Composing for a Networked, Pulse-Based, Laptop Orchestra*

DAVID OGBORN

Department of Communication Studies & Multimedia, McMaster University, 1280 Main Street West, Hamilton, Ontario L8S 4M2, Canada
E-mail: ogbornd@mcmaster.ca

Guided by the idea of participatory culture, networked pulse synchronisation and live coding have been core approaches in the activity of the Cybernetic Orchestra, an electronic performance ensemble at McMaster University in Hamilton, Canada. Following general discussion of the way in which networked pulse-based music and live coding work within this orchestra, there is specific discussion of a number of compositional models and practices that have been found effective, including code-sharing, instruction-scores, code as material, and physical performance.

1. INTRODUCTION

The Cybernetic Orchestra is an electronic performance ensemble at McMaster University in Hamilton, Canada, formed to investigate and expand the possibilities of laptop orchestras and related performance/creation situations. The orchestra meets for weekly rehearsals, with participation open to the entire university community (students, alumni, employees). I have directed the Cybernetic Orchestra since its inception during the 2009–10 academic year.

A guiding idea has been the thesis that laptop orchestras are sites for the construction of participatory cultures – in the sense advanced by Henry Jenkins (and others) as cultures with 'relatively low barriers to artistic expression and civic engagement, strong support for creating and sharing one's creations, and some type of informal mentorship whereby what is known by the most experienced is passed along to novices' (Jenkins et al. 2006). This is closely related to the pursuit – championed by Wessel and Wright in a classic article of NIME research – of a 'low entry fee with no ceiling on virtuosity' (2002). Laptop orchestra activities can be assessed from this double perspective. On the one hand, the 'entry fee' to the orchestra should be kept as low as possible, making the orchestra a place that is as immediately attractive as possible to a wide pool of potential participants. On the other hand, it should be a place where participants from a wide range of backgrounds are able to orient themselves and develop as electronic musicians in advanced, unconstrained ways.

The day-to-day practice of the Cybernetic Orchestra emphasises performance and improvisation over composition, although the orchestra does perform fixed compositions created outside of the weekly rehearsals. This makes it possible for people to join the orchestra, gain experience and contribute as performers before potentially making an additional contribution as composers. Within this basic framework, two broad approaches are consistent features of our rehearsal and performance activities: network pulses and live coding. We have explored both of these approaches in free improvisation settings, in fixed compositions and in hybrid situations that blend the improvised with the composed. In what follows, I describe some of the models and practices that have evolved in the orchestra, with an emphasis on the way in which particular compositional practices can facilitate wider participation in the activities of the orchestra, while also allowing the development of virtuosity and specialisation.

2. NETWORKED PULSES

The creation of music with a regular pulse has the potential to be a 'hot button' issue in electroacoustic discourse, given a history in which important traditions of 'high art' electroacoustic practice have tended to avoid pulse structures, and a countervailing set of important milieus (both popular and niche) in which both audience and performer tend to privilege such structures. Recently, artistic practices blending the concerns of 'high art' traditions with the beats and pulses of electronica have become more common, although, as Ben Neill (2002) notes, the influence is sometimes frustratingly unidirectional. The practice of the Cybernetic Orchestra has been to embrace pulse-based structures, as a way of connecting with participants' prior experience (often substantial) with other and popular musics as listeners and creators.

This embrace of pulse-based structures has not simply led to the reproduction of these other and popular

*The author would like to thank the members of the Cybernetic Orchestra (2010–11) for their creativity and enthusiasm, and research assistants Alyssa Lai, Stephanie Moore and Rob Petti for their invaluable assistance. This ongoing research has been supported by funding from the Social Sciences and Humanities Research Council of Canada (SSHRC) Image, Text and Sound Technology programme, as well as by the Arts Research Board of McMaster University.

musics, however – nor, indeed, to the production of any music based primarily in notes, rhythms and metre. What happens instead is that the rapid underlying cycles of pulses give the orchestra (individual members and the group as a whole) a way of controlling moments of synchrony (things that happen at the same time tend towards perceptual fusion), material distinctions (things that happen at different times and places tend to stream separately) and density (things that happen only on a certain pulse versus on sets of pulses or in the space between pulses). This control has a flexible granularity, in that one can specify something as simple as 'make this sound on every downbeat' or as involved as 'begin this sound a gesturally controlled short period of time after every 13th pulse, with no regard to the length of the underlying pulse cycle, and continue the sound until 2.5 seconds after a separate gestural control on brightness passes a certain threshold'.

