Foreword

Barry Vercoe

It is indeed a pleasure to peruse this volume, to see so many composers and authors joined in a similar purpose of making their insights and experiences public and to feel that the computer music community will surely benefit from such broad-based disclosure. It is never easy to have more than one living composer present at a single concert of their collected works, and to the extent that these contributions represent composing time lost and thoughts and insights given away free, the richness and evenness of this volume suggests that these composer/authors must all have been ensemble performers first. Of course, every ensemble has its taskmaster, and I stand in awe of what Richard Boulanger has done to bring this one to the concert stage.

This field has always benefited most from the spirit of sharing. It was Max Mathews's willingness to give copies of **Music 4** to both Princeton and Stanford in the early 1960s that got me started. At Princeton it had fallen into the fertile hands of Hubert Howe and the late Godfrey Winham, who as composers imbued it with controllable envelope onsets (**envlp**) while they also worked to have it consume less IBM 7094 time by writing large parts in a BEFAP assembler (**Music4B**). Looking on was Ken Steiglitz, an engineer who had recently discovered that analog feedback filters could be represented with digital samples. By the time I first saw Music4B code (1966–1967) it had a **reson** filter—and the age of subtractive digital sound design was already underway.

During 1967–68 I wrote a large work (for double chorus, band, string orchestra, soloists and computer-generated sounds), whose Seattle Opera House performance convinced me that this was a medium with a future. But on my arrival back at Princeton I encountered a major problem: the 7094 was to be replaced by a new machine called a 360 and the BEFAP code would no longer run. Although Godfrey responded by writing a Fortran version (**Music4BF**, slower but eternally portable), I took a gamble that IBM would not change its assembler language again soon, and wrote **Music 360**. Like Max Mathews, I then gave this away as fast as I could, and its

super efficiency enabled a new generation of composers with limited budgets to see computer music as an affordable medium.

But we were still at an arm's length from our instrument. Punched cards and batch processing at a central campus facility were no way to interact with any device, and on my move to the Massachusetts Institute of Technology in 1971 I set about designing the first comprehensive real-time digital sound synthesizer, to bring the best of Music 360's audio processing into the realm of live interactive performance. After two years and a design complete, its imminent construction was distracted by a gift from Digital Equipment Corporation of their latest creation, a PDP-11. Now, with a whole computer devoted exclusively to music, we could have both real-time processing and software flexibility, and **Music 11** was the result.

There were many innovations in this rewrite. First, since my earlier hardware design had introduced the concept of *control-rate* signals for things like vibrato pitch motion, filter motion, amplitude motion and certain envelopes, this idea was carried into the first 1973 version of Music 11 as **k-rate** signals (familiar now to Csound users). Second, envelopes became more natural with multi-controllable exponential decays. Indeed, in 1976 while writing my *Synapse,* for Viola and computer, I found I could not match the articulation of my soloist unless I made the steady-state decay rate of each note in a phrase be a functional inverse of the note length. (In this regard string and wind players are different from pianists, who can articulate only by early release. Up to this time we had all been thinking like pianists, that is, no better than MIDI.) My **envlpx** opcode fixed that.

This had been my second gamble that a particular machine would be sufficiently common and long-lived to warrant assembler coding, and Music 11's efficiency and availability sustained a decade of even more affordable and widespread computer music. Moreover, although the exported code was not real-time, our in-house experiments were: Stephen Haflich connected an old organ keyboard so that we could play the computer in real-time; if you played something reasonably metric, the computer would print out the score when you finished; if you entered your score via our graphical score editor, the machine would play it back in real-time (I made extensive use of this while writing *Synapse*); if you created your orchestra graphically using Rich Steiger's **OEDIT**, Music 11 would use those instruments. Later, in 1980, student Miller Puckette connected a light-sensing diode to one end of the PDP−11, and an array-processing accelerator to the other, enabling one-dimensional conducting of a real-time performance. Haflich responded with a two-dimensional conducting sensor, using two sonar cells from a Polaroid camera. This was an exciting time for real-time experiments, and the attendees at our annual MIT Summer Workshops got to try many of these.

Meanwhile, my interest had shifted to tracking live instruments. At IRCAM in Paris in 1982, flutist Larry Beauregard had connected his flute to DiGiugno's 4X

audio processor, enabling real-time pitch-following. On a Guggenheim at the time, I extended this concept to real-time score-following with automatic synchronized accompaniment, and over the next two years Larry and I gave numerous demonstrations of the computer as a chamber musician, playing Handel flute sonatas, Boulez's *Sonatine* for flute and piano and by 1984 my own *Synapse II* for flute and computer—the first piece ever composed expressly for such a setup. A major challenge was finding the right software constructs to support highly sensitive and responsive accompaniment. All of this was pre-MIDI, but the results were impressive even though heavy doses of tempo rubato would continually surprise my **Synthetic Performer**. In 1985 we solved the tempo rubato problem by incorporating *learning from rehearsals* (each time you played this way the machine would get better). We were also now tracking violin, since our brilliant, young flutist had contracted a fatal cancer. Moreover, this version used a new standard called MIDI, and here I was ably assisted by former student Miller Puckette, whose initial concepts for this task he later expanded into a program called **MAX**.

