音楽情報科学 23－10
（1997．12．16）

Scotch, twice: convincing human/machine improvisation

Harry Castle

University of California at San Diego
hcastle@man104nfs.ucsd.edu

Abstract

This paper discusses elements of the piece "Scotch, twice" in view of their impact on an improvisational realization of the work. The piece functions within a particular interactive framework that influences its outcome. The paper talks first about the hardware configuration of the system and the performance capabilities made available through software, then the classification of the system vis a vis other models for interactive systems, and finally concludes with a discussion of the emergence of an improvisational performance from within this framework.

"Scotch, twice" における人とマシンによる即興演奏の確信

ハリー・キャッスル

カリフォルニア大学サンディエゴ校
hcastle@man104nfs.ucsd.edu

この論文では、作品の即興的実現への影響という点から「Scotch, twice」という作品の要素について論じる。この作品はその結果に影響を及ぼす特別な相互作用の枠の中で機能している。ここでは、最初にシステムのハードウェアの輪郭とソフトウェアを通して可能になる演奏能力について語られ、次に他のインタラクティブ・システムのモデルとの相対的なシステムの類別、そして最後にこの枠組みの中からの即興演奏の可能性についての論議でまとめる。

Introduction

"Scotch, twice" is an improvised duet that functions within a particular interactive framework. Fundamentally, one performer sits at a disklavier piano and the other sits at a computer which is connected to the piano via MIDI. The performer at the keyboard improvises as a piano player, while the performer at the computer captures groups of MIDI note events from the piano player, reformulates them, and deploys them at will across the keyboard. This paper discusses elements inherent in this system in general and in the piece in specific both as they relate to a given performance, and as they act to influence a performance. I will first describe the performance capabilities made available by the software, then discuss the system in terms of the attributes that Rowe establishes for the classification of interactive systems [1], and conclude with a discussion of the strengths of this system as a foundational framework for improvisation.

Description of the Pengstrument Program.

The program running on the Amiga 3000, "Pengstrument," evolved from an earlier program that I wrote to facilitate experimentation with serialism and total-control techniques. The program is built upon a real-time scheduler that I wrote in the early 1980's which takes advantage of the Amiga's multi-tasking capabilities to allow for the asynchronous playing of concurrent event streams. In this particular piece all events are MIDI note events, but the scheduler could also be used to trigger sample playback, graphical events, etc. A single note stream, or stratum, is built out of four lists containing 1) MIDI note numbers, 2) MIDI velocity values, 3) note durations in milliseconds, and 4) time intervals between events. Values are selected sequentially from each of the lists, and the four values combined to describe a single note event. It is significant that these lists need not contain the same number of entries. If each list has the same number of entries, continuous playing will yield a sequence which repeats verbatim. If the lists are of different lenghts, however, the entries will pair up differently as each list restarts at different times, producing an iso-rhythmic effect that involves all four parameters. The following examples illustrate this:

1) note list: A, Eb, G
velocity list: f, ff, p
result: A(f), Eb(ff), G(p),
A(f), Eb(ff), G(p),
A(f), Eb(ff), G(p), etc.

2) note list: A, Eb, G
velocity list: f, p
result: A(f), Eb(p), G(f),
A(p) Eb(f), G(p),
A(f), Eb(p), G(f), etc.

Notice that in the first example the note and velocity (dynamic) lists each contain three items, so that when they are combined to form note events the same note is always paired with the same velocity value. In the second example the lists are of different lengths so the pairings are not always the same. the Eb for instance is at first p, then f, then p, etc.

For "Scotch, twice," the software employs a MIDI record feature which monitors the incoming MIDI stream from the piano and parses it into pitch, volume, duration, and time interval values. These values are then copied into their respective lists. The record feature also provides the flexibility to record values into only those lists chosen by the performer. Recording into all four lists simultaneously will allow for verbatim repetition at playback, i.e., all notes will be paired with their original velocity, duration and interval time. It is often desirable, however, to record subsequently only the pitch information into the pitch list, retaining the velocity and time information from a previous recording. This will result in the accumulation of lists of differing lengths and will add complexity and variation to the sequence when it is played back.

