A REAL-TIME MIDI COMPOSER AND INTERACTIVE IMPROVISER BY MEANS OF FEEDBACK SYSTEMS

Sergi Jordá
Laboratorio de Informática y Electrónica Musical
Centro para la Difusión de la Música Contemporánea
Santa Isabel 52
28012 Madrid
Spain

ABSTRACT

PITEL is a software environment for polyphonic real-time composition and improvisation, based on non-linear recurrence. It is written in the language MAX on the Macintosh II, and can generate, under the control of a mouse conductor or composer, up to eight voices in the form of MIDI data, while optionally listening and reacting to one or more MIDI players.

INTRODUCTION

Music has a big deterministic component as present events strongly depend on many previous events. Since Hiller's beginnings in algorithmic composition, Markov chains seemed a reasonable way to deal with music memory and correlation. However, given the big amount of calculations and choices caused by any successive increment of the chain order, Markov music usually presents a very short correlation, sounding much better over brief, than over long fragments. This approach also brings a new problem, with the election of the transition probabilities, which always favour a particular musical style (1).

We wanted our tool to be as flexible as possible (not oriented to any special musical aesthetic), and show a much larger correlation. Believing that basic organization properties are quite independent from the media they are applied (biology, ecology or even music...), we decided to build a feedback system with no specific musical knowledge (2)(3). We will see that even when such systems are completely deterministic (with no noise or no random decisions), the big amount of parameters which can be evolved and the complexity of their relationships, make them completely unpredictable (though appreciably organized).

MUSICAL FEEDBACK RELATIONS AND AUTOCORRELATION

Let's simplify, and consider a monophonic musical voice as completely described by three independent successions: a pitches succession not_n, a durations successions dur_n, and an intensities succession vel_n; at any discreet time i, the sound is determined by the not_i, played with an intensity vel_i, during a time dur_i.

Although not always explicitly declared, special fragments of these successions (i.e. an 8 or 12 bars theme, a rhythmic motif, etc.) have a special role in the structure of many (if not all) kinds of music, as many internal properties of a fragment are closely related to the previous one. We will name N, the size of one of these fragments, and we will call it the periodicity parameter or more simply the period.

If we write the relation: $X_i = F(X_{i-N}, X_{i-k})$ with $0<k<N$, we are meaning that at any instant i, the value X_i depends on the value X that occupied the same position in the previous period, and on another value which occurred somewhere between these two.

Writing it for our three successions, we will have:

a) $not_i = F(not_{i-N}, not_{i-k})$
b) $dur_i = F(dur_{i-N}, dur_{i-k})$

where the period *N*, will indicate the memory size of our musical tune (directly related with the autocorrelation window), and the value *k* will be related to the inertia of the system, so as *k* becomes smaller, our music will also be more influenced by the more recent events.

We could consider our polyphonic composer as various simultaneous monophonic voices generators of this kind, and then, for eight voices, we would have therefore a total of 24 relations (8 notes, 8 durations and 8 velocities relations). These would generate however eight simultaneous but independent voices, which is surely not the best musical environment we can imagine!

We can however make voices interdependent, by crossing these relations, which can be achieved by simply picking X_{i-k} from a different voice. We will have then:

$Xa,i = F(X_{a,i-N}, X_{b,i-k})$ where *a* and *b* indicate two different voices.

The actual value (which can be a pitch, a velocity or a duration) of the voice *a*, will consequently depend on the value it had on the same position on the previous period, and on another value, more or less recent, picked from another voice *b*. According to that, when starting from initial seeds of size *N* (recorded, loaded or randomly generated) of notes, velocities and durations values, each voice can keep generating and outputting new values, while moving in time their N-values correlation windows. We have created a multidimensional recurrent system with N-elements memory.

THE MAX IMPLEMENTATION

Following these principles, a first (and not very friendly) version of this program was developed by the author, on *Lattice C* for the *Commodore Amiga*. Our new version for the *Macintosh II*, written on the language *MAX*, the real time, object-oriented language created by Miller Puckette, makes use of all of its interactive facilities (4).

In the programs main window, the composer-conductor can change at any time with the mouse, the parameters *N* (period) and *k* (inertia) for any of the eight voices, in a range from 1 to 127 (maximum memory allowed). There is also an 8x8 matrix for modifying the dependencies between these voices. He can also select for every voice a different kind of relation *F*, choosing between a set of different functions.

