Live Coding the Audience Participation

Sang Won Lee
University of Michigan
snaglee@umich.edu

Gerog Essl
University of Michigan
gessl@umich.edu

ABSTRACT

In this paper, we discuss live coding in the context of audience participation performances. A live-coder can modify the distributed musical instruments that large numbers of audience members use by sending executable code text to each participant using cloud-based messaging mechanisms. We discuss how this idea was realized as part of the web-based mobile audience participation piece *Crowd in C[loud]* (Lee, Carvalho Jr, and Essl 2016). We introduce basic probabilistic schemes to allow sectioning off large scale audiences through live coding. While the demonstrated realization of this idea is simple, we believe that not only this structure can support a wide range of large-scale audience participation pieces under the control of live coding performers but also the concept of live coding the array of multiple machines can push the boundary of live coding music.

1 Introduction

The combination of live-coding and audience participation offers a novel and exciting playground for rich artistic expression. In this paper we introduce this concept from the example of a concrete piece called *Crowd in C[loud]* where a live coder interactively codes the large-scale structure and evolution of an audience participation piece using mobile devices through web technology, leveraging a cloud service for messaging. The piece distributes a simple and limited musical instrument to an audience, and they can play it in a web browser that supports Web Audio API. We then discuss how the performer of the piece uses live coding to improvise on the crowd-powered musical instruments. We stretch our previous work of live coding the mobile music instrument (Lee and Essl 2013) in which live coding musician(s) changing the sound, the interface and mapping of the musical instrument on the fly by sending the code text over a wireless network. This work is a scaled version of the previous work where a programmer can change the instruments that are being played by participants on the fly to shape the music. We briefly review the related works of audience participation and live coding and introduce how we live-coded the audience participation piece.

2 Audience Participation and Mobile Phones.

It has been a long-standing endeavor for musicians to create musical performances which involve the audience as part of the music making process. For example, in popular music, audiences also often participate by making sounds directly such as singing, clapping, or stomping feet (e.g. *We will rock you* by Queen). In Jean Hasse's *Moths*, the audience was instructed to whistle along to a conductor's gestures and a graphical score (Hasse 1986). In contemporary computer music, additional forms of audience participation have been explored. For example, the audience can play the role of a composer who influences the piece on stage collectively (e.g. voting, averaging). Another example would be that separate groups of instrumentalists play music that influenced by the live audience. Freeman's works fall into this category, where the outcome of participation is a real-time music notation (Freeman 2008; Freeman and Godfrey 2010). In these pieces, the audience influences the composition indirectly, rather than generates sound directly. Having audience as a composer requires one or more intermediate steps to structure the diversity of audience input and to make the resulting music coherent to the composition. This approach makes the audience comfortable in participating in the music performance without any musical background.

The emergence of mobile smartphones has made audience participation easier. Audience members already have participatory technology in their possession which offers networking and rich sensor capabilities. Levin (2001)'s *Dialtones* used mobile network dial-up to let a concert hall filled with ringtones of cell phones that the audience has, allowing the audience members to indirectly participate in the performance. As smartphones offer greater computational power with network capability, numerous projects have contributed to building mobile-based infrastructure to support smartphone-based audience participation, such as *TweetDreams* (Dahl, Herrera, and Wilkerson 2011), *massMobile* (Weitzner et al. 2013) or *Swarmed* (Hindle 2013).

Mobile phones can be used for sound generation. Hence, mobile phones serve a role that is rather close to a traditional instrument, where the performance interface and the sounding mechanism are co-located. Hence, using smartphones audience members can participate as performers of mobile phone instruments. The author's previous work, *echobo* exemplifies this approach where the sound of the piece is coming from the audience seats (Lee and Freeman 2013a). This participation model has the advantages that each participant has individual outcome coming from their hands so that it is clear how their participation is contributing to the music.