The above describes a single stratum within the Pengstrument program. There are 16 such strata, each of which may be recorded into or recalled for playback at any time, concurrent with whatever else is going on. The potential for building textures of layered complexity provides a powerful tool for the performer. Each of the strata may be distorted by transposition and by stretching or compressing independently the values in each of the four lists for any stratum. Stretching means multiplying all values by a number greater than 1.0, so, for instance, stretching the note duration by

2.0 would double the duration of each note while not altering the tempo. This might case the notes to begin to overlap producing a legato effect. As a final convenience, there are eight "storage bins" provided, each of which stores all of the current settings of all of the strata. The storage bins are also available at all times for immediate recall.

System Classification

"Scotch, twice" is realized through an interactive system. The hardware configuration in conjuction with the specific set of capablilities afforded by the software make up what I have been referring to as "the system." The system provides some capabilities/constraints, and additionally some agreements are arrived at between the two performers that then characterize a given performance as a piece. The physical relationships within the system are fairly simple: a disklavier piano is connected to a computer via MIDI. Both performers are capable of playing the piano (one from the keyboard and the other from the computer), and the computer may record anything that is played at the piano by either performer. The general attributes of the system are probably best understood in terms of Rowe's classifications of interactive systems, i.e., 1) score-driven vs. performance-driven, 2) transformative, generative, or sequenced, and 3) acting according to an instrument or a player paradigm[1]. As Rowe emphasizes, these are not either/or categories but rather classifications with attributes that may appear concurrently and in varying amounts.

Classification under the first category is fairly straightforward, as the system here is clearly best described as performance-driven. There are no "predetermined event collections" [1], no score to be anticipated or realized. Unlike some performance-driven systems, however, there is almost no analysis performed on the incoming MIDI stream. The MIDI information is parsed into constituent components for subsequent transformation and playback, but is in no way categorized in terms of its broader perceptual aspects such as phrase lengths, density, etc. All such determinations are left to the computer performer and are therefore made in real-time within the context of the piece as it exists at that moment.

Regarding the second category, response methods, this system has elements of all three of the attributes listed above. The system is transformative in that it allows the computer performer to apply a variety of transformations to the recorded material either before or as it is being reintroduced. The nature and degree of the transformations are left to the discretion of the performer. Small transformations (simple transposition or dampening of intensity) may leave the original character of the fragment relatively unchanged, while larger distortions may modify it beyond recognition. As distinct from some other transformative methods, however, this system does not apply algorithmic transformations to the input, and is not capable of applying an ongoing transformation to the MIDI stream as it moves directly from the input to the output.

There is a generative side to the software as well, in that it generates note events from the raw materials that have been accumulated. As outlined in the discussion of the software, the rules used to construct a melodic line are not at all complex and in fact even allow for verbatim playback of recorded material. There is nothing in the software approaching artificial intelligence or heuristic methods. Nevertheless, when a stratum has been constructed which has, say, note and duration rows of different lengths, the note stream produced can be complex and difficult or impossible to anticipate as a direct result of the generative behavior of the software.

If sequenced techniques are understood to mean initiating playback (with perhaps some modification) of a stored sequence in respose to real-time input, then this system uses little, strictly

speaking, in the way of sequenced response methods. Generally speaking, however, the constituent components of the material are sequenced and may be started on command. The performer can therefore at times initiate a known sequence in response to real-time input. This can result in situations where, for instance, the pianist and computer performer spontaneously construct phrases together, perhaps with the pianist playing some variant of a familiar gesture and the computer performer recognizing it and responding by invoking a stratum which adds some kind of a tail to each phrase.

Improvisation within the system.

The preceeding discussions reveal aspects of the software and of the overall system that bear directly on the results of a given performance. The software is clearly not designed to solve any problems for the performers, it merely changes the angle from which some problems may be attacked. The facility inherent in the system does, however, encourage certain kinds of thinking and in that way will influence the results. Specifically: there are unpredictable qualities built into the software, there is an overall bias towards thinking about the music gesturally, and it is necessary to be constantly shifting ones focus between high and low levels of detail.

