extramuros: making music in a browser-based, language-neutral collaborative live coding environment

David Ogborn
McMaster University
ogbornd@mcmaster.ca

Eldad Tsabary
Concordia University
eldad.tsabary@concordia.ca

Ian Jarvis
McMaster University
lostinthegroove@gmail.com

Alexandra Cárdenas
University of the Arts in Berlin
tiemposdelruido@gmail.com

Alex McLean
University of Leeds
alex.mclean@icsrim.org.uk

ABSTRACT

The extramuros software was developed to explore live coding and network music, bringing live coding musicians together around shared text buffers. Originally developed to support a globally distributed live coding ensemble, the extramuros software has found additional application in projecting laptop orchestra performances to remote sites, in zero-installation workshop and performance settings, and in facilitating the efficient display of code by an ensemble. As the software works by connecting shared text buffers to audio programming languages through specific network connections augmented by pipes, it is a language-neutral approach. This paper describes the overall architecture of the extramuros system, relating that architecture to perennial network music concerns for bandwidth, security, and synchronization. Examples of early use in workshops, rehearsals and performances by laptop orchestras and other small telematic ensembles are discussed, leading to a concluding discussion of directions for future work.

1. Introduction

Music is deeply collaborative, and network music (Bischoff, Gold, and Horton 1978; Kim-Boyle 2009; Fields 2012) wherein musicians of different types and genres collaborate (usually but not always over the Internet) is a burgeoning field of research. The main currents of research include a focus on extending traditional musical performance over the network (Alexandraki and Akoumianakis 2010), as well as exploring the network as an inherent part of a "new" instrument (Gremo 2012; Cáceres and Renaud 2008). Live coding can also be deeply collaborative. Code is a highly efficient means of communicating intentions, with efficiency potentially understood as any or all of efficiency in comprehension by a human reader, efficiency in transmission over a network, or efficiency in translation into a sounding result. The extramuros software was developed to explore live coding and network music, bringing live coding musicians together around shared text editing interfaces ("shared buffers").

In the basic operation of extramuros, a server application is run on one computer. Some number of performers use a web browser to connect to the server and edit shared text, with each person's editing more or less immediately visible to all the others. At any computer where the sounding result of the code is desired (for example, all of the individual laptops in a globally distributed laptop ensemble), a client application is run that receives evaluated text from the server and sends it to the audio programming language of choice over standard UNIX-style pipes. Anyone can edit anyone else's code at any time, and all code is sent to all listening clients (and thus, potentially executed on audio programming languages connected to all listening clients).

As the extramuros system simply collects text that is piped to another application, it is "language-neutral". This neutrality is especially pertinent in the specific context of the live coding community, which uses a large and growing number of distinct languages, with individual artists and researchers bringing forward new languages continually (whether as ephemeral experiments or as more persistent pieces of community infrastructure). extramuros' language-neutrality allows it to be a site that brings together "partisans" of different languages, because their familiarity with the extramuros web-based interface can represent a bridging factor next to their unfamiliarity with a given language. The extramuros server does not evaluate or render any code, but rather just sends such text code back to the local clients that pipe it to an existing audio programming language. Thus, as new text-based languages enter use, extramuros will continue to be immediately useful with them. Both Tidal (McLean and Wiggins 2010),] and SuperCollider (McCartney 1998) have been used heavily with current versions of extramuros.

This paper discusses some diverse applications of extramuros, the system's software architecture and how that architecture responds to some of the perennial challenges of network music, and then presents some qualitative feedback from early workshop deployments. The paper concludes with a discussion of directions for future development of this, and related, systems.

2. Distributed Ensembles and Other Applications

extramuros was originally developed with the intent of allowing a distributed ensemble of live coding performers to rehearse, perform and learn together, connected by the Internet. It was first used in an ongoing series of "shared buffer" performances with the language Tidal (McLean and Wiggins 2010), and then subsequently used in the first distributed algorave (Collins and McLean 2014) performance (Colour TV, a.k.a. Alexandra Cárdenas and Ashley Sagar, at the 2014 Network Music Festival).

While supporting such globally distributed ensembles was the primary intent behind the creation of extramuros, the approach taken here supports a number of applications or situations that go beyond this, including "projected ensembles" and "zero installation ensembles". The software also allows for a comfortable relationship between large ensembles and the live coding practice of sharing one's screen.

