Free as in BEER: Some Explorations into Structured Improvisation Using Networked Live-Coding Systems

Scott Wilson, Norah Lorway, Rosalyn Coull, Konstantinos Vasilakos, and Tim Moyers

The Birmingham Ensemble for Electroacoustic Research
Music Department, University of Birmingham
Edgbaston, Birmingham B15 2TT, UK
www.birmingham.ac.uk/facilities/BEAST/research/Birmingham-Ensemble-for-Electroacoustic-Research.aspx
s.d.wilson@bham.ac.uk
norah.lorway@gmail.com
roziz_coull@msn.com
konstantinos.vasilakos@gmail.com
oysterhouse82@yahoo.com
beerensemble@contacts.bham.ac.uk

Abstract: Much improvised music that has developed since the advent of free jazz has been concerned with the imposition of structure, often through systems of directed improvisation, or through the use of rule-based approaches (e.g., game pieces). In this article, we explore the possibility of a networked live-coding system as a structural intervention mechanism *par excellence*, through the discussion of two pieces from the repertoire of the Birmingham Ensemble for Electroacoustic Research.

The relationship between live coding and improvisation is generally understood in terms of the latter being an implicit aspect of the former, or as the former being a specific case of the latter (see, for example, the discussion in McLean and Wiggins 2010). Live coding provides one fertile solution to the problem of interface design for musical performance, with rich implications for improvisational practice. Unlike most computer music performance systems, which tend to consist of relatively fixed configurations of graphical user interface (GUI) widgets and/or gestural controllers, "code as interface" allows radical intervention and reconfiguration of musical systems while they are running. On the one hand, it becomes possible to do many previously unimaginable things; on the other, it is often not possible to do them very quickly.

In terms of this flexibility at least, live coding might be seen as an improvisational interface *par excellence*, albeit one often lacking in agility and deftness. Like much post–free-jazz improvisation, however, the practice of live coding can present problems in terms of creating formal articulation. Moreover, as in free improvisation, a substantial amount of live-coded music consists of a gradual flow in which layers are added and subtracted, usually with some modifications in the interim. Post–free-jazz improvised traditions have developed various strategies for dealing with this issue. In exploring these approaches, we might consider examples drawn from the works of John Zorn and Anthony Braxton.

Arguably Zorn's best-known work, *Cobra*, composed in 1984, is—like many of his early compositions—a "game piece." It consists of a number of rules about how and when musicians may contribute to the piece and interact with one another. These different "improvisational strategies" are indicated using color-coded cue cards. A prompter holds these up in advance, so that performers can prepare, and then cues their activation with a downbeat (Brackett 2010). Braxton's "Ghost Trance" compositions (composed between 1995 and 2006) consist of a pulse layer (usually notated with a nonspecific clef and lacking details such as tempo or articulation) that can be combined with precomposed materials and improvisation. All three are subject to a set of (sometimes optional) rules about how to combine them and a detailed system of directed improvisation (what Braxton calls "Language Music"), which can modify musical materials while in use (Fei 2001). Braxton (2008, p. 2) refers to these works

Computer Music Journal, 38:1, pp. 54–64, Spring 2014
doi:10.1162/COMJ_a_00229

as a "unique activity forum," and they might be understood equally well to constitute a toolkit for creating and structuring a performance as to be compositions in the traditional sense. One thing that these approaches have in common is the use of intervention mechanisms to facilitate aspects of musical interaction that can be difficult to realize in strictly "free" performance (e.g., coordination, interruption, and other forms of formal or structural articulation). Interestingly, both of these examples allow a certain gap of time between changes being chosen and being executed, something that live coding practice sometimes take to extremes.

The concept of freedom in music, and particularly in improvised music, can be articulated in various ways, with free jazz (i.e., jazz freed both from the constraints of precomposed limitations and from limits on individual expression) representing only one concrete choice. As in the Braxton and Zorn examples, the sorts of freedom that we have chosen to explore in the music we will discuss in this article are different. Rather than a freedom that valorizes virtuosic individual expression and a radical subjectivity freed from any preconceived musical structure, we explore something more akin to a social contract—i.e., the freedom of agents to choose, at least for a time, to limit their expression within constraints that are mutually agreed upon and are mutually (and hopefully musically) beneficial. Intersubjective compositional limitation is intended to provide a shared space in which the musical community can craft its expression.