My observation is that the focus of orchestra members' attention remains to a very large extent on the type, qualities and morphologies of the sounds they produce. Indeed, it may well be the case that the pulse structure heightens participants' attention to aspects of their performance other than metre. With the scope of free choice about when an underlying event begins somewhat reduced, more attention is available for thought about what the identity of that event will be. The occasional use of predetermined pitch structures in fixed compositions or improvisations can facilitate sound-based musical creativity in the same way. With a somewhat reduced expectation to create pitch structures expressively, the focus of attention moves elsewhere. From an analytical point of the view, the explicit text (for instance, a code, patch, score and/or performance instructions) can be misleading about the real content of a moment. The compositional intention, delivered in one way or another to a laptop orchestra of 16 people, to 'create a low E-flat on beat 8' will inevitably produce 16 different sounds, with the extent of the differences (and the way those differences appear to an audience) a function of the specific performance practice of the orchestra.

Within a general context of engaging with pulse-structures without necessarily or directly engaging with popular or traditional musical genres, the distinction between entrainment (brought about by pulses and the expectation that they will continue), distraction (when the periodicity is longer than 5 seconds or so, leading to a different type of listening) and boredom (when material is too predictable) represents a useful conceptual tool (Emmerson 2008). By aiming for an interplay between entrainment and distraction, we can sidestep genre expectations without discarding the potential enjoyment, interest and perceptual orientation effects of beats.

The natural technological support for creating pulse-based music with a laptop orchestra is the network distribution of a pulse cycle controlled from a central computer. This software structure is found in numerous compositions and instruments created for laptop orchestras. Since the Cybernetic Orchestra has employed this structure so frequently, I have developed a draft protocol, espBeat, which aims to facilitate pulse-based performance and improvisation along two distinct axes. On one axis, espBeat addresses the technical problems that arise when distributing pulses over a wi-fi network with the Universal Datagram Protocol (UDP) – principally missed beats (dropped packets) and poor timing (variable network latency). The protocol accomplishes this by generating timing within cycles on each machine locally, by broadcasting the beginning of each cycle in redundant bursts of identical information, and by regenerating new cycles automatically in the absence of new information from the server. At the same time, on another axis, espBeat presents a growing collection of standardised software handles to specifying time relative to cyclic pulse structures (this aspect is less developed at present). Software implementations of the espBeat protocol for various audio programming environments are part of the Cybernetic Orchestra's collection of laptop orchestra tools, espTools, and are available through the esp.mcmaster.ca website.

3. LIVE CODING

If pulse-based structures have been the usual form of the orchestra's sonic production, live coding has been our pre-eminent mode of performance. Orchestra members create simple (or not so simple) programs that control audio signal processing and communicate over the network (often to wait for the next pulse in the bar, or a specific pulse in the bar). Most of our live coding efforts have made use of the ChucK language (Wang and Cook 2004), both for its intrinsic advantages from the standpoint of on-the-fly programming and because the orchestra has sometimes experimented with pieces created for the Princeton Laptop Orchestra, many of which are coded in ChucK (Smallwood, Trueman, Cook and Wang 2008).

I would advance the thesis that live coding has the potential to be a form of participatory culture *par excellence*, both within and beyond the specific context of the laptop orchestra. At first blush, it may seem strange to connect live coding and participatory culture. After all, 'classical' formulations of the possibilities and opportunities of live coding have tended to emphasise features suck as risk (and, by implication, virtuosity in the management of that risk), competition, flexibility and universality (i.e. abstraction) (Collins, McLean, Rohrhuber and Ward 2003). But while code can certainly be an exciting space for the unfolding of virtuosity, it can just as readily serve as a vehicle for the unrestricted sharing and informal mentorship that is

a constituent element of participatory cultures. For instance, in a workshop or rehearsal setting, the code of a leader or selected participant can be displayed or projected, and then copied, imitated, altered and discussed. It is often just as easy for the members of a group to lean over and copy, imitate, alter or discuss the code they see on their neighbours' screens.

The text and visuality of the code produces 'meaning effects' for an audience, but it also produces 'sharing effects' or 'participation effects' within and for an ensemble. It produces these effects, moreover, without the requirement for any additional software or hardware – often an important consideration when dealing with people new to electronic music performance, or situations where software and hardware have to be deployed very quickly. Each additional layer of complexity brings with it an additional cost in time spent explaining and troubleshooting, and the sum of these additional costs works against a performance achieving the 'escape velocity' wherein participants understand, manipulate and perform the constituent elements more flexibly and expressively. For the sighted and literate, the visual display of code (including the mere display of code on a nearby screen) leverages substantial prior experience of parsing and manipulating text. Text is already a surprisingly robust network technology. Thus, while network protocols for the direct transmission of code during a performance can certainly be adopted (and form an important area for future exploration), it is not necessary to adopt such sophisticated mechanisms in order to reap the participatory benefits of text-based code in a laptop orchestra situation.