2.1. Projected ensembles

While not the original target of development, extramuros can be used for the "projection" of an ensemble's performance to a second location. For example, a laptop orchestra (Trueman et al. 2006; Smallwood et al. 2008; Wang et al. 2008) whose performance practice involves routinely playing together in the same room, can perform through the web browser interface, with the sounding result of that performance rendered both at their location and at another site, or sites.

The Cybernetic Orchestra at McMaster University (Ogborn 2012b; Ogborn 2014) employed this technique to participate in the 2014 Network Music Festival in Birmingham, UK. The orchestra performed a live-coding roulette in SuperCollider on a circular array of 8 loudspeakers, "projecting" the code from a closed location in Hamilton, Canada to the public festival site in the UK. As only small pieces of text code were traveling from Canada to the UK, this projection of 8-channel audio plus high resolution video required almost no bandwidth at all.

2.2. Zero installation ensembles

An unanticipated benefit of the approach taken here was the utility of extramuros in workshop and educational settings where participants arrive with diverse laptops lacking the specific audio programming environments used in the activities. Because, in such settings, people are together in a room, only a single computer needs to have an appropriately configured audio programming environment (and hardware) with the other computers accessing this through the browser interface. Moreover, the utility of this zero-installation aspect of the approach is not limited to workshop and other "entry-level" settings, but extends to any collective performance setting where quick and robust installation is required.

The Cybernetic Orchestra has taken advantage of this zero-installation aspect in at least three distinct ways: (1) to give live coding workshops to members of the community, (2) as a way of getting new members of the orchestra live coding during their first minute at a rehearsal, and (3) as a mode of performing together in public. In early February 2015, the orchestra performed at the world's first ever algoskate, an event combining live-coded dance music as in an algorave with ice skating. The orchestra's 8 performers were distributed in a long line alongside a skating rink, with each performer placing their computer on top of a large 200-watt, 15-inch sound reinforcement style speaker. All but one of the computers were connected to a local area network via Cat6 Ethernet cables (run through the snow) with the one computer connected to the network via Wifi. Because the event was part of a busy winter festival (the city of Hamilton's WinterFest) this outdoor setup had to take place very quickly (in less than an hour). The fact that the members of the ensemble only had to open their computers and web browser freed their attention to run cables through the snow to speakers and switches. During the algoskate, one member of the ensemble put on their skates, connected to the orchestra's network via WiFi and performed Tidal live coding on their smartphone (with zero configuration) while skating around the rink, thus entering the history books as the first person to skate and live code at the same time!

2.3. "Show Us Your Screens"

In the above-mentioned algoskate, another benefit of the extramuros approach was on display (pun intended). The web browser interface provides a single, unified visual method of sharing code with the audience. While the classic TOPLAP

demand to "show us your screens" is easy enough to make in the case of a solo or duo performer, it becomes more fraught as the number of performers increase. When everyone is coding Tidal or SuperCollider through a shared web browser interface, this interface provides a single visual object that can be shared with the audience, with no requirement for elaborate or bandwidth-hungry network video compositing.

