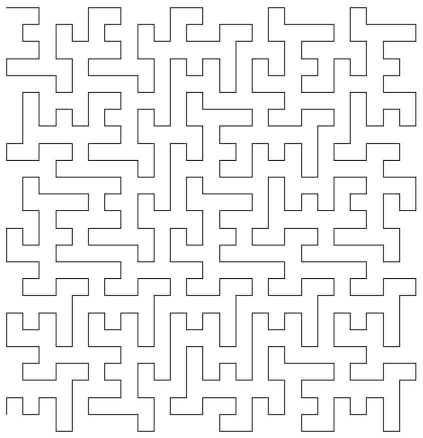

Figure 9. The McKenna E Curve

To reduce symmetry, Figure 9 was compressed toward the upper right corner as shown in Figure 10.

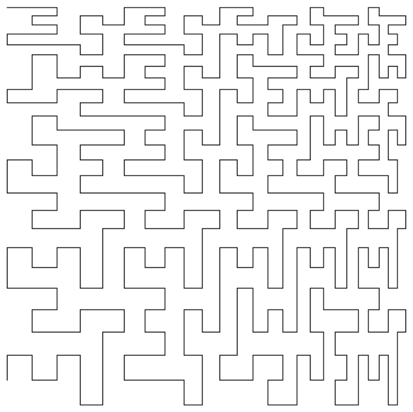

Figure 10. Compressed toward upper right corner

Next the image was twisted, warped and stretched by changing the turning angle to 107 degrees to give the image in Figure 11.

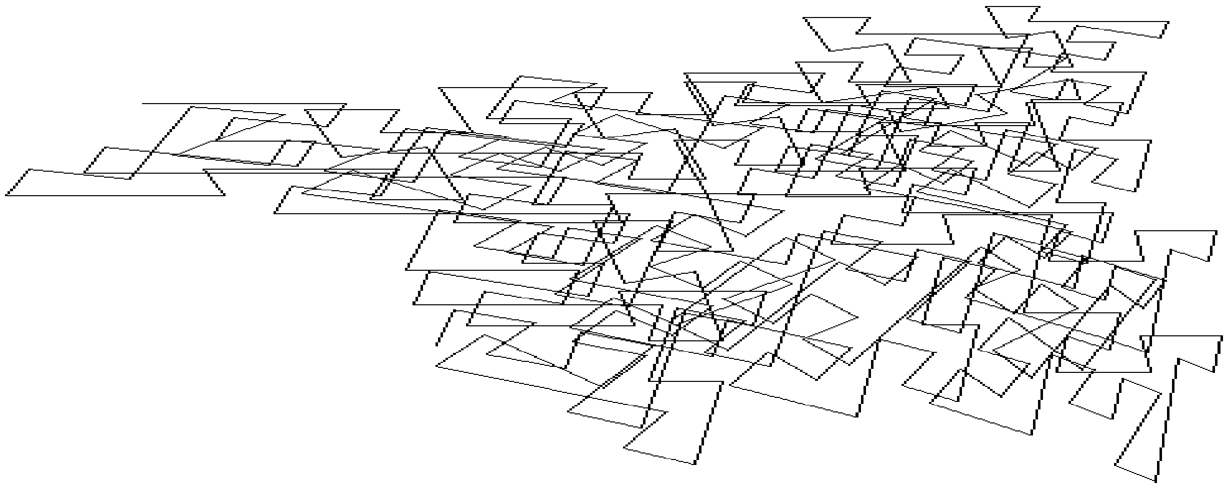

Figure 11. Twisted, warped and stretched

Finally, Figure 11 was unraveled into a continuous line (Figure 12) to form the pitch and time contour for the solo part.

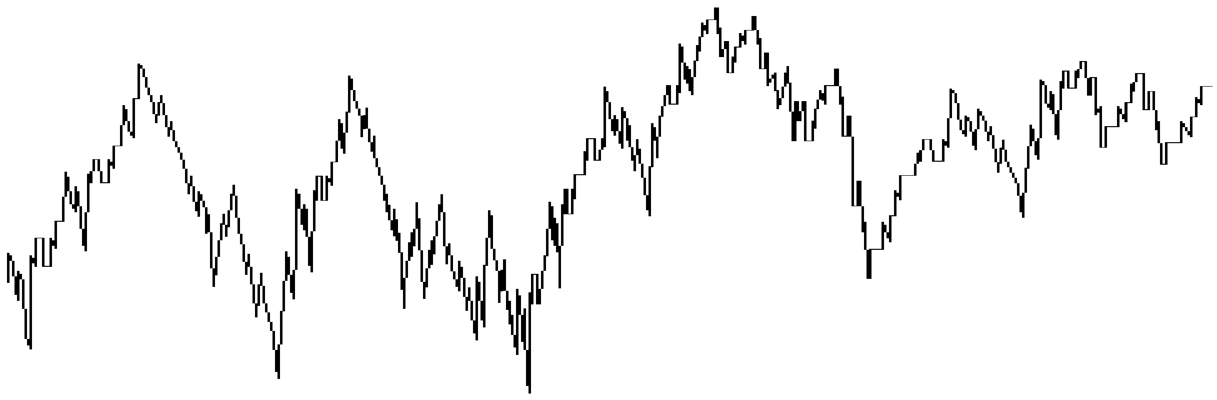

Figure 12. Raveled into pitch time contour

The atonal accompaniment material in "Goss" is generated with an algorithm borrowed from my earlier pieces, "Fractal Mountains" and "Mountain Song." The violinist plays the solo part on a Zeta violin that is outfitted with a MIDI translator. A Macintosh computer "listens" to the notes from the solo part and generates the accompaniment automatically. The close harmonization of the solo part is also automatic using notes from the scale. The soloist exercises control over the harmony and accompaniment with two foot pedals and with dynamic balance that is tied to bow pressure.

Conclusion

Since this paper is in the nature of a composer's notebook the conclusion, I suppose, is that some interesting music was made and the process was fulfilling both intellectually and artistically. This fulfillment is due, in large part, to the nature of the tools that are available to the present day composer. Inexpensive, yet powerful, computers, real time synthesis hardware, and sophisticated software contribute to the growing role of intuition and fantasy in the making of computer music. It appears possible at last to balance art and technology as we pursue our creative dreams.

References

[1] Nelson, G. L. "Fractal Mountains" in *Computer Music Currents*, vol 10 (WER 2024-50), Wergo Schallplaten, Mainz, 1992.

[2] Dannenberg, R. B. "MIDI Toolkit," Center for Art and Technology, Carnegie Mellon University, Pittsburgh, 1991.

[3] Prusinkiewicz, P. "Score generation with L-systems," Proceedings of the 1986 International Computer Music Conference, pp. 455-457, Computer Music Association, San Francisco, 1986.

[4] Nelson, G. L. "MPL: A Program Library for Musical Data Processing," *Creative Computing*, 3:2 (April 1977), pp. 76-81.

[5] Puckette, M. & Zicarelli, D. "MAX: An Object Oriented Programming Environment for Music and Multimedia," OpCode Systems, 3950 Fabian Way, Palo Alto, CA 94303, (415) 856-3333.