One might raise the point that this is hardly unique, and is in fact true of most music-making. More interesting are the possibilities that "code as interface" provides, in combination with the intervention and formal articulation that networked music systems so superlatively provide. In this article we will explore some concrete examples of the possibilities afforded by this configuration. Two pieces in particular will be discussed. The first, *Telepathic,* provides a common formal procedure and time base. The other, *Pea Stew*, imposes a limit on material, which is entirely generated from a fully meshed audio feedback network interconnecting the performers.

BEER Basics

In 2011, the Birmingham Ensemble for Electroacoustic Research (BEER) began as an exploration of the potential of networked music systems for structured improvisation. Working primarily in the SuperCollider (SC) language (available online at supercollider.sourceforge.net), BEER uses the Republic (de Campo et al. 2012) and, more recently, the Utopia (Wilson, Rohrhuber, and de Campo 2013) extensions to support code sharing and other network activity. In both cases we use the JITLib (Just in Time Library) classes in SC for basic live coding functionality. The ensemble varies in size from three to five players (occasionally augmented by guests) in performances, and when possible, performers use individual stereo loudspeaker pairs. This approach gives each performer a small stereo field to work within, and differs somewhat from the more "instrumental" approach taken by groups such as the Princeton Laptop Orchestra (PLOrk), which favor single hemispherical loudspeakers for their instrument-like acoustic dispersion. In some ways, this provides the spatial advantages of stereo amplification in that performers can project a field of sounds, rather than function as a point source, while avoiding the risk of each performer's contributions being made anonymous within the single stereo field that would be produced by a shared public-address system. This practice is also reflective of our desire to move beyond the limits of the instrumental conception of electronic music software design that characterizes much laptop orchestra practice. (Instrumentality has its uses, but why limit yourself to it?)

BEER has experimented with the common live coding practice of projecting code (a practice pioneered by groups such as Slub; see slub.org for video documentation), although we have not done this consistently. One reason is simply that this raises additional technical demands, which can be difficult to meet in some situations. The second is that, although we appreciate the way in which code projection can enhance an audience's experience by making clearer connections between performer activity and the resulting sound, it is nontrivial to implement with five or more performers. We

feel that some of the existing solutions to multi-performer code projection, such as periodically switching between different performers' screens (notably by the group Benoit and the Mandelbrots, see www.the-mandelbrots.de), undermine this effect somewhat by interrupting an individual's coding in progress. We have, however, experimented with an alternative approach, using a custom class called *BEERWatcher* class, which captures and projects each bit of code as it is executed. This avoids the interruption problem, but sacrifices the occasionally fascinating aspect of observing a programmer's thought process by watching them edit and correct code before it is executed. In any case, our experiments in this area are ongoing, but it is our hope that even in cases where we do not project code, the music itself and other aspects of the performance experience are of sufficient interest.

As with other improvising live coding ensembles using the Republic software (e.g., PowerBooks_UnPlugged, see www.pbup.net/s), the practice of code sharing is intended to lead to a greater unity of musical result and to facilitate the creation of newly improvised material that is sonically or behaviorally related to what other performers have already produced.

Compositions (if they can be properly called that) are developed in workshops and performances, with the entire ensemble contributing to the development of what might initially have been individually designed and conceived systems.

Telepathic

Telepathic is a piece for an unspecified number of players, which makes use of a networked system to facilitate coordination and formal articulation. Live coding performers improvise within the limitations imposed by this framework.

Implementation

The backbone of *Telepathic's* networked system is a shared master clock, which broadcasts its beats, tempo, etc., to interested listeners on the network.

