Preparation for Interactive Live Computer Performance in Collaboration with a Symphony Orchestra

Timothy Madden, Ronald Bruce Smith, Matthew Wright, & David Wessel

CNMAT, UC Berkeley, 1750 Arch St., Berkeley, CA, 94709

email: {tjmadden,smith,matt,wessel}@cnmat.cnmat.berkeley.edu

Abstract

This paper describes the design, implementation, and use of an interactive computer-based instrument in the context of a composition for orchestra and live electronics. We will begin with an overview of the piece from a compositional point of view, emphasizing the musical goals for the electronics aspect. Then we will discuss the extremely demanding technical requirements of integrating live electronics with a full orchestra. We will describe the interactive instrument in detail, emphasizing the novel mapping of performer's gestures to computer-generated sound events and the novel OpenSound Control-based structure of the software.

1. Introduction

This paper describes the design, implementation, and use of an interactive computer-based instrument for Ronald Bruce Smith's composition *Constellation* for orchestra and live electronics (Smith, 2000). *Constellation* was premiered on November 10, 2000 by the Berkeley Symphony Orchestra conducted by Kent Nagano. The concert was held in Hertz Hall on the UC Berkeley campus.

2. Musical Goals

Harmonic and inharmonic spectra (from bell-like spectra) form the basis of the harmony in *Constellation*. From this, several layers of harmony and spectral reinforcement were composed. First, spectral components were subjectively selected by the composer, resynthesized using additive synthesis and orchestrated using acoustic orchestral instruments. Secondly, certain pitches (components) from a given spectrum were selected as principal melodic tones. Those pitches were then highlighted through both the orchestra and electronics by reinforcing selected spectral components of the harmonic spectrum of the orchestral instrument(s) performing the selected pitch(es) of the resynthesized timbre.

With respect to harmonic movement in the composition, interpolations between spectra were calculated using IRCAM's Patchwork software (Laurson and Duthen, 1990, Assayag, 1993). The interpolations were then subjectively altered by the composer, especially with respect to the voicings of the harmonies that the interpolations yielded. Those harmonies were then resynthesized to form timbres that were subsequently treated by the layering and reinforcement techniques outlined above.

The electronic instrument in this piece was used to produce several kinds of sounds. Some of the inharmonic spectra described above were synthesized directly with additive synthesis. Other inharmonic spectra, such as percussive bell-like tones, were synthesized using exponentially decaying sinusoids. Similar synthesis techniques produced glockenspiel-like tones. Many of the notes in the orchestral parts were doubled by harmonic, synthetic timbres, also produced with additive synthesis. Other parts used sample playback.

3. Implementation

3.1 Goals

The implementation of this instrument was determined in large part by the special and severe constraints of working with a symphony orchestra. We refused to freeze the electronics part on tape and require the conductor to wear headphones with a click track; instead we made all of the electronics performable by a musician sitting in the orchestra watching the conductor. Rehearsal time was extremely limited, so it would have been a disaster to have to wait for a computer to reboot; therefore high reliability was essential. Limited rehearsal time and the lack of any sound check give us no time to adjust the volumes of the electronic parts in context with the orchestra, so we made all gains controllable in real-time by both the performer and the

composer sitting in the hall. The conductor needed to be able to jump from section to section in rehearsal, so the electronics needed to be able to quickly go to any event in the score.

3.2 Hardware

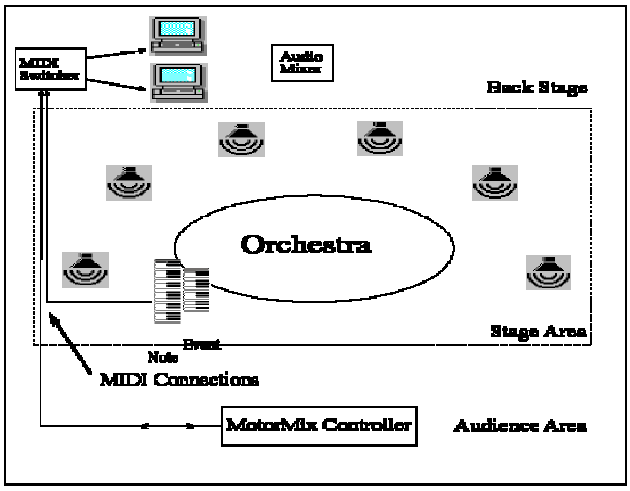

Figure 1. In this schematic diagram of the concert set up, the audio connections are not shown. They run between the multi-channel DAC's on each of the computers and the audio mixer. The audio mixer was configured to switch its source rapidly and silently from one computer to the other if one failed. Likewise, the MIDI switcher was available to redirect MIDI in the event of computer failure. The composer controlled the gains of the audio sources from his position in the center of the audience with the motorized faders of a MotorMix Controller. During rehearsals the gains were adjusted and set. During the performance they were fine-tuned by overriding the settings sent from the computers to the MotorMix. The note keyboard was equipped with a pedal for tapping in the tempo given by the conductor.

