THE COMPUTER AS ACCOMPANIST

PANELISTS:

William Buxton (Moderator)
Computer Systems Research Institute
University of Toronto
Toronto,Ontario
Canada M5S 1A4

Roger Dannenberg
Computer Science Department
4212 Wean Hall
Carnegie Mellon University
Pittsburgh Pennsylvania
USA 15213

Barry Vercoe, Director
Experimental Music Studio
E15-483
MIT
Cambridge MA
USA 02139

INTRODUCTION (William Buxton)

One of the most interesting developments in computer music in the past year has been the introduction of a new class of program, the computer accompanist. These programs do just what their name suggests: they play the accompanyment to a solo played by a live performer. On the surface, such programs appear to be no more than yet another version of "music-minus-one" records. What is different, and of intense interest from the perspective of human-computer interaction, is that this class of programs not only listens to the soloist, but exercises some degree of musical intelligence in making decisions about their performance. The programs behave like a human accompanist in that they are governed by the tempo and dynamics of the soloist. And, like a good accompanist, they do their best to adapt in a musical fashion when the soloist makes mistakes, plays extra notes, or leaves notes out.

This panel presents the work of the two people who pioneered this field, Barry Vercoe and Roger Dannenberg. Vercoe's presentation will takes the form of a video-taped demonstration, a brief talk, and some musical examples. Dannenberg will supplement his verbal presentation with a live performance/demonstration.

Outside of novelty value, why is this work of interest to the CHI community? I believe that it forces us to reconsider our preconceptions of what constitutes a user interface. These systems use no keyboard or CRT in their primary mode of operation. The computer "listens", "understands", and responds sonically. The systems introduce uncommon technologies in the signal processing modules employed. These perform real-time event detection (with pitch and amplitude analysis) on the acoustic signal generated by the soloist. And, in their handling of mistakes by the soloist, the systems also provide a dramatic demonstration of adaptive knowledge-based programming.

For more information about the implementation of computer accompanists, see Dannenberg (1984), Vercoe (1984), Bloch (1985), Lifton (1985), and Vercoe (1985). For more information on computer music in general, see Roads and Strawn (1985) and Abbott (1985). The latter is a special issue of *ACM Computing Surveys* devoted to computer music. In that issue, the article by Pennycook (1985) specifically addresses user interface issues. Finally, there are two main sources for current information on computer music. One is the *Computer Music Journal*, editted by Curtis Roads and published quarterly by the MIT Press. The other is the proceedings of the annual International Computer Music Conference (ICMC). These proceedings (and a quarterly newsletter) are available from the *Computer Music Association*, P.O. Box 1634, San Francisco, CA, 94101-1634, USA.

CONTRIBUTION: Barry Vercoe

THE SYNTHETIC PERFORMER IN REHEARSAL:
Teaching a Computer to Play by Ear

The most effective way to teach young musicians is to have them learn by doing. Although the underlying framework of music can be clarified by lessons in music theory, a real sense of what music is about is built up only through the experience of live performance. Exactly what is communicated, and how humans learn from this experience, is not much understood, but when it works it is clear that an important transfer of information and procedural knowledge has taken place.

Teaching a computer to play requires a similar transfer of knowledge. While the fundamental rules governing pitch,

rhythm, tempo and loudness are easily communicated, the subtle variances that characterize a good performance can only be hinted at from without. Low-level details of nuance are normally the consequence of high level controls suggested by a conductor, a teacher, or a concerto soloist. A computer entering into such skilled collaborative activity must be able to deal with the semantics of real-time performance, both as input and output.

During the last two years, in work done at MIT (Boston) and IRCAM (Paris), I have developed a convincing real-time model of music understanding and response. The system has three major parts:

1. Perception and cognition of musical audio stimuli.

2. Organizing a performance response.

3. Learning from the experience.

Perception involves two main activities: pitch/envelope detection, and score matching. Pitch detection methods have varied with circumstance. In tracking live flute we used a combination of audio signal and fingering information (from optical sensors on the keys), obtaining accurate pitch in about 35 milliseconds. In violin tracking we used acoustic information only, but the richer results took longer to settle. In both cases the event sequence was matched onto a score of expected events. Differences between expectation and perception are of two kinds: those due to tempo changes (slow changing but strongly indicative of the future), and those of stylistic affectation (rubato, and other time shifts that have only local yet highly significant expressive import).