Pengstrument was built to function as a tool to produce unanticipated richness and as such there are degrees of unpredictability (though usually without random number generators) at every level. Starting from the bottom, a simple multiplication will produce results that are unpredictable at the time of performance. For instance, a sequence of MIDI note values when compressed (multiplied by a number less than 1.0), will also be transposed down. As an extreme example take the note values 20, 30, 40. If compressed by a factor of 0.1, the resulting list of notes is 2, 3, 4. The interval between each note has gone from a minor 7th (10 half steps) to a semitone, but the entire sequence has also been transposed down dramatically. This can be corrected for by changing the transposition value for the stratum, but as Pengstrument does not display actual note values on the screen the best you can do is approximate.

The sequence of note events that Pengstrument generates becomes increasingly complex as one varies the relative lengths of the lists within a single stratum. This, combined with the ability to concurrently play multiple strata asynchronously, provides a great deal of power to generate music of lush complexity. Interestingly enough, this complexity, while not entirely predictible, nevertheless has a character all its own. Even an individual stratum, significantly transformed, will have identifiable and memorable rhythmic and melodic attributes. The performers attention then is directed towards remembering, manipulating, combining, and reintroducing some or all of the accumulated strata in a performance.

A natural outgrowth of the above 'constraints' is that both players are encouraged toward gestural thinking and global structural considerations. The computer performer can apply general transformation settings to a stratum without hearing it, but must eventually turn it on to know exactly what is there. As an example, after capturing a sequence of notes played in an upper register spanning, say, an octave. the performer may decide to compress the notes to within an interval of roughly a fifth, dampen their velocities, stretch the notes out in time by a factor of 10 (slowing them down), and transpose them down into a low register before reintroducing them for playback. The result would be to produce long, soft sustained bass notes in support of the upper registral activity. It is not possible to know in advance exactly what the pitches will be, but the computer performer can proceed to micro-manage the transposition or note compression in an attempt to get specific pitches. In any event, all subsequent manipulations must be done after the

material has been brought out into the open. The significance being that all grooves and patterns that are discovered happen within the context of the duet -- both players are making adjustments at the same time so that anything arrived at is the product of mutual effort. Both players have a memory of it and a hands-on association to it that will inform their reactions to it and similiar materials as the piece evolves.

Improvisation within the piece.

As stated earlier, the hardware and software configuration constitute the improvisational framework, the system. Additional constraints or agreements external to the system serve to characterize a piece, although as an improvisation that piece may have many and varied realizations. There are only two such decisions that characterize "Scotch, twice," but they are rather important. Both share the quality that they are not rules that dictate how or when the players are to play, but are rather additions to the system that futher shape the performance environment.

The first and most obvious is that only a single piano is used. If one performer is playing a note and the other attempts to play the same note, nothing will happen. They are sharing the same 88-note pitch set and that is all they have to work with. Even more significantly, all sound emanates from a single acoustic source, fusing the contributions of the two performers right there at the soundboard. The result is a built-in bias towards the blending of two voices into one.

The other, less obvious choice, is that of allowing the computer to use only material introduced within the current performance. This is an artificial constraint as the software has the ability to save and recall accumulated data from one performance to the next making it possible for the computer performer to begin playing immediately drawing on whatever material had been loaded from disk before the performance. By starting out with all strata empty, the onus is on the pianist to introduce the first musical ideas. It is usually not long, however, before the computer performer is quite well armed. During the development of a piece, both performers are working with similar materials and are free to identify and amplify on anything that they choseto treat as thematic material. This arrangement is biased toward an inescapable degree of thematic cohesion. The stylistic nature of the materials are not suggested by the piece in any way, and are wholly dependent upon the predilictions of the performers, i.e., what they like to play and what they like to hear. The players' improvisational sensibilities and cooperative treatment of texture, density, color, etc., become, appropriately, the central focus of a performance.

---

[1] Rowe, Robert, "Interactive Music Systems: Machine Listening and Composing", The MIT Press, Cambridge, Massachusetts, pp 6-8.