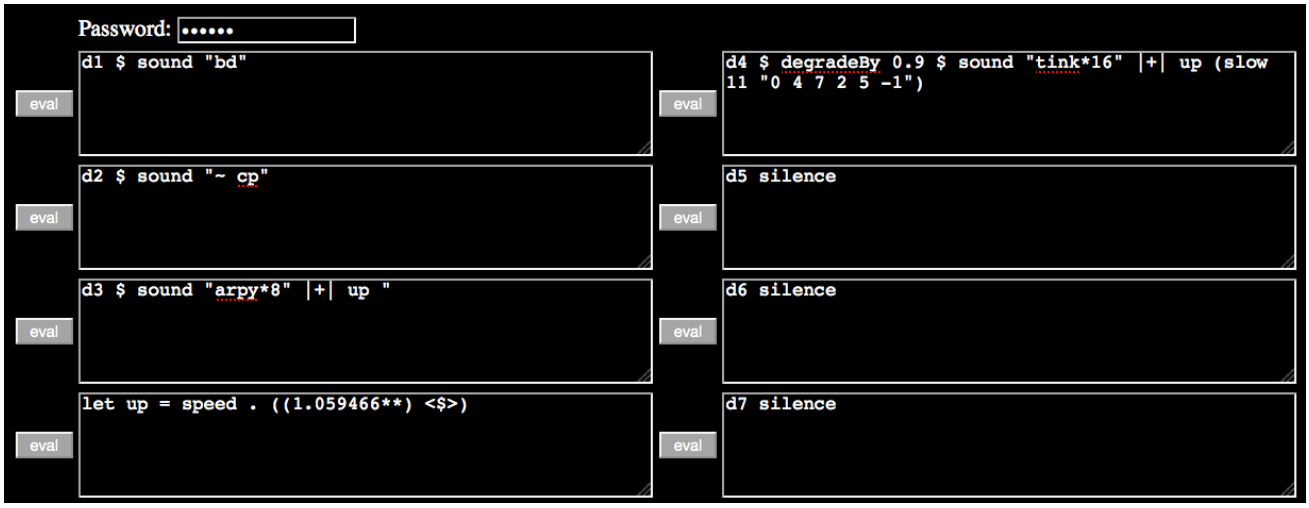

Figure 1: *A typical extramuros browser window with 8 text editing regions*

We can speculate that this type of application could have an impact on the evolution of live coding in large ensemble, "laptop orchestra" contexts. While many laptop orchestras are connected in some way with live coding, few have made it a central and defining activity. This could be in part because of the lack of an established practice of making a unified visual experience of live coding activity with so many people. While systems like Republic (Campo et al. 2012)), Utopia (Wilson et al. 2014), or EspGrid (Ogborn 2012a) allow performers in an ensemble to see what other performers are doing, copy their code, etc, they don't necessarily or inherently provide a unified visual focus and so performers tend to be in a situation of guessing what the other performers are doing (until closer investigation through their system's affordances provides the answer). Without special care, this becomes a situation of laptop performers closeted behind their screens, not fully co-present in the virtual space of the codework.

3. Network Music Challenges and Architecture

Performing collaboratively in networked contexts raises many challenges. The extramuros architecture responds to common network music challenges around security and timing/synchronization, while the fact that small pieces of text and operations on text are the primary pieces of transmitted information also makes the software quite economical in terms of bandwidth requirements. Before discussing the nature of these security and synchronization considerations, it will be helpful to outline the architecture of extramuros in slightly more detail. The extramuros server and clients are implemented in node.js. The share.js library (an operational transform library) is used for collaborative text editing, with common web browsers providing readily available virtual machines for the collaborative editing of text code. The 0mq library is used for reliable and scalable connections from the single server application back to the, potentially quite numerous, audio "render" clients. The rendering of code into audio results is out-sourced via UNIX pipes to external audio programming languages:

When a performer edits any of the editable text regions provided in the browser by the extramuros server, this involves back and forth communication between the browser and the server via websockets, mediated by the share.js library. This ongoing communication ensures that all browsers who have that page open see the current state of all the editable windows, and can potentially intervene and change them. When a performer clicks an "evaluation" button next to any given editable region, this triggers an HTTP request to the server application, which if successful, triggers the output of text in the client application that is then piped to some audio programming language.

3.1. Security

Information security issues in network music come in two forms: sometimes common security measures represent obstacles to the performance of network music, while at other times it is required that network musicians implement additional security measures in order to protect their performances.

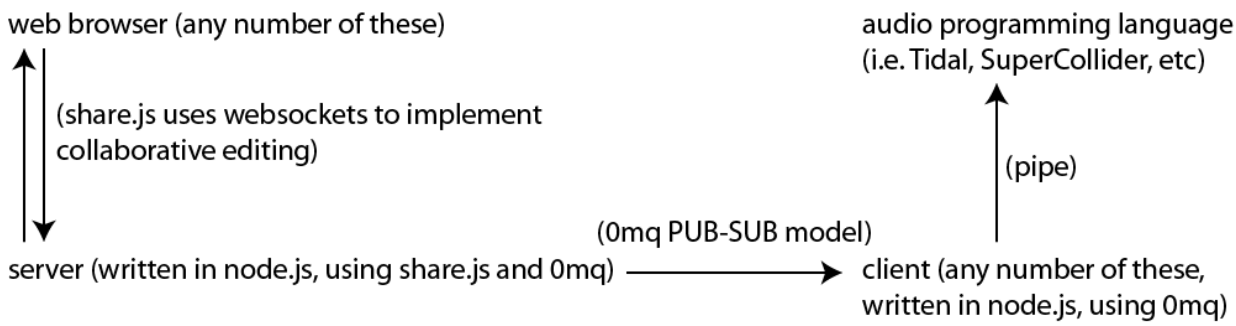

Figure 2: *Basic extramuros architecture diagram showing direction of data flow during editing and evaluation*

