Crowd in C[loud] : Audience Participation Music with Online Dating Metaphor using Cloud Service

Sang Won Lee
Computer Science and Engineering
University of Michigan
2260 Hayward Ave
Ann Arbor, MI 48109-2121
snaglee@umich.edu

Antonio Deusany de Carvalho Junior
Universidade de São Paulo
Rua do Matão, 1010, São Paulo, SP, Brazil
dj@ime.usp.br

Georg Essl
Electrical Engineering & Computer Science and Music
University of Michigan
2260 Hayward Ave
Ann Arbor, MI 48109-2121
gessl@umich.edu

ABSTRACT

In this paper, we introduce *Crowd in C[loud]*, a networked music piece designed for audience participation at a music concert. We developed a networked musical instrument for the web browser where a casual smartphone user can play music as well as interact with other audience members. A participant composes a short tune with five notes and serving as a personal profile picture of each individual throughout the piece. The notion of musical profiles is used to form a social network that mimics an online-dating website. People browse the profiles of others, choose someone they like, and initiate interaction online and offline. We utilize a cloud service that helps build, without a server-side programming, a large-scale networked music ensemble on the web. This paper introduces the design choices for this distributed musical instrument. It describes details on how the crowd is orchestrated through the cloud service. We discuss how it facilitates mingling with one another. Finally we show how live coding is incorporated while maintaining the coherence of the piece. From rehearsal to actual performance, the crowd takes part in the process of producing the piece.

1. INTRODUCTION

Web Audio significantly lowers the level of complexity for creating networked interactive music application and cloud technologies allow for creating scaleable audience participation in this context. In this paper, we introduce a web-based audience participation implementation as used in the music piece, *Crowd in C[loud]*. A distributed musical instrument is implemented entirely for a web browser, enabling an audience to easily participate in music making with their smartphones. This web-based instrument is designed to encourage the audience to play music together and to interact with other audience members. Each participant composes a short musical tune that serves as a musical profile of that particular participant. Once a profile is submitted, they can browse other people's profiles and play a pattern they like forming pairs. These musical profiles serve a metaphor for online dating websites.

To realize the network capability of the musical application, we made use of a cloud service to exchange data. It allowed us to create a networked ensemble without the hassle of configuring and developing a server program. A computer mediator and performer can actively progress the music by orchestrating the crowd through live coding on the console of the web browser.

In this paper, we describe the inspiration for musical aesthetic of the piece, justify the design choices that we make, and introduce the technical details of the instrument. Lastly, we share our experience of the rehearsal process and describe the performance of the piece.

Web Audio Conference WAC-2016, April 4–6, 2016, Atlanta, USA.

2. BACKGROUND

Mobile smart phones are an attractive platform to enable audience participation in musical performance, as today it is sensible to assume that many audience members do have their own mobile device at hand. Realizing contemporary audience participation can has taken two routes: Developing native applications or using web technologies.

In recent years, developing (or repurposing) a native application for smartphones has been a common approach [17, 21, 25]. This is attractive because musicians and developers can design an instrument choosing from a full range of interactivity and can utilize the full computational power of a mobile phone. However, such an application often limits participation to a set of people who use a certain operating system (e.g. iOS). In addition, there are the challenges of downloading a native app and setting up the network configuration inside the app.

On the other hand, the web browser is a popular choice for audience participation because it alleviates the aforementioned problems; no additional installation is required, the web browser runs on multiple platforms and it is easy to distribute or update an application. It has been used in numerous previous works even before the time of Web Audio. Freeman's Graph Theory [11] and Piano Etudes [12] constitute such examples. In these cases, participants directly and indirectly took part in the process of composing music piece via web browsers prior to a live performance. Instead of customized websites, a pre-existing social network system (e.g., Twitter) that runs on any web browsers can be re-purposed for real-time participation [8].

massMobile, a general framework for audience participation, facilitates rapid development of various participatory applications and enables plug-and-play setup on mobile web platform [29]. Another framework, SWARMED, implements a captive portal so that any audience member can, with minimal configuration, connect to the web-based musical interface [14]. Early usage of web browsers in audience participation without web audio was limited regarding native sound synthesis on mobile devices. In that sense, the audience acts as a composer, influencing the piece on stage, rather than a performer generating sound from a mobile phone [21].