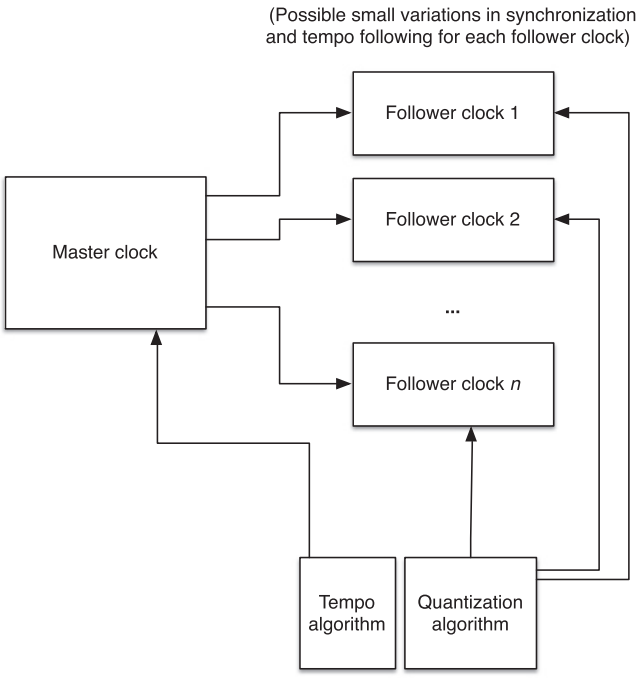
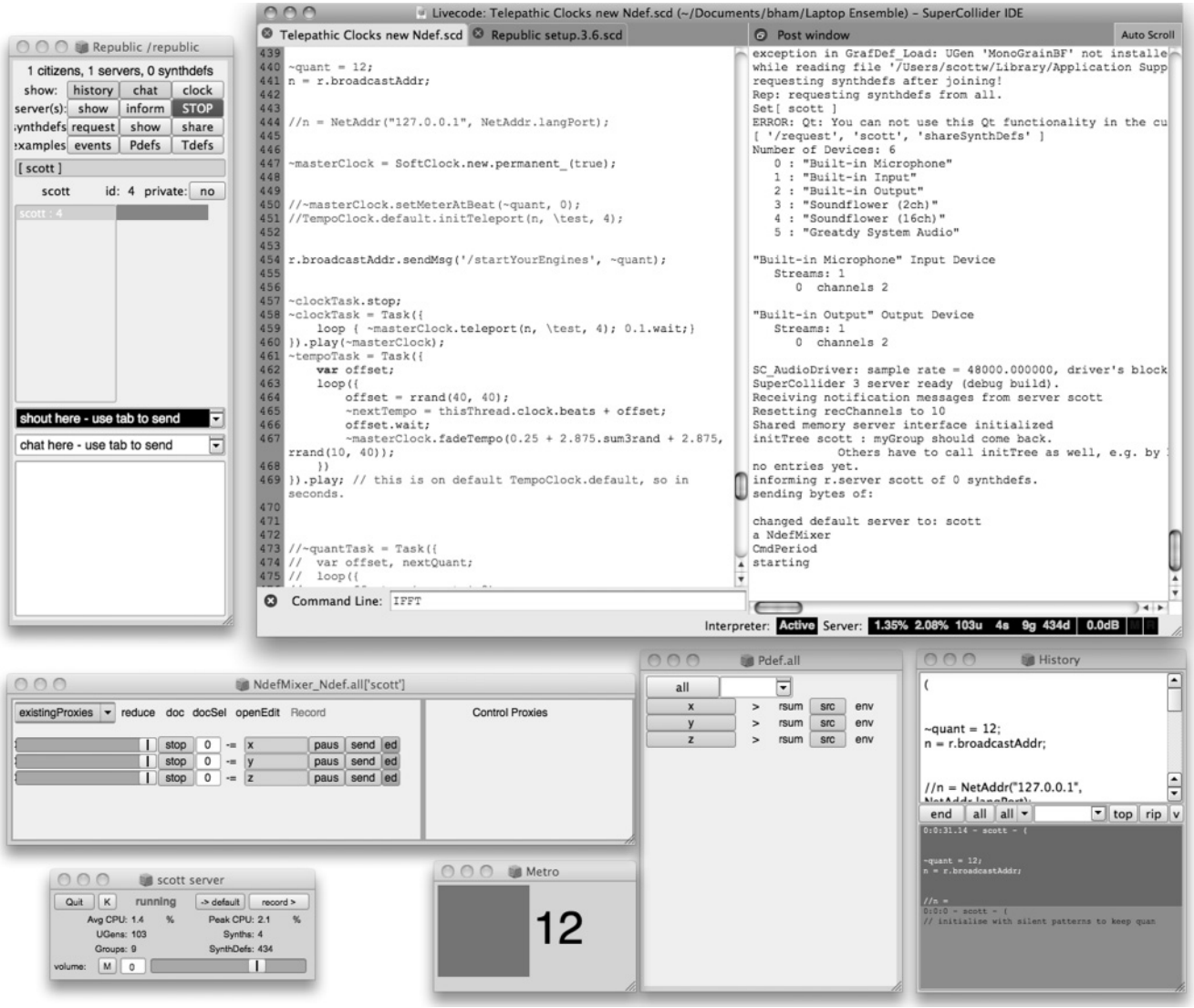


Figure 1. Networked clock configuration scheme for Telepathic.

