DISTRIBUTED PERFORMANCE IN LIVE CODING

Ben Swift, Henry Gardner
School of Computer Science
ANU College of Engineering and Computer Science
{ben.swift, henry.gardner}@anu.edu.au

Alistair Riddell
Photography and Media Arts
ANU College of Arts and Social Sciences
alistair.riddell@anu.edu.au

ABSTRACT

Distributed composition is an extension to current live coding practice, in which audience members are given control of certain aspects of the performance. The scope of this control is dictated by the live coder as the performance progresses. In this way, the role of performer shifts from the live coder alone, to be distributed amongst participating audience members. The live coder acts as a compositional overseer. This offers new possibilities for creativity and engagement in a live coding performance.

1. INTRODUCTION

Live coding is primarily an improvisational performance practice. During a performance a live coder must create musical structures (such as generative processes) and manipulate their associated parameters over the course of the performance [16]. The common live coding practice of code projection (projecting a view of the performer's code onto the screen during the performance) means that audience members can see the processes underlying the performance as well as hear the music which is produced [10]. While not every audience member at a given live coding gig has the necessary background to understand the code on screen, those who do are free to examine and appraise the performer's decisions on algorithmic choice and parameter selection.

The notion of performance in live coding incorporates elements of both composition and improvisation. This paper describes a performance paradigm in which audience members are, at the discretion of the performer, given direct control over some of the parameters in the performer's code. Audience members can then use a control device to change these parameters over time, while the live coder continues to work on other parts of the code. The role of the performer is therefore distributed amongst the live coder and the audience participants. We believe that this concept of *distributed performance* is a promising extension to current live coding practice, and suggests a new performance modality for exploration as the practice of live coding matures.

At present, we have a working distributed performance system implemented using *Impromptu* [17] as a live coding environment and the MRMR OSC Controller [12] application on the Apple iPhone (or iPod Touch) as a control device. Section 2 of this paper discusses some of the opportunities and conditions associated with distributed performance in live coding, and Section 3 describes the details of our implementation as well as the results of our first concert.

2. WHY RELINQUISH CREATIVE CONTROL?

2.1. Interaction Model

The experimental computer music group the *Hub* were pioneers in the use of technology to facilitate new interactions between performers. Tim Perkins, one of the founding members of the *Hub*, reflects in their CD liner notes

> [our] emphasis has been on connections between musicians, the excitement of using computers to define a new social context for music making, as well as exploring the possibilities of systems too complex for direct control.[8]

Similarly, incorporating audience interaction into the practice of live coding offers new possibilities for distributed control of complex musical systems. While live coders sometimes perform in groups, ranging in size from pairs [16, 11] up to whole orchestras [15, 13], these are static, egalitarian set-ups, with a pre-set number of performers having a more or less equal role in the performance. In contrast, distributed performance in live coding allows audience members to control specific aspects of the musical performance, with the live coder changing the roles and relationships of the audience participants on-the-fly. The live coder acts as an overseer, dynamically distributing creative control in response to the changing demands of the performance.

Distributed performance is not a musical democracy or a chaotic free-for-all where each participant acts independently of all others. The live coder is still in control of the overall shape of the piece, but he chooses particular parameters to 'distribute'; those which he knows to be suitable for audience control (see Figure 1). The performer also chooses the type and range of the control given to the participant, depending on the control affordances of the interaction device in use. The live coder determines the role of the audience as the code is written. This improvisational dimension fits well with the idea of live coding being an improvisational practice, the live coder creating and responding to musical ideas like a modern concerto artist [3].