Firewalls are a common obstacle in network music performances, as the latter often take place in institutional or domestic contexts where access to the configuration of the firewall is prohibitive. In the case of extramuros, whose server listens for incoming TCP connections, firewalls are rarely a problem when the server application is run at publicly reachable location: all of the connections from other points/hosts to the server are "outbound" from the perspective of those other points/hosts, and firewalls tend to be configured to allow outbound connections on arbitrary ports without special reconfiguration. Provided a copy of the extramuros server is running somewhere, this allows potentially as many as all of the working/rendering/listening locations to be behind home or institutional firewalls. (At the first public "shared buffer" performance with extramuros, one of the performers was working on headphones at a café in a theme park, where one has to presume there would be no possibility of negotiating a momentary reconfiguration of the firewall!).

Use of extramuros entails open access to a web page requires some security measures in order to prevent sabotage. For example, on one occasion, an extramuros/tidal workshop was given to first year students at Concordia University in two groups. Once Group 2 was live coding, a mysterious code was hijacking the music with selfish, intrusive sounds, which were not generated by any of the participants. Apparently a mischievous student from Group 1 was having "some fun" from an unknown location. Identity still unknown, he/she apologized anonymously in the chat window.

Security is addressed primarily through a rudimentary password system. The web browser interface includes a field to enter a password. This password is included in the transmission back to the server whenever one of the buttons to evaluate code is pressed and the server quietly ignores any such evaluation requests that arrive without the correct password. If no password is entered, an in-browser alert pops up to remind the performer to enter a password, a measure suggested by early experiences, where reloading the browser window and forgetting to re-enter the password was a common mistake.

The current mechanisms do potentially open the door to security problems on the audio rendering/listening/client computers, and more attention to the security implications of the system are without a doubt an important direction for future work on the system. A part of the "solution" may be to establish practices and guidelines in each of the languages used with the system so that languages are addressed in a "secure mode", i.e. with no access to the file system outside of a certain folder or set of user permissions.

3.2. Synchronization

In the broadest sense, synchronization (i.e. latency) in extramuros is addressed by the fact that the same code is rendered at each render location. Depending on the nature of the structures deployed, the result at each client location could be more or less identical. When it is a matter of specifying patterns (in other words, quantized information at a relatively low frequency) as in the case with typical uses of Tidal, in most cases the different locations will render the same thing. There is the possibility that a piece of code might arrive just before a cycle boundary (i.e. a quantization point) on a given client machine and just after the same boundary on another client machine, so that briefly people will hear different things at each location. However, in a live coding context it is likely that a new evaluation will arrive a moment later with a similar relationship to the cycle boundary and then the identity of the two listening experiences (at least as signals at the DAC, not of course as identical acoustic and psychoacoustic experiences) will converge again.

In other code situations, without a high-level pattern focus and quantization, it is possible for the signals produced to diverge more. Consider, for example, the following SuperCollider/JITlib situation:

```
~a = { SinOsc.ar(0.1) } // someone makes a 0.1 Hz LFO on all render machines
~b = { SinOsc.ar(0.5) } // then a moment later, another LFO on all render machines
~c = { ~a.ar * ~b.ar } // then, the product of the two LFOs
```

The shape of oscillator ~c in the code above is highly dependent on the specific amount of time that elapses between the evaluation of ~a and ~b. Depending on the use to which ~c is put, this could result in substantially different sonic results. Collective live coding with the current extramuros architecture might take account of this dynamic by avoiding those kind of interdependencies between synthesis nodes:

```
~c = { SinOsc.ar(0.1) * SinOsc.ar(0.5) } // should be very close on all machines
```

However, this represents a rather unnatural constraint, and suggests that fine-grained time synchronization mechanisms, like EspGrid, could be quite important to the future development of networked live coding. Triggering code evaluation in the browser could schedule a packet of code to be evaluated at a more precise moment in the short-term future, as there are numerous situations where the sonic result can be quite different depending on small differences in the time of execution of code:

```
~a = { SinOsc.ar(1000) }
~b = { SinOsc.ar(1000) }
~c = { ~a.ar + ~b.ar } //
```

In the example above, the spectral energy at 1000 Hz could go from doubled (effectively, energy of ~a plus energy of ~b) to completely phased out, depending on a mere 0.5 millisecond change in the time of execution of ~b relative to ~a. At 5000 Hz, a 100 microsecond difference in time of execution (4 samples at 44100 Hz) can either double or eliminate the energy. The implementation and use of sample-accurate networked synchronization methods thus has a special relevance to projects like extramuros, that mirror the execution of code to multiple points.

