Encoding Musical Knowledge in Software Performance Tools

Arne Eigenfeldt
School for the Contemporary Arts
Simon Fraser University
Burnaby, BC CANADA

Introduction

Traditionally, electroacoustic composers have had to choose between two essentially divergent alternatives: if you were interested in the creation or manipulation of complex timbres, you would work in the studio; if you were interested in performance or improvisation, you could work in live electroacoustics, but at the expense of complex timbres.

Is such an exclusive choice still necessary today? Many of the signal processing algorithms formerly limited to non-realtime computation are now available for performance with today's dual core processors, and virtual synthesizers can produce timbres as almost as rich as any studio-based process.

If that's the case, why are a majority of works at electroacoustic festivals studio-created compositions? Many of those pieces that do feature a computer on stage may use a live performer whose sound is being processed live. However, many of these works are extensions of the tape plus performer idiom, rather than a continuation of the experimental tradition of live electronics.

The experimental approach to live electroacoustic music has now been relegated to the late night concerts. There seems to be several reasons for this separation. The works on these performances tend to be much longer (20 minutes or more), while the accepted duration of a daytime concert is seldom beyond 12 minutes. Many of the late night works are improvisational, while the daytime concert works follow the established EuroAmerican tradition of the fixed time object: composition in which the relationship between all musical elements is carefully considered, and the resultant object does not vary in performance.

A Brief History

The music heard on the "late night" electroacoustic concert, often called "laptop music", has been influenced as much by popular electronica as by the art music tradition. This latter tradition can be traced back to Cage's live electroacoustic music in the late 1930s, but many of its models arose in the 1980s. The development of MIDI, the availability of commercial synthesizers, and the introduction of personal computers facilitated a growth in live electronics, if for no other reason than composers no longer had to solder anything.

But there was still that tradeoff - although it was a lot easier to work in real-time, the technology still limited the complexity of timbre. This, coupled with the note-based sensibility of MIDI, forced composers to focus their efforts on elements other than timbre for complexity, most obviously pitch and rhythm.

Randomness and improvisation

The already existing relationship between interactive music and improvisation, established by such pioneers as David Tudor, David Behrman, Joel Chadabe, et al, also strengthened during this time. Todd Winkler noted that this connection is due to the fact that many processes in interactive music tend to model activities found in improvisation, namely *listening*, *interpreting*, *composing*, and *performing*[1].

In such systems, compositional algorithms can generate material in which musical parameters can be altered by the composer/performer or by constrained random procedures, both of which allow for the shaping and influencing of musical output. The use of such random procedures allows music to be unpredictable, yet remain within a defined range. As Truax suggests, complexity in musical systems is a direct correlation to its unpredictability; in this respect, music that is randomly generated (or influenced) is, by definition, complex[2].

Constrained random generation has been used to imply the spontaneity of improvisation. As Winkler suggests, compositional algorithms of many flavors can easily deal with discrete data, such as MIDI pitches. However, since computers have been able to process sound in real-time, the gestures in live EA are no longer note-based, but sample-based. The richness of sample playback and processing now places live EA on the same timbral level as studio based composition - but without the time for reflection.

Today's Laptop Performer

I've noted in my own laptop performances certain tendencies, tendencies that are also evident in the majority of laptop music I've heard:

- the use of distinct layers, based upon sample playback;

- layers are introduced one at a time, "launched" by the composer;

- the extensive use of looping of material;

- live processing, independent within the layers. Foreground layers are usually processed by the composer through direct control;

- background layers processed using constrained random methods; for example, panning and filter cutoff frequencies.

Here is an example of my own music, somewhat typical of my "non-academic" musical pursuits:

http://www.sfu.ca/sca/sounds/arne/Shameface.mp3

---

[1] Winkler, Todd. "Strategies for Interaction: Computer Music, Performance, and Multimedia" *Proceedings of the 1995 Connecticut College Symposium on Arts and Technology*, 1995.

[2] Truax, B. (1994) "The inner and outer complexity of music", *Perspectives of New Music*, 32 1, (1994), p. 176-193.

One could argue that because each layer progresses on its own through constrained random procedures, their interaction is “complex”, because, although general tendencies can be predicted, details cannot.

While this hypothesis can be argued theoretically, I suggest that the musical results tend to be limited, and rather homogeneous.

For example,

- the layers never truly interact;

- only one layer at a time changes in any meaningful way;

- the amount of variation is consistent, particularly in the background layers;

- significant change occurs slowly, since each layer needs to be altered on its own.

Returning to the musician model

What would be the necessary tools to overcome these limitations? If we return to the roots of live EA, and that of improvisation, can we more closely examine the model.

How do musicians act in an improvisational setting? (certainly not randomly). Could the computer behave more like a real musician? Could it be trusted to make more high level decisions?

In order to do this, we need to move beyond simple constrained random procedures, and attempt to encode musical knowledge into software.