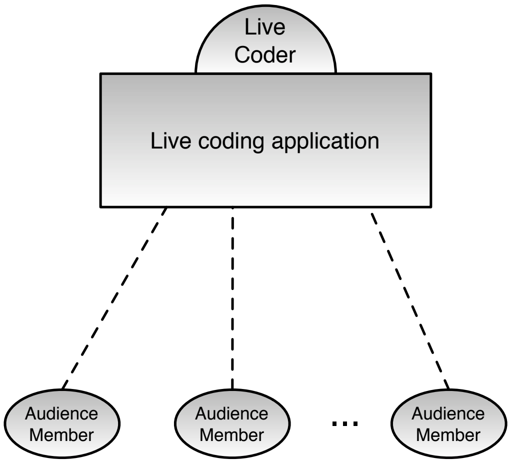

Figure 1. A distributed performance set-up

It is important to point out that the idea of outsourcing some control to an audience member is not designed to address a perceived shortcoming in current live coding practice. Clearly, a work of art is not necessarily poorer for the fact that it is the vision of a singular skilled artist rather than a collaborative effort involving the viewing public. However, sharing some control with the audience in a live coding performance gives rise to some interesting possibilities

1. increased audience engagement
2. the introduction of a social dynamic between different audience members, each in charge of a different part of the code
3. the introduction of musical material beyond the control of the live coder, to which he/she must react
4. a new modality of live coding performance

2.2. Engaging the Audience

Mechanisms for audience participation in a performance can lead to an enhanced performance experience [2, 18, 14]. During a performance, live coders often set up generative processes which create ongoing streams of musical material [4]. These processes may be governed by a small number of parameters which control the musical material they generate. Allowing an audience member to control the trajectory of these parameters over the course of a performance gives them a share in the creative process.

Furthermore, the projection of the code for the audience is designed to facilitate audience engagement [3]. The code is an abstraction, a symbolic representation of the performance. In live coding this symbolic representation is made visible to the audience, in contrast to other forms of improvisational performance where the decisions and creative processes of the performer are inscrutable. The live coder can take advantage of this 'transparency of creative process' by inviting the audience to modify this symbolic representation (that is, the code), resulting in a deeper engagement with the performance.

2.3. Introducing a Social Dynamic

With more than one audience performer, the relationship between the various participants can be explored [9]. As the live coder delegates control of certain parameters out to different audience participants, each audience member can observe the gestures and control manipulations of their peers in the audience. Each participant is responsible for a certain aspect of the whole performance, and the interplay between the participants is a key factor governing the overall aesthetic of the piece. Furthermore, any audience participants who have not chosen to participate directly (that is, who do not have a control device) can observe the behaviour of the performers distributed throughout the audience. The divide between performer and audience present in many conventional music performance practices is removed, and this adds further incentive for all audience members to engage with the performance.

There is also scope for more complex interaction between participants in the audience. The mapping of a participant's control to a parameter need not be one-to-one, control may be given to two (or more) audience participants to control collaboratively. The co-location of the audience members can be utilised, such as by giving two participants a pop-up instruction to find each other and synchronise control gestures. The activity of individual participants and the relationships between them can be displayed visually, alongside the projected code. These relationships, such as groupings amongst participants, can be controlled dynamically by the live coder, or even to a participant to which the ability has been delegated.

2.4. Responsive Coding

In any improvisational practice, creative decisions are made continuously as the performance unfolds, often in response to material unanticipated by the performer [7]. In jazz, for instance, this 'new material' can be introduced by other members of the ensemble, while in live coding, novel material may be the result of stochastic processes. Indeed, responding to the music as it unfolds is one of the challenges of live coding, and can also be a catalyst for inspiration.

Incorporating audience control is an opportunity for the live coder to respond to the choices of the other participants in the performance. This adds a responsive element akin to jamming in a free jazz ensemble, as the music produced by the participants may provide inspiration for the live coder to create and modify the code in new ways. The asymmetry in the degree of control possessed by the live coder (a great deal of control) and the participant (limited control, as dictated by the live coder) means that the situation is not quite the same as an instrumental jam, where each musician is limited only by their instrument and ability. However, if participants could enter a 'code entry' mode and send actual snippets of code to the live coder's code buffer, then this interaction between coder and audience participants begins to resemble that of a more traditional improvisational ensemble.