Each member of the ensemble has a corresponding follower clock, which attempts to match the master clock's tempo. These are implemented using the *Listening Clocks* quark, developed by Rainer Schütz, Alberto de Campo, and Julian Rohrhuber (see the discussion in Schütz and Rohrhuber 2008). This allows for certain variations in behavior in terms of how clocks follow and converge, perhaps modeling an ensemble-like behavior rather than attempting to maintain strict synchronization. (They provide the suggestively named parameters "confidence" and "empathy.") This allows performers to make use, at least optionally, of a shared time base, and to easily schedule events that exhibit rhythmic relationships to material already playing. An algorithm creates periodic sweeps to new tempi in the master clock, which the follower clocks attempt to match (see Figure 1).

Each performer controls up to three streams of events. A second algorithm imposes a quantization process, which aligns all changes in event streams to multiples of some number of beats (see further discussion in the next section). As a general formal principle, this quantization cycle gradually decreases

Figure 2. An example GUI setup for Telepathic.



over the course of a performance from an initial value (typically 12 beats) to a value of one beat, after which the piece is ended with either an extreme accelerando or an extreme ritardando, which in either case leads to the clocks stopping at the close of the piece. A "metronome" GUI indicates the passage of beats, as well as the current cycle length. Beats that begin a new cycle are distinguished by color from other beats. Figure 2 shows the basic GUI setup.

As noted earlier, BEER performances generally involve the practice of sharing and modifying code created by each member. The use of concrete material (i.e., sound files) in this context can be problematic, however, because of the particular way in which the SuperCollider language abstracts sample buffers and the multitude of ways in which sound files might be shared. For this reason, we make use of a custom SuperCollider class called BEERffers. This is essentially a global dictionary that

allows programmers to look up objects representing local sample buffers by a label (generally generated from the name of a sound file). It allows performers to query the available files. Furthermore, and most importantly, by abstracting away local file paths, any shared code that references buffers in "BEERffers style" is made reusable on any machine. In practice, we have worked by copying sound files from a shared folder to a local location and then using the BEERffers class to manage them. In principle, however, this approach also allows shared materials to be streamed over the network, loaded from a single network share, or acquired in some other fashion, because it separates reference from location.

Playing Telepathically

Telepathic has one important aspect in common with some of the works of John Cage, in that it treats form and content as separable entities (Pritchett 1996). Because we are free to use different materials each time the piece is performed (and do), *Telepathic's* identity as a work of art might lie not so much in the specific materials used in any given realization, but rather in the common formal trajectory and likely behaviors imposed or facilitated (or encouraged?) by the tempo and quantization algorithms.

These include the following:

1. *Rhythmic interaction based on a shared pulse*. In principle, this is not required, but it is very easy to achieve. The quantization mechanism limits when changes in the stream specifications can occur, but does not constrain the output of the streams themselves. Therefore, rhythmic patterns that do not correspond to the current cycle length, polyrhythms, and even poly-tempi relationships (providing a tempo is specified relative to the master clock tempo) are all possible and straightforward. In practice, complex rhythmic behavior may result.
2. *Synchronization of mid-level formal change*. The quantization algorithm makes it very likely that significant changes in streams made by multiple players will occur simultaneously, particularly in the early stages of a realization when the cycle lengths are longer. Commonly, live-coding ensembles rely upon numbers of performers to compensate for the relatively long gestation period that can be required for setting up a substantial musical change. Although this can make performances seem more nimble, with sufficient change occurring as multiple performers bring slowly implemented developments to fruition at different times, as in free jazz this encourages the emergence of a kind of "layering" form. The quantization algorithm inverts this situation by forcing changes specified in the same cycle to occur simultaneously, as if by "mind-reading." In this sense, it functions in a manner similar to Braxton and Zorn's directed improvisation strategies, albeit without the same immediate intention.
3. *A gradual intensification as the piece progresses*. The decreasing duration of the cycles both increases the rate of mid-level formal change and decreases the likelihood of musically meaningful synchronization of that change. (General alignment of pulse is of course still possible and likely.) Subjectively, this tends to result in a perception of gradual intensification, in a manner perhaps analogous to the effect of increasing the rate of harmonic rhythm while maintaining the underlying pulse in tonal music. We have found that, as performers become aware of this trajectory, they come to pace their patterns of coding to accommodate it. This feel for changes in a larger-scale rate of change, as distinct from subdivision of pulse, has interesting effects on coders' perception of their live coding practice, and might find parallels in things like Javanese gamelan performers' experience of changes in irama:

> The concept of irama consists of two aspects: the rate of temporal flow and temporal density. Temporal density is the primary factor in irama. It is a process

> in which time-space in the structure of a *gendhing* [composition] is being contracted or expanded. A contraction of time-space means a decrease in the number of pulsations (i.e., temporal density) of certain instruments; and an expansion means an increase in the number of pulsations (Sumarsam 1995, p. 156).

4. *A likelihood that materials will transition between rhythm and texture*. The changes in tempi imposed by the tempo algorithm are randomly determined, and it is not possible to predict when a tempo sweep will begin, exactly how long it will last, or what its end state will be. The tempo changes are also potentially extreme, with target tempi ranging from 15 beats per minute (BPM) to 360 BPM. Thus, a rhythmic pattern consisting of thirty-second notes at a 60-BPM quarter-note pulse might cross the threshold of pitch during a tempo sweep. In practice, it is more common for clearly distinguishable streams of events to fuse into a texture in perceptual terms. This also has an effect on people's live coding practice, because thinking "across" that threshold and maintaining a general awareness of potential future tempo changes can be musically rewarded. Conversely, failing to account for the CPU load implications of a potential increase in event density can lead to musical disappointment!

Broadly speaking, we feel that *Telepathic* is interesting both in terms of the extent to which the musical results are a product of the limitations that the system imposes, and in the ways in which it changes performers' relationships with their live coding practice.

Pea Stew

Pea Stew is a work based on audio feedback, with an arbitrary number of performers sharing audio streams over a wireless or wired network. Performers intervene in the signal chain using live-coding techniques to process incoming streams.

Precedents

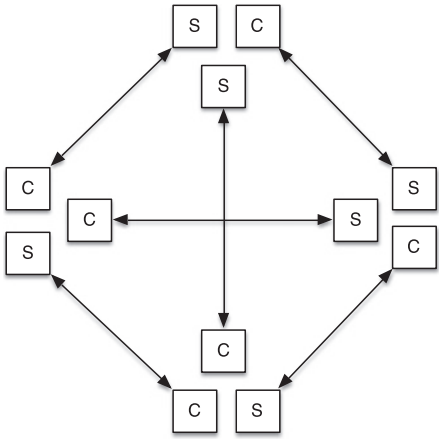
There is of course a large body of "feedback music" (for an overview, see Sanfilippo and Valle 2013). Of particular relevance to *Pea Stew* are works such as David Tudor's *Rainforest IV*, which makes use of a series of objects that serve as resonators for electronic signals (in some versions with signals circulating in "networks" of objects, cf. Driscoll and Rogalsky 2004); *Global String* by Atau Tanaka and Kasper Toeplitz, which uses a wide-area network as part of resonant system involving a multi-site art installation (Tanaka and Bongers 2001); SoundWIRE, a sonification tool which uses network latency as the delay component in a Karplus-Strong algorithm in order to make quality of service variations audible (Chafe et al. 2000); and Toshimaru Nakamura's "No-Input Mixing Board" approach, which uses feedback loops and processing to manipulate sound originally deriving from circuit noise, rather than an external source (Meyer 2003).

As is likely apparent from its name, however, *Pea Stew* is most immediately inspired by Nicolas Collins's *Pea Soup*. Originally an analogue work, *Pea Soup* consists of one or more feedback loops, each made up of a microphone, a limiter, a phase shifter controlled by an amplitude follower, filtering, and a loudspeaker. This creates a "site-specific architectural raga," in which feedback is controlled, and different patterns of pitches emerge due to the phase shifter changing the resonances of the system (Collins 2011).

