Code As Performance Interface - A Case Study

Alberto de Campo, Adrian Vacca
Les Duchamps / goto10
Graz, Austria
Amsterdam, Holland
adc@inode.at, vacca@goto10.org

Hannes Hoelzl, Echo Ho
earweego
Cologne, Germany
Beijing, China
ear@earweego.net, eskimotion6@gmail.com

Julian Rohrhuber, Renate Wieser
Academy of Fine Arts
Hamburg, Germany
rohrhuber@uni-hamburg.de, re.nate@web.de

ABSTRACT

This paper discusses different notions of what styles of interaction different electronic music (or more recently, computer music) instruments afford to performers, and considers in detail the case of code as the primary performance interface. Rather than attempting a full taxonomy of existing practices, we describe in detail the strategies developed in the ensemble we play in, *powerbooks unplugged* [7].

One concern addressed is the question of who is interested in which decisions to be made in a musical performance; composers, performers, and audience may have diverging interests. While code interfaces tend to reduce conventionally theatrical aspects of music performance (the lack of which in 'laptop performance' some consider problematical), we contend that for audiences mainly concerned with musical questions, plenty of interesting artistic decisions are made live, along with new forms of communication which we consider valid objects of interest. In fact, we find that Finally, shared code reduces the experiential distance between musicians and the audience: Much of what we do when performing is listening, guessing what might come next, and being joyfully disappointed.

A performance at NIME by *powerbooks unplugged*, using code as interface in the specific ways described, may serve to demonstrate the issues discussed in this paper.

Keywords

Musical Interfaces, Just In Time Programming, Live Coding, Realtime Open Source Improvisation

1. INTRODUCTION

Many different approaches to performing have been taken in the history of electronic music so far:

Electronic instruments closely modeled on acoustic instruments (e.g. MIDI pianos), using similar physical interaction (e.g. the Ondes Martenot), or - more distantly - touchless sensing (Theremin); from a different angle, the 'diffusion' tradition, i.e. interpreting a fixed tape piece by re-spatialisation, typically with a mixing desk as interface (with especially early electronic and computer music falling into the tape music category); playback mechanisms for musical structures with added 'expression' controls[1]. STEIM has cultivated its own strong tradition, focusing on instrumental performance by developing idiosyncratic personalized instruments in close collaboration with the performers.

While using manual dexterity to its fullest potential for refinement, there is musical life outside motor skills as well. New interactive programming environments offer new possibilities of higher level, and highly computer-specific interfaces and interaction modes: One avenue we are exploring is that of (re-)writing text that describes how sound is to be generated (i.e. code) in real time. Such an approach unifies the tradition of algorithmic sound synthesis and composition with the idea of high-level, discourse-like interaction with an environment consisting of a text editor, an interpreted programming language and (not mandatory, but highly desirable) other participants.

Rather than discussing an elaborate taxonomy of these possibilities, we go into the practice we have developed in some detail. We hope to show in a performance at NIME how this style of artistic practice manifests itself musically.

The paper begins with a glance at common models of musical interfaces, discusses which decisions which participants find interesting to make in performance, then covers code as an interface from the practice of *powerbooks unplugged*.

2. COMMON MODELS OF COMPUTER MUSIC INSTRUMENTS

Two models of electronic music instruments are well established: the traditional (acoustic) instrumentalist model and the higher-level control 'conductor' model. In both cases, electronic and computer means characteritically allow for decoupling between physical control interface and sound generation.

2.1 The traditional instrumentalist model

In acoustically played music, a performer generates every sound by physical energy; in terms of control, a dense stream of gestures physically creates every detail of every musical event (in immediate interaction with the physics of the instrument played) and all its nuances.

[1]An interesting acoustic variant is the barrel organ, played with amazing expressivity by Pierre Charial.

Electronic music instruments can follow this model closely, and some do:

Classic examples are the Theremin and the Ondes Martenot, but many MIDI instruments adhere to the model very closely too. Such instruments allow for long-term learning, and attaining more and more refinement of the embodied fine motor skills relevant to music (in classical music terms, virtuosity). One can say that the performer is in full control, and that success depends completely on motor skills.

Such instruments can be discouraging for beginners, such as the violin, or inviting for easy exploration, such as the piano. For computer-based instruments, a violin-like steep initial learning curve does not seem a good idea; the historical rate of development of computers may make the instrument technically obsolete (require reimplementation on newer platforms) before the first performer has ever learned to play it well.