3 Live Coding Something Else: The Audience Participation

A long-standing research program of the authors is the expansion of the way of live coding is utilized. Traditionally, live coding of music is concerned with writing code that generates sound on the fly. In a series of projects, we attempted to apply the idea of live coding to "something else" other than generative music. We proposed live patching on a touchscreen of a smartphone (Essl 2010), live coding the real-time notation (Lee and Freeman 2013b), live coding the musical instrument (Lee and Essl 2013), live sketching the graphical user interface (Yang and Essl 2015) and live writing (instead of programming) (Lee and Essl 2015; Lee, Essl, and Martinez 2016).

In particular, we introduced a live coding performance practice where an instrument performer played a mobile music instrument while the mobile music instrument application was being live coded on the fly by on-stage programmers (Lee and Essl 2013). In this case, the outcome of live coding is not generative music, but a dynamically changing mobile phone instrument that is performed at the same time by a mobile phone musician. The sonic result of the performance can be drastically different from typical live coding music given that the mobile phone musician can inject immediate expressive gestures into the performance. A related performance idea uses a glove-based interface (Baalman 2015). The solo performance engages both in live coding sound synthesis and dynamically developing a mapping of the glove-based instrument while performing the instrument by hand gestures. Combining ideas from live coding the NIME, we realized an audience participation music piece. Or the existing work has been expanded to the scale of audience. We already proposed earlier that scaling this kind of performance practice to a larger number of participants could be an exciting next step (Lee and Essl 2014). The project discussed in this paper is a direct outgrowth of tackling this challenge. The technical detail of how the performer changes the scale will be discussed in the later sections.

4 *Crowd in C[loud]*: Audience Playing Largely in C

We summarize the audience participation piece, *Crowd in C[loud]*, which is a networked music piece composed and developed for audience participation at a music concert (For more detailed motivation regarding the aesthetic of the piece, see (Lee, Carvalho Jr, and Essl 2016)). It draws on the idea of the piece, *In C* by Terry Riley, where musicians (with various instruments) were guided to play pre-composed melodic variations based around the C chord (Riley 1964). In *Crowd in C[loud]*, each participant uses web browsers (typically on their smartphones) that support Web Audio API. They are instructed to play a short snippet (or tune) composed by other audience members (including themselves) for a random amount of time, which follows the basic concept of *In C*. The aggregated result of each playing a short tune creates a heterophonic texture of chance, largely in centered around the C chord. The mobile phones that audience members use are connected via a wireless network using a cloud service (PubNub[1]) to a laptop that is controlled by an on-stage performer. For more detail regarding the network structure that utilizes a cloud service, see (Carvalho Jr, Lee, and Essl 2016).

Accessibility in audience participation is especially essential to motivate people to participate in the piece with clarity and musicality regardless of musical backgrounds. *Crowd in C[loud]* incorporates two design decisions to achieve this goal. First, the interaction design in the instrument is loop-based where a participant needs to place musical notes on the screen, and the pattern of five musical notes will create a tune that is looped indefinitely based on where the notes are. Thus a user does not need to make playing gesture constantly to generate tones and the instrument will produce sound automatically. This concept is useful for a live coder to change the instrument itself while the musician can ensure that the instruments are constantly being played. Besides, the musician need not worry about overall sound too sparse (or silent) due to low participation. Second, the piece uses the metaphor of online dating for browsing the composed tunes of others. A tune composed by a participant serve as a personal profile on an online dating website. At the beginning of the performance, once a participant finishes the composition, he or she can browse, and play, what other audience members have composed. Browsing tunes composed by others mimics an online-dating website (such as Tinder[2]) where a user creates a personal profile and then scans other member profiles that include pictures and written descriptions about themselves. The networked instrument creates a temporary social media that lasts until the end of

[1]http://www.pubnub.com
[2]http://www.gotinder.com

the performance where each tune is a musical profile of a participant. Also, the collection of each tune composed by individuals serves as musical phrases found in Riley's *In C*. The metaphor of online dating profiles promotes collective creativity as browsing other profiles inspires participants to enhance their compositions with new ideas, which may be musical or visual.