Implementation

Pea Stew uses the JackTrip extension (available online at code.google.com/p/jacktrip/Jack) to the Jack audio system (www.jackaudio.org) in order to share audio streams over a network. JackTrip requires an individual pair of client and server "devices" for each connection, which makes configuration non-trivial. In order to simplify setup, we have written a custom SuperCollider class to

Figure 3. A Republic-JackTrip network topology for four players. An "S" represents a JackTrip server device, a "C" a client device. Each cluster of (in this case) three devices represents an individual node on the network, i.e., a performer's computer.

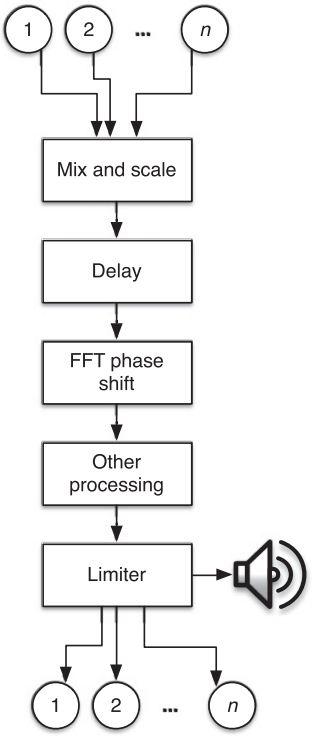


deal with this. Called RepublicJackTrip, it allows for the straightforward creation of a network of audio streams between an arbitrary number of users, automating the allocation and creation of JackTrip clients and servers. (Groups of sizes from 2 to 10 have been tested with successful results.) RepublicJackTrip implements a fully meshed network topology (see Figure 3).

In the current configuration, each player runs a basic processing node that collects and scales the signals sent from other players. These are summed and then fed into a delay. The output of this is then sent to a process based on a fast Fourier transform. This does the phase shifting à la *Pea Soup*, but on a bin-by-bin basis, with each bin's phase being shifted according to its magnitude. In testing, this seemed to allow for more complicated sonic results than a single phase shift. ("More complicated" refers here to the greater number of pitches in play—which may or may not be desirable—and the patterns of pitch movement. It does not represent an aesthetic judgment.) The phase shift may be "flipped" by a performer, reversing its direction, either for all bins, or on a bin-by-bin basis. Doing this can create more variety and contrast between different performers' outputs.

At this point in the signal path, performers intervene, using live coding techniques to alter the sound. Processing can be serial or parallel, and, in practice, we have not limited the sorts of processing used. The output of this processing is then sent to

Figure 4. Signal path for each player's computer in Pea Stew. *The numbers at top and bottom represent JackTrip receives and sends from the* n *other players. "Other processing" can include filtering, enveloping, pitch shifting, etc.*



a limiter in order to keep the overall signal level in control.

The output from each player's processing is sent to all other performers via the JackTrip connections and played over one or more loudspeakers (see Figure 4).

Playing *Pea Stew*

As an digital audio network lacking inputs, *Pea Stew* is theoretically noiseless. That means that a performance cannot rely on system noise as an initial source à la Nakamura, and the network must be primed with some signal. We have found low-level pink noise or impulse streams to be useful for this purpose. Once the signal is introduced, players turn up one or more of their inputs until the system starts resonating, at which point the noise source is generally removed. Although it is relatively easy to set the gain low enough at some node to cause the resonance to stop, in such cases it is generally not

necessary to re-prime the system. Because the signal is merely inaudible, rather than silent (at least not until enough time has lapsed that rounding errors reduce samples to a value of zero), increasing the gain will cause the signal to return. That said, in some cases we have found it interesting to inject a new signal at some point in a performance, as this may activate new modes of resonance.

Once seeded, a performance involves improvisation through live processing of the audio stream. In addition to live-coding techniques, the code for the basic processing node provides a GUI interface that allows performers to scale the input from each of the other players, weighting it to different degrees. This also allows for the effective reconfiguration of the network topology in real time, since by muting the correct inputs any possible topology can be achieved. Because each connection between players can be bi-directional or mono-directional, the number of possibilities is large, and with larger numbers of performers, two or more isolated sub-networks (perhaps with signal paths of different lengths) can be created. In performance, we have found varying the network topology to be very useful. The GUI allows for negative gains, i.e., phase inversion of another performer's audio stream as it passes that point in the network. The GUI also allows the performer to control a number of parameters, such as the lag time for the phase shift, and the input delay time. By lengthening the latter, a player can change the resonance of the system, effectively lowering the fundamental frequency of all audio paths passing through that node. The value of the delay time is initially set randomly, in order to avoid equivalent resonances between pairs of nodes.