Music and AI

The exploration of artificial intelligence in music isn’t new, of course. But unlike many previous attempts, I was interested in exploring it through a purely musical, rather than conceptual, approach - I am a composer, rather than a programmer or AI researcher. I also wanted to pursue music apart from the modernist tradition: atonal, arhythmic gestural music (which I believe is easier to model). I also felt I had to limit my goals- the creation of complex rhythm outside of melody or harmony, and to explore the more elusive quality of “groove” and its compositional development.

This became *Kinetic Engine*, which premiered as an installation at ICMC New Orleans. It is written using tools I am most familiar with - Max/MSP.

A paper describing Kinetic Engine in detail is available here:

http://www.sfu.ca/~eigenfel/research.html

I won’t describe it in detail, but instead focus upon a few aspects on its application of encoded knowledge.

A brief disclaimer: The encoded knowledge within *Kinetic Engine* is my own: I view this software as a first step in creating an intelligent extension of my own compositional aesthetic. At the risk of sounding blasé, my own training as a composer suggests that I have some knowledge as to what determines interesting music, so I am relying upon that knowledge. Nor am I attempting to model a specific style of rhythm. I have not attempted to extract a rule base from existing music, nor use a database of stored rhythms. Lastly, none of my research is based upon cognitive models: I make no claims that humans think or act in this manner. I am interested in producing software whose *outcome* resembles intelligent decision making. As such, my research can be considered behaviour-based AI.

A brief description of *Kinetic Engine*

First and foremost, the software has an understanding of what makes an interesting rhythm, using rules that I have determined through self-analysis. Using fuzzy logic to construct patterns beat by beat, patterns are built in relation to a global density control, and continually tested using various algorithms to determine their rhythmic "soundness". Internal patterns and repetitions (i.e. hemiola) may emerge, and are encouraged.

The latest version of *Kinetic Engine (*currently under development*)* is based upon intelligent multi-agents that determine independently how to react to the current context. Their actions (and reactions) are dependent upon "personalities" set when the agent is generated. For example, whether the agent will tend to syncopate; how responsive it is to global change; how willing it is to upset a stable system. This reflects the notion of a group of individuals improvising within an ensemble, reacting to one another but having musical personalities. Although the agents determine their own actions, they communicate portions of their current state to the other agents, which can inform the other agent's decisions. This reflects the notion that musicians may not know each others personalities, but they can hear what each other are playing. Anything "heard" or "displayed" can be reacted to.

Agents can also exhibit "social" behavior by forming connections with other agents. If connections are established, agents alter their patterns to form rhythmic polyphony (i.e. interlocking patterns) or heterophony (similar rhythms). Agents can exist as distributed agents, networked onto different machines. An upcoming performance will involve 12 different computers placed in a circle, communicating via a network.

Knowledge-based timbral selection

One aspect of encoded knowledge that I'll describe a little deeper is the timbral selection method. A method that I used in earlier beat-generating software, also used extensively in laptop music is based upon constrained randomness. Samples are organized into folders of timbral similarity (in this case, a certain drum struck in a variety of ways); the software can then choose randomly from within the folder, creating timbral variation, but limited in its variety. At some point , a higher level timbral change will be required, and the composer will most likely have to select a new folder. Having several instances of such a model active will further necessitate higher level control. One limitation that results, however, is that no timbral combination will result that isn't predetermined.

In order to allow more flexibility in timbral selection, without arbitrariness, onecould move to a learning based system, in which the software could be taught as to which timbres could be combined with which other timbres.

However, I would argue that, from a musician's point of view, this kind of information is implicitly known.

If a drummer is given a role within an ensemble (for example, low frequency bass drums), that player could look over many available instruments, and have a good idea which instrument would produce that kind of sound.

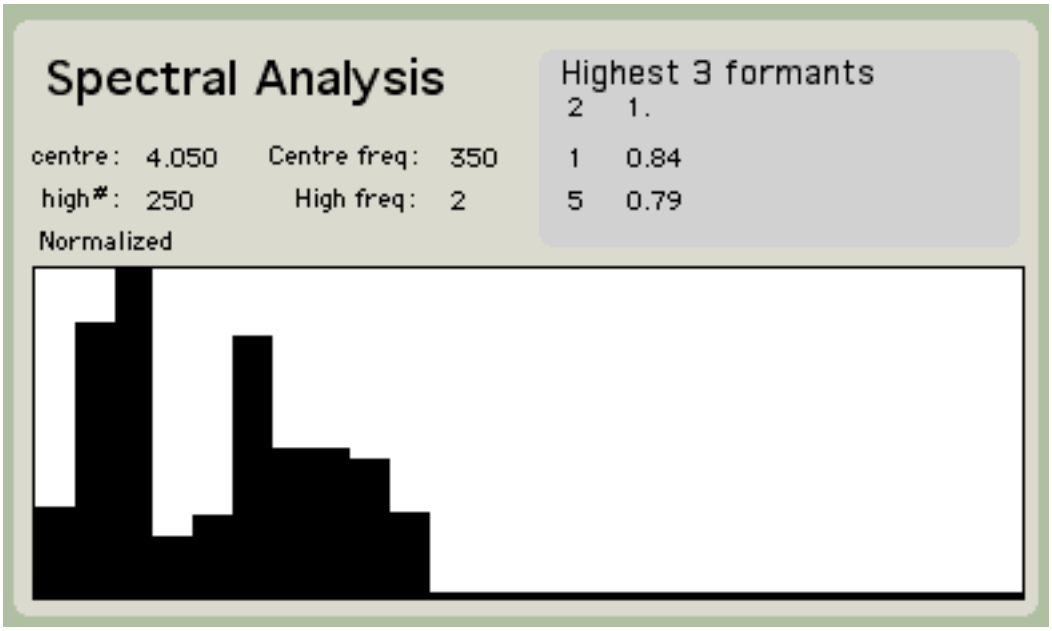
This is the type of analysis used in *Kinetic Engine*: the software knows something about each instrument, and makes a decision about which one to choose based upon this knowledge coupled with recognizing its own role in the ensemble.