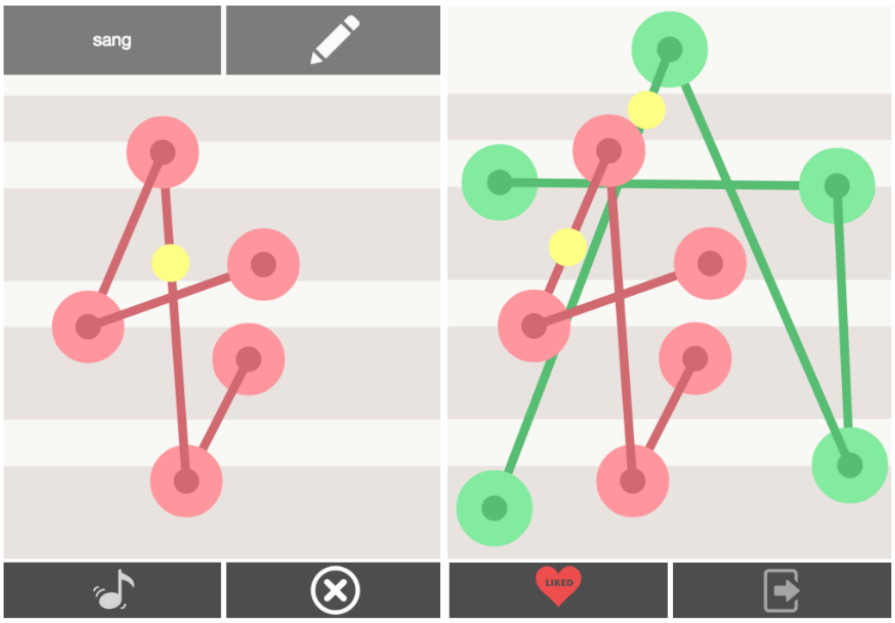

Figure 1: *Loop-based Instrument in Crowd in C[loud] (Left) Browsing a composition (or an musical profile)of a participant. (Right) Mingle Mode : playing two tunes at the same time*

The model of performance in *Crowd in C[loud]* is musician-audience pairing. In this model, there is a separate musician performing the piece on stage, at the same time with the audience members. The stage performer is a live coder who serves the role of a meta-performer who can control the progression of the music by changing the sound synthesis algorithm, the interface and the mapping of the distributed musical instruments. For example, the meta performer can alter the scale in which the instrument tuned to a different chord scale (e.g. C Minor, Pentatonic Scale) on the fly. This will be the chord progression of the music. In the meantime, this performer cannot generate sound at all on his/her end. Rather, the performer only controls the harmonic flow of the piece produced by the crowd. The performer's laptop and the audience's smartphones are connected via a cloud service, of which latency is typically around 100ms. The typical challenge of synchronizing clocks of connected machines for network music is less of its concern as the piece has no strict tempo.

We decided to live code to apply such changes to the audience's mobile musical instruments from the performer's laptop. While there can be multiple ways for the artist to control the instrument remotely such as GUI elements on a web page or MIDI Controller, live coding the audience participation expands the expressivity that the performer can have and enables algorithmic ways to control the large-scale audience.

We successfully presented *Crowd in C[loud]* in two public performances. The piece was, premiered at the University of Michigan's Mobile Phone Ensemble Concert 2015 and the presented at the Web Audio Conference 2016, Atlanta. The performance video footages of the latter are available online at the following links

Video 1: https://youtu.be/8nnrKJ4Ap0c

Video 2: https://youtu.be/gtg4p75-7wY

In the next section, we discuss the live coding aspect of *Crowd in C[loud]* in detail.