The large number of musicians and dedicated hobby players seems to indicate a human tendency to benign addiction to such motor skills; apart from artistic impulses, one explanation is of course the Flow nature [3] of the experience of playing music. An interesting connection to obsessive behaviour is discussed by Favilla and Cannon in [4], especially for idiosyncratic self-made electro-acoustic instruments.

Wright in [14] makes a very good case for this approach in electronic music: Musicians being highly trained to achieve micro-time articulation (the notion of grooves vs. quantized music, down randomizing functions to 'humanize timing') should also be able to attain this level of expression with electronic instruments. Especially for metric music (which Wright is very interested in), any human-computer interfaces for music require rather low latency (ideally below 10msec), and as little jitter as possible: one can adapt to some constant lag, but one cannot learn rhythmic subtlety if jitter destroys one's high-precision timing.

A major center of advancement for the performance of experimental electronic music has been the Studio for Electro-Instrumental Music (STEIM); Joel Ryan in [10] both reports on work STEIM has supported in this area, and argues a more general notion really well: Considering the advanced and highly refinable motor skills that are characteristic of the human species, the *performance* of electronic music benefits enormously from ways to connect 'finger space' to interesting sound spaces.

Similar arguments for high-resolution 'intimate control' also made by Wessel and Wright [13] holds just as well for higher-level control, as follows.

2.2 The Conductor Model

A second starting point for musical performance and control interfaces is the idea of letting a computer take care of the more 'mechanical tasks' of music performance (such as playing all the events in a fixed composition in the correct order), while giving the performer control of higher-level parameters, often considered to be responsible for 'expression'. The precedent in traditional music practice often given for this model is the conductor, who only globally shapes what the instrumentalists play.

One attraction to this model is that uses the computer as a labor-saving device; thus initial playing may well seem easy with such a system. We find that while this model has some ovious attractions, it also has problems:

The idea that a musical performance can be separated into 'just the mechanics' and 'just the expression' seems quite elusive, and runs counter to the experience of playing acoustic instruments well. More precisely, the fundamental problem is how to model expression: the entire range of spontaneous expressive behavior a performer/conductor might want to use would have to be programmatically formalized beforehand, in order to map an 'expression' recognized in the sensor data to the relevant 'expressive' performance parameters. If one suddenly has an idea that a specific passage should be realized differently, that must have been prepared, or it will be inaccessible.

Secondly, it is not easy to imagine ways to play together in a group with such an underlying model, it seems rather solipsistic, if not dictatorial in tendency.

Wessel and Wright [13] aruge that interfaces with sufficient 'control intimacy' for low-level articulation are also very useful for higher control, e.g. flying in spaces of musical processes; Ryan [Territory] addresses similar ideas, e.g. navigating landscapes of previous performances. Both argue for potentially unlimited learning of more and more advanced and subtle skills with such interfaces, i.e. allowing for the equivalent of virtuosity being attainable in the long run. Long phases of free exploration (the equivalent of babbling in speech acquisition) help to habituate such skills for a feeling of intimate embodied knowledge of the instrument.

2.3 Pragmatic Approaches

Many computer/electronic musicians end up using whatever is pragmatically simple: affordable controllers, from simple gamepads to MIDI faderboxes, keyboard/fader combinations; 'laptop artists' are considered problematic by some when they just sit still, and do only a few mouse-clicks, as the stereotype goes. This is allegedly 'boring' and 'confusing' for an audience; artists are exhorted to consider "visual/corporeal aspects ... to make the performance convincing for the audience" [11].

We consider this visual entertainment point of view to be mainly an extra-musical concern; before discussing text as interfaces, let us consider which intra-musical decisions artists may make during performances, and how audiences may relate to them.

3. DECISIONS IN A PERFORMANCE

In music played on acoustic (and electronic, if they follow the same model) instruments, a common factor is the physical level:

3.1 Physical Micro-decisions

A constant stream of nuanced detail is generated by hands, mouth, body; there is direct physical coupling between action and sound, and potentially a high correlation of visual and auditory experience. The flow experience performers may have may communicate well to an audience. However, there is often a tendency toward exaggeration for show purposes, histrionics by (rock) guitar solos, classical hero-pianists, star conductors, and somewhere along this the line, air guitar championships. [2]

3.2 Interpretation of a Given Work

[2]One is sometimes reminded of the quote: *If you can fake sincerity, you've got it made.* (often attributed to Frank Zappa, and many others.

