David Zicarelli
Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music
Stanford University
Stanford, California 94305, USA
ddz@CCRMA.Standord.edu

Communicating with Meaningless Numbers

Because this symposium is being held in honor of John Pierce, I thought it would be appropriate to discuss a topic related to human-computer communication in computer music, as he has made so many major scientific contributions to the field of communication. The computer's role in music has traditionally been to produce complex output from descriptions which are simplified in some way. While the instrument (in a Music-*N* sense) might be a rather complex computer program, the score roughly approximates the amount of detail a traditional composer might specify to an orchestra. In recent years, computers have become available to a far larger audience which also includes performers, and of course the communication between a performer and his or her instrument is of a completely different character than the communication a composer sets down in a score.

In performer-instrument communication, the instrument is not reminded of what to do by the low-bandwidth channel of a few terse marks on a page; rather, the performer is continuously engaged in control of the instrument. The detail and complexity of this control is such that it is never completely articulated, hence the use of marks on a page. A common method of controlling so-called real-time synthesizers has been to trigger sounds from a keyboard. Keyboard technique allows a certain degree of accent, phrasing, and articulation; and while many listeners can tell the difference between a real violin and a violin simulation being triggered and released by a keyboard, the important difference is not in sound but in the performer's motivation. The central issue is that it takes more effort to learn how to make a good sound on a violin than it does to trigger the start of a recording or simulation of a violin—as described by Michel Waisvisz (see Krefeld 1990). Effort is expended in the development of a relationship with an instrument over the course of time, which results in a substantial amount of complexity under fluent management by a performer. While a skill situated in one's nervous and motor systems can be referred to by marks, it can never be totally articulated, certainly not by a composer whose involvement in the musical process is to specify an arrangement of these marks.

Computer Music Journal, Vol. 15, No. 4, Winter 1991,
© 1991 Massachusetts Institute of Technology.

With the development of controllers such as the Mathews-Boie Radio Drum (Mathews, Boie, and Schloss 1989), we now have ways to measure some of the gestures people make when they play traditional instruments. There has, however, been little discussion of the computational architecture and resources necessary to support a situation in which the audio output might actually represent a reduction in the amount of data transmitted when compared with the gestural input.

Suppose we have 16 continuous channels of control data occurring at approximately 1000 numbers per second. If we are to use these data in some musically clever way, we need to be able to perform a computation on each channel (which might range from detecting the onset of a "note" to updating a parameter in a synthesis algorithm) in less than 160 μsec. This figure assumes (unrealistically) that the CPU can spend all of its time watching and processing only this set of control data. What appears to be the case with current technology is that most of this 160-μsec interval is spent just fetching the data, either across a slow bus, or addressing a serial chip using an interrupt polling scheme. The ideal situation would be that the CPU could access these control data as easily as reading an address in memory. This is typically accomplished by *direct memory access* (DMA) operations, but such operations are typically rather unintelligent for musical needs. DMA was originally intended to service disk drive transfers, where large blocks of data are moved into or out of memory at once.

The situation with gestural controllers is a bit different. We would, for example, often like to know the last few values of a gesture, maintained in a kind of circular queue. Although I am not an expert in such matters, it would seem that an architecture of multiple processors sharing memory (where one processor handled the fetching and formatting of incoming control signals for the other) is the most flexible means for handling many simultaneous channels of information. We are used to thinking about shared memory for synchronous environments such as digital signal processors, but there is no reason why continuously sampled control signals cannot be considered for a synchronous environment as well.

One way of handling gestures is to preprocess the incoming data, looking for features and events that are determined to be relevant, then communicating a reduced amount of information to the host computer. While this may work when we have a thorough understanding of the relationship between gesture, composition, and synthesis parameters, we currently have only the sketchiest understanding of how continuous gestures could be used effectively in an artistic sense. Indeed, we don't really have a complete understanding of how continuous gestures control acoustic instruments.

Typically, these data reduction processors are microcontrollers that need to be programmed in assembly language, making the user feel that the gestural data reduction scheme is a black box, and discouraging any experimentation with the basic signal to discover how it might be employed to make the total instrument more responsive. How a gesture is interpreted in software, as for example how the computer determines when a note has been played and something should be started or triggered, is extremely important in determining an instrument's feel, as much (I would claim) as the kind of synthesis used to make a sound. Giving a musician access to the complete gesture is a way to open up the instrument so that it can evolve along with the musician's technique and compositional ideas.

As an illustration of why gesture analysis is best considered an element of the overall control exercised by the CPU, it should be noted that Max Mathews didn't devise his notion of the reset plane and trigger plane for extracting hits from the radio drum until he bypassed the microcontroller that turned the drum's signals into serial data and digitized the analog signals directly on his PC host computer (Mathews, Boie, and Schloss 1989). He then used the analog-to-digital converter card's DMA capability to transfer the signals directly into the PC's memory, where he had complete control over how they should be interpreted. Given the direct access to the continuous gesture, and a scheme fast enough to use the data for rudimentary analysis, other people could invent their own schemes for extracting features such as hits from the drum. Without in any way slighting Mathews's reset and trigger plane algorithm, there are probably at least 10 different ways of interpreting hits on the radio drum that are yet to be discovered, all of which will serve the needs of different performers.