5 Live Coding the Audience Participation in *Crowd in C[loud]*

5.1 Code Text Transmitted over Cloud Service.

As briefly mentioned, the piece utilizes a cloud service for a performer to communicate with audience smartphones or vice versa. For example, if an audience member composed a tune, the data will be submitted to the performer's laptop by publishing a message with data to the particular channel ("performer" in this case) via the cloud service. Similarly, the live-coded text is transferred from the performer's laptop to the audience mobile phones using the cloud service. For example, the following message entered on a performer's web browser javascript console will send the code text to all the smartphones of the audience.

```
publishMessage("audience",{type:"script",script:"console.log('hello world')"});
```

The web application on the audience's side is written to take the value of the property with the key "script" and evaluate the string value whenever the message type is "script"; in this case, it will print `'hello world'` message in the console of audience mobile phones. The first argument of the function `publishMessage` represents the channel that the message should be published to, which is `audience` channel that the whole audience devices are subscribed to. Therefore, remote live coding is enabled as if the code text that the performer enters runs in the javascript console of the audience's web browser. For example, the instrument is initially tuned to C Major scale with the base note of middle C (midi:60), but the scale can be changed by sending code message since it is going to refer a global variable named `baseNote`.

```
publishMessage("audience",{type:"script",script:"baseNote = 72;"});
```

Note that this does not trigger any sound by itself, but after the code sent is evaluated in an audience's smartphones, all sound will have a base note of high C (midi: 72) from that point, which will change the overall sound of the music immediately. In this way, the performer not just uses audience's smartphone as an array of speakers to generate sound but enable interactivity of audience participation with the power of live coding. Another interesting global variable that we often utilized in *Crowd in C[loud]* is the `scale`. By default, it uses the major scale specified in a javascript array. The following codes will change the scale to minor scale or pentatonic scale respectively.

```
publishMessage("audience",{type:"script",script:"scale=[0,2,3,5,7,8,10,12];"});
publishMessage("audience",{type:"script",script:"scale=[0,2,4,7,9];"});
```

The combination of `baseNote` and `scale` will determine the chord scale that each audience member plays the instrument so that the performer can live-code in a way that overall aggregated texture sounds in one chord (or even one note when `scale=[0]`). These scale changes were the major musical gesture to structure the music and to allow a coherent progression of the piece throughout the performance.

One caveat here is that the performer needs to know what kinds of code text can quickly change the mapping of the musical instruments, hence necessitating rehearsal and preparation. While we believe this is still live coding, we realize that what can be live-coded should be understood in advance to the performance over the practice and rehearsals. The examples introduced above are simple but efficient for a performer to change the chord progression of the piece, sustaining the interaction scheme of the instrument for the audience. Potentially code text can be any javascript code that can be evaluated remotely and a performer will be able to run more interesting algorithms or interaction. Here's an example code:

```
setTimeout(function(){
  alert("Are you having fun?");
}, 5000 * Math.random());
```

This code snippet set a timeout for alerting a message with random interval (between 0 and 5 seconds). The code will create an alert message on the web page that audience plays and the way that the warning message will halt the synthesis of Web Audio API until it is cleared. This can create a fade-out effect when the alert message appears with the random interval and the fade-in effect as each participant clears the alert message. This is only one example that how live coding on the audience participation can be useful to make the participation engaging and interactive as well as to use it to shape the music, especially as a product of both live coding and user interaction. With enough preparation on the web application in advance, the on-the-fly interaction with the performer and the audience can be dynamic. This typing process can be minimized by having a separate code input editor that will be published to the audience automatically with a shortcut (e.g. `shift-enter`) as a typical live coding editor (instead of web browser console). In this setup, the performer can just type the value of `script` property (without `publishMessage` call), and the audience can better read the code on projection.