In the tradition of network music, users have deployed web browsers in the collaborative music making endeavors [6, 7]. The Web Audio API [26] accelerated emerging trends of web browser-based music applications where sound is synthesized and generated directly from web pages.

Recent efforts push the performance of web-based audio applications towards the level of native audio applications [22, 5]. The majority of music performances presented in the first Web Audio Conference involves audience participation (or audience involvement) [1, 18, 24, 27, 28]. This reflects the web audio community's strong focus of collaborative music making with the audience. This mobile approach differs from previous approaches where the audience influences music indirectly and sound comes from a stage. *Crowd in C[loud]* draws upon the ideas of existing audience participation works and introduces a new venue for networked web-based music app using a cloud service.

Previous works in audience participation have often relied on a local network with a server developed, configured, and managed by the developers [17, 29, 14, 21]. The use of a cloud-computing infrastructure in the context of computer music requires no physical server on the spot and the cloud can be used in various settings — remote-networked performances, online collaboration systems, audience participation, and locally networked ensembles.

A series of tools to support cloud computing in the context of collaborative composition. Computer Music Cloud is a system for music composition in the cloud with a data exchange protocol [2]. A textual representation of music notation [3] and a computing architecture designed for online music composition system [4] further add to this tool set.

Hindle's "CloudOrch" [15] presented a system that implemented a virtual soundboard on the cloud and, without a physical soundboard or mixer, interconnected multiple audio inputs and outputs. Using the cloud soundboard frees a musician from carrying about high performance machines for computationally heavy music performances. The author conducted a follow-up study in the deployment of such scaled music applications and resource allocation for many computers in cloud [16].

As opposed to using cloud servers, a commodity cloud service provides a convenient option to build a real-time network among computers. It makes the network configuration abstract to users and can be used to distribute data via cloud data centers located globally. An evaluation the efficiency of a cloud service in computer music applications sending control signals measured latency (83ms) between devices located in North America and South America through the Pusher[1] cloud service [9]. Using the same cloud service, SuperCopair provides real-time shared document support for multi-performer live coding in Supercollider [10].

[1]Pusher cloud service: http://www.pusher.com/

3. CROWD IN C[LOUD]

As the name of the piece suggests, the crowd (audience) plays the musical instrument in C Major. This is directly inspired from the piece *In C*, by Terry Riley [23]. In this piece, musicians (with various instruments) were guided to play pre-composed melodic fragments in sequence for a random amount of time. As it is up to each musician to decide how many times to play one fragment, the collective outcome of the ensemble creates a heterophonic texture of chance. Similarly, in *Crowd in C[loud]*, each audience member plays a series of short snippets composed by herself and by other audience members. The interface provided will first guide a participant to compose a short `"tune"` that has five musical notes in C major. Once the participant finishes the composition, he or she can browse, and play, what other audience members have composed. It is thus quite similar to Terry Riley's *In C*, in that one determines for oneself how long to play a tune. The difference is that there is no pre-composed fragments but each audience member will contribute to the piece by submitting a short melody. In this way, participants will have their own tunes and a chord scale become the common ground upon which the entire audience plays. In addition, there is a separate musician performing the piece on stage at the same time with the audience members in *Crowd in C[loud]*. The role of the musician is a meta-performer who can control the chord scale in which the audience members are playing. For example, the meta performer can, on the fly, change the instrument tuned in C major scale to a different chord scale (e.g., C Minor, Pentatonic Scale). This performer cannot generate sound at all on his/her end but only controls the harmonic flow of the piece as generated by the crowd. The interplay between the musician and audience members ensures that each audience member will play individual patterns while a musician can progress the piece by changing chord scales. This performer-audience pairing model comes from a previous work of *echobo* [21] where audience members played a simplified key instrument on smartphones with the chord progression determined by a performer on stage and synchronized over a mobile network.

3.1 Loop-based Instrument

The web-based musical instrument we developed for the piece contains a simple interface that can loop a five-note melody in a specified scale (C major scale in the beginning). There are five circular notes (or "`note dots`") that are connected by lines. And there is a circular play head (or "`the play dot`") in yellow which travels over five red (or green) note dots, triggering a tone whenever it reaches a note dot. The play dot moves at a constant speed so that a melodic pattern (or "`Tune`") will be looped consistently. This ensures that the instrument will generate sound without any user involvement as long as the user turns up the volume and stays on the page. This way, the musician need not worry about being too sparse or silent due to low participation. The expressive range of the instrument depends on where a player places note dots on a screen. First, the vertical position of note dots determines the pitch of the notes. The interface visualizes pitch difference with alternating white and gray divisions in the background. Secondly, the horizontal position of a note dot determines the timbre of the tone; the note dots placed on the leftmost side of the screen will generate pure sine tones while the note dots on the other end will play a tone that combines various oscillators (sine, sawtooth, square, and triangle waves with different detune parameters).