2.5. Conditions of Distributed Performance

Certain conditions must be considered when extending a live coding environment to accommodate distributed performance. Coding a performance in real-time is already a cognitively demanding activity, and introducing audience control into the performance can add to the cognitive load of the live coder. For this reason, it is important to customise the live coding software environment to make the distribution of control as simple and effective as possible. A selection of different interaction modes, based on the affordances of the interaction device and the requirements of the performance, can be designed ahead of time. These pre-set modes can then be pushed to a (possibly random) participant with a simple function call in the code. The distributed performance infrastructure should be pre-prepared so that the live coder can spend their time writing musical (rather than boilerplate) code. Some ideas regarding this 'interaction infrastructure' in the live coding environment are discussed in Section 3.1.3.

Feedback is crucial in distributed performance to ensure that the participants are aware of exactly what they are controlling, and the scope of that control [5]. If the participants cannot determine which aspect of the music they are influencing, they will quickly lose interest in the interaction. Feedback can either be auditory (incorporated into the musical performance) or visual (alongside/overlaid on the projected code).

For visual feedback to be useful, each participant must register a name (or other unique identifier) which can be used to tag any visual feedback applicable to them. This can be done in a simple configuration step upon joining the performance with their interaction device. The system can then give meaningful visual feedback, such as a graphical overlay of the participant's name attached to the particular parameter they are controlling in the projected code.

In live coding, there are certain parameters in the code which have immediate and noticeable effects (this is described by John Croft as *aesthetic liveness* [6]), such as timbral parameters (filter cutoff and resonance, etc.). If these parameters are outsourced, they require minimal visual feedback, as the effect of any participant manipulation is sonically obvious. There are also some parameters whose effects are subtler (such as reverb) and less immediate (changing the parameters of a generative process). Outsourcing these parameters requires more informative visual feedback.

3. DISTRIBUTED PERFORMANCE: OUR APPROACH

3.1. Technical Specifications

We are in the process of implementing a distributed performance system for use in our own live coding practice. Our distributed performance set-up revolves around a single live coder performing on a laptop (using *Impromptu*), and a flexible number of audience participants interacting wirelessly with the live coder using an Apple iPhone (or iPod Touch [1]) running the MRMR OSC Controller application. The basic architecture of the system is shown in Figure 2.

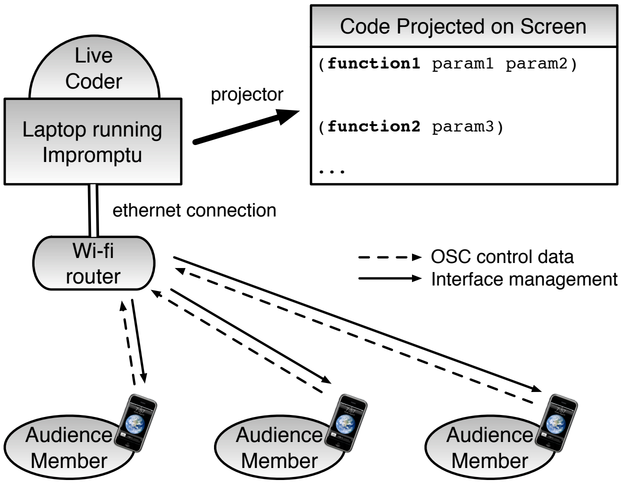

Figure 2. Our live coding set-up

3.1.1. The iPhone as an interaction device

Each iPhone connects to the network over 802.11g wi-fi. The live coding laptop is connected to the same local network using an ethernet connection. As a control device, the main affordances of the iPhone are its 3.5" multi-touch screen and built-in 3-axis accelerometer. The accelerometer, combined with the device's small size, means that the iPhone can be used as a gestural controller.

The iPhone was chosen for two main reasons:

1. While far from ubiquitous, the iPhone is a popular phone, which in our experience has reasonably high ownership rates amongst the type of crowd which would attend a live coding gig. By using a 'BYO controller'

set-up, we eliminate the need for custom interaction hardware, allowing us to have several participants without the expense of purchasing and maintaining a fleet of our own interaction devices.

2. The iPhone provides a standardised development environment for our interaction software. We are currently using an open source software controller called the MRMR OSC Controller [12] to send control data to the live coding environment. In the future, we would like to have interaction clients available for other mobile OSes (such as Symbian, WebOS or Android), but currently our efforts are focused on the artistic practice of distributed performance in live coding, and the iPhone and MRMR has proved valuable in getting the system up and running in a short period of time.