On returning to MIT in 1985 it was clear that microprocessors would eventually become the affordable machine power, that unportable assembler code would lose its usefulness, and that ANSI C would become the *lingua franca.* Since many parts of Music 11 and all of my Synthetic Performer were already in C, I was able to expand the existing constructs into a working **Csound** during the Fall of that year. Once it was operating, I received additional help from students like Kevin Peterson and Alan Delespinase and later from Bill Gardner, Dan Ellis and Paris Smaragdis. Moreover, thanks to the internet and ftp/public, my continuing wish to share the system even as it gained further maturity would take even less of my time.

The step to **Real-time Csound** was a simple one. With the right constructs already in place owing to my long-time interest in interactive performance, and computers now fast enough to do floating-point processing on a set schedule, I only had to use the DAC output pointer to implement blocking I/O on a fine time-grid to achieve tight interactive control. I took that step in 1990, and demonstrated it during the ICMC paper *Real-time Csound: Software Synthesis with Sensing and Control* (Vercoe and Ellis 1990). For me, the only reason for real-time is controllable performance, and Dan Ellis illustrated this by controlling a Bach synthesis by tapping arbitrary drum patterns on the table that held the microphone. The sensing also introduced Csound's new **Spectral Data Types** (see my chapter in this volume). With a sufficiently powerful machine (at the time a DECstation), both sensing and controlled high-fidelity synthesis had finally become possible.

But not all of us can command such a powerful central processor and today's interest in deft graphical control and graphical audio monitoring can often soak up the new cycles faster than technology creates them. At the 1996 ICMC in Hong Kong, I demonstrated an alternative architecture for both software and hardware with

Extended Csound. This is the first stage of an orderly progression towards multi-processor fully-interactive performance. In the current version, Csound is divided between two processors, a host PC and a DSP-based soundcard. The host does all compiling and translation, disk I/O, and graphical-user-interface (GUI) processing, such as Patchwork (editing) and Cakewalk (sequencing). The DSP does all the signal processing, with sole access to the audio I/O ports; it also traps all MIDI input with an on-chip MIDI manager, such that each MIDI note-on results in an activated instrument instance in less than one control period.

The tightly-coupled multi-processor performance of Extended Csound has induced a flurry of new opcodes, many of them tailored to the internal code of the DSP I am using (a floating-point SHARC™ 21060 from Analog Devices). The new opcodes extend the power of Csound in areas such as real-time pitch-shifting, compressor-limiting, effects processing, mixing, sampling synthesis and MIDI processing and control. Since this volume is not the place for details, the curious can look at my paper in the 1996 ICMC Proceedings. I expect to be active in this area for some time.

Meanwhile, the fully portable part of Csound is enjoying widespread use, and this volume is a testament to the ingenuity of its users. Some are familiar names (Stephen Beck, Richard Dobson, Brian Evans, Michael Clarke, John ffitch, Richard Karpen, Jon Nelson, Jean Piché and Russell Pinskton) whom we have come to lean on out of both habit and dependency; some are newer lights (Elijah Breder, Per Byrne Villez, Michael Gogins, Andrew Horner, Alan Lee, Dave Madole, David McIntyre, Hans Mikelson, Michael Pocino, Marc Resibois, and Erik Spjut) we are attending to with growing interest; others are fresh faces (Bill Alves, Mike Berry, Martin Dupras, Rajmil Fischman, Matt Ingalls, Eric Lyon, Gabriel Maldonado, and Paris Smaragdis) likely to leave long-term images. The range of topics is broad, from sound design using various synthesis and modeling methods to mathematical and signal processing constructs we can learn from, and from compositional strategies using existing Csound resources to some powerful lessons on what to do when what you need is not already there. The whole is amply supported by CD-ROM working examples of ports to other machines, user-interfaces, personal approaches to composing, and most importantly, compositions that speak louder than words.

I am impressed and somewhat humbled by the range of thought and invention that Csound has induced. Gathering all this energy into one place creates a singularity that defies real measurement, but I am certain the effects of this volume will be much felt over both distance and time. On behalf of the readers, my thanks to all the contributors. Finally, my hat off to Richard Boulanger for all that composing time lost (you should have known better), and my best wishes to Csound users as you continue to redefine your own voice.