

Development of a Versatile Interactive Performance System

Douglas Geers* and Maja Cerar†

*School of Music, University of Minnesota
geers001@umn.edu

†Music Department, Columbia University
msc48@columbia.edu

Abstract

This paper describes the development of an interactive performance system begun in 2001 and built using Max/MSP. Initially created for performance of a multimedia theater work, the system has been further developed since that time for use as an instrument for a variety of chamber music and improvisational situations.

1 Introduction

Since 1999, the authors have engaged in a series of collaborative performances, combining violin (Cerar) with computer performance (Geers). The authors have experimented with a number of approaches, and since 2001 have been developing a Max/MSP-based software system for collaborative performance. The system consists of a reconfigurable set of modules for sound synthesis and signal processing, initially created for interactive performance of a multimedia theater work, *Gilgamesh*. In the following pages, we will describe the initial design and use of the system, and then will describe how it has been subsequently developed.

2 Initial Design and Operation

During the first two years of the authors' collaboration, the degree of electroacoustic interactivity in the works created and performed grew gradually: Initially, pre-composed electroacoustic tracks were cued during violin performance, and the violin's sound was not processed beyond the use of reverb. However in 2001 the authors began to explore MSP and add real-time interactive musical components into the performance system.

The MSP performance system, then called *EA-7*, was originally designed for *Gilgamesh*, a seventy-minute multimedia work by Geers in which all the music is performed on one violin and one computer. In *Gilgamesh*, the computer sounds took on a role akin to that of the orchestra in a concerto: answering the violin, accompanying it, and at times acting as the primary voice of the music. For *Gilgamesh*, the *EA-7* computer instrument had three main

functions: live sound synthesis, live sound processing, and playback of pre-composed materials. Numbered cues were placed at specific locations throughout the written *Gilgamesh* score indicating changes to live processing and synthesis settings (DSP) and for cueing short sound files to play back (SF) (Figure 1). During performance of the piece, the computer musician adjusted the active modules and their settings at each of these designated moments to specific, pre-composed values.



Figure 1. Score with DSP presets marked.

The computer sounds created for *Gilgamesh* fall into three categories: pre-composed segments of various lengths, live synthesized events, and timbral coloring (processing of violin and synthesized materials). *EA-7* was used in the creation of many of the studio-composed cues; but here we will focus on the live synthesis and processing, since these are the tasks for which the instrument was primarily designed.

Twelve individual synthesis, processing, and playback patches (hereafter called *modules*) were built in MSP, and these were combined to create the live performance system *EA-7*. Each patch was self-contained, and the path of signal flow through the modules was completely and constantly variable. This was possible via a system of named MSP *send~* objects that allowed the audio output from any module to be routed to any of the others. To change the destination of one module's output, its output *send~* was renamed to match the desired destination.

2.1 Modular Design

The computer instrument modules built for *EA-7* were additive synthesis, granular synthesis, audio file playback (three of these), digital delays, waveshaping, flanger, tremolo, moving resonant filter, moving low-pass filter, and comb filter. Moreover, some of the modules were

connected to the MSP **fiddle~** object so that they could respond based on particular characteristics of the violin performance. This combination of DSP modules was chosen to create the most rich and varied sound possible with the computing power of the single Macintosh G3 available for performance at that time (2001). The instrument has since evolved considerably, as will be discussed below, thanks to increased CPU speeds and additional components that have been added to Max/MSP.

Beyond processor-load issues, the DSP modules used were chosen specifically for the ways that they could complement and enhance the sound of the violin itself, which can produce an enormous range of sound qualities even without any added processing. The violin provided a rich sonic foundation and enlivened the processing, when it was used, and enabled *Gilgamesh* to sustain interest through its seventy-minute length. The processing complemented the violin's sound in that the ways and degree to which it masked or did not mask the acoustic violin's own timbre was varied widely. Filters, for instance, were useful for pulling out components of the violin sound while not covering it, whereas waveshaping distortion produced rich partials that could mask the brighter components of violin notes. Thus combinations of the multiple signal processors were used to create a kind of orchestration derived from the sound of the violin, and the authors discussed and jointly decided what timbral results were successful, collaborating to create a good balance of violin vs. synthetic sound. Cerar, for instance, disliked mechanically regular patterns of sound modulation; and thus the delay lines, LFOs, etc., used in *Gilgamesh* were built to avoid these simple repetitive patterns.

In addition to signal processing, *EA-7* also possessed modules for sound file playback and real time additive synthesis. Of these, the additive synthesis module was noteworthy because it created twenty-eight-note harmonies in which each pitch had its own amplitude envelope and panning trajectory. The entire *Gilgamesh* piece was structured around these harmonies, and so their realization was crucial to the work. However, because the additive synthesis module put a heavy processor load on the Macintosh when used, much of *Gilgamesh* employs harmonies played from sound files or which are manifest via the violin's melodic content. Use of the live additive synthesis module was restricted to sections of the work when real time synthesized harmonic reaction to the violin was of primary concern.