On a more musical level of concern, a connoisseur audience is often interested in interpretive decisions:

A pianist in classical setting makes decisions on details that bring out the structure and the subjective emotional meaning of a piece; the 'text' of the composition itself is usually not touched. Even if the pianists hands are not seen, an audience can follow and appreciate these aspects quite well.

3.3 Improvisation

In many musical cultures, improvisation within given frameworks is common; this often includes inventing new melodies on a known (e.g. in Jazz, harmonic) background, showing virtuosity in fluency of musical ideas; forms of accompaniment may be just as sophisticated.

Depending on cultural context, conforming to a highly detailed 'mood' of a particular piece may be most desirable (e.g. in classical Indian music); in other contexts, it may be creative ways to drastically redefine a trivial known reference point (e.g. John Coltrane's reshaping of *My Favorite Things*).

3.4 Who decides what?

Generalising dangerously here, composers often display an interest that performers get to decide rather little (and some composers even have a reputation for that); independently minded musicians may want to make more decisions themselves, some in preparation, others in the performance situation; audiences may or may not care how these decisions happen, depending on their level of knowledge of the music being played.

For well-informed audiences, a better question may be:

3.5 Which aspects of live music are interesting to follow?

Answers to this question will be highly culture-dependent; imagine the answers you might get from audiences of a Western classical solo recital; a traditional jazz concert; a free improv concert; a boy group or a Death Metal concert; a New Music concert; an academic electronic music concert; a noise artist club gig; a Gamelan performance in a Javanese village; and so on.

Here are some terms we would expect to come up in audience answers:

Causality (who does/did what), [8]

Communication (who reacts to what),

Expectation/Surprise (where is the music going),

and one can imagine others.

Note that these topical answers are not necessarily dependent on the presence of visually readable physical instruments and associated gestures.

4. CODE AS INTERFACE

The organisation *toplap* [12] has formulated a manifesto of live coding; and some of its members are exploring the idea of live coding as musical performance, to the point of experimenting with regular practice disciplines (Collins, Olofsson, Mme Bourbaki, and others). [see paper submitted by Collins at NIME2007]

The authors play in the ensemble *powerbooks unplugged*, whose overall concept developed from a few admittedly idiosyncratic tenets, formulated as follows:

4.1 PBUP Latento

The laptop is the next guitar, i.e. *the* new folk instrument

The laptop is a complete instrument as is

The laptop can also be the entire interface

The laptop in its current physicality is best used while it historically exists - now [3]

Code and music belong to everyone

So far, we have found these constraints both liberating and challenging. While not ruling out more physical interactions with laptops, for this paper we focus on code as the performance interface of choice.

The music we play is highly communicative group improvisation, based on a body of code created in rehearsals and concerts (collected later), rewritten on the fly, and communicated back and forth continuously.

We describe here strategies that work well for us; other live coding practitioners work with different concepts (slub [6], klippAV [1, 2], and others [12]

4.2 Contextual Considerations

4.2.1 Staging causality

The complaint that laptop music is boring shows that the stage (with its actors) serves as an interface between music and audience, which enhances focus. The physical actions of single musicians serve as easy to read cues. The absence of synchronous physical gestures in algorithmic music could be addressed by projecting code; however, this is on a different level than activity cues. In fact we (players) all listen to the algorithm like the audience; by contrast, the stage represents a producer-consumer and performance-perception split.

4.2.2 Audience Immersion

We address this by sitting in the audience space: the empty stage is a visual cue emphasizing the performers' immersion in the audience. The activity we perform is well-known (typing), and audience and performers share the same bodily posture. The softness of the sounds avoids causing feelings of forced immersion by acoustic power. The physical proximity and peculiar weaknesses of the laptop loudspeakers emphasize the physical reality of the instrument; this is often more difficult with instruments amplified through loudspeaker systems.

4.2.3 Network and Sharing

Since all members can create sound structures that are spread over any subset of the laptops used, they are delocalized actors. It is not essential to know who does what; rather it is all about listening to and understanding the subtleties of algorithms for everyone. Competence and performance are delocalized as well; they live in the space between machines, audience and performers.

[3]Who knows what portable computers may be like in 20 years?