NON-LINEAR FEEDBACK AND BITWISE LOGICAL OPERATORS

We have talked about relations of the kind: $z = F(x,y)$ (where recurrence appears by the fact that the output *z*, becomes a future input), but we have not so far discussed about the kind of functions *F* we have implemented.

The first function we used was the obvious linear relation, $z = (x+y)/2$ (notice the similarity with the Karplus-Strong algorithm, for plucked-string sounds synthesis). This function, which

acts like a digital filter with no inputs, makes very correlated music, but can only be used over a short time, as it ineluctably averages the values, ending in a constant note, with constant velocity and constant duration, and bringing therefore any composition to its entropic death *(5)*.

Focussing on non-linear relations (and given the program speed needs), we started studying functions constructed with the logical bit wise operators *(and, or, xor* and *one's complement)* available in the C programming language. These operators led us to an unexplored and surprising domain. Hardly predictable, logical bit wise operators may present very different behaviours; depending on the rates *N/k*, strange attractors appear, showing from near chaotic regimes to quasi-periodic behaviour.

Not any logical relation can be adequate. Some boundary properties have to be imposed, and in order to determine their organizing/disorganising qualities or their bandwidth, many different tools (information theory, transforms analysis...) shall be done to the resulting signals (i.e. notes or durations sequences). Statistical studies applied to the Cartesian product of the data domain (typically 0 to 127) are also interesting and clarifying. This field is barely scratched, and a lot of work is still to be done, but we have tried to order the -up to now- existing functions in a intuitive entropic (organizing/disorganizing) progression, leaving the identity function $(X_i=X_{i-N})$ in the middle.

COMBINING PITEL WITH EXTERNAL SIGNALS - IMPROVISING WITH HUMANS

Another important feature of the program, is the ability to interact or respond to incoming MIDI data. Any of the eight voices can be plugged to MIDI IN, and receive data from a specific MIDI channel (e.g. a player with a wind controller MIDI device, or an acoustic instrument with a pitch to MIDI converter). If any of the remaining voices are set to depend (via the 8x8 matrix) on these external ones , they will be influenced by what is being played live, establishing a dialogue between the computer and the human players.

MUSICAL FILTERS AND ENHANCEMENTS

The feedback mechanisms we have described are quite simple and possess no musical knowledge at all, but the compositions or improvisations *PITEL* generates, have however a surprising human flavour, and keep a good balance between foresight and surprise, as the listener can usually feel the periodicity of the cycles, but never anticipate their contents.

We have however implemented three additional MAX windows, with some filters, that bring a deeper control over the musical results. We should notice however that these filters only affect the data send to MIDI OUT, and not the data stored in memory, and used in future calculations.

a) **Time Control Window**: For each voice, durations can be quantized, averaged with a variable duration (according to an also variable weight), or even compressed or expanded to produce a staccato/ legato effect. Each voice can also have a silence ratio (O100), which statistically determines which notes will sound or not.

b) **Tone Control Window**: For each voice, pitches can be incremented/decremented by octaves, until they fu in a specific musical range.

Voices can also be transposed up to +/-24 halftones. A parameter we call *tonal freedom* ranging from 1 to 12, limits the number of correct notes on a voice, according to a given mode (if the freedom is 12, all notes will be correct, but if it was 1, only the tonic would be allowed).

c) **MIDI Control Window**: Any MIDI channel can be assigned to any voice. Each voice has also a MIDI volume slider, a pan dial, a mute/unmute button, and can send any program change number.

d) **Master Control Window**: From this window, the composer can start, pause or stop the program, and change the tempo, and the time signature.

CONCLUSION

The fact that *PITEL's* main algorithmic processes lack of musical terms or knowledge, gives it a great flexibility for composing in completely different musical styles. Given its real time properties, it can be considered as a musical instrument, and like those, it requires some training. Besides from live performances, it can also be used as a tool for studio composing, as the data generated can be edited in any MIDI sequencer.

Further developments shall necessarily include a deeper comprehension and control over logical bit wise feedback processes (which up to now remain quite wonderfully mysterious!).

REFERENCES

(1) Hiller, L., "Composing with Computers. A Progress Report", Computer Music Journal *5(4):* 7-21.
(2) Prigogine, 1. and Stengers, 1., "Order out of chaos", Bantam Books Inc., 1984.
(3) Beyls, P., "Musical morphologies from self-organizing systems", Interface, Vol. 19 (1990): 205-218.
(4) Puckette, M. and Zicarelli D., "MAX Development Package", Opcode Systems, 1990.
(5) Jordà, 5., "Slow Death", a composition for MIDI wind quintet.