Performers have used a variety of processing techniques, including distortion and modulation, pitch shifting, granular techniques, and creating rhythmic effects through application of amplitude envelopes. As one might expect, filtering is a very powerful tool in this context, and can radically change the output of the system. That said, we have found it to be a technique that is easily overused, and small occasional changes can be equally or more rewarding.

Playing *Pea Stew* presents challenges somewhat different from those encountered in normal live coding situations. The sound produced is truly collective, and although the output heard at each performer's loudspeaker (or loudspeakers) will be different (often quite surprisingly so), any individual change made is likely to have an effect on the entire network, or at least on any sub-networks of which the performer is currently a part. Even straightforward processing techniques are rendered unpredictable (notably because most processing is recursive along some path), and the experience has been likened by one member to "trying to push around a room-sized blob of jelly." Each action a player takes has some effect, but it is often impossible to anticipate exactly what it will be. It thus differs from *Telepathic*, which is more traditional in the forms of autonomy allowed to performers. (*Telepathic* could be understood as several performers working independently within an agreed structure. *Pea Stew* might be better characterized as several performers simultaneously playing a single instrument.)

Although *Pea Stew* tends to force performers outside their comfort zone in terms of the predictable use of knowledge and skills, the experience of playing it is often delightful and surprising in its indeterminacy. The familiar patterns of live-coding improvisation, in which one usually works with individually derived and known (or at least somewhat predictable) material, are broken. The "material" of a performance of *Pea Stew* is at once collective and collectively limited, but derived in such a way that those limits are not entirely knowable (see Figure 5). We have found that once one surrenders the aspiration to control one's musical material (and perhaps the aspiration towards demonstrable virtuosity!) working in this way can be most rewarding.

Future Work

Telepathic continues to be refined in workshops and performances. In particular, we are interested in extending the capabilities of the system for the sharing of materials. In a general sense, simply varying the basic parameters (rates and ranges of quantization cycles, ranges of tempi, etc.) also provides plenty of scope for variations on the general

Figure 5. BEER performing Pea Stew *at the Bramall Festival of Music. (Photograph by Aaron Croston, Birmingham, UK.)*

concept of the piece, and we continue to explore these possibilities. The piece is currently being reworked using the new Utopia network music library (Wilson, Rohrhuber, and de Campo 2013), which allows for a more modular approach to networked music applications.

Development of *Pea Stew* is also ongoing, and has already involved many hours of "workshopping" and testing, as well as a number of performances. In future versions of this piece, we would like to experiment with several different aspects.

One of these is the imposition/generation of musical form (most particularly in the sense of enabling the perception of musical structure), which, as noted earlier, is one of our major interests. One possibility for this would be to move through a number of randomly or predetermined topologies in a piece. Another might be to make use of slightly different basic processing for different sections. (We have tried a number of variations in the phase-shifting algorithm, and they do exhibit different characters.)

Another possibility would be to allow processing that is not recursive (i.e., processing whose output is only to the loudspeakers, and is not sent to other nodes of the network). Currently this is only possible in cases where one performer is receiving audio from the others, but where the others have set the input from the first performer to zero. Although, in some senses, this would depart from what we feel is the collective spirit of the piece, it would allow performers greater control over their own sound.

One final possibility to explore would be performances with large numbers of players. In practice,

we have found four or five to be more satisfactory (although different in character) than two or three performers, with a good balance between stability, complexity, unpredictability, and the ability to influence the collective result. It seems likely that large ensembles would further diminish the influence of individual performers, but would provide more opportunities for interesting sub-networks. Network performance might be a concern, of course, and it might be worth testing this on a wired network rather than our more usual wireless one.

We also continue to refine other pieces in our repertoire and to develop new ones that explore different concepts of intervention or limitation. One such work in progress requires performers to live code a number of streams specified by type, using categories that need not be mutually exclusive—for example, "rhythmic," "noisy," "melodic," or "energetic." This extends the shared clock and quantization approaches used in *Telepathic* by implementing a control algorithm that determines which streams generated by which players are audible at any given moment, in the process generating characteristic rhythmic structures in some sections of the piece.