4.2.4 Fear of Code

Code is still a problem for many people; they seem to prefer distance from it. This may be based on potential embarrassment for not being competent at this particular 'specialist knowledge'; often it may also be a variant of the cliche that intellectual capacities are seen as opposed to perceptual and emotional capacities. Coding as a casual thing to do in the audience space plays with this barrier, and may raise interesting questions to consider.

4.3 Just In Time Programming - JITLib

Given our platform of choice SuperCollider3 [5], it is very easy to write short scripts that create brief sound phrases or textures (often on the order of 10 to 30 seconds long). The library JITLib that comes with SC3 provides a number of elegant constructs for this style (see [9] and the JITLib documentation). While such a phrase evolves, one can listen to it, read its code, make changes to it, and replace it with a new variant at an appropriate moment.[4]

Some examples with code follow; these are only meant to convey the flavor and flow of this performance style; full explanations would exceed the scope of this paper.

4.4 Granular textures

Streams of events in time can be expressed very generally in SC3 with the Tdef (Task definition) construct, which keeps tasks organized by name; e.g. a granular texture:

```
(	// hatRit
Tdef(\hatRit, {
	var freqr = rrand(2000, 12000);
	100.do({ arg i;
		b.addresses.choose.sendMsg("/s_new",
			"hatXLine", -1, 0, 0,
			\freq, freqr * rrand(0.5, 1.5),
			\sustain, 0.03,
			\amp, 10,
			\rq, 0.006
		);
		(0.025 * (1.02 ** i)).wait;
	});
}).play;
)
```

Defining a variant at the same name replaces the old task,

```
(	// hatRit
Tdef(\hatRit).set(\curve, 1.02);
Tdef(\hatRit).set(\rq, 0.03);
Tdef(\hatRit, { |e|
	var freqr = rrand(2000, 12000);
	100.do({ arg i;
		b.sendMsg("/s_new",
			"hatXLine", -1, 0, 0,
			\freq, freqr * rrand(0.5, 1.5),
			\sustain, 0.03,
			\amp, 10,
			\rq, e.rq ? 0.006
		);
		(0.025 * (e.curve ? 1.02 ** i)).wait;
	});
}).play;
)
```

[4]This can be seen as a continuation of motivic development as in western classical music, as a form of genetic algorithms with the performers' aesthetic preferences as fitness functions, or as multi-path looped version of the surrealist technique *cadavre exquis*.

And one can change variables while a task is running:

```
Tdef(\hatRit).set(\curve, 1.03).play;
```

4.5 Patterns

Keeping patterns around by name works very similar to Tdefs, with Pdefs (pattern definitions); except one can take full advantage of the rich library of algorithmic patterns available in SC3:

```
Pbindef(\melody,
	\instrument, \grainXLine,
	\degree, Pstutter(
		Pseq([2, 2, 3, 2], inf),
		Pseq([
			[1, 4, 6], [2, 5, 7], [4, 6, 8], [5, 7, 9],
			[4, 6, 8], [2, 5, 7], [2, 4, 6], [1, 3, 5]
		], inf)
	),
	\octave, 5,
	\root, 4,
	\amp, [0.01, 0.005, 0.01] * 2,
	\dur, Pseq([0.5, 0.25, 0.5, 0.5, 0.5, 0.5, 0.25, 0.5, 0.5], inf),
	\sustain, 0.2
).play;
```

4.6 Continuous Synthesis processes

Continuous sounds are written with nodeproxies, and as a convenience, these can be written as variables in a special environment called ProxySpace:

```
p = ProxySpace.push;
~snd.play;

~snd = { SinOsc.ar([400, 407] * 0.9, 0, 0.2) };
```

Changing the sound process fades out the old and fades in the new:

```
~snd = { SinOsc.ar([400, 437] * 0.9, 0, 0.2)
	* LFPulse.kr([1, 1.3]) };
```

Sounds can be 'plugged together' very much like analog synthesizer modules:

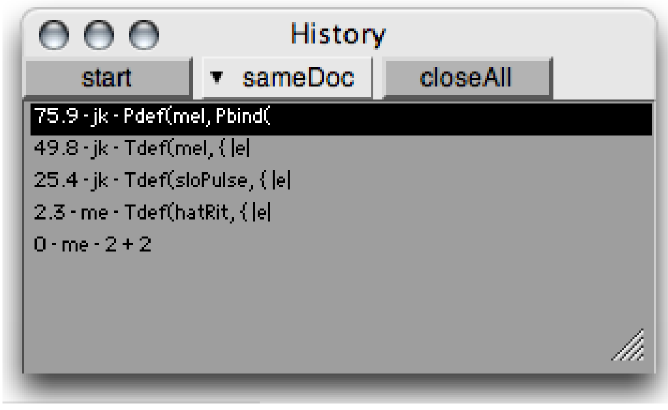
```
~ctl = { LFPulse.kr([1, 1.3] * MouseX.kr(1, 30, 1)) };
~snd = { SinOsc.ar([400, 437] * 0.9, 0, 0.2) * ~ctl };
```

4.7 Well-tempered Randomization

Given the need for fast decisions in performance, the rich implementation of randomness in SC3 is very helpful: Using random numbers, patterns, decisions, collection methods, and unit generators, one can delay decisions by making random first passes, and subsequently fine-tuning sounds incrementally very easily.

4.8 History

Out of the desire to make the history of a performance accessible immediately for shared evolution of musical ideas, we created a History class. Once History has been started, it records every code strings that is executed on the machine, as well as code being sent from the machines of other players, and keeps them in timed order. Thus a complete script of a performance can be stored and replayed if desired. A simple GUI provides access to all code strings since History has started: Every line represents one code string, with a time tag since History began, an ID for the author/sender, and the first line of code.

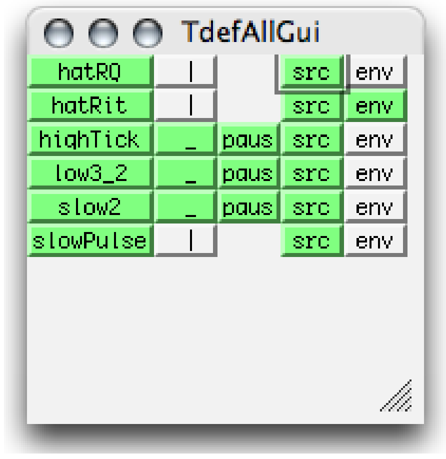


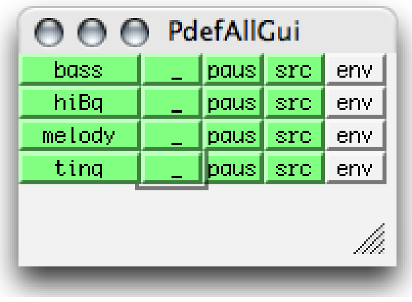
```
75.9 - jk - Pdef(mel, Pbind(
49.8 - jk - Tdef(mel, { |e|
25.4 - jk - Tdef(sloPulse, { |e|
2.3 - me - Tdef(hatRit, { |e|
0 - me - 2 + 2
```

Selecting an entry opens a text window and puts the code into it, so one can begin rewriting.

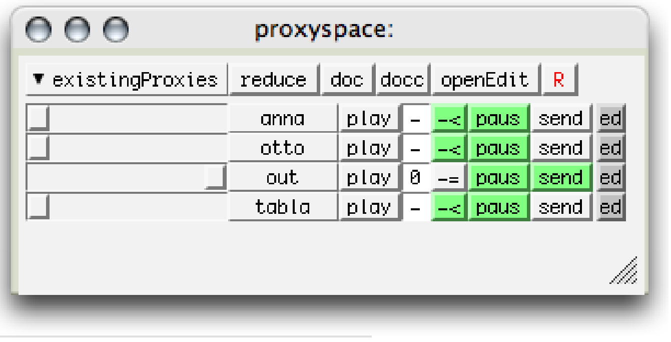
4.9 More Just In Time GUIs

Task and pattern definitions also have GUI support to reduce typing latency: TdefAllGui and PdefAllGui provide names and state information on all present Tdefs and Pdefs. They can be started and stopped by GUI, one sees whether a Tdef is empty or not [src], and whether it has an environment [env] for variable access or not. Clicking on these button opens a text window with the relevant code posted, ready for editing.