Sound synthesis of the instrument is entirely realized using Web Audio API oscillators. The instrument implements a JavaScript object for each tone (or "`voice`"). Each time the play dot reaches a note dot, the program creates a voice instance that contains a set of oscillators. The voice instance includes a JavaScript object that contains a Gain node and implements an ADSR envelope. The interval between the two consecutive notes is determined by the length of the line in between and the duration of each note is proportional to the interval. A tune can be archived with the position data of five note dots in order and the archived data can be later shared with other audience members to reproduce the tune in other devices.

Note that the duration of one's tune can be arbitrarily long and is not exactly the same as the tune of other audience member. We embrace that asynchronicity among ensemble members and leave the temporal expressivity of a tune up to each player. It is similar to the original version of Terry Riley's *In C* where there was no pulse. We find that having the synchronized global pulse and quantized beats gives the audience a different experience and achieves a different style of music, the exploration of which we leave to future work.

We wanted to design the instrument to be extremely accessible so the audience to pick it up in a few seconds but still be good enough for a participant to be musically expressive. However, simplicity can induce limited flexibility and expressive range and hence hamper long-term engagement. Indeed, the mixed use of note dot locations for multiple parameters (timbre, pitch, and time) constrains the expressive space of the instrument. For example, one cannot play two consecutive notes of the same timbre and same pitch with a long interval in our interface. While we could have made an effort to build a musical instrument that achieves *low entry and no ceiling* [30], we take a different approach to encourage the participation. We find it acceptable to develop a constrained musical instrument [13] in the hope that participants will discover a diversity of playing style via social interaction with other audience members.

3.2 Online Dating Metaphor

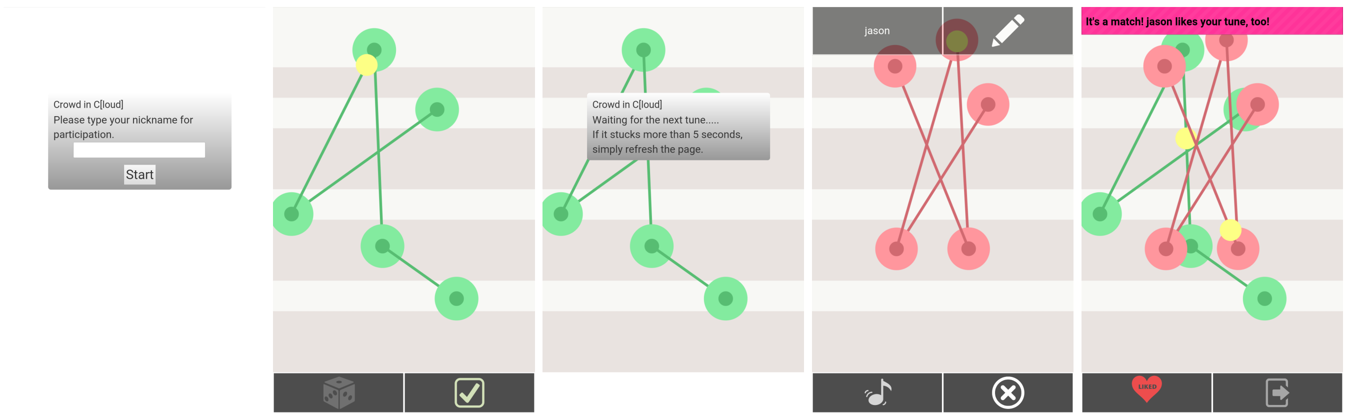
As discussed earlier, the musical instrument provided to audience members has clear limitations; it can only loop five notes with different pitch choices and timbre variations. In turn, once the user finishes the composition of a short snippet, the user is able to browse other people's composition. This grows out of the idea of an online-dating website (such as Tinder) where a user creates a personal profile and then browses other member profiles that include pictures and written descriptions about themselves. Similarly, the networked instrument creates a temporary social network that will last until the end of the performance where a short tune is used as a musical profile. Lastly, the collection of each tune composed by individuals serves as musical phrases such as found in Riley's *In C*. The difference here is that the number of musical phrases is the same as the number of participants and each participant can change the short composition on the fly.