Precise mechanisms for the controlling the logical time of execution of code, however, will not exhaust the broader question of coordinating the state of distributed machines. When code is executed in an extramuros browser window that calls upon any form of pseudo-random generator, the behaviour of the multiple client languages in the extramuros network will diverge. Intuition suggests that this is the type of challenge (i.e. random processes distributed over network architecture) that may ultimately be addressed by new languages designed from the ground up around the idea of collaborative, networked editing.

4. Workshop Feedback

On several occasions, electroacoustic (EA) majors at Concordia University (Montréal) live-coded in Tidal with extramuros in pedagogical settings, including a first year EA studio class, a second year EA ear training class, and a Concordia Laptop Orchestra (CLOrk) session. In an anonymous survey conducted with CLOrk members afterwards, students commented that live coding in this setting was “logical” and “flexible”, that it produced “instantaneous results”, and that it had “almost limitless potential” for “collaborative, digital expression” and “possibilities for composing [that] are quite vast.” Students also commented in the survey that this experience was “fun” because it “produced sound right away,” but that their ability to express themselves was between “very limited” to “none at all”, due to their inexperience with text-based programming. A student noted, “It would take a lot of time to feel flexible or expressive.” (This response is, of course, a response not only to the extramuros interface but also to the initial experience of programming audio operations in a text-based environment.)

According to students’ survey responses, this minimal expressive ability also led to minimal interactive ability in their first extramuros/Tidal tryout. Nonetheless, the visibility of the shared code on every laptop screen succeeded in generating at least some interactive engagement, as evident in these students’ survey comments: “it was interesting to have ensemble code-visibility” and “viewing the code of others gave me ideas about what gaps to fill, or when to be quiet.” Commenting on musicality, students noted that the effortless synchronization “worked” and that it was “a big plus.” One student felt that “abrupt entrances were annoying,” which were likely due to the inability to hear the code until it is evaluated. The student proposed that “some type of auditioning options” could be useful.

The majority of the students commented favorably on the effects of this first experience on their interest in live coding. Comments ranged from simple approvals, such as “it’s pretty cool,” “it has great potential,” and “Yeah, I would totally try it again” to insightful realizations such as “Yes, I never had a special interest towards live coding, but now I realize the amount of focus and the interesting symbiotic relationship with code.” While acknowledging the practical advantages of the zero installation and quick access allowed by extramuros, students asked to be tutored through the installation process in order to be able to practice at home and be more prepared, expressive, and flexible in upcoming group sessions.

5. Conclusions and Future Work

A major limitation of the current system is that it provides little in the way of feedback on the success or failure of code evaluations, or other states of the audio rendering environment (for example, audio levels). To remedy this, a mechanism is being created whereby arbitrary Open Sound Control messages arriving at the (single) server application, will come back to all of the web browser environments as calls to a JavaScript stub function. This should allow the development of distributed error messages, visual monitoring of audio levels, as well as more sophisticated forms of visualization of collective live coding activity.

It is quite common in the community that currently exists around extramuros for informal communication, during rehearsals and performances, to take place in an IRC channel or other external chat mechanism, but these mechanisms are not always strongly visible/present to participants. While the extremely spartan nature of the software has helped it to come into existence and support a number of artistic outputs, in the longer term the facilitation of communication and co-presence is doubtless an area of the software where considerable improvements could be made. Depending on the performance context it may be desirable to have video, audio, and/or data streams from each of the collaborators (Akoumianakis et al. 2008).

The language-neutrality of extramuros, based on the use of text code, is a project strength, and enables it to be used with diverse live coding communities, and also potentially as a thread bridging those communities. At the same time, text code comes with all kinds of cognitive demands that could be lessened through the exploration of structure editing techniques. These could be incorporated into extramuros without sacrificing the underlying language-neutrality in the form of something like the Emacs "major modes" for given languages, with the various structure constraints and affordances all calculated in the browser, and plain text still distributed to the audio rendering/listening clients as in the present architecture.

5.1. Acknowledgements

Special thanks to Research and High Performance Computing Services at McMaster University, to the McMaster University Arts Research Board for supporting the project Live Coding and The Challenges of Digital Society (of which this work is a component), and to Ash Sagar, Holger Ballweg, Scott Wilson, Mike Hodnick, and all of the members of the Cybernetic Orchestra and the Concordia Laptop Orchestra for using and discussing extramuros. Kudos also to the creators and contributors of node.js, share.js and 0mq!