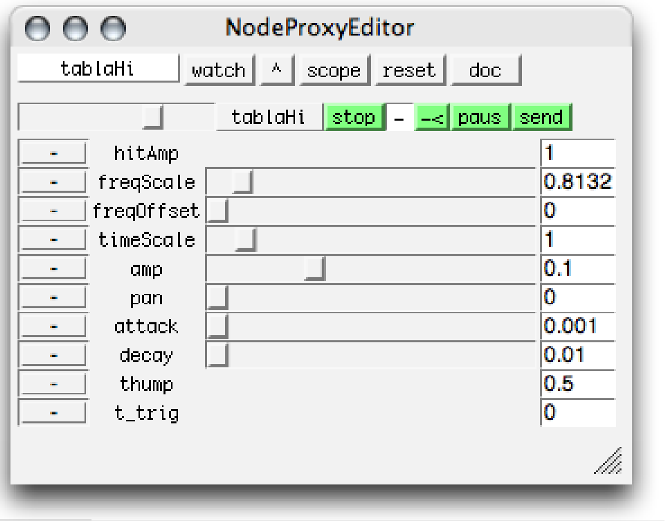




The state of a ProxySpace can be displayed with the ProxyMixer class, showing all current audio and control processes and their playing state, and making the most frequent adjustments available, e.g. start, stop, volume mixing, and output configuration of several processes:



Every individual process (a.k.a. NodeProxy) can be displayed and edited with a NodeProxyEditor, which automatically creates gui elements for all parameters with defined ranges:



hitAmp	1
freqScale	0.8132
freqOffset	0
timeScale	1
amp	0.1
pan	0
attack	0.001
decay	0.01
thump	0.5
t_trig	0

5. CONCLUSIONS

Considering laptops as full-blown musical instruments creates interesting new perspectives. Several current computer music languages offer a multitude of interaction possibilities well worth extended exploration, also for realtime performance. Especially extensible systems allow for creating support to reduce reaction times (i.e. typing latency), while still keeping full access to every corner of the current state open.

Choosing code as an interface is an interesting constraint, and steers one toward choices one would not easily make with acoustic instruments. While high-resolution gestural input is certainly interesting in its own right, the communicative aspects of a running discourse via shared code, and the changes such an approach suggests for the performance situation, creates plenty of musically interesting objects of attention for audience and performers alike.

6. ACKNOWLEDGMENTS

We would like to thank James McCartney for writing SC3 and making it open source; all the generous contributors to SC3; the Academy for Media and Arts Cologne sponsoring the Warteraum series; and the Academy for Fine Arts Hamburg for sponsoring the symposium where toplap was founded.

7. REFERENCES

[1] N. Collins and F. Olofsson. klippav.org. http://klippAV.org , 2004-2007.

[2] N. Collins and F. Olofsson. klipp av: Live algorithmic splicing and audiovisual event capture. *Computer Music Journal*, 30(2):8–18, 2006.

[3] M. Czikszentmihalyi. *FLOW – The Psychology of Optimal Experience*. Harper Perennial, 1991.

[4] S. Favilla and J. Cannon. Fetish: Bent leather's palpable, visceral instruments and grainger. *Contemporary Music Review*, 25(1/2):107 117, 2006.

[5] J. McCartney. SuperCollider3. http://supercollider.sourceforge.net, 2003-2007.

[6] A. McLean and A. Ward. *http://slub.org/* .

[7] Powerbooks Unplugged. *http://pbup.goto10.org* .

[8] J. Rohrhuber and A. de Campo. Waiting and uncertainty in computer music networks. In *Proceedings of the ICMC 2004, Miami*, 2004.

[9] J. Rohrhuber and A. de Campo. Algorithms today - notes on language design for just in time programming. In *Proceedings of the ICMC 2005, Barcelona*, 2005.

[10] J. Ryan. Some remarks on musical instrument design at steim. *Contemporary Music Review*, 6(1):3–17, 1991. also available online: *http://www.steim.org/steim/texts.phtml?id=3*.

[11] W. Schloss. Using contemporary technology in live performance: The dilemma of the performer. *Journal of New Music Research*, 32:239 – 242, 2003.

[12] Various. Toplap - (temporary...) organisation for the (proliferation...) of live (art...) programming (note multiple acronym expansions). *http://toplap.org* .

[13] D. Wessel and M. Wright. Problems and prospects for intimate musical control of computers. *Computer Music Journal*, 26:11 – 22, 2000.

[14] M. Wright. Problems and prospects for intimate and satisfying sensor-based control of computer sound. In *Proceedings of SIMS 2002, Santa Barbara*, 2002.