5.2 Orchestrating the Crowd using Live Coding.

Live coding the whole audience's musical instruments at the same time and in the same way, is not any different from live coding one musical instrument (Lee and Essl 2013). Embracing the large-scale of the group, it will be more exciting to have fine controls over as if the audience was an ensemble which plays various instruments. Using simple probability embedded in the code text, one can split the whole audience into groups. For example, consider the following code:

```
if ( Math.random() < 0.7)
{
    // 70% probability to enter this if-block.
    baseNote = 64; // F Major Code
    scale = [0,4,7,12];
}
```

If this code snippet was sent to an audience member's smartphones, the instrument would be in F Major Scale with the 70% probability (because `Math.random()` will generate a random number between 0 and 1). This is useful to apply a certain change to the portion of the audience. As largely, approximately 70% of the audience will run the if-block while the others will not, one can split the audience into two groups that play different tunes. The portion will be determined by the number that is inside the if-condition parenthesis (0.7 in the above example). This probability is one of the optional property in the `publishMessage` function so that any code text can have a certain probability of that code text being evaluated. For example, the following code can be used to have half of the audience to be silent by changing the value of the flag(`sound`) that is used in the instrument application to determine whether to run sound synthesis.

```
publishMessage("audience",{type:"script",script:"sound=false;",probability:0.5});
```

Once this kind of code with probability value runs, the whole audience will be split into two groups based on the state of the musical instrument, the one in which the code text is executed, and the other which the received code text is not evaluated. This is useful to be musically expressive in a way that a performer can assign certain musical roles to certain groups; one group playing background chord sound, the other group playing a monophonic melody. Besides, often, a performer wants to keep track of the specific group that is in a certain state to keep the group to run a series of code text. This cannot be done just by specifying the probability because it will keep choosing a new subset of the audience each time regardless of previous states. For example, suppose a musician runs code that changes the program to state `A` with 0.5 probability. The audience is then split into two groups, one group in state `A` and the other group still in the previous state (let's say `S`). If the liver coder runs another code `B` with 0.5 probability, there will be three groups: `A` (approx. 25%), `S` (approx.25%) and `B`(approx. 50%) as the group `B` will be randomly selected from the whole audience. In fact, what the performer actually wanted could be the group who was in state `A` to be in state `B` (50:50 of `B` and `S`). This can be done by randomly assigning a device a group in advance and then a performer can run code text only if the group matches. More specifically, one `publishMessage` can be used to split the audience into N groups in a certain ratio using the random number and create a variable that specifies a group number. See the following example.

```
if ( Math.random() < 0.3){
    groupNumber = 0;
}
else{
    groupNumber = 1;
}
```

As that particular variable is a global variable, the variable `groupNumber` can be later used t run code text for the devices that belongs to the group with simple if statement (e.g. `if (groupNumber == 0) { // run something }`). While using probability 0.3 will pick new devices each time, using `groupNumber` after that will allow the specific set of devices can only run the code text multiple times.

5.3 Challenges of Live Coding in Numerous Remote Machines

The way we employed live coding to perform an audience participation in *Crowd in C[loud]* is simple: code text is sent to smartphones and evaluated to make changes in the application that the audience is using at the moment. This approach was effective enough to successfully realize the piece in practice. However, this approach poses some challenges. One challenge for a live coder is that the code text sent is going to be evaluated in remote machines, not the one that the live coder is in control. The immediate problem is that the error will also be remote. Syntax error messages can be collected as responses from mobile phones and be presented to the live coding musician. In the case of semantic errors that do not necessarily give an error message, the dislocated sound may make it difficult to hear the error due to the collective nature of the sound that each smartphone makes. One practical choice a live coder can have is to "preview" the code result in one device dedicated for testing so that a live coder can monitor sound and see(or hear) potential errors.

More difficult challenges come from the scale of the code run. If a musician sends code text to hundreds of devices, there is no guarantee that the code will run without problems on all one hundred devices consistently. Many factors are not in a live coder's control, including browser compatibility, the difference in its versions, different form factors, the difference in the state coming from the individual user interactions, all of which can cause diverse results if not carefully written. While, typically, it is almost impossible for an audience participation music piece to have no flaw, it will be beneficial to design the system as inclusive as possible and to have measures to mediate the responses from the remote machines. At this moment, we do not have an easy solution to this other than a simple case of syntax errors by collecting error messages from the devices and present the list of unique error messages (with its count) to the live coder. However, it will be beneficial to develop the mediation strategy to give the live coder brief one summarized message that monitors remote machines so that one can understand any unwanted consequence of code run.