2.2 Modular Interface

The *EA-7* system was designed in a modular fashion in hopes that many of its elements and possibly all of them could be “broken out”, reorganized and recombined, and reused in future projects. In fact, nearly everything visible on the main *EA-7* screen was the “face” of a module imported into the final interface via the Max **bpatcher**

object (Figure 2). Within each instrument module, the visual controls and the processing “engine” objects were separated so that when the module was imported into the main patch the **bpatcher** window could be sized to show only the module's controls. Thus the *EA-7* interface presented a large and reconfigurable array of controls without displaying the visually distracting processing algorithms.

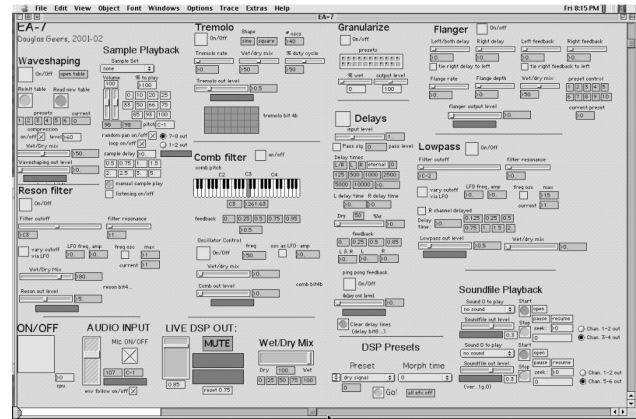


Figure 2. Main EA-7 interface (2002).

In the main *EA-7* patch, the twelve modules were arranged on the computer screen as a variety of rectangular “faceplates” in a manner akin to old modular synthesizers such as the Buchla 100 Series. This similarity of visual layout was intentional, because the authors were trying to design an instrument rather than an automated patch (although it was also capable of automated use, as described below). Moreover, the controls on each module's faceplate were updated in real time to reflect the instrument's current state. Although this was not necessary for the module to function, it was desired that the computer performer know the state of the instrument in order to facilitate a successful performance.

Because of limited screen space, the interface to the additive synthesis engine (to play the 28-note harmonies mentioned above) was separated from the other modules and was scrolled to when it was the center of the computer performer's musical attention (Figure 3).

2.3 Instrument Function

In performance, a wireless lavalier microphone is placed on the violin and fed into the computer. The choice to use a microphone rather than a MIDI violin or mounted pickup on the violin was made in order to most accurately preserve and capture the timbral qualities of the acoustic violin, for the reasons described in section 2.1 above. Listening to the violin, the software (led by the MSP **fiddle~** object) gathers information regarding the violin's pitch and rhythmic patterns, and then uses this data to generate its own material in response. However, as the computer musician leads the

computer through its "score" of activities during a performance, he/she may also use the modules' controls on the screen to guide or tweak the computer's musical expression, enabling subtle sonic changes and quite precise synchronizations with the violin performance or other events onstage. Between the musicians' ability to anticipate and react musically to each other, the computer's "listening" via the *fiddle~* object, and the harmonic materials shared by the computer and violin parts, the violin and computer can always be related to each other through both gesture and harmony.

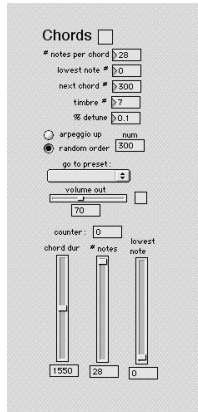


Figure 3. Harmonic playback interface.

In order to quickly adjust numerous settings at once on the *EA-7* instrument, an extensive system of messaging and preset values for each audio module allows the computer musician to initiate multiple simultaneous parameter changes with a single keystroke or push of a button via the "DSP Presets" module (see Figure 4). The computer musician increments the numbered DSP setting (as seen in the score excerpt in Figure 1) at appropriate moments in the score by using this module. When the user hits the DSP Presets module's "Go" button (or the spacebar on the computer keyboard), each parameter of every audio module is directed to change its settings to new values, interpolating from the current values over a specified amount of time associated with that preset. To keep the display coherent for the computer performer, all on-screen graphic interface objects interpolate in synch to display the new values. The use of *EA-7* in *Gilgamesh* takes much advantage of the fact that the modules interpolate between values rather than jumping from one to the next, because long interpolation times (as long as thirty seconds) were used to create patterns of evolving timbre. Thus the presets themselves became markers along a much wider trajectory of electroacoustic sound.

