Strange Attractors: A Virtual Instrument Algorithm for Acoustic Instruments

Stephen David Beck, Ph.D.
Electro-Acoustic Music Studies
School of Computer
Louisiana State University
Baton Rouge, LA 70803-2504
Internet: MUBEC9@LSUVM.MSNC-LLSU.EDU

Abstract

Composers have long been searching for ways to integrate acoustic instruments with synthesized or computer-processed sounds, particularly in the area of score following. While score following is certainly a logical model for the coordination of computer and performer, I have been most interested in developing interdependent processes between acoustic and computer instruments. My search has been for a logical and musical link between the intuitive processes of acoustic performance practice and real-time computer music synthesis. To do this, I began thinking about a different paradigm to treat these two media, one which is event-based rather than time-based.

The main thrust of this paradigm is to treat the computer sounds as extensions of the acoustic instrument—a "virtual" instrument which behaves and sounds according to rules and algorithms that satisfy an underlying aesthetic of acoustic viability. The initial product of this research is Strange Attractors v.1.06 (1990) for bassoon and computer-controlled synthesis and signal processing.

This paper will discuss virtual instruments, the nature of the Strange Attractors algorithm and how the algorithm's temporal and translation features create a reactive computational music environment responsive to the instrumental subtleties of a human performer.

Searching for a New Paradigm

Clearly, the quest for interactive environments has been going on for quite some time. While the work of these programmers like David Zicarelli, Miller Puckette and Roger Dannenberg (and many others) has afforded us many tools with which to perform and compose interactive computer music, the paradigms by which these tools should be governed are far well defined, and in some instances not defined at all. As a composer, it is very important to me to be fully aware of what and how these interactions are taking place. In my own search for an interactive paradigm, I considered the entire process of sound making, deconstructing the process from its most general aspects down to its most specific.

While there has been much success in previous-compositions combining acoustic and synthesized media, there are problems with the models upon which these interaction paradigms are based. The first of these paradigms is the absolute time paradigm, in which the coordination between acoustic and synthesized media is governed by the elapsed time of a composition. Unfortunately, this paradigm fails to recognize that while music is a time-dependent art form, the art of musical structure is the manipulation of perceived time, not of actual elapsed time. Classical ideas, like rhythmic and harmonic acceleration, are very successful not because they manipulate absolute time, but rather, because they manipulate perceived or relative time.

The second paradigm is the computer-伴奏 paradigm, in which the computer tracks the sounds of a live performer, determines the pitch(es) played, finds that location in a predetermined score, calculates the tempo, and then scales it in internal tempo to that of the performer. Some schemes allow for the repetition and storage of past performances to create an averaged tempo map which the computer then also follows. Much like the rehearsals between live performers, this paradigm simulates the temporal nuances that separate computer performances from human performances. And while this paradigm is most closely modelled after the musical rehearsal process, it does not directly address the issue of translating idiosyncratic acoustic performance parameters into idiosyncratic synthesis performance parameters, except in the area of tempo. It may allow for such translations to occur, but it does not specifically address this issue.

Defining the Acoustic Instrument

I began as the most fundamental part of the music-making process—the physics and mechanics of the instrument. While the mechanism by which each instrument class produces sound may be different, there are certain consistencies in the acoustic output of musical instruments which connect all instrument classes, these being the relationship between performance input and sonic output. For example, it is intuitively obvious that certain performance techniques on a particular instrument will elicit certain kinds of acoustic responses based upon the mechanical response of the given instrument class. But changes in the amount of energy put into the vibrating system will affect the spectral energies in similar ways. We must also speak of the spectral changes

ICMC 332
due to changes in frequency. These changes are well documented and are an important part of how we identify and track different instrument classes. This use of these basic acoustic principles of instrumental mechanism in computer-based synthesis is a concept which I refer to as acoustic stability.

While performance techniques are not always translatable between instrument classes and the effects of frequency-based spectral modulations vary from instrument class to instrument class, the general idea was clear — performance energies (measured amplitude and frequency) are directly translated into spectral responses idiomatic to the given instrument class, and changes in those energies elicit more generalized changes in spectral response.

As such, an event-based paradigm for real-time interaction should have the ability to translate performance energies, whether pitch-related or amplitude related, into synthesis and decision processes which are idiomatic to a computer-based instrument. These transitions need not be direct translations (i.e. strict pitch translation from data input to synthesis output) nor do they need to be associated with similar or even exact acoustic parameters. Regardless of how the performance parameters are translated between input and output, the process as a whole could be described as a "virtual instrument" paradigm.

**Defining The Virtual Instrument**

Much like virtual reality, in which an imaginary three-dimensional space is created and moved through within the memory of a computer, a virtual instrument is an algorithm/process which behaves like an acoustic instrument but does not physically exist. In its simplest form, the virtual instrument consists of two main simple principle process objects: the performer object and the synthesis object.