5.4 Opportunities Beyond Sending Code Text

The current idea is based on the performer-audience pairing model where audience playing the musical instruments and the musical instruments are live-coded by one performer. In this case, the collaboration between the audience and the musician is mediated with an interactive music application. However, it should be clear that the approach of live coding the musical instrument is not the only way to have the expressive power of programming language in the audience participation music. New models can be developed depending on the interaction scheme, the networked structure, the devices that the live algorithms run, and the sound source. Here we suggest a few performance concepts, drawing ideas from the existing works of live coding.

The most naive idea may be live coding text can be broadcasted to the mobile devices so that mobile phones will play generative music controlled by the live coder. In this case, the mobile phones are used as an array of speakers and the audience's participation will be minimal. The synchronization between devices may be necessary based on the style of music but often the composition is written to be tolerant to the asynchrony. Although not live coding was involved, the piece *Fields* well represent the style of music that can be accomplished with this model (Shaw, Piquemal, and Bowers 2015).

The second model that will be particularly exciting for the live coding communities is the shared editor where audience members can access to the editor and live-code together. This is exactly opposite to the current model in a way that code-text is from the audience and sound is coming from a central computer. In the performance of *Shared buffer*, the shared editor was open to the audience and some audience members were able to pick up the syntax quickly and made changes in the share text buffer(Ogborn et al. 2015). This model, however, will not scale well enough when the increase in the number of participants can cause conflicts and errors. Also, it will be challenging to ask for the audience to live-code in general, outside the live coding communities.

Lastly, we believe the audience themselves can be the live-coded machines; to read the code text on the projection screen, to run the code and to perform the piece. This is the scaled version of the piece Encoding the Marimbist by (Magnusson and Eacott 2015) where a marimba player was asked to read the code on projection and interpret the code text and play as instructed. For example, maybe ICLC audience will be able to read the following code on the editor and perform a piece?

```
bpm = LISTEN_TO_THE_METRONOME;

while (true) {
    if(Math.random() < 0.5)
        clapYourHands();
    else
        stompYourFeets();
    beat(1); // proceed time by one beat
}

if( MyFirstNameContains([“g”, “s”]) )
  whistle.seq([60,62,67,69], [1/2]).loop(nearestBeat);
```

In this case, a live coder plays a role of a conductor and code text is the conducting gesture. The gesture does not make any directive sound by itself but it guides the participants to perform. Besides, each machine that interprets the code will inherently have randomness in an interpretation of the same code due to various reasons: the ambiguity of the code, malicious behaviors of the participants, social barriers in doing something in public, and mob psychology.

6 Conclusion

In this paper, we introduced a new live coding performance practice to perform an audience participation music piece, *Crowd in C[loud].* The way we accomplish the piece includes an onstage performer live coding the instruments that the audience is playing. Both the style of programming and music are drastically different from what one would expect in a live coding music. However, it takes the full expressivity of programming language can take and shares the aesthetic of live-algorithmic manipulation of the control. We wish that this idea of live coding the audience participation will keep challenging the live coding community to push the boundary of what live coding does.

References

Baalman, Marije. 2015. "Embodiment of Code." In *Proceedings of the First International Conference on Live Coding*, 35–40. ICSRiM, University of Leeds. doi:10.5281/zenodo.18748.

Carvalho Jr, Antonio Deusany de, Sang Won Lee, and Georg Essl. 2016. "Understanding Cloud Service in the Audience Participation Music Performance of Crowd in c[loud]." *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)* 1001: 48109–42121.