Allowing participants to browse other tunes is expected to motivate people to play the instrument in various ways. First, it motivates a user to compose a tune to express oneself, to attract more people, and to find a (musical) match among the participants. This resembles a self-presentation strategy in online dating sites where participants post photographs and a written description that represents themselves well. Secondly, browsing the tunes inspire participants to discover new styles with which to play the instrument. For example, suppose one created a tune with ascending tones in C major and then later discover a tune that uses note dots to visually draw a certain object. Later, one may find another tune that have five note dots in one place close enough so that it creates a very dense rhythmic pattern. Lastly, we bring the joy of playing together. In `MINGLE` mode, discussed below, one can play his or her own tune with another tune. When two tunes are looped on the same screen, a user is allowed to modify his or her own tune to musically match that of the other. It is a metaphor for the situation where two people on an online-dating website start a conversation, meet off-line, and explore the possibility of being a match. We are planning to analyze the interaction of the participants to investigate whether this socially connected ensemble actually inspired each other.

While there are many different levels of interaction in online-dating websites, we borrowed the simplest model from a popular online dating application, Tinder. On Tinder, one can browse profiles, press the like button, and start to chat when it's a match. The musical instrument can be in one of the five different states: `NAME`, `EDIT`, `WAIT`, `CHECK`, and `MINGLE`. Each state is used to design different interfaces and determine what other states a user can reach out from and go to. The five states are described below.

- `NAME`: When an audience member first visits the link provided http://bit.ly/crowdinc, the member is prompted to type a unique screen name that will be used throughout the performance (Figure 1a). Once the participant submits a valid screen name, the web page will be redirected to the `EDIT` state.
- `EDIT`: A participant composes a tune in this state by dragging and dropping the note dots. The play dots will continue while editing so one can hear the current tune (Figure 1b).
- `WAIT`: This is a transient state where the instrument is waiting for a message from the cloud service after a request for data. The incoming message contains data for another member's tune (Figure 1c).
- `CHECK`: This is a state where a participant can browse the tunes of the other members (Figure 1d).
- `MINGLE`: This is the state where a participant can play two tunes at the same time (Figure 1e). The note dots in green are the tune composed by the user and the note dots in red are the tune composed by another audience member. In this mode, one can freely move green note dots to make two tunes sound differently and explore a new musical pattern with the combination. The red dots cannot be modified.

The interface is designed to notify social interaction by broadcasting messages. For example, when a participant named John "likes" Jane's musical profile by pressing the heart-shaped button in `MINGLE` mode, Jane will receive a


Crowd in C[loud]
Please type your nickname for participation.
Start

Crowd in C[loud]
Waiting for the next tune.....
If it stucks more than 5 seconds, simply refresh the page.

(a) NAME state (b) EDIT state (c) WAIT state (d) CHECK state (e) MINGLE state

Figure 1: Screenshots of Audience Interface in Five States.

message saying "John likes your tune!" Later, Jane browses more tunes and "likes" John's pattern. John will then receive in the top banner saying "It's a match! Jane liked you back!" (Figure 1e).

On the other hand, the performer's interface (Figure 2), which is also a web page, is used to display the list of screen names that are currently participating in the performance. The performer program calculates the number of likes received and the number of participants playing the pattern at the moment under each individual screen name. This interface works like a score-board when projected on a screen at a concert hall. On the top right corner, it shows the screen name of the participant whose tune is most liked and the screen name of the participant whose tune is played most at the moment (named "most crowded"). This projection helps audience members realize that the nature of the participation is social and it also helps a non-participating audience engage with the piece by looking at how their friends and families are doing.

4. CROWD MUSICKING IN CLOUD

The performer interface and the audience interface are available at the following sites. Performer : http://bit.ly/performerinc, Audience : http://bit.ly/crowdinc. To try the demo, press the "`Go Live`" button in the performer interface and use multiple devices to play in the audience interface. Currently there is only support for a single performer using the performer interface.