The performer object is responsible for interpreting the input and making decisions based on the input regarding how the virtual synthesis output should be. Such a decision process could be as simple as note for note, dynamic for dynamic. If the performer object received a message meaning A440 at mf, it would then pass that information along to the synthesis object, where an A440 at mf would be synthesized according to virtual instrument's overall process. This is an example of how an instrument controller (i.e. a MIDI keyboard, wind controller, or Analog-to-MIDI converter) could be implemented in the virtual instrument paradigm. This straightforward translation is certainly not a profound implementation of the virtual instrument, but it illustrates the basic nature of the performer object.

The synthesis object is responsible for handling all tasks with respect to real-time synthesis and signal-processing. It would receive the "response" from the performer object and perform the necessary tasks to synthesize a sound representative of the data provided. Using the previous example, the synthesis object would receive a "play A440 at mf" from the performer object, and then send that data via MIDI to an off-board synthesizer. In a software synthesis-based system, the synthesis object itself would call software routines designed to synthesize its sound. In both configurations, the object and its sound are directly linked together.

Because my own research has been limited to commercially available MIDI synthesizers and signal-processors, the synthesis object portion of the virtual instrument paradigm was restricted in what it could actually do in terms of on-board synthesis. This seemed to be a logical way to deal with synthesis that could ultimately be ported to other computer platforms which could handle on-board real-time synthesis.

**Non-Trivial Implementations of the Virtual Instrument Paradigm**

The example used to first illustrate the nature of the virtual instrument paradigm is rather trivial. It is simply a null method which passes the data it receives directly to the off-board synthesizer to which the virtual instrument is connected. There are other parameters which can be used, and other translations which can be implemented to make the virtual instrument non-trivial.

The performer object could choose based on a number of arbitrary selection methods to transpose all or some of the input data. This includes the mapping of input data on to preset or evolving data maps, using transition tables to determine output data for both pitch and volume, using volume data to control pitch or pitch data to control volume, etc. Because the virtual instrument is an abstract paradigm for interactive systems, the possibilities of transformations are truly limited only by the imagination of the composer.

To implement this virtual instrument paradigm, I began work on an application which modeled these basic ideas. The end result would be used to create a work for bassoon and interactive computer, as well as to create a MIDI-based object-oriented tool set for constructing virtual instrument applications.

**Strange Attractions: A Virtual Instrument Algorithm**

Initially, my concern with interactive systems was with the understanding of acoustic performance energies, their translation into acoustic sound production, and how acoustic performance data could be translated into synthetic sound production in ways which were consistent, inherent and endemic to computer and synthesis technology. Because of the general limitations of Analog-to-MIDI converters, I was limited in using only pitch and velocity information as representations of performed pitch and intensity. As such, I decided on a translation scheme that treated pitch and intensity separately and independent of one another.
Pitch was handled in such a way that the Strange Attractor instrument would respond with pitches based wholly on what the acoustic performer would play, but with an increasing randomness to the pitch class. The length of the pitch class buffer is dependent on the number of NoteOn events which have transpired, with the buffer length equal to one at the start, and gradually increasing over the course of whole performance. When the buffer length is at one, the pitch selection process can only select the pitch class that was just played. As the performance progresses, the buffer gets larger, providing a wider range of selection possibilities and therefore an increased randomness to the Strange Attractor's performed pitch. When the buffer becomes full and a new pitch class is added to it, the most previous pitch class will then be removed from the buffer, thus keeping the buffer current with the acoustic performer and the output increasingly random but clearly related to the current pitch material.

Velocity is translated much more straightforwardly. First, the velocity of the NoteOn message is applied directly to the pitch the performer object sends to the synthetic object. More interestingly, the velocity parameter is used to determine the apparent spatial location of the synthesized sound. Clearly, spatial placement is something very easy for computers to synthesize, but nearly impossible to do with acoustic instruments (used from running around stage really quickly). This aspect was both intriguing and conceptually consistent with what had been set out to do.

A loud acoustic performance is translated into a spatial placement of the synthesized sound on stage with the acoustic performer. The quieter the acoustic performance, the further back the spatial placement. This apparent spatial placement is achieved through real-time control of a DSP-7 digital mixer and signal processor, utilizing the principal of distance localization cues as outlined by John Chowning in his article on simulating moving sound sources. Radial placement of synthesized sounds is done through random panning.

Certainly, the translations of Pitch and Velocity used here provide an adequate, but simple interactive process. Yet, the need to apply more to the process was clear. One aspect of the Apple MIDI Manager which I found interesting was its ability to write MIDI events into the future, and to have them actually sent at the assigned time. I decided that this feature could easily be manipulated to create response delays between the timeStamp of the received NoteOn message and the timeStamp used for the synthesized sound. These delays, along with the pitch selection algorithm, could create an interesting quasi-canonic process.

Ultimately, a routine was created that would increasingly delay the virtual instrument's response to a NoteOn message, or increasingly re-synchronize the response to the original message. An event counter was created to determine when the process would switch between the delay and sync mode. With each event, the timeStamp of the response would be increased by a factor determined by the formula

\[
\text{delay} = \text{last\_delay} + \text{random} \times (\text{delay\_factor})
\]

where last\_delay is the delay of the previous event, and random (delay\_factor) is a value between 0 and delay\_factor milliseconds.

