Roger B. Dannenberg
Carnegie-Mellon University
Pittsburg, Pennsylvania, USA
RBD@cs.cmu.edu

A Perspective on Computer Music

It is a pleasure to be part of this 20th anniversary celebration of *Computer Music Journal*. The *Journal* has great significance to me and to many readers because it was our first introduction to the field of computer music. I saw a copy of Volume 1, No. 1 at an activities fair at Rice University where I was an undergraduate. I wrote for a subscription and back issues, and received a handwritten note from none other than founding editor, John Snell, explaining that my subscription would start with No. 1, as No. 2 was not yet in print.

This was an exciting time. Microprocessors were just becoming available and digital synthesizers were beginning to appear, although nothing had reached the consumer market. Julius Smith was also at Rice, and we played in a band together. As a freshman, I met a graduate student who had invented "Karplus-Strong" plucked-string synthesis years before the well-known publication (Karplus and Strong 1983). He used a Digital Equipment Corp. PDP-11 minicomputer attached to a vector graphics display. The *x* and *y* deflection signals were diverted to a stereo amplifier to form a makeshift computer music system. The grad student had it playing two-part inventions in real time with very nice harpsichord-like timbres. This was perhaps the first "physical model" synthesis system.

Computer Music Journal played a critical role in disseminating ideas from research, in fighting the isolation that comes from working in a small interdisciplinary area, and in defining the emerging field of computer music. While many of the early practitioners received their introduction to computer music through courses and residencies, many of us "learned the ropes" from the *Journal*'s pages. *Computer Music Journal* is still a major resource in the field, even with the advent of new journals and conference proceedings.

Computer Music Journal, 20:1, pp. 52–56 Spring 1996

Where Were We?

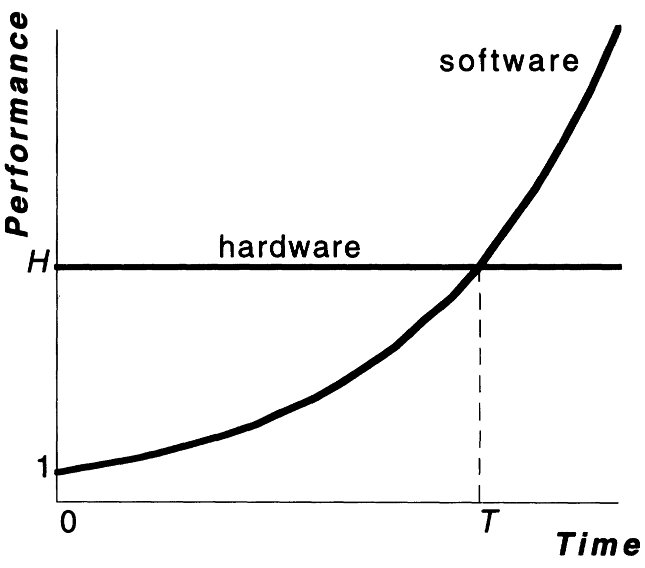
In 1976, it was practical to design and build special-purpose hardware at the gate and register level. One could wire components together point-to-point with inexpensive hand tools. A lot of effort went into hardware design, as this was the way to get high performance. In some cases, high performance is simply the research goal, but it also gives researchers a working environment comparable to future commercial systems. Custom-built hardware is thus a way to discover and solve problems early on.

Unfortunately, many hardware-based projects failed to deliver results, and many more never developed adequate software environments. It is not unusual in any field for projects to end before they reach a mature state. We have seen many systems come and go through the years.

There were some success stories as well. The Samson Box at Center for Computer Research in Music and Acoustics (CCRMA), the SSSP at the University of Toronto, and the 4 Series at Institute de Recherche et Coordination, Acoustique/Musique (IRCAM) all were developed to a point where composers could use them on a routine basis to obtain interesting musical and research results. In particular, the SSSP developed interactive graphical sequencers and patch editors, and explored the use of hierarchical data structures for music representation.

Throughout the late 1970s and 1980s, computers grew more powerful at an exponential rate. Processor speed is still doubling approximately every 18 months. Much of the speed increases result from denser circuitry, and it became impractical to fabricate digital systems by hand using low-density integrated circuits and simple interconnection schemes. Only a large investment in application-specific integrated circuits, gate arrays, and multilayer printed circuit boards is likely to have any performance payoff today.

Figure 1. Hardware and software performance as a function of time (years), as anticipated by someone starting a project at time 0. Software performance accelerates through porting to faster computers, while hardware performance is fixed at the time of design and construction.



As a consequence, we have seen synthesis-hardware development taken over by commercial manufacturers. Yamaha, for example, realized very early that a heavy investment in Very Large Scale Integration would have a high return, and soon low-cost consumer products surpassed all but the most expensive research machines in power if not flexibility.

Where Are We Now?