The fact that most of the members of the Cybernetic Orchestra have not tended to come from backgrounds with extensive experience in either general-purpose or audio programming has led this group to develop live coding practices and aims in a way that emphasises code's potential to share ideas quickly and efficiently. In the course of an individual session, the directionality of code sharing tends to change in unpredictable ways. Sometimes a director or more experienced coder needs to *demonstrate* a particular technique that the group needs – whether to deal with the basic requirements of a specific composition, or piece of hardware, or simply in order to expand the group's basic stock of techniques. At other times, the sharing of code is rhizomatic and 'peer-to-peer', as code ideas proliferate around the orchestra according to a complicated topology of physical proximity, the interplay of personalities and the situationally specific appeal of particular sounds

I hesitate to conceive of these two types of code sharing as top-down versus bottom-up. As someone directing or facilitating laptop orchestra sessions one of the things I am very keenly aware of is the delightful extent to which people tend to imitate rather than copy. When a code model is presented from the 'centre' outwards, there is invariably a spectrum that emerges – some follow the model literally or very closely, others change it systematically or in subtle nuances, while others still do something that responds to the model on some level but doesn't implement it. The idea from 'the top' never reaches 'down' unchanged. Conversely, there are moments when it is clear that lots of enthusiastic, informal, neighbour-to-neighbour sharing is happening around the ensemble, and yet the ideas shared in such moments do not necessarily constitute stable objects receiving additional votes in some kind of formal democracy. Ideas from 'the bottom' don't need to be aimed 'up' to produce interesting effects for direct participants or audience members.

One consequence of this emphasis on code sharing is a departure from the idea that live coding should begin from a blank screen. Live-coding pioneer Alex McLean writes, 'The development of software becomes part of the art in a very real sense; at the beginning of a typical live coded performance there is no code and no audiovisual output, but the output grows in complexity with the code' (McLean, Griffiths, Collins and Wiggins 2010). In the practice of the Cybernetic Orchestra we have often done exactly the opposite, starting from screens full of code, upon which orchestra members then begin to operate, transforming it in various ways and with various intentions (some specific models of operating on given code are discussed below).

Another interesting and particular consequence of the combination of pulse-based music with live coding is that participants, once they have some familiarity with the semantics of the code, essentially face an ongoing choice to opt in or opt out of the pulse. Especially in ChucK, which makes the passage of time such an explicit and central part of its notation, orchestra members are continuously making statements about time and about the pulse. The code to let a freely chosen duration pass is not very different from the code to wait for the next downbeat!

```
6::second => now; // let six seconds pass
Esp.downbeat => now; // wait for the next
downbeat
```

This is a promising area for the development of virtuosity and specialisation within our specific live-coding culture. A basic rhythmic idea is easy to 'put on the table', both in terms of informal communication and in terms of code. For example, a first orchestra member puts a certain sound on the table, repeating it on the global downbeat. Then a second orchestra member puts a second sound on the table, at first placing it squarely on beat 8, motivated by an intention to distinguish their sound clearly from the first member's sound. But now more nuanced or

virtuosos responses are also possible. The second player could immediately recognise that if their sound were to happen 25 milliseconds earlier it would connect in an interesting way with a rhythmic detail within the sound made by the first player, so they adjust their code to specify '25 milliseconds before beat 8'. Later, the first player could add an anticipatory event to their downbeat event, and this anticipatory event could combine rhythmic specifications in terms of fractions of the pulse with direct times in milliseconds.

4. COMPOSITIONAL MODELS AND PRACTICES

The following are models and practices that can be incorporated in the text of compositions (or improvisational plans) and that have been found to facilitate participation in networked pulse-based live coding.

4.1. Code-sharing

Examples of code in a given environment can form part of the score or performance instructions of a composition. This has several beneficial consequences, beyond the sheer fact that such code samples form part of the identity of the composition (in the same way that the provision of a fixed instrument does for many other laptop orchestra compositions). In the first place, it facilitates the participation of orchestra members new to live coding, because they can get a working sound immediately, then concentrate on making changes to code, without the extra pressure of having to build something up 'from scratch'. Secondarily, it encourages the informal sharing of code within the orchestra by starting from a code-sharing situation. In this model, the live coding activity of orchestra members potentially becomes more about *transforming* code than (supposedly) producing it from scratch.