Each timbre is analyzed beforehand for its amplitude level within 25 frequency bands. The above example is a low pitched drum, as evidenced by energy only in the first three bands.The results, however, are left purposefully crude: only the three highest bands and their relative ratings are stored. The purpose is not to identify a drum based upon its timbre, but instead to offer a method of organizing the timbres through comparison, similar to how a musician might classify drums: i.e. low, but bright; dull mid-range, etc.

African_Breketa

36	36	normal	2	1.0000	0	0.7900	1	0.6900
38	38	normal	2	1.0000	0	0.7900	1	0.6900
40	40	normal	1	1.0000	0	0.9400	2	0.7200
41	41	normal	1	1.0000	0	0.9800	2	0.8700
43	43	normal	0	1.0000	1	0.6300	2	0.5400
45	45	normal	0	1.0000	1	0.6300	2	0.5400
47	47	normal	0	1.0000	1	0.9900	2	0.8900
48	48	normal	0	1.0000	1	0.9700	2	0.8700
50	50							

Several samples, their playing style (normal) the highest amplitude frequency band (0,1,2 in these cases), and their amplitudes



An example analysis of a low pitched drum. Only the three highest frequency bands (2, 1, 5) are kept. Note that spectral change over time is not stored, but maximum energy over the duration of the sample.

Fuzzy relationships

Once each sound's timbre has been determined, related timbres (those within a folder, such as "tabla") are rated in "closeness" to each other using fuzzy logic. Thus, given one method of striking the instrument, which other method is timbrally "close" (a slight variation) and which is dramatically different (a large variation)? When the software is performing, a separate parameter determines the level of change required (small, medium, large); timbral change can now occur intelligently based upon this database.

60	norm	62	0.43	64	0.15	67	0.10	69	0.10	65	0.05
62	norm	60	0.43	64	0.15	67	0.10	69	0.10	65	0.05
64	norm	67	0.38	69	0.38	65	0.24	60	0.15	62	0.15
65	norm	67	0.39	69	0.39	64	0.24	60	0.05	62	0.05
67	norm	65	0.39	64	0.38	69	0.36	60	0.10	62	0.10
69	norm	65	0.39	64	0.38	67	0.36	60	0.10	62	0.10

The fuzzy relationship database for a set of sounds. Note 60 has a .43 closeness to note 62 (the closest timbre), and a .05 closeness to note 65 (the most remote timbre).

Lastly, all samples are rated for potential as low, medium, or high frequency timbres - the individual players in *Kinetic Engine* assume these roles. In this way, the software is free to choose from all possible sounds, but will always make a musical intelligent decision. For example, if the player is playing low frequency timbres (i.e. the bass drum player), it can choose any timbre that exhibits these frequencies. Thus, it can choose

the Argentinian Bom, but would not play the side stick samples. Completely unexpected combinations of drums result, but combinations that follow the prescribed principles.

Future Directions for Encode Timbral Knowledge
This example of encoded knowledge works very well in performance, due mainly to the limited nature of sample data. All the timbres have a short duration, and any timbral change that occurs is essentially ignored.

However, it suggests to me the potential for such methods to be employed in more live acousmatic works (if such a medium exists). Earlier, I suggested that many laptop performances have longer time frames; one reason for this is the time it takes to determine which new sound to add: staring at a menu full of sounds and trying to identify the best candidate at the moment.

If each sample had already been analyzed (obviously in more detail than in my case), the software could dramatically limit the potential choices, based upon the current context (i.e. how many other layers are active?) and existing timbral relationships (i.e. if a currently playing sample has a dominant energy band coming up, disallow samples with similar spectrums).

Future Directions
The use of agents to achieve complex rhythms has been extremely encouraging in achieving *musical* results. Version 1 tended to produce more homogeneous rhythms, since it wasn't agent based, but hierarchical; as such, rules were determined from above. Version 2 more closely models the individualism of improvisers: the agents work together to create interesting rhythms, but their own personalities interact in extremely complex and unpredictable ways.

The next step is to incorporate similar processes to control melodic and harmonic creation and development. Also, more intelligent performance algorithms. At the moment, it may sound like the music is intelligently composer, but not necessarily performed.