Now, we seem to be at a turning point. Although MIDI synthesizers and audio processors are the dominant form of synthesis, processors are just crossing the point where they can provide real-time synthesis in software. The situation as it looks to a researcher is shown in Figure 1.

In this graph comparing hardware and software performance, notice that hardware performance is constant—once a piece of hardware is designed, it doesn't get any faster. However, hardware is inherently faster than software (the hardware speedup factor is H).

Now consider software performance. Assuming that software is written in a portable language (C or C++ seem to be the favored choices now), the performance will increase exponentially. Regardless of the development platform, one need only recompile the software on the latest CPU to benefit from advances in hardware technology.

The question for the designer, then, is, How long before software catches up to my hardware? Using the graph,

$$H = 2^{T/1.5}$$

which can be rearranged to derive the following:

$$T = 1.5 \cdot \log_2 H.$$

Let us assume $H = 16$, that is, a hardware approach would run 16 times faster than software. In this case, $T = 6$ years; that is, in 6 years software will begin to outperform hardware you build today.

The second question is, How long will the system take to develop? Three to six years is not unusual for a small research team. Projects always take longer than expected. Obviously, the development time should not be greater than T.

Finally, one must ask, What kind of speed advantage will make a qualitative difference in research? A good rule of thumb is it takes an order of magnitude change to create a qualitative change as opposed to a quantitative change. With a development time of 3 years, H must be 40 or more (!) to realize a 10-fold advantage of the hardware over the software solution.

This logic leads to the conclusion that the days of special-purpose hardware are over. Even digital signal processor-based systems will soon be regarded as ineffective for research. Of course, there is still room for low-cost, high-volume consumer products, but even here, software signal processing is becoming increasingly common. For example, software synthesis is being used to cut hardware costs on sound cards for PC-class machines.

Where Are We Going?

It seems clear that software-based computer music systems are the way of the future. I expect this will lead to several interesting changes in perspective. First, new synthesis methods that involve adaptive filtering, learning, and logical computation will receive more attention. Historically, synthesis algorithms have been based on hardware capabilities, leading to fixed sequences of numerical operations. Given that synthesis will now be software based, it makes sense to consider new schemes that take advantage of the full capability of a general-purpose processor. The most important capability is to perform logical tests and change computation accordingly. Thus, synthesis can move away from mathematical conceptions to more computational ones. An example is the computation of bow-string or reed-mouthpiece collisions in physical models. Another is the assembly and control of computational elements as in resNET (Hamman 1994), where procedural control is tightly integrated with sample generation.

Traditionally, control and synthesis are treated separately in computer music systems. This is a reasonable model with roots in the concept of the performer and instrument, in the score and orchestra dichotomy of the Music-N language family, and now in MIDI. This model leads to "a separation between models of sound material and models of musical design" (Di Scipio 1994). However, I think that software synthesis will begin to break down some of the distinctions, enabling a reintegration of control and synthesis. Composers will happily break free of the limitations of fixed algorithms in VLSI accessed through the MIDI bottleneck. Composers will be freer to exercise artistic creativity at the timbral level.

As machines continue to grow in power, it is hard to imagine what research and compositional issues we will face more than a few years from now. Let us take as a baseline that current high-end personal computers deliver roughly 100 million instructions per sec (MIPS). We can expect about 1 billion MIPS in 20 years, and much more from high-end workstations. Disk capacity is increasing at about 40 percent per year (recently 60 percent), so in 20 years, we should expect personal computers to ship with at least 1 terabyte of on-line storage, enough for 5,000 or so compressed CDs. Also, RAM will be measured in gigabytes rather than megabytes (we used the term kilobytes 20 years ago).

Control

With future computation rates, all synthesis algorithms currently in use will run in real time. There will certainly be new software that will demand even higher computation rates, but the point is that there will be an enormous potential to realize sounds. Are we ready for that potential? In my opinion, we have a long way to go. We spend inordinate amounts of effort optimizing code and microcode, inventing new synthesis techniques, and working at higher levels of control as in algorithmic composition. However, there is relatively little effort invested in dynamic timbre control.

There seems to be a missing link in computer music research that spans the time scale from about 10 msec to 250 msec. This is the realm of articulation, vibrato, slurs, and performance effects. These are tied to longer time-span concepts of phrasing, bowing, and breathing, and to lower time-span concerns of signal control, synthesis, and haptic feedback.

Chris Chafe's (1989) work on string synthesis is a perfect illustration of the type of research that I would like to see more of. In this work, a string quartet score was processed to develop a set of control events such as bow-string contact, bow reversal, finger placement, and vibrato. These events were then used to drive the "synthesis" of smooth control functions for bow position, pressure, and velocity. These complex, continuous functions were then used to control a simple string model, resulting in a rich performance.

In a similar spirit, Brad Garton has developed "expert systems" that control string synthesis models to produce idiomatic "string folk band" and solo rock guitar music (Garton 1992). As in Chris Chafe's work, much of the richness of the result is due to "control rate" information rather than "audio rate" synthesis algorithms or "note rate" compositional structure. These works construct stylistically appropriate strumming and fretting—controls that enhance and complement the synthesis technique.