We used the Max/MSP environment (Zicarelli, 1998) for all of the software for this piece. The entire program ran redundantly on two Macintosh G3 computers, so that if one crashed during a rehearsal or performance, we could instantly switch to using the other. The performer played two MIDI keyboards, including volume pedal, pitch bend wheel, and sustain pedal. The composer sat in the hall with a CM Automation MotorMix motorized fader bank, finely adjusting the levels and the envelopes of the synthesized timbres during rehearsal and performance. We used MOTU 2408 sound I/O hardware with each Macintosh, both connected to the speakers via a Mackie 1604 mixer. Synthetic sound was diffused from an array of six Meyer UPL-1 loudspeakers situated in a horseshoe shape around and behind the orchestra. Figure 1 shows the physical layout of this equipment for the performance.

/ScoreEvent/midinote-off
/ScoreEvent/midinote-on
/bell1/cutoff-freq
/glockies/midinote-off
/glockies/midinote-on
/harmtones/osc[1-8]/glissdir
/harmtones/osc[1-8]/glissmag
/harmtones/osc[1-8]/init
/harmtones/osc[1-8]/mute
/harmtones/osc[1-8]/play
/harmtones/osc[1-8]/volume-pedal
/midi-routing/note-gate
/midi-routing/note-gate/glockies
/midi-routing/note-gate/ronbell
/midi-routing/note-gate/sampler
/midi-routing/volume-pedal-gate
/midi-routing/volume-pedal-gate/bell1-freq
/midi-routing/volume-pedal-gate/harmtone-volume
/ronbells/mute
/ronbells/play/bell2a
/ronbells/play/bell2b
/ronbells/play/bell2c
/ronbells/play/bell2d
/ronbells/play/bell3a
/ronbells/play/bell3b
/sampler/midinote-on
/sampler/midinote-off
/sampler/multisample
/sampler/speaker
/sampler/master-volume

Table 1: OSC Address Space of the Patch

3.3 Software

The two MIDI keyboards played by the live musician had very different effects on the Max/MSP patch. One of the keyboards, called the "event keyboard," was used to set the software into states to synthesize specific sounds. For example, if a section of the score called for a bell tone, the musician pressed a specific key on the event keyboard to

1, /harmtones/osc*/glissdir 0.;
2, /midi-routing/volume-pedal-gate/harmtone-volume 1;
3, /harmtones/osc{1\,4}/speaker/number 2;
4, /harmtones/osc[2-3]/speaker/number 5;
5, /harmtones/osc1/play 1760. 2.3;
6, /harmtones/osc2/play 3078.530762 2.3;
7, /harmtones/osc3/play 3960.205811 2.3;
8, /harmtones/osc4/play 4841.90625 2.3;

Tab;e 2: OSC Commands Comprising Score Event 2

trigger an event, in this case, the synthesis of a bell sound. The Max/MSP patch is essentially stateless except for whatever was configured by the most-recently-played key on the event keyboard; in other words, each new event supercedes the previous event and does not rely on previous events having been played. The use of the event keyboard made possible the ability to move effortlessly and swiftly to any section of the score during a rehearsal since the computer could be set to any state at any time.

The second keyboard, called the "note keyboard," was played like a traditional piano, with each key triggering a note at a given pitch. The note keyboard can play different timbres at different times, thanks to events that reconfigure the mapping of the note keyboard.

Unique to the software was the use of Open Sound Control (OSC) (Wright and Freed, 1997, Wright, et al., 2001) to control all aspects of the patch. All of the sound synthesis capabilities of the software were arranged in an OSC address space. For example, to play the "glock" instrument at some pitch at some loudness, the OSC command is `/glock/playnote` *`note loudness`*. Table 1 lists the entire OSC address space for this patch. The only way for the patch to make a sound is for the control part of the patch to send an OSC message to the synthesis part of the patch.

All OSC messages went through a central OSC dispatcher that routes each message to the correct place. This gave us an extremely useful debugging tool: seeing every message the patch was sending itself. The typical software architecture of a large Max/MSP patch, with multiple `send` and `receive` objects, makes it very difficult to trace all of the intra-patch communication.

Since any event in the score (each corresponding to a note on the event keyboard) requires several simultaneous messages to be sent to the patch, an electronic score or database of OSC commands was used to store these messages. In this way, when a note on the event keyboard was pressed, several OSC commands were issued from the electronic score. We used Max's `coll` object to store the sequence of OSC commands corresponding to each event. Table 2 shows the collection of OSC messages that implement event number two in the electronics score. The first event disables glissandi for all of the additive-synthesis-generated "harmtones". The second event remaps the MIDI volume pedal to control the volume of the harmtones. The 3rd and 4th assign the output of the first four harmtone voices to particular speakers; note the use of OSC pattern matching. The 5th through 8th events cause the harmtones to synthesize tones with given frequencies and loudnesses.