Performance response (e.g. a piano accompaniment) is difficult to do right. It cannot be event driven, or it will always lag and never be able to match or lead. Acceptable response appears to require a close model of the physiological processes involved in actual human performance, including score look-ahead, gradual focus on a forthcoming event, then an anticipatory action decision. The action decision window (to mobilize the motor actions for the event) is roughly one-tenth of a second before real sound -- much earlier than this and the response to change is sluggish, much later and the event may not happen on time. This appears to be the tolerance permitted skilled, collaborative chamber music ensembles when they create an aurally interesting performance. The same tolerance allowed of a computer performer provides an important amount of control computation time. In my model, the neural and physiological processes that are invoked to produce a single, sensitively performed note are modelled by suddenly spawning 10-20 active object modules, each responsible for some facet of the event, and all competing cooperatively during this 100 ms window for limited CPU resources. This has enabled a combined Listen/Perform model to present adequate public performances of such literature as a Handel flute and harpsichord sonata and a Brahms sonata for violin and piano. In these performances the soloist is live, and the accompanist entirely synthetic.

Despite the success of the Listen/Perform model, it becomes evident that the micro-structure of expressive time-shifts is not easily parsed and responded to on a single hearing. That is the stuff of which rehearsals are made, and without knowledge of past performances our model is essentially that of *sight reading on the concert stage*. An important advance has been made in modeling the *learning* process of music rehearsal. The technique involves keeping a record of everything that is heard, and making inferences over time about what to expect and how to deal with it. Expressive time-shifts tend to vary from one performance to the next, some shifts being more consistent than others. As hinted above, ensemble togetherness requires that slow moving tempo changes and faster moving expressivity be separable. With hindsight they are, and the purpose of rehearsal is to bring this benefit into the present. The computational strategy is to use rehearsals to develop a record of each time-shift, its average size, and variance. This creates a modified score of what to expect, and a sense of how much one should infer from departures. During subsequent performance runs, all interpretive decisions are based on this extracted knowledge. This technique has led to quantitative improvements in the performance and synchronization of musical expressiveness.

The Synthetic Performer in Rehearsal provides an important model for human interactive systems. First, it shows that the relation between a skilled practitioner and a computer assistant involves an important layer of procedural knowledge that can be acquired only from practice. Any composer would find it impossible to imbue his data base (musical score) with that kind of detail. Secondly, it is very apparent that the best channel for communicating data is that in which in naturally occurs. In musical applications, traditional graphic and gestural interfaces are of limited usefulness. The true medium is sound.

The lesson here is a simple one. As we instinctively know with a young child: if you want your computer to be musical, just keep singing to it.

This work was supported by an award from the Guggenheim Foundation.

CONTRIBUTION: Roger Dannenberg

Introduction

Music provides a rich domain for the study of user interfaces, real-time control, and multi-media interactive computer systems. Just as pointing devices and graphics displays have led to new interface paradigms, so have real-time computer music systems. These systems deal with sound as the most important mode of output, and piano-like keyboards or musical instrument interfaces are common input devices. Both input and output are therefore unorthodox by computing standards, and input and output are inherently real-time.

Music is also interesting from a computer interface standpoint because it contains a variety of abstract concepts like tempo and melody, and interesting tasks including editing and accompaniment. These concepts and tasks are "real" in the sense that there is a wealth of tradition and knowledge against which to relate our work.

Computer Accompaniment

Computer Accompaniment is a good example of taking the user interface beyond the desktop paradigm. The task is similar to that of a human accompanist, who listens to another performer, reads music, and performs another part of the music in a synchronized fashion. The score contains a description of the music to be played by each performer. In my model, the *soloist* considers only his, her or its part of the score and determines the tempo of the performance. The *accompanist* dynamically adjusts its timing to match that of the soloist.

The *accompanist* can be described (and implemented) as a collection of concurrent tasks. The first task, called the *listener* is a preprocessor of input from the soloist. Listing in this context means converting sound into a symbolic form to be used by the next task. A single melodic line from, say, a trumpet or flute is

processed in real time to obtain time-varying pitch information, which is then quantized to obtain the discrete pitches of a musical scale. Alternatively, a music keyboard whose output is inherently symbolic can be used. In either case, the listener task sends a schematic representation of the soloist's performance to the next task.