Another characteristic of current implementation of communicating gestures, particularly of MIDI, is the way in which meaning is attached to a communication channel. For instance, MIDI has note-on messages, and the typical temptation is to treat these messages as though they were limited to the specification of notes rather than as magnitudes coming from an abstract set of impact-sensitive switches. When using multiple parallel channels to transmit gestures, each channel would represent a single aspect of musical gesture. Specifying what happens on a polyphonic keyboard is rather naturally expressed as a discrete event, but there is a big difference between an incoming single event and the complex possibilities and contingencies of starting, controlling, and specifying a sound in a digital synthesizer. Environments such as MAX (Puckette 1988), which allow the flexible reassignment of the individual bytes of a MIDI message, allow us to see past the supposed meaning of note number as meaning pitch, or velocity as meaning how hard the note is to be played. When numbers lose their meaning, their potential control applications increase dramatically. Streams of numbers are the universal language of control, so I feel it's important that we strive to keep our communication channels as meaningless as possible and send numbers with little or no surrounding context.

F. R. Moore (1987) points out how easily multiple continuous controllers needed to represent the playing of a violin will overload a single MIDI cable. A scheme in which each gestural aspect had its own channel means that the incoming signal has no semantic baggage associated with it. This is at best three times more efficient than MIDI (which requires that each message be distinguished by type and channel), and has the added benefit that computer users will tend to avoid associating the signal's effect with its source.

We need to do all we can to encourage our computer systems to be as open to individual whim and interpretation as possible. The computer, by its very nature of repeatability, often hides the assumptions of software and hardware architecture that drive us to certain kinds of aesthetic statements. The very process of filtering complexity for musicians often causes additional perceived complexity to arise, after the musician realizes that he or she needs to take apart the conceptual filter in order to understand what is really going on. The transmission of musical tradition through notation and musical instruments themselves should be our conceptual model for the proper way to deal with complexity.

We should provide computer-based instruments which are as rich in opportunity, customization, and reward as traditional instruments, and, if they are good enough for musicians to care to use them for a sufficient amount of time, the complexity will disappear in much the same way that the complexity of the musical score disappears for those who are familiar with its notation, or the complexity of speaking a language disappears for those who are fluent in it.

By preserving as much detail about musical gestures as possible, and building computer systems that can respond to this detail completely and openly, we allow for the possibility of the control of musical complexity by a corresponding complexity in the gestures the musician is able to produce. We tend to believe that computers should eliminate having to learn skills as wonderful and complex as playing a musical instrument, that we should be able to replace unintelligent manual tasks with intellectual mental tasks. I think this will turn out to be a fundamentally misguided notion. Instead, we should recognize the eloquence of the musical performer, and reestablish a balance between the body and the intellect in the design of our computer systems.

Discussion

Max Mathews: I certainly concur with your thoughts that it's useful to generalize about number streams that come from these various sensors, and I think the way you think about them is very similar to the way I like to think about them. The thing that kept me from using MIDI for most of my work is the range of the numbers used as values 0 to 127, and in pitch sense that just isn't enough. The issue you raised at the end of your talk, the physical reaction of the sensor back on the finger or whatever part of the body is actually the sensor, is one of the most potentially interesting dreams that at the moment, I think, is almost completely unrealized; and I think there should be a lot of work there in the next decade leading to some very nice instruments. I think the only person I know who's really seriously worked on that very much is Claude Cadoz and his group over in France. [See their article in *Computer Music Journal* 14(2)—Ed.]
David Zicarelli: Even if you have 256 numbers for pitch someone isn't going to be satisfied. How about thinking of pitch this way—if two people talk to each other and they both understand what the other person means by pitch then we have pitch. So if two devices could talk to each other and they both agree, then the number that you throw at one device makes its pitch something that you can predictably relate to P if they tell you what the numbers are for P. I don't see why it matters how many bits you use. If all we did was decide that if we only had seven bits then they mean X, but if you had another range it means Y, then we don't need to invent a standard for how many bits there are to represent the pitch.
Miller Puckette: At IRCAM we are representing pitch as MIDI key numbers but using floating-point

numbers, so that the value 60 corresponds to middle C and 60.5 is a quarter tone up from that; so far that's fine. People understand it.
David Wessel: The implication of what you've said for the designing or implementation of future communications protocols is that we use neutral ones. I'm hearing neutrality, and I'm hearing perhaps something that could easily take advantage of rather standard hardware in terms of either a parallel or multiplexed connection. You don't care if it's multiplexed, it's just the channel concept that has to be there. So, in the music industry then, we need to be encouraging people to think about rather inexpensive, high-speed, but neutral data transmission schemes. I'm just reformulating what I'm hearing, and I think I agree with that very strongly.
David Zicarelli: It's slightly more difficult to make a device speak MIDI than it is to speak a continuous quantity of some kind that gets transmitted at a synchronous rate. The real jump in implementing this kind of scheme will be if someone makes a chip that does continuous synchronous data transmission. Then everyone can put it in their particular widget and they have a simple and reasonable interface to the analog world or the computer world.

References

Krefeld, V. 1990. "The Hand in the Web: An Interview with Michel Waisvisz." *Computer Music Journal* 14 (2):28–33.
Mathews, M., R. Boie, and A. Schloss. 1989. "The Radio Drum as a Synthesizer Controller." In *Proceedings of the International Computer Music Conference.* San Francisco: International Computer Music Association.
Moore, F. R. 1987. "The Dysfunctions of MIDI." In *Proceedings of the International Computer Music Conference.* San Francisco: International Computer Music Association.
Puckette, M. 1988. "The Patcher." In *Proceedings of the International Computer Music Conference.* San Francisco: International Computer Music Association.