A natural area for further work is in drumming. In popular, jazz, and ethnic music, drumming appears to operate within highly constrained worlds consisting of idiomatic rhythms and phrases. This is not to say that drumming is easy or that it is easy to automate, but this seems like a natural area for research. A good start was made by Bilmes (1993), who demonstrated that deviations from the beat (as opposed to say, changes in tempo or dynamic) account for "swing," and that these deviations are not random. We need the experience of building a dozen or so "artificial drummer" systems in different styles to really get at the issues. This area seems ripe for machine learning techniques.

Others have studied rubato (e.g., Repp 1990) or micro-timing and how it relates to musical structure, and this work is part of the big picture of generating appropriate control. Piano has been a favorite vehicle for these studies since attack time and velocity capture nearly all of the expressive content. The next step is to study instruments with continuous control: brass, woodwinds, and strings. Some interesting ideas have been put forth by Manfred Clynes (1987) on the relationships among shapes, emotion, and composers, and new formalisms and approaches undoubtedly wait to be discovered.

It seems that there should be more emphasis than ever on languages for expressing control-level information. My own work on the Nyquist language (Dannenberg 1993) has finally evolved to a releasable form. Nyquist is mentioned here because *all* synthesis and signal processing operators can be applied to construct or manipulate control functions. Also, functions can be hierarchically composed and inherited to separate various concerns such as phrase structure, metrical structure, and note structure. Nyquist is intended to support the kind of continuous control research that I am advocating. Kyma (Scaletti and Hebel 1991) and GTF (Desain and Honing 1992) are important related systems.

Conclusion

Computer music has made great strides in the last 20 years. Over the long run, the exponential growth of hardware tends to dominate all other terms in the equation, so the most notable change is the incredible growth in processing power and storage capacity. Along the way, a brief window of opportunity for VLSI-assisted computer music opened, leading to a flood of MIDI-related activity in the computer music community. This activity is near its peak, and the next "wave" will be a return to software synthesis on ever-more-powerful personal computers. Real-time synthesis will be increasingly common.

A neglected area of research is the musical control of sound synthesis. There is a gap between the level of samples, which is addressed by sound synthesis, and the level of "beats," which is addressed by composition. The realm of control in the 10 msec to 250 msec range is underexplored, and many exciting discoveries will be made there.

References

Bilmes, J. 1993. "Timing Is of the Essence: Perceptual and Computational Techniques for Representing, Learning, and Reproducing Expressive Timing in Percussive Rhythm." Masters Thesis. Cambridge, Massachusetts: MIT Media Laboratory.

Chafe, C. 1989. "Simulating Performance on a Bowed Instrument." In M. Mathews and J. Pierce, eds. *Current Directions in Computer Music Research*. Cambridge, Massachusetts: MIT Press, pp. 185–198.

Clynes, M. 1987. "What Can a Musician Learn about Music Performance from Newly Discovered Microstructure Principles (PM and PAS)?" In A. Gabrielsson, ed. *Action and Perception in Rhythm and Music*. Publication no. 55: Stockholm, Sweden: The Royal Swedish Academy of Music.

Dannenberg, R. 1993. "The Implementation of Nyquist, a Sound Synthesis Language." *Proceedings of the 1993 International Computer Music Conference*. San Francisco, California: International Computer Music Association, pp. 168–171.

Desain, P., and H. Honing. 1992. "Time Functions Function Best as Functions of Multiple Times." *Computer Music Journal* 16(2):17–34.

Di Scipio, A. 1994. "Formal Processes of Timbre Composition Challenging the Dualistic Paradigm of Computer Music: A Study in Composition Theory (II)." In *Proceedings of the 1994 International Computer Mu-*

sic Conference. San Francisco, California: International Computer Music Association, pp. 202–208.

Garton, B. 1992. "Virtual Performance Modeling." In *Proceedings of the 1992 International Computer Music Conference*. San Francisco, California: International Computer Music Association, pp. 219–222.

Hamman, M. 1994. "Dynamically Configurable Feedback/Delay Networks: A Virtual Instrument Composition Model." In *Proceedings of the 1994 International Computer Music Conference*. San Francisco, California: International Computer Music Association, pp. 394–397.

Karplus, K., and A. Strong. 1983. "Digital Synthesis of Plucked String and Drum Timbres." *Computer Music Journal* 7(2):43–55.

Repp, B. 1990. "Patterns of Expressive Timing in Performances of a Beethoven Minuet by Nineteen Famous Pianists." *Journal of the Acoustical Society of America* 88:622–641.

Scaletti, C. and K. Hebel. 1991. "An Object-based Representation for Digital Audio Signals." In G. DePoli, A. Picialli, and C. Roads, eds. *Representations of Musical Signals*. Cambridge, Massachusetts: MIT Press, pp. 371–389.