The iPhone and iPod Touch possess an internal speaker, and the iPhone (but not the Touch) also has a built-in microphone. These audio interfaces also offer interesting possibilities for interaction and feedback, although they have not been incorporated into our system at this time.

3.1.2. The MRMR OSC Controller

The MRMR OSC Controller is available as a free download from the online App Store. Upon start-up, it automatically detects and connects to the live coder's laptop using Bonjour Zeroconfig networking. The MRMR software allows the live coder to present a unique interface (called a *patch*) to each iPhone participating in the performance. Each patch is a combination of widgets: buttons, sliders, multi-touch zones, text input fields and accelerometer data (see Figure 3). The participant can use these widgets to send control data back to the live coder using the Open Sound Control (OSC) communication protocol.

One key feature of the MRMR application is that a patch can be modified (or changed completely) at any time by the live coder. Each widget can be represented as a formatted string, specifying they widget's type, size, position and label. A patch is simply a collection of such strings, which is interpreted by the MRMR App to present the appropriate UI to the participant. Strings representing new widgets can be pushed to any device at any time, and the UI will be updated instantaneously.

This allows the control options presented to each audience participant to be changed during the performance. For instance, if a particular piece has three distinct movements, then the interfaces can be updated to reflect the different needs of each movement. Also, different participants may be given different interfaces, allowing the potential for users to be given different roles in the performance. In this regard, the MRMR OSC controller is well suited to the ever-changing needs of the live coder in distributed performance.

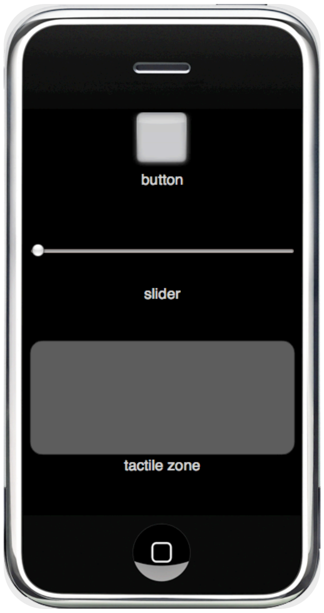

Figure 3. An example MRMR patch

3.1.3. Coding with participation in mind: distributed performance in Impromptu

As we continue to develop our system, we also hope to learn how to best customise a live coding environment to allow for distributed performance. Currently, we are using a client-server interaction model, in which the connections between the iPhones and the live coding laptop are managed automatically. Impromptu, which is a Scheme-based live coding environment, manages a list of the names and IP addresses of all connected devices, and this list is available to the live coder at any time.

As the code is written during a performance, the live coder can (with a function call) push a MRMR patch or read control data from a given device by device name *or* IP. Given the immediacy required in live coding, it is desirable to allow control data from a given iPhone to be accessed as succinctly as possible, and we are currently investigating several possible Scheme forms for doing this. In particular, we are currently using the iPhone control data to vary certain parameters in the live coded Scheme functions.

3.2. *inMates*: Lessons From Our First Concert

The *inMates* concert in April 2009 was the inaugural performance of our distributed performance system. The system is still under active development, and the *inMates* performance served as a means to evaluate the technical and artistic progress of the system, as well as suggest further refinements to our distributed performance approach.

The *inMates* performance took the form of a virtual drum circle. Each distributed performer was assigned control of a certain type of drum, either a djembe, taiko or conga. The participants did not have the ability to hit the drum directly, but could control the beating of their drum along five dimensions. Each participant determined the period, (and phase offset) of their drum pattern, quantised to 32nd-note increments. The participants could also control the volume of their drum, and the 'type' of drum hit (for example, a heel or a slap hit on the conga). Each of these parameters could be controlled by a slider on the iPhone. They could also choose to stop playing, dropping in and out of the circle as they desired, by toggling a button on the iPhone's screen. All up, each performer had five parameters under their control.