4.1 Network Structure - Cloud Service

We utilized a cloud service to exchange data among audience members and to orchestrate a chord scale of the crowd. The performer interface and the audience interface are two static web pages hosted on a university web server. Once both web pages are downloaded to a device, there is no dynamic interaction between the device and the web server. The performer interface runs on a laptop and the audience interface typically runs on a participant's smartphone. The performer interface maintains the relevant data (in local JavaScript data structure) regarding all the participants' data (tunes) and all the information needed to display on the scoreboard. Although the performer interface is a web page running on a local machine, it acts as a server in the traditional sense. The only difference is that the server (a performer's laptop) and the clients (audience's smartphones) communicate via a cloud service with minimal network configuration, which is already hard-coded inside the JavaScript file.

After comparing many cloud services, we chose the PubNub cloud service.[2] PubNub provides better bandwidthand reliability.While we used a free plan from PubNub for the development and rehearsal, we purchased the cheapest paid plan to obtain a dedicated key that will allow more than 100 participants in on the session for the actual performance.

PubNub follows a pub-sub paradigm for data communication in JavaScript. Any number of JavaScript web application using the same application key can publish (or send) messages to certain channels or subscribe (or listen) to one or more channels. There are three types of channels used in the application — `performer`, `audience`, and `<uuid>`. `performer` is a channel that only the performer program listens to and it is used when audience programs make a request to retrieve certain data such as a tune object. `<uuid>` stands for universal unique id and is given to each participant as soon as the page is load on the device. It is used to transfer data from the performer to each individual, which is the response to the request mentioned previously. Lastly, all clients subscribe to the `audience` channel and it is used to broadcast a message or to change the chord scale of participants instruments.

4.2 Orchestrating the Crowd via Live Coding

Changing the chord scale in an audience application is pre-written as a JavaScript function and the performer can send a signal to the `audience` channel to call up the function to make changes in the entire crowd. While there are many ways to signal the function call in audience members' devices (buttons, knobs, sliders, keys), we chose to live code on the JavaScript console of the web browser. A performer can type the following line on the console to change the chord scale of the whole crowd.

```
publishMessage("audience",type:"scale",baseNote:60,scale:[0,3,7,12]);
```

`publishMessage` function is written to broadcast a message with a JavaScript object to a specified channel (`audience` in this case). `type` indicates that the message is to change the scale; the parsing function in the audience

[2]PubNub Cloud Service: http://www.pubnub.com

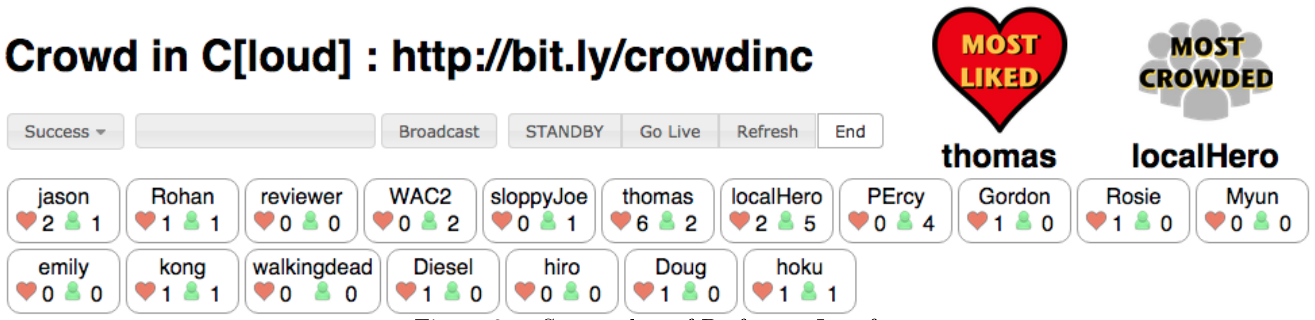


Figure 2: a Screen-shot of Performer Interface

application expects `baseNote` and `scale` within the object. And then the function call in the audience JavaScript program will change the scale to C Minor $(C, E\flat, G, C)$ starting from middle C, of which the MIDI note number is 60. Whenever a performer sends this kind of scale-control message, they are informed with a message on the top banner that, in a few seconds, disappears automatically.

