ON CONVINCING HUMAN/MACHINE IMPROVISATION

Harry Castle
University of California at San Diego
hcastle@ucsd.edu

Scott Walton
University of California at San Diego
swalton@ucsd.edu

Abstract

This paper describes an interactive system for two performers, collaboratively developed and demonstrated by the authors. 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. Custom software allows the computer performer to selectively record, reformulate, and play back material introduced by the pianist. This paper discusses the hardware and software that make up the system, and the compositional and performative strategies inherent in the design that influence the outcome of a particular performance.

I. Introduction

Our goal in this collaborative venture was to create an interactive hardware/software environment within which we could develop improvisationally-based compositions. Our first step was to conceive a list of priorities for the environment which would provide us with a set of unique challenges and advantages. The two most important decisions were to not use synthesized sound: all musical activity would be in the domain of the acoustic piano; and to utilize only one disklavier, necessitating the joint negotiation of the same physical and aural space.

It was important that the computer performer have the ability to generate the same sense of spontaneity and immediacy of expression that is assumed with an accomplished improvising instrumentalist. This was achieved through the use of Pengstrument, a reconfigurable custom software program written by Castle that allows for, among other things, the real-time recording, reformulation, and playback of MIDI note streams. The skeleton of the system was thus established: a disklavier piano connected via MIDI to a computer running Pengstrument. Following is a discussion of the performative capabilities of the Pengstrument software, followed by a discussion of the compositional decisions we have made that influence the outcome of any given performance.

II. Description of the Pengstrument Program

Pengstrument, originally written to run on the Commodore Amiga, evolved from an earlier program written to facilitate experimentation with serialism and total-control techniques. The program runs on top of a real-time scheduler written by Castle that takes advantage of 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. Within Pengstrument, a single note stream, or stratum, is built out of four circular queues containing respectively: MIDI note numbers, MIDI velocity values, note durations in milliseconds, and time intervals between events, also in milliseconds. Values are selected sequentially from each of the lists, and the four values are combined to describe a single note event. It is significant that these lists need not contain the same number of entries. If each list does have the same number of entries, continuous playing will yield a sequence which repeats verbatim. If the lists are of different lengths, however, the entries will pair up differently as each list recycles, 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), Verbatim Playback
A(f), Eb(ff), G(p), etc.

2) Note list: A, Eb, G
Velocity list: f, p
Result: A(f), Eb(p), G(f), Varied Playback
A(p), Eb(f), G(p), 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.

Pengstrument is endowed with 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, perhaps producing a legato effect. As a final convenience, there are eight "storage bins" provided, each of which stores all of the current settings for all of the strata. The storage bins are also available at all times for immediate recall.

III. Inside the Musical Environment

The musical environment we have constructed is best described as performance-driven (Rowe, 1993). There are no "predetermined event collections" (Rowe, 1993) and 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 length, 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 any given moment.

Using Rowe's terminology, the response methods within the environment would be characterized as transformative. The computer performer is able to apply a variety of transformations to the recorded material either before or as it is being reintroduced, and 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 travels from the input to the output.

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

As Lewis says, "notions about the nature and function of music are embedded right into the structure of music software" (Lewis, 1997). Software is central to our setup, and assumptions regarding what is musically valuable to us are reflected in the musical environment we have constructed. When a stratum has been accumulated which has, say, note and duration rows of different lengths, the note stream produced can be complex and difficult or impossible to anticipate. It is, however, possible to anticipate generally what will be produced, and the computer performer can micro-manage a stratum's parameters once it is out in the open. As a result, both players are encouraged toward gestural thinking and global structural considerations, and all grooves and patterns are discovered within the context of the duet: both players simultaneously adjust to events as the piece evolves, arriving at a product of mutual effort.

The inherent malleability of interactive software environments is sometimes viewed by composers as something to be held under control, and a tight leash is kept on the amount of improvisational choice allowed to a performer. Improvisers, however, tend to prefer non-hierarchical environments in which individuals can work together without the control structures that supposedly ensure successful music-making. This encourages a multiplicity of viewpoints and a relative autonomy for each of the participants. In our case the individual hardware and software components are relatively simple, making pre-eminent the improvisational skills of the performers. By distributing musical intelligence among performers and machines one can allow music-making to remain the "imperfect and social process that it is" (Trayle 1991).

IV. Improvisation Within the Environment

As stated earlier, the hardware and software configuration constitute the improvisational framework or environment. Additional constraints or agreements external to the environment may serve to characterize a piece, although as an improvisation that piece may have many and varied realizations. To date, we have been performing essentially one such piece, "duosolo," which is described by two such constraints. Both share the quality that they are not rules that dictate how or when the players are to play, but are rather additions that further define 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. Perhaps even more significantly, all sounds emanate from a single acoustic source, fusing the contributions of the two performers at the sound board of the piano. 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 by 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 choose to treat as thematic material. This arrangement is biased toward an inescapable degree of thematic cohesion. Nevertheless, the stylistic nature of the materials are not suggested by the piece in any way, and are wholly dependent upon the predilections 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.
V. Summary

The preceding discussion reveals aspects of the software and of the overall environment that have a direct bearing on the development of a given performance. The software is clearly not designed to solve any problems for the performers, but rather to establish a framework for their interaction. The environment was designed specifically for improvisation, and as such has attributes that may or may not be useful in achieving other musical goals. There is a measure of indeterminacy built-in to the setup, and each performer has different tools with which to respond to the unexpected. The overall arrangement encourages gesture development, global form development, and thematic cohesion throughout a given performance. To date we have only begun to explore the possibilities inherent in this system and are looking forward to experimenting with other extra-environmental constraints and whatever else the future may bring.

----------------------------------------