3.4 Synthesis

Because the computer must synthesize a large variety of sounds throughout the piece, including bell-like tones, harmonic tones, glockenspiel-like sounds, and various percussive sounds, several synthesis methods were employed. The glockenspiel-like sounds were synthesized with digital resonators (using the `resonators~` MSP object) (Jehan, et al., 1999) from analysis data of a marimba sound stored in Sound Description Interchange Format (SDIF) (Wright, et al., 1999, Schwarz and Wright, 2000). To avoid simply reproducing transpositions of the marimba sound, the sound was filtered to add many irregularly-spaced formants to the sound. Also, randomness was added to the spectral envelope to make each attack sound slightly different. To prevent high notes from sounding too metallic, the decay time was inversely scaled with fundamental frequency so that high notes decay faster than low notes. In order to make the timbre change with loudness, a brightness model based on exponential functions was employed.

Harmonic tones were produced with additive synthesis using MSP's `sinusoids~` object (Jehan, et al., 1999). Small time-varying deviations were added to the partials to enrich the sound. Also, variance in odd/even partial content, formants, and exponential brightness models were used.

Bell tones were synthesized with digital resonators using predefined lists of frequencies, amplitudes, and decay rates. The decay rate can be altered in real time using the MotorMix.

Finally, the patch created some percussive sounds using samples. In some instances, prerecorded MIDI sequences

were used to play long passages of notes. In order to keep the tempo of the MIDI sequence synchronized with the live orchestra conductor, the keyboardist tapped a foot pedal in a tempo established by the conductor thus setting the speed of the MIDI sequence. A novel MIDI sequencer, allowing real-time tempo changes, was created for this purpose.

The MotorMix, used by the composer to adjust the loudness of the electronic sounds, features motorized faders which can be controlled by the Max patch to store a preset mix. A problem when storing a preset mix is synchronizing the mix with the live performance. To solve this problem, a database was created to store the MotorMix settings. The event keyboard, which triggers state changes in the patch, was used to reference the database, and thus control the MotorMix. During the rehearsal, the composer could adjust the faders on the MotorMix and store the settings in the database indexed by the note played on the event keyboard. In this way, the mix could be synchronized to the live performance.

4. Conclusions

We now detail the successes and failures of the design of the electronics for this piece. The use of the event keyboard to control the electronic sounds proved to be a success. The written electronics part played by a live musician resembled a piano score, allowing a professional pianist to play the electronics part with relatively little rehearsal time. A slight failure was that there is no current standard for the layout of the conductor's score when live electronics is involved. During the rehearsals, the conductor was a bit uncomfortable with location of the electronics part which was placed in the written score where one would notate a keyboard part. We found that it is best to keep the standard orchestra layout of all of the acoustic parts, and add electronic parts to the top or bottom of the written score.

The Max patch was created by musically knowledgeable engineers in close collaboration with the composer. The composer described the desired sound, the engineers designed software to approximate the composer's desires, and the process would iterate until both the engineers and composer were satisfied. The process of close collaboration seemed to be successful.

The use of OpenSound Control was a success in that it offered a simple unified interface to control the patch. In orchestra rehearsal, several unforeseen problems with the Max patch arose, requiring modifications of the patch during rehearsal time. The OpenSound Control interface allowed us to solve unforeseen problems without wasting valuable rehearsal time.

5. References

Assayag, G. 1993. *CAO: Vers la partition potentielle*. Paris: IRCAM.

Jehan, T., A. Freed, and R. Dudas 1999. Musical Applications of New Filter Extensions to Max/MSP. *Proceedings of the ICMC*, Beijing, China.

Laurson, M. and J. Duthen 1990. A compositional environment based on Preform II, Patchwork and Esquisse. *Proceedings of the International Computer Music Conference.*

Schwarz, D. and M. Wright 2000. Extensions and Applications of the SDIF Sound Description Interchange Format. *Proceedings of the International Computer Music Conference*, Berlin, Germany, pp. 481-484.

Smith, R. B. 2000. “Constellation for orchestra and live electronics,” Canadian Music Centre, 20 St. Joseph Street, Toronto, ON, M4Y 1J9 Canada.

Wright, M., A. Chaudhary, A. Freed, S. Khoury, and D. Wessel 1999. Audio Applications of the Sound Description Interchange Format Standard. *Proceedings of the Audio Engineering Society 107th Convention.* (http://cnmat.CNMAT.Berkeley.EDU/AES99/docs/AES99-SDIF.pdf)

Wright, M. and A. Freed 1997. Open Sound Control: A New Protocol for Communicating with Sound Synthesizers. *Proceedings of the International Computer Music Conference*, Thessaloniki, Hellas, pp. 101-104.

Wright, M., A. Freed, A. Lee, T. Madden, and A. Momeni 2001. Managing Complexity with Explicit Mapping of Gestures to Sound Control with OSC. *Proceedings of the International Computer Music Conference*, Habana, Cuba.

Zicarelli, D. 1998. An Extensible Real-Time Signal Processing Environment for Max. *Proceedings of the International Computer Music Conference*, Ann Arbor, Michigan, pp. 463-466.