4.2. Instruction scores

A basic model for composing with/for a live coding orchestra is to provide sets of instructions about what happens in each section of a multi-section form. The live coding activity then aims to realise the instruction, and may do so in a great variety of different ways: for instance, 'Continuously create long (15 seconds or more) crescendi–decrescendi combinations with low (60–140 Hz) sine tones of slightly unstable frequency, beginning to slowly fade out when the theremin player decides to join the texture.' Characteristic of this sort of instruction is the fact that it describes timbres and times, but also something for each orchestra member to listen to.

The composition *Seventy* (by David Ogborn) for marimba and a live-coding laptop orchestra employs a combination of the code-sharing and instruction-score modalities. The authoritative text of the piece consists of a page of instructions describing eight moments or sections of the form, describing both the kinds of gradual changes orchestra members should be making to particular live coding layers and the things they should be listening for in the overall texture (and from the live marimba player). Three examples of simple ChucK code are provided, one for each of the three pulse layers that each orchestra member controls at various points in the piece. This allows for a low entry fee – someone could participate in this piece with minimal knowledge of the ChucK syntax – but it also allows for 'no ceiling on virtuosity' – there is no hard-coded constraint on the complexity or expressivity of the way in which they depart from the given example code. Indeed, in a given performance there is no requirement that they start from the shared code at all, although I would say that I consider this extreme departure to be more acceptable if they have worked with the group previously and with the given code in order to be 'plugged in' to the (evolving) identity of the composition. Someone could even perform the piece with a hardcore 'start from a blank screen' aesthetic, or using other live coding environments.

4.3. Code as material

It is also possible to provide code in advance not as an example of a way of achieving a given end but as a specific piece of material to be transformed in specific ways. This model facilitates participation, as orchestra members do not need to begin with a large vocabulary in any given live coding environment – they just need to understand what they are to do to the specific code in front of them. At the same time, by making these changes and working with this code they are learning the consequences of the changes they make and thus building a generalised live coding vocabulary that will allow the development of virtuosity.

4.4. Physical performance

An additional model that allows for the identity of a live coding composition to be established, while encouraging both participation and virtuosity, is to include elements of physical, theatrical, gestural performance in the composition. In many ways we have only begun to explore the countless possibilities opened up by this model. Such gestural performance communicates well to an audience – and thus presents a viable alternative to the default live-coding visual modality of code projection. Gestural performance also forces orchestra members to become more aware of their body and posture in relation to the laptop, and introduces an element of physical activity into the laptop orchestra rehearsal, with benefits for attention and health.

The composition *Programme réduit* (by David Ogborn) employs both physical performance and the 'code as material' model. A single piece of ChucK code is provided to the orchestra members, which waits for a downbeat from our networked pulse server then sounds a full bar of 16 semiquavers with a simple sawtooth oscillator, with the frequency of the oscillator on each semiquaver controlled by a simple algorithm to randomly select a base pitch class and an overtone. At the beginning of the piece, this random selection covers a wide range of possibilities, but as the piece progresses the live coding orchestra members gradually reduce this range of possibilities so that the piece 'converges' on a single low pitch.

The physical-performance element of the piece consists in the fact that each time an orchestra member adds a new 'shred' (ChucK terminology for an independently executing thread, likely to be making its own contribution to the sound output from each device) they are to clap very obviously, loudly and dramatically on the downbeat, potentially giving rise to the illusion that their large clap gesture is triggering the burst of synthesised sounds that also starts on the downbeat. Anecdotally, some audience members and even some electronic musicians have been fooled into thinking the computer is responding to the clap! The piece ends when, after the pitches have converged on that single low tone, the orchestra members choose an opportune moment to cease live coding and simply clap on each downbeat. The piece thus ends with a complete transition out of the modality of live coding into gestural/physical performance. A quiet hi-hat-like pulse is started at the beginning and end of the composition in order to keep people oriented within the metre.

```
// Programme Reduit, by David Ogborn, 2011

// setup the signal network
SawOsc s => LPF lpf => Envelope e => PRCRev r => dac;
24 => Std.mtof => s.freq;
[24,26,28,30,31] @=> int bases[];
[1,2,3,4,5,6,8,10] @=> int overtones[];

// you might change oscillator gain, filter frequency, and reverb
0.288 => s.gain;
2800 => lpf.freq;
0.019 => r.mix;

// wait for global downbeat and attack (probably don't change this)
Esp.downbeat => now;
1 => e.target;
2::ms => e.duration;
Esp.beat => now;

// choose random pitches for remaining 15 pulses of the bar
for(1=>int x;x<16;x++) {
   bases[Std.rand2(0,4)] => int b;
   // *** PERFORM HERE, range=0 to 4
   overtones[Std.rand2(0,7)] => int o;
   // *** PERFORM HERE, range=0 to 7
   (b => Std.mtof) * o => s.freq;
   Esp.beat => now;
}

// decay (no point in changing this)
0 => e.target;
40::ms => e.duration => now;
4::second => now;
```