Live coding JavaScript code on the console expands the flexibility of what a performer can do on the crowd's machines. For example, a performer can send any kind of text string that can be evaluated in a JavaScript application on the audience side using a `script`-typed message. See the following three examples:

```
publishMessage("audience",type:"script",script:"soundEnabled=false;");
publishMessage("audience",type:"script",script:"refresh();");
publishMessage("audience",type:"script",script:"alert('hello');";
```

The first example will set a variable `soundEnabled` to false, which is a boolean variable in the audience program state to determine whether to switch on/off the sound synthesis (all device will be muted!). The second example will run the function `refresh();`, which is readily available in the audience program to refresh the page (so everyone is forced to start over!) The third example, maliciously enough, will show an alert box on all smartphones and the web audio synthesis will be halted until the user clicks the okay button.

In orchestrating the crowd, a musician may want to change only the subset of the crowd to produce diverse sound, for example, half the audience playing in C Chord the other half playing the F note only. We included the `probability` property used with `scale`- and `script`-typed messages to achieve a partial code run. If the performer includes `probability : <a float number>` to the JavaScript object in messages as above, it will run the parsed action with the probability of the given number. For example, if `probability : 0.5` was attached at the end of an object, the code will run with a 50% chance, so that the performer can make only (roughly) half of the crowd take the change. The model of one live coder controlling crowd-scale computer networks was proposed in [20] and we believe this is the first realization of the idea. While the potential of live coding has not yet been explored other than changing the scale, we believe there are novel musical styles and aesthetics that we can achieve with this live coding of large-scale machines. We plan to use this feature to live code the web-based instrument [19] to achieve diverse styles of music by changing more than just the chord scale (timbre, interface, mapping), while leveraging human computation of individual users for musical expression within this dynamic scenario.

5. CROWD IN C[LOUD] IN ACTION

We premiered *Crowd in C[loud]* at the Winter Final Class Concert [3] in Stamps Auditorium at the University of Michigan North Campus. The audience consisted predominantly of students and local citizens, who had little background in computer music but were casual smartphone users.

The program note included a shortened link (bit.ly and QR code) and a set of step-by-step instructions that described how to participate so the audience could access the web page and try the interface before the concert began. As long as the participants had devices that could run a web-audio-enabled web browser with any connectivity (mobile or WiFi), they would be able to participate in the piece. There were no additional steps needed such as joining a designated WiFi network, downloading a native app from app stores or typing an IP address, which for casual users may be challenging.

For the performance, the first author composed a short piece and played the role of performer. The performance was started by the performer giving a sign to the audience and pressing the "Go-Live" button at the top of the interface. For the first few minutes, the performer did not intervene to change the global scale. Rather, he communicated with the audience broadcasting chat messages to explain the instrument (timbre, pitch mapping), to encourage participation, and to introduce what the projection screen showed using the message broadcasting. The performer started to change the scale occasionally for the latter part of the performance. The performer interface included frequently used code in a textarea so that the performer could quickly copy and paste in a contingency of possible errors in typing code or scale. At a certain point, the performer changed the global scale to one note so that all device would generate a one-pitched sound. This was to facilitate the playing of a simple melody by quickly running the series of code that had different base note numbers. Later, the crowd was divided into two groups using the `probability` option and one group played a sequence of unified notes while the other half played a background chord.

The performance was well received by the audience and we got positive feedback an various levels. Video footage showed participants, ranging from young kids to the elderly, actively engaged with the instrument and occasionally watching the projection screen.

[3]The video footage of the first performance are available in two following links.
https://youtu.be/wCXhGotDtFs?t=248 : audience seats
https://youtu.be/8RWgXoM2BCA?t=263 : stage

6. CONCLUSION

In this paper, we have introduced audience participation music that uses a distributed instrument that runs on the web browser using Web Audio API. The metaphor of collaborative music making comes from the social interaction model of online dating. In addition, we hope that this work convinces the web audio community that the cloud service adds a convenient option to support networked ensemble with minimal server-side programming. While web audio has an infinite amount of opportunities to host existing music applications, we find this piece meaningful in that it draws upon many ideas from the web both aesthetically and technologically.

7. ACKNOWLEDGMENTS

We would like to thank the students from winter class of 2015 Mobile Music Ensembles from University of Michigan to help during the rehearsals. Thanks CAPES (Brazil) for sponsoring one of the authors during the research.

8. REFERENCES

[1] J. Allison. Traversal. *WAC - 1st Web Audio Conference*, 2015.