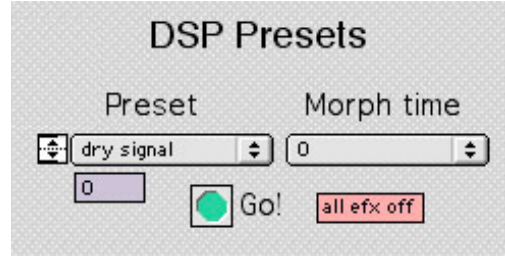


Figure 4. Controls for Automated Parameter Evolution.

For its initial use in the performance of *Gilgamesh*, the DSP presets of *EA-7* were programmed to follow the order of events in that piece, but they can be and have been reprogrammed for other compositions. The use of the DSP Presets module means that users do not need to operate the individual module's controls when performing with *EA-7*. However, one may use the parameters of presets as "jumping-off" points for real time manual performance; and indeed, after composing *Gilgamesh* the authors did adapt *EA-7* for use in their completely improvised performances.

Moreover, since several sections of *Gilgamesh* may be played as standalone concert works and since these pieces are sometimes played when the composer is not present, the authors decided to create a simplified *EA-7* interface so that unfamiliar users would not become confused by the numerous onscreen modules. For these situations the individual modules' interfaces were replaced by a single, simple control interface consisting of an incrementing menu, basic audio I/O faders with labels, and an emergency mute switch (Figure 5).

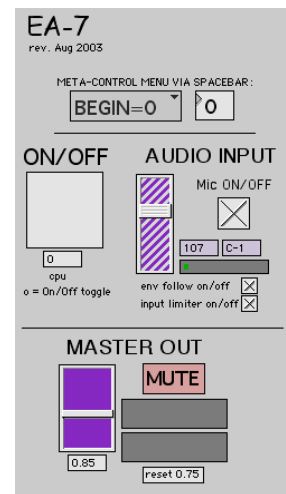


Figure 5. Simplified *EA-7* interface.

3 Further System Development

Since the creation of *EA-7*, the system has been extensively updated and numerous modules have been added. The modular organization of the system has proven

quite effective, allowing for relatively easy upgrading or replacement of individual instrument modules. Thus the instrument has evolved in an organic fashion, with more interesting and effective modules taking the places of older ones, based on compositional situations and judgments of effectiveness from performance experiences.

The current form of the instrument, now named *Hop* (Figure 6), contains a reconfigurable set of twenty-seven instruments. *Hop* and *Hop Jr.*, a reduced version of it containing only six modules, are freely available for download at <http://www.dgeers.com/hopjr.html>.

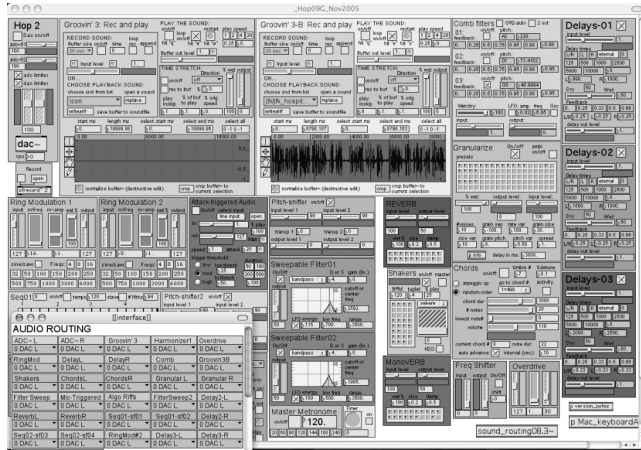


Figure 6. Hop instrument (2005).

Hop has been used extensively for improvisation, including many performances as part of the authors' laptop, laptop, and violin trio *Sønreel*. *Hop* has also been used for performance of *Twisting Pairs* (2005), for improvising chamber ensemble and computer, as well as for completely notated concert works, including a work for big band and computer, *Memory Dust* (2003), and others for violin and laptop in a manner akin to *Gilgamesh*, including *Shadow* (2005) and *Obsessive Currents* (2005).

Although it is a direct descendant of *EA-7*, *Hop* looks and functions quite differently. The DSP Presets module has been replaced by a programmable list of routing presets, and the instrument now can listen to and react to and/or process two unique audio streams simultaneously. It could technically handle more audio inputs, but the authors have found that it is generally enough work for a laptop performer to manipulate two audio streams at once. In addition, the output of any of *Hop*'s modules may be routed to the input of any other module at any time via a new Audio Routing Interface (lower left corner of Figure 6). In *EA-7*, rerouting of audio paths was possible but required some programming to implement. *Hop*'s Audio Routing Interface maximizes the instrument's sonic flexibility and potential, and thus greatly facilitates improvisational performances.