Dahl, Luke, Jorge Herrera, and Carr Wilkerson. 2011. "TweetDreams: Making Music with the Audience and the World Using Real-Time Twitter Data." In *NIME*, 272–75. Citeseer.

Essl, Georg. 2010. "UrMus-an Environment for Mobile Instrument Design and Performance." In *In Proceedings of the International Conference on New Interfaces for Musical Expression (NIME).* Citeseer.

Freeman, Jason. 2008. "Glimmer: Creating New Connections." In *Transdisciplinary Digital Art. Sound, Vision and the New Screen*, 270–83. Springer.

Freeman, Jason, and Mark Godfrey. 2010. "Creative Collaboration Between Audiences and Musicians in Flock." *Digital Creativity* 21 (2). Taylor & Francis: 85–99.

Hasse, Jean. 1986. "Moths." *Visible Music, Euclid, OH.*

Hindle, Abram. 2013. "SWARMED: Captive Portals, Mobile Devices, and Audience Participation in Multi-User Music Performance." In *In Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, 174–79.

Lee, Sang Won, and Georg Essl. 2013. "Live Coding the Mobile Music Instrument." *Proceedings of the International Conference on New Interfaces for Musical Expression* 1001: 48109–42121.

———. 2014. "Models and Opportunities for Networked Live Coding." In *Live Coding and Collaboration Symposium*, 1001:48109–42121.

———. 2015. "Live Writing: Asynchronous Playback of Live Coding and Writing." In *Proceedings of the First International Conference on Live Coding*, 74–82. ICSRiM, University of Leeds. doi:10.5281/zenodo.19322.

Lee, Sang Won, and Jason Freeman. 2013a. "Echobo: A Mobile Music Instrument Designed for Audience to Play." *Proceedings of the International Conference on New Interfaces for Musical Expression* 1001: 48109–42121.

———. 2013b. "Real-Time Music Notation in Mixed Laptop–acoustic Ensembles." *Computer Music Journal* 37 (4). MIT Press: 24–36.

Lee, Sang Won, Antonio Deusany de Carvalho Jr, and Georg Essl. 2016. "Crowd in c [Loud]: Audience Participation Music with Online Dating Metaphor Using Cloud Service." In *Proceedings of the 2nd Web Audio Conference (WAC-2016), Atlanta.*

Lee, Sang Won, Georg Essl, and Mari Martinez. 2016. "Live Writing : Writing as a Real-Time Audiovisual Performance." *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)* 1001: 48109–42121.

Levin, Golan. 2001. "Dialtones-a Telesymphony."

Magnusson, Thor, and Greta Eacott. 2015. "Encoding the Marimbist." Performance.

Ogborn, David, Eldad Tsabary, Ian Jarvis, Alexandra Cárdenas, and Alex McLean. 2015. "Extramuros: Making Music in a Browser-Based, Language-Neutral Collaborative Live Coding Environment." In *Proceedings of the First International Conference on Live Coding*, 163–69. ICSRiM, University of Leeds. doi:10.5281/zenodo.19349.

Riley, Terry. 1964. "In c." Composition.

Shaw, Tim, Sébastien Piquemal, and John Bowers. 2015. "Fields: An Exploration into the Use of Mobile Devices as a Medium for Sound Diffusion." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, edited by Edgar Berdahl and Jesse Allison, 281–84. Baton Rouge, Louisiana, USA: Louisiana State University. http://www.nime.org/proceedings/2015/nime2015_196.pdf.

Weitzner, Nathan, Jason Freeman, Yan-Ling Chen, and Stephen Garrett. 2013. "MassMobile: Towards a Flexible Framework for Large-Scale Participatory Collaborations in Live Performances." *Organised Sound* 18 (01). Cambridge Univ Press: 30–42.

Yang, Qi, and Georg Essl. 2015. "Representation-Plurality in Multi-Touch Mobile Visual Programming for Music." In *In Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, 1001:48109–42121.