[2] J. Alvaro and B. Barros. Computer music cloud. In S. Ystad, M. Aramaki, R. Kronland-Martinet, and K. Jensen, editors, *Exploring Music Contents*, volume 6684 of *Lecture Notes in Computer Science*, pages 163–175. Springer Berlin Heidelberg, 2011.

[3] J. L. Alvaro. Stringscore: Composing music with visual text. In *Proceedings of the Sound and Music Computer Conference*, 2012.

[4] J. L. Alvaro and B. Barros. A new cloud computing architecture for music composition. *Journal of Network and Computer Applications*, 36(1):429 – 443, 2013.

[5] B. M. J. C. M. Anand Mahadevan, Jason Freeman. Earsketch: Teaching computational music remixing in an online web audio based learning environment. *WAC - 1st Web Audio Conference*, 2015.

[6] A. Barbosa. Public sound objects: a shared environment for networked music practice on the web. *Organised Sound*, 10(03):233–242, 2005.

[7] P. Burk. Jammin'on the web-a new client/server architecture for multi-user musical performance. In *Proceedings of the International Computer Music Conference*, 2000.

[8] L. Dahl, J. Herrera, and C. Wilkerson. Tweetdreams: Making music with the audience and the world using real-time twitter data. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 272–275, 2011.

[9] A. D. de Carvalho Junior, G. Essl, and M. G. de Queiroz. Computer music through the cloud: Evaluating a cloud service for collaborative computer music applications. In *Proceedings of the International Computer Music Conference*, Denton, Texas, 2015.

[10] A. D. de Carvalho Junior, S. W. Lee, and G. Essl. Supercopair: Collaborative live coding on supercollider through the cloud. In *International Conference on Live Coding*, 2015.

[11] J. Freeman. Graph theory: interfacing audiences into the compositional process. In *Proceedings of the 7th international conference on New interfaces for musical expression*, pages 260–263. ACM, 2007.

[12] J. Freeman. Web-based collaboration, live musical performance and open-form scores. *International Journal of Performance Arts and Digital Media*, 6(2):149–170, 2010.

[13] M. Gurevich, P. Stapleton, and A. Marquez-Borbon. Style and constraint in electronic musical instruments. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2010.

[14] A. Hindle. Swarmed: Captive portals, mobile devices, and audience participation in multi-user music performance. In *Proceedings of the 13th International Conference on New Interfaces for Musical Expression*, pages 174–179, 2013.

[15] A. Hindle. Cloudorch: A portable soundcard in the cloud. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2014.

[16] A. Hindle. Orchestrating your cloud-orchestra. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2015.

[17] H. Kim. Moori: interactive audience participatory audio-visual performance. In *Proceedings of the 8th ACM conference on Creativity and cognition*, pages 437–438. ACM, 2011.

[18] T. Kita. Smartphone jam session with audience. *WAC - 1st Web Audio Conference*, 2015.

[19] S. W. Lee and G. Essl. Live coding the mobile music instrument, 2013.

[20] S. W. Lee and G. Essl. Models and opportunities for networked live coding. *Live Coding and Collaboration Symposium*, 2014.

[21] S. W. Lee and J. Freeman. echobo: A mobile music instrument designed for audience to play. 2013.

[22] J. Monschke. Building a collaborative digital audio workstation based on the web audio api. *WAC - 1st Web Audio Conference*, 2015.

[23] T. Riley. In c. Composition, 1964.

[24] S. Robaszkiewicz and N. Schnell. Soundworks–a playground for artists and developers to create collaborative mobile web performances. *WAC - 1st Web Audio Conference*, 2015.

[25] C. Roberts and T. Hollerer. Composition for conductor and audience: new uses for mobile devices in the concert hall. In *Proceedings of the 24th annual ACM symposium adjunct on User interface software and technology*, pages 65–66. ACM, 2011.

[26] C. Rogers. Web audio api. 2012.

[27] T. S. Sébastien Piquemal. Field #2. In *WAC - 1st Web Audio Conference*, 2015.

[28] B. Taylor. Pearl river (2013 2015). *WAC - 1st Web Audio Conference*, 2015.

[29] N. Weitzner, J. Freeman, Y.-L. Chen, and S. Garrett. massmobile: towards a flexible framework for large-scale participatory collaborations in live performances. *Organised Sound*, 18(01):30–42, 2013.

[30] D. Wessel and M. Wright. Problems and prospects for intimate musical control of computers. *Computer Music Journal*, 26(3):11–22, 2002.