References

Akoumianakis, Demosthenes, George Vellis, Ioannis Milolidakis, Dimitrios Kotsalis, and Chrisoula Alexandraki. 2008. "Distributed Collective Practices in Collaborative Music Performance." In *Proceedings of the 3rd International Conference on Digital Interactive Media in Entertainment and Arts*, 368–375. ACM. http://dl.acm.org/citation.cfm?id=1413700.

Alexandraki, Chrisoula, and Demosthenes Akoumianakis. 2010. "Exploring New Perspectives in Network Music Performance: the Diamouses Framework." *Computer Music Journal* 34 (2): 66–83.

Bischoff, John, Rich Gold, and Jim Horton. 1978. "Music for an Interactive Network of Microcomputers." *Computer Music Journal* 2 (3) (Dec): 24. doi:10.2307/3679453.

Cáceres, Juan-Pablo, and Alain B. Renaud. 2008. "Playing the Network: the Use of Time Delays as Musical Devices." In *Proceedings of International Computer Music Conference*, 244–250. http://classes.berklee.edu/mbierylo/ICMC08/defevent/papers/cr1425.pdf.

Campo, A. de, Julian Rohrhuber, Hannes Hoelzl, Jankees van Kampen, and Renata Wieser. 2012. "Towards a Hyperdemocratic Style of Network Music." In *Paper Presented at the SuperCollider Symposium.*

Collins, Nick, and Alex McLean. 2014. "Algorave: Live Performance of Algorithmic Electronic Dance Music." In *Proceedings of the New Interfaces for Musical Expression Conference, London, UK*. http://nime2014.org/proceedings/papers/426_paper.pdf.

Fields, Kenneth. 2012. "Syneme: Live." *Organised Sound* 17 (01) (Apr): 86–95. doi:10.1017/S1355771811000549.

Gremo, Bruce P. 2012. "Tele-Media and Instrument Making." *Organised Sound* 17 (01) (Apr): 73–85. doi:10.1017/S1355771811000537.

Kim-Boyle, David. 2009. "Network Musics: Play, Engagement and the Democratization of Performance." *Contemporary Music Review* 28 (4-5): 363–375. doi:10.1080/07494460903422198.

McCartney, J. 1998. “Continued Evolution of the SuperCollider Real-Time Synthesis Environment.” In *Proceedings of the ICMC*. http://quod.lib.umich.edu/cgi/p/pod/dod-idx/continued-evolution-of-the-supercollider-real-time-synthesis.pdf?c=icmc;idno=bbp2372.1998.262.

McLean, Alex, and Geraint Wiggins. 2010. “Tidal—Pattern Language for the Live Coding of Music.” In *Proceedings of the 7th Sound and Music Computing Conference*. Vol. 2010. http://server.smcnetwork.org/files/proceedings/2010/39.pdf.

Ogborn, David. 2012a. “EspGrid: a Protocol for Participatory Electronic Ensemble Performance.” In *Audio Engineering Society Convention 133*. Audio Engineering Society. http://www.aes.org/e-lib/browse.cfm?elib=16625.

———. 2012b. “Composing for a Networked, Pulse-Based, Laptop Orchestra.” *Organised Sound* 17 (01) (Apr): 56–61. doi:10.1017/S1355771811000513.

———. 2014. “Live Coding in a Scalable, Participatory Laptop Orchestra.” *Computer Music Journal* 38 (1): 17–30.

Smallwood, Scott, Dan Trueman, Perry R. Cook, and Ge Wang. 2008. “Composing for Laptop Orchestra.” *Computer Music Journal* 32 (1): 9–25.

Trueman, Daniel, Perry Cook, Scott Smallwood, and Ge Wang. 2006. “PLOrk: the Princeton Laptop Orchestra, Year 1.” In *Proceedings of the International Computer Music Conference*, 443–450. http://www.scott-smallwood.com/pdf/plork_icmc2006.pdf.

Wang, Ge, Dan Trueman, Scott Smallwood, and Perry R. Cook. 2008. “The Laptop Orchestra as Classroom.” *Computer Music Journal* 32 (1): 26–37.

Wilson, Scott, Norah Lorway, Rosalyn Coull, Konstantinos Vasilakos, and Tim Moyers. 2014. “Free as in BEER: Some Explorations into Structured Improvisation Using Networked Live-Coding Systems.” *Computer Music Journal* 38 (1): 54–64.