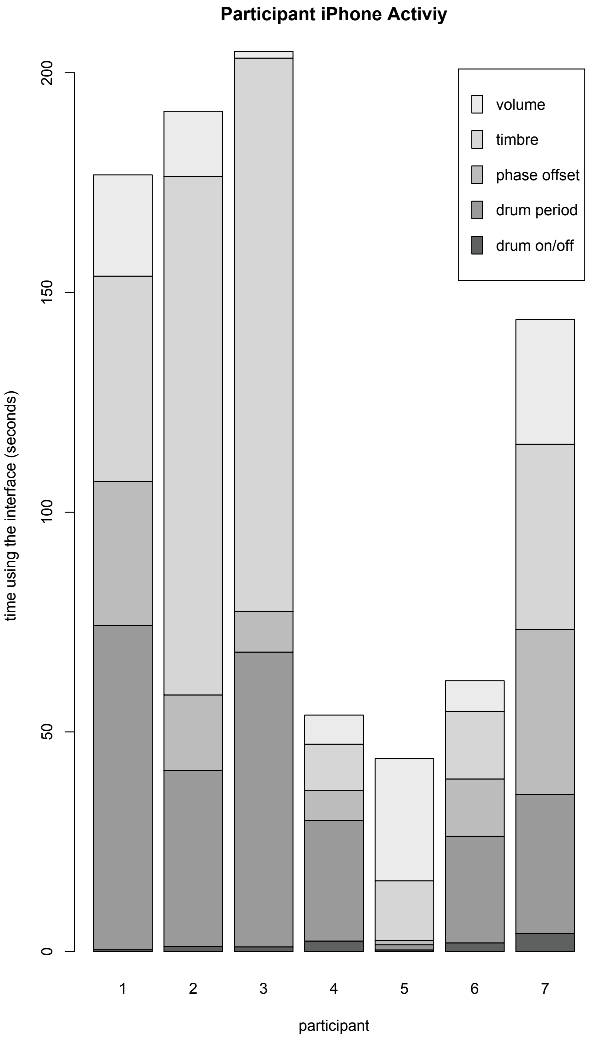

Figure 4. Audience interaction data from the *inMates* concert

In total, there were seven audience participants at the *inMates* concert. The performance lasted 25 minutes and all interaction data was logged. The interaction breakdown, showing the amount of time each participant spent manipulating each different parameter, is shown in figure 4.

Given the preliminary nature of this concert, we do not present any particular statistical conclusions from the interaction data gathered. However, it is interesting to note the different interaction styles of the different participants. Quantifying these differences will be a goal of future concerts. The primary aim of this concert was to test the system under load. In this regard, the concert was a success, with no audio dropouts or packet loss, despite the simultaneous interaction of the seven participants.

Following the concert, an informal feedback session was held amongst the participants. Overall, the participants enjoyed the the ability to influence the performance directly through the iPhones. One participant commented that *"There were some moments where I was really enjoying the drumming and hoping it would continue, but then somebody changed something and that moment was gone."* Participants commented that there were moments where they were particularly enjoying controlling the drum to which they had been assigned.

At times, participants found it hard to ascertain which drum they were controlling. Unfortunately, a planned visual feedback element was not completed in time for the performance, and some participants felt this could have helped them determine which aspects of the overall sound they were controlling. Overall, the participants felt that they would like to participate in future performances.

In future concerts we hope to refine the experience for both live coder and audience participants, including investigating the differences in audience interaction for participants with different levels of musical expertise. The possibility of audience members attending numerous concerts and developing their own particular interaction techniques is also an exciting one.

4. CONCLUSION

Our work on distributed performance in live coding has proven to be a fruitful source of inspiration for our creative practice. The ability to distribute control gives live coders a new source of intentional, sensitive creativity to harness, and allows the audience to engage more deeply with the creative process of the performance.

Live coding, as a discipline, cannot yet boast the same rich history of innovation and extended techniques that many conventional instrumental practices can. However, distributed performance is one opportunity to extend the current landscape of live coding performance, where the live coder and the audience are deeply interconnected in the act of performance. We hope to increasingly incorporate elements of audience involvement in our live coding.