Some of the instrument modules present in *Hop*, beyond those inherited from *EA-7*, include: (1) live audio recording, editing, scrubbing, and speed alteration; (2) a bank of tunable comb filters with algorithmically generated tuning combinations; (3) audio pitch transposition, (4) frequency shifting, (5) convolution, (6) reverberation, (7) ring modulation, (8) automated moving filters (filter type is selectable), (9) stochastically performed physically-modeled percussion, (10) audio files played back stochastically (based on note detection), and (11) algorithmically generated pitched material that may be constrained to programmed scales or to sets of pitches produced by a cellular automata tracking what has been recently played, with stochastically configurable rhythmic patterns and densities.

The instrument modules of *Hop* may be swapped in and out of it in numerous combinations depending on performance interest in a manner similar to the function of plug-ins in mixing software such as Pro Tools or Digital Performer. Moreover, since each *Hop* module is a Max/MSP *bpatcher*, they may be freely rearranged onscreen at any time. This functional and visual flexibility has proven useful in the continuing development of the instrument.

3.1 Tactile Control

As the instrument has grown more complicated, and in order to further increase its usefulness and success as an instrument capable of sophisticated musical interaction in chamber music and improvisation situations, the authors have explored the use of a number of MIDI control surfaces, including the Peavey 1600x and the Oxygen 8. Currently, the authors use the Evolution UC-33 controller, which has 47 programmable faders and buttons and multiple programmable "scenes" (Figure 7.) We chose the UC-33 because it is relatively inexpensive with a large number of programmable controls, but also because it is lightweight and USB powered (more on this below).



Figure 7. Evolution UC-33 MIDI controller.

3.2 System Limitations

The interface design and methods of control of the current software instrument are not yet completely satisfying. The complexity of the instrument allows for sophisticated electroacoustic music to be created and developed, and the detailed array of controls and parameter

value indicators is useful but at times momentarily confusing. One possible solution is to allow for modules' interfaces to be hidden and shown at various times during performance, so that the computer performer may concentrate on whatever subset is of most interest at the moment. However hiding some modules may cause the performer to lose track of them during performance.

Regarding the MIDI control surface: The use of a controller with a large number of physical control points allows for very quick attention to multiple parameters, and the authors prefer to use a control surface that minimizes the need to map single physical controllers to multiple software destinations. That is, we prefer to have many immediate controls and fewer "scene changes." This is why we have also eschewed physical controllers that are visually interesting but possess fewer points of control.

However, although MIDI control surfaces such as the UC-33 provide more controls, their larger arrays of faders and knobs can also cause a performer to lose his/her orientation at the control surface. Moreover, larger control surfaces are less convenient to transport, which is a serious consideration for those who want an electroacoustic instrument to be as portable as a violin. As those who fly often with their gear will attest, these seemingly peripheral and non-musical aspects of one's instrument--such as the size and weight of a controller and power/voltage issues--in reality often have great impact on how, when, and where performances can happen.

4 Further Plans

The authors are very interested to continue developing musical dialogue between violin and computer(s), and are currently in development of a new multimedia theater work inspired by our experience with *Gilgamesh*. Regarding the instrument itself, the authors have discussed the musical advantages and disadvantages of the fact that, as the system is currently configured, the computer performer, not the violinist, has complete control of the processed violin sound. With that in mind, we plan to create a version of *Hop* that can be controlled via a MIDI foot pedal board so that the violinist too can choose how to process her audio signal. Ideally, we would employ a second computer for this task. We also would like to improve the subtlety and flexibility of the performance-tracking elements of the instrument, while retaining the ability to map them to various audio parameters.

Concerning the problem of the complex interface, one solution may be the development of meta-controls that would act upon numerous module parameters simultaneously and systematically. This is being considered, but would seem to require a considerable amount of programming and may reduce the system's current easy reconfigurability.

Finally, the authors are considering a move to a radically different control interface, possibly created ourselves or in

collaboration with an instrument designer. We are concerned though that any new control be at least as robust as the current one and that it provide a variety of options so that the software instruments may be accessed to serve spontaneous creative intuition onstage.

References

- Kimura, Mari. 1995. "Performance Practice in Computer Music", *Computer Music Journal*, 19(1): 64-75.
- Lee, M. A., and D. Wessel. 1992. "Connectionist Models for Real-Time Control of Synthesis and Compositional Algorithms." *Proceedings of the 1992 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 277-280.
- Lippe, Cort. 1997. "Real-Time Interactive Digital Signal Processing: A View of Computer Music." *Computer Music Journal* 20(4): 21-24.
- Puckette, Miller S., Theodore Apel, and David D. Zicarelli, 1998. "Real-Time Audio Analysis Tools for PD and MSP." In *Proceedings of the 1998 International Computer Music Conference*, ed. M. Simoni. San Francisco: International Computer Music Association.
- Wessel, D. 2002 "Live interactive computer music performance practice." *Journal of the Acoustical Society of America*, Vol III, Issue 5, pp. 2348-2348.