4.5. Improvisation with soloists

In most rehearsals and performances of the Cybernetic Orchestra we include a 'free improv' with the networked pulse server activated. In order to highlight the spatial nature of the orchestra to the audience, and in order to introduce distinct sections and moments into the form, we interrupt the jungle of freely evolved collective texture with moments where all but one member of the group finds a way to be silent, as quickly as possible. When mobility around the hall and visibility are good, I as the orchestra director have wandered the hall, physically stood next to each orchestra member and made a signal to the rest of the orchestra to 'find the nearest exit' (so to speak) and listen to the player I am standing next to. This gesture can easily be combined with pointing at a second or third player before making a 'relaxation' type gesture that acts as a cue to the orchestra that they should all re-enter the texture as quickly as possible.

As an alternative method of implementing this focusing mechanism when visibility or mobility (for the director) represent challenges, we've also used a networked chat protocol (espChat) in the same way: 'only Rob', 'now add Elise', 'everyone back in', 'let's gradually fade away to end this one' and so on. Typically, some friendly chatting takes place over espChat in addition to the chatting strictly required by the performance…

5. FUTURE WORK

There are a number of other models for ways of treating code that can be postulated, and which will no doubt be explored in future activities of the orchestra. For instance:

1. Using code as a mechanism for circumventing comprehension, creating a given piece of code of great complexity such that the nature of transformations to the code is difficult to intuit by mental projection, and must be experienced to be understood.
2. Transcoding: taking a given text (of any language or nature) and following systematic practices to

recode it into viable sound synthesis code. In addition to exposing novel sound structures, orchestra members will develop comfort and familiarity with the syntax of the language into which they are transcoding.

3. Code as textual communication: facilitated by multiple projection systems, or some other system of displaying the coding activities of everyone in the group, exploiting the possibilities of code for symbolic and expressive communication that does not have any meaning from a sound synthesis standpoint. For instance, naming objects in a code environment in a systematic way over the group (rather than using the usual 's', 'x', 'y2', etc. to name variables) or further developing the idea of 'playing the comments' in such a way that the comments can be part of the sound synthesis program (if uncommented) or part of a free visual, literary poetry.

The practices documented in this article have been developed around a laptop orchestra of 8–16 people. The majority of these orchestra participants have been current university students, although across a range of departments, disciplines and backgrounds, and with some non-student participants. The major direction of our future research will be around scaling these practices up to accommodate larger and yet more diverse groups. This involves technological development (making changes and adding capabilities to the espTools in ways that facilitate large groups, and porting the espTools to ubiquitously available mobile devices) as well as the refinement and expansion of social and compositional practices.

REFERENCES

Collins, N., McLean, A., Rohrhuber, J. and Ward, A. 2003. Live Coding Techniques for Laptop Performance. *Organised Sound* **8**(3): 321–30.

Emmerson, S. 2008. *Pulse, Meter, Rhythm in Electro-acoustic Music*. Paris: Electroacoustic Music Studies Network http://www.ems-network.org/ems08/papers/emmerson.pdf.

Jenkins, H., Katie, Clinton., Ravi, Purushotma., Alice, J.Robinson. and Margaret, Weigel. 2006. *Confronting the Challenges of Participatory Culture: Media Education for the 21st Century*. Chicago, IL: The John D. and Catherine T. MacArthur Foundation.

McLean, A., Griffiths, D., Collins, N. and Wiggins, G. 2010. Visualisation of Live Code. Paper presented at the conference Electronic Visualisation and the Arts, London.

Neill, B. 2002. Pleasure Beats: Rhythm and Aesthetics of Current Electronic Music. *Leonard Music Journal* **12**: 3–6.

Smallwood, S., Trueman, D., Cook, P.R. and Wang, G. 2008. Composing for Laptop Orchestra. *Computer Music Journal* **32**(1): 9–25.

Wang, G. and Perry, Cook. 2004. On-the-fly Programming: Using Code as an Expressive Musical Instrument. *Proceedings of the 2004 International Conference on New Interfaces for Musical Expression (NIME)*, Singapore.

Wessel, D. and Wright, M. 2002. Problems and Prospects for Intimate Musical Control of Computers. *Computer Music Journal* **26**(3): 11–22.