The process would alternate between the two modes every fifty NoteOn events by switching the delay\_factor to its negative value. Every complete "flip-flop" cycle would increase the rate at which the distance localization cues would increase or decrease by increasing the value of the delay\_factor. A minimum delay value was set at 0 milliseconds for obvious reasons, and a maximum limit of three seconds delay was needed to prevent the process from running out too far into the future.

The resulting temporal process behaves by oscillating between two temporal points (0 second delay to 3 second delay), with the paths between those points being unpredictable in a general sense, but not in an exact sense. This kind of temporal bifurcation is suggestive of the Strange Attractor concept in chaos theory; hence the name of the virtual instrument, Strange Attractor.

To increase the coherence and elegance of this overall process, the length of the pitch class buffer was linked to the flip-flop cycle, which would increase by one with every flip-flop. The synthesized pitch classes were also transmitted to a harmonizer which would transpose the acoustic sound by an interval equal to the distance between C and the transmitted pitch class. The volume of the harmonized sounds was determined inversely by the velocity of the acoustic sound.

Four of these virtual instruments, which would run simultaneously and independently, were assembled into one setup to create a rich and complex acoustic performer would be controlling four voice harmonies through his own acoustic performance as determined by the composed musical score. This homophony will slowly distill down to four voice polyphony (or quasi-homophony) as the random delay factors increase, the distance between acoustic event and synthesized response, only to have the voices reintegrate themselves into homophony at either the 3 second delay maximum or the 0 second delay minimum.

ICMC 334
Writing Music For A Virtual Instrument

In approaching the actual composition, Strange Attractions v1.0h, I treated the virtual instrument with the same concern and detail that I would for an acoustic instrument; identify the strengths and weaknesses of the instrument and then exploit those areas to demonstrate and manipulate the performer's own abilities. As such, I wanted to highlight the temporal variations and quasi-cyclic nature of the virtual instrument, while at the same using the flip-flop oscillation of processing to control the structure of composition.

By keeping track of the number of 1/16th notes in the events of the composed score, I was able to know where in the algorithm would be in the processing cycle. At the very beginning of the work, I knew that the computer would always respond to the acoustic performer with an octave variant of the pitch class written in score, and in synchrony with the performer. A loud and rhythmically awkward motif was used to establish that the computer was not playing according to some preset sequence, but rather in actual coordination with the bassoon. As the opening section transpired, the computer would slowly, almost imperceptibly, become unsynchronized with the four computer voices responding almost but not quite in time to the bassoon. To highlight this, I wrote a passage which continued the opening motif, but more densely and with a rhythmic acceleration followed by a rapid resolution, that would be echoed about 1/2 second after the bassoonist stopped.

By this point, it was clear that the computer was listening to the bassoon, reacting to the bassoon, but the process governing the temporal coordination was mysterious to the listener. Short and sparse staccato motifs helped to delineate the connection between the real and virtual instruments, but at the same time created a strange illusion of polyphony that equally disoriented the listener's perception of who was playing what.

By using a synthesis patch that slowly simulated a bassoon sound, this disorientation between what was real (acoustic) and what was not real (synthesis and signal processing) became the overriding raison d'être of the piece. But more importantly, I found myself creating a motivic and rhythmic structure that clearly was dependent on the Strange Attractor algorithm. In the same sense that composers write for a given instrument's technical capabilities, I too, was writing music idiosyncratic to both the bassoon and the virtual instrument.

Summary

My success with both the concept of virtual instruments and the work, Strange Attractors v1.0h led me to create a set of C-language MIDI objects which will allow me to further develop this paradigm, construct more sophisticated interactive algorithms, as well as teach the virtual instrument concept to my students. The object set currently includes object-based interfaces to the Apple MIDI Manager, a chain of command structure based on the symphony orchestra model, and a single object superclass which allows complex command structures to be handled through a common method.

This set of objects will be ported to the NeXT platform this summer and expanded to include real-time graphic capabilities for interactive multimedia. These objects will be discussed in more detail after further and more extensive development.

In thinking about event-based interaction between acoustic and synthetic instruments, I have attempted to treat the computer as an abstract musical instrument which must be performed by or through a live performer. In developing an abstract response algorithm not dependent on score following or pattern matching, I have had to focus my musical writing for the virtual instrument as articulations of virtual processes. Different musical materials will be articulated as differently on these virtual instruments as they would on acoustic instruments. Experiments with improvisation have shown a great potential for cohesiveness and musical interactions, due to the physical and control processes and the manipulability of the algorithm's output. As such, I have given acoustic performers new parameters in which they can explore their musical expressiveness and given myself new imaginary instruments for which to write music.

Bibliography

Dennenberg, Roger 'The CMU MIDI Toolkit.' Center for Art and Technology, Carnegie Mellon University, Pittsburgh, 1986.

ICMC 335