5. REFERENCES

[1] “Apple iphone.” [Online]. Available: http://www.apple.com/iphone/

[2] L. Barkhuus and T. Jørgensen, “Engaging the crowd: studies of audience-performer interaction,” *CHI ’08: CHI ’08 extended abstracts on Human factors in computing systems*, Apr 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1358628.1358785

[3] A. Blackwell and N. Collins, “The programming language as a musical instrument,” *Proceedings of PPIG05 (Psychology of Programming Interest . . .* , Jan 2005. [Online]. Available: http://www.cogs.susx.ac.uk/users/nc81/research/proglangasmusicinstr.pdf

[4] A. Brown and A. Sorensen, “Interacting with generative music through live coding,” *Contemporary Music Review*, Jan 2009. [Online]. Available: http://www.informaworld.com/index/909056573.pdf

[5] X. Cao, M. Massimi, and R. Balakrishnan, “Flashlight jigsaw: an exploratory study of an ad-hoc multi-player game on public displays,” *CSCW ’08: Proceedings of the ACM 2008 conference on Computer supported cooperative work*, Nov 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1460563.1460577

[6] J. Croft, “Theses on liveness,” *Organised sound*, Jan 2007. [Online]. Available: http://journals.cambridge.org/abstract_S1355771807001604

[7] C. Gould and K. Keaton, “The essential role of improvisation in musical performance,” *The Journal of Aesthetics and Art Criticism*, vol. 58, no. 2, pp. 143–148, Apr 2000, improvisation in the Arts. [Online]. Available: http://www.jstor.org/stable/432093

[8] S. Gresham-Lancaster, “The aesthetics and history of the hub: The effects of changing technology on network computer music,” *Leonardo Music Journal*, vol. 8, pp. 39–44, Jan 1998. [Online]. Available: http://www.jstor.org/stable/1513398

[9] C. Heath, P. Luff, D. Lehn, J. Hindmarsh, and J. Cleverly, “Crafting participation: designing ecologies, configuring experience,” *Visual Communication*, Jan 2002. [Online]. Available: http://vcj.sagepub.com/cgi/content/abstract/1/1/9

[10] A. Mclean, N. Collins, and J. Rohrhuber, “Live coding in laptop performance,” *Organised sound*, vol. 8, no. 3, p. 321, Jan 2003. [Online]. Available: http://dialnet.unirioja.es/servlet/articulo?codigo=862522

[11] C. Nilson, “Live coding practice,” *Proceedings of the 7th international conference on New interfaces for musical expression*, pp. 112–117, 2007.

[12] E. Redlinger, “Mrmr osc controller.” [Online]. Available: http://mrmr.noisepages.com

[13] J. Rohrhuber, A. de Campo, R. Wieser, and J. van Kampen, “Purloined letters and distributed persons,” *Music in the Global Village Conference (Budapest)*, Jan 2007. [Online]. Available: http://www.wertlos.org/articles/Purloined_letters.pdf

[14] J. Sheridan, N. Bryan-Kinns, and A. Bayliss, “Encouraging witting participation and performance in digital live art,” *BCS-HCI ’07: Proceedings of the 21st British CHI Group Annual Conference on HCI 2007: People and Computers XXI: HCI...but not as we know it*, vol. 1, Sep 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1531294.1531297

[15] S. Smallwood, D. Trueman, P. Cook, and G. Wang, “Composing for laptop orchestra,” *Computer Music Journal*, vol. 32, no. 1, pp. 9–25, Mar 2008. [Online]. Available: http://muse.jhu.edu/journals/computer_music_journal/v032/32.1smallwood.pdf

[16] A. Sorensen and A. Brown, “aa-cell in practice: An approach to musical live coding’,” *Proceedings of the International Computer Music Conference*, Jan 2007. [Online]. Available: http://impromptu.moso.com.au/extras/aa-cell-icmc07.pdf

[17] A. Sorensen, “Impromptu.” [Online]. Available: http://impromptu.moso.com.au

[18] R. Taylor, P. Boulanger, P. Olivier, and J. Wallace, “Exploring participatory performance to inform the design of collaborative public interfaces,” *CHI EA ’09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, Apr 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1520340.1520561