This second task, called the *matcher*, compares the actual performance to the expected performance as indicated in the score. The objective of the comparison is to find a correspondence between the performance and the score, thereby relating real time to the timing indications in the score. Since the soloist or the listener task can make mistakes, the matcher must be tolerant of missing notes, extra notes, or notes whose pitch is wrong. Furthermore, the timing of notes will vary from one performance to the next. To deal with this kind of "fuzzy" match, a real-time adaptation of dynamic programming is used. The output of the matcher is a sequence of reports that occur whenever the matcher is fairly certain that a note performed by the soloist corresponds to a particular note in the score.

The third task, called the *accompanist* controls the timing of the accompaniment. Note that the content of the accompaniment is determined by the score, so timing is the only dimension that varies from one performance to the next. Typically the accompanist will output commands that mean something like "the violin should now begin playing C-sharp", and a synthesizer handles the actual sound generation. The main problem in this task is to adjust the timing of the accompanist in a musical fashion.

In our implementations, a "virtual clock" is used to schedule accompaniment events. The clock speed and offset relative to real time is adjusted according to information from the matcher task. Adjustments must be made carefully if they are to sound musical, and we have taken a rule-based approach to programming this task. For example, one rule says that if the virtual clock is behind the soloist by a moderate amount, it is better to catch up by playing very fast than by skipping part of the accompaniment.

Implications

This work has several implications for research in user interfaces. The first is that interfaces can move beyond the command/response model to one in which computers and humans actively participate in a cooperative task. The real-time nature of accompaniment demands this level of participation by the computer, but one can imagine other areas where interfaces could be improved by making them more active. Help systems, computer-assisted instruction, and computer-aided design applications come to mind.

The second implication is that in a very high-level interface, it is possible for the user to become less direct in his or her interaction with the computer. The human input side of the interface is maintained (or even enhanced) by having the computer *monitor* human activity and perform actions with no explicit request. In this way, the computer becomes less a slave and more an assistant.

Finally, this work illustrates that sound and music can be important elements in a user interface. Even in non-musical applications, sound provides a high-bandwidth communication channel that is seldom utilized. Modern music synthesizers are inexpensive, easy to interface to computers, and open many avenues of research.

Computer accompaniment is protected by U. S. patent pending.

Acknowledgements

I would like to thank Bill Buxton for encouraging me to look at accompaniment from a user-interface perspective and also for working with me, Josh Bloch, and Cherry Lane Technologies to produce the first accompaniment system I am proud to demonstrate. I would also like to thank the Computer Science Department and the Center for Art and Technology at Carnegie-Mellon University for supporting this work.

REFERENCES

Abbott, C. (Ed.) (1985). *ACM Computing Surveys*, **17(2)**, Special Issue on Computer Music.

Bloch, J. & Dannenberg, R. (1985). Real-Time Computer Accompanyment of Keyboard Performances, *Proceedings of the International Computer Music Conference*, Vancouver, August 1985, 279 - 289.

Dannenberg, R. (1984). An On-Line Algorithm for Real-Time Accompanymnet, *Proceedings of the International Computer Music Conference*, Paris, October 1984, 193 - 198.

Lifton, J. (1985), Some Technical and Aesthetic Considerations in Software for Live Interactive Performance, *Proceedings of the International Computer Music Conference*, Vancouver, August 1985, 303 - 306.

Pennycook, B. (1985), Computer-Music Interfaces: A Survey, in Abbott, C. (Ed.) (1985) *ACM Computing Surveys*, **17(2)**, 267 - 289.

Roads, C. & Strawn, J. (Eds.)(1985). *Foundations of Computer Music*. Cambridge: MIT Press.

Vercoe, B. (1984), The Synthetic Performer in the Context of Live Performance, *Proceedings of the International Computer Music Conference*, Paris, October 1984, 199 - 200.

Vercoe, B. & Puckette, M. (1985), Synthetic Rehearsal: Training the Synthetic Performer, *Proceedings of the International Computer Music Conference*, Vancouver, August 1985, 275 - 278.