As noted earlier, the question of whether or not to project code, and how that might best be done, remains an open one for us. Given the sort of work we are undertaking, other ways of enhancing audience experience and perception might also be worth considering (for example, making imposed musical structures and points of formal articulation visually explicit).

Conclusions: Free as in BEER

As noted in the introduction, we feel that a networked music system is a small-scale example of a useful social contract. As performers and participants in what is a social—as well as artistic—activity, we agree to certain restrictions and responsibilities in exchange for the new possibilities that arise from that agreement. We have presented examples of such contracts, in the form of the two pieces described here; many others are possible. Limitations, in the form of networked systems, not only afford us the benefit of things like greater coordination, but might also free us from our own limiting habits, forcing us into musical territory that we might otherwise never have the opportunity to inhabit. (Again, we find commonalities with Cage's desire to avoid the musical limitations of his own taste; the separation of form and content is an effective basic tool for this.) A common, if not particularly nuanced, critique of free improvisation is that performers often rely upon their "licks" (i.e., they are really only free to do what they already know how to do). Although we feel that this is too simplistic a description of what actually happens in instrumental improvisation, we recognize the danger it describes and that this is equally true of live coding as a practice.

Live coding does not free us from the limitations of our "instruments" (these instruments are arguably just more flexible and less specified; thus, not necessarily a strength). It does afford us, however, the opportunity to avoid the narrowly conceived instrumentality that typifies much interface design for live electroacoustic performance, particularly within the context of laptop ensembles. To that freedom, we add the freedom to choose to limit ourselves, as well as our habits, behavior, and expression—choices that we feel can enhance live coding as an improvisation practice. We offer the proposition that we are all free to design tools that will allow us to enjoy the musical freedom that such limitation brings.

(Recordings of the pieces discussed in this article can be found at soundcloud.com/beer-ensemble.)

References

Brackett, J. 2010. "Some Notes on John Zorn's *Cobra*." *American Music* 28(1):44–75.

Braxton, A. 2008. Liner Notes to *QUARTET (GTM) 2006*. Important Records Imprec184.

Chafe, C., et al. 2000. "A Simplified Approach to High Quality Music and Sound over IP." In *Proceedings of the COST G-6 Conference on Digital Audio Effects*, pp. 159–164.

Collins, N. 2011. *Pea Soup: A History*. Available online at www.nicolascollins.com/texts/peasouphistory.pdf. Accessed 12 August 2013.

de Campo, A., et al. 2012. "Towards a Hyperdemocratic Style of Network Music." Paper presented at the 2012 SuperCollider Symposium, 12–19 April, London. (The Republic quark is available at quarks.sourceforge.net. Accessed 12 August 2013.)

Driscoll, J., and M. Rogalsky. 2004. "David Tudor's *Rainforest*: An Evolving Exploration of Resonance." *Leonardo Music Journal* 14:25–30.

Fei, J. 2001. Liner notes to *Composition No. 247*. Leo Records LR306. Reproduced at www.jamesfei.com/247.html. Accessed 12 August 2013.

McLean, A., and G. Wiggins. 2010. "Live Coding towards Computational Creativity." In *Proceedings of the International Conference on Computational Creativity*, pp. 175–179.

Meyer, W. 2003. "Toshimaru Nakamura: Sound Student." *Perfect Sound Forever*. Available at www.furious.com/perfect/toshimarunakamura.html. Accessed 12 August 2013.

Pritchett, J. 1996. *The Music of John Cage*. Cambridge, UK: Cambridge University Press.

Sanfilippo, D., and A. Valle. 2013. *Computer Music Journal* 37(2):12–27.

Schütz R., and J. Rohrhuber. 2008. "Listening to Theory: An Introduction to the Virtual Gamelan Graz Framework." In *Grazer Beiträge zur Ethnomusikologie/Graz Studies in Ethnomusicology*, vol. 22. Aachen: Shaker, pp. 131–194.

Sumarsam. 1995. *Gamelan*. Chicago, Illinois: University of Chicago Press.

Tanaka, A., and B. Bongers. 2001. "Global String: A Musical Instrument for Hybrid Space." In *Proceedings of cast0//Living in Mixed Realities*, pp. 177–182.

Wilson, S., J. Rohrhuber, and A. de Campo. 2013. *Utopia Network Music Library*. Available online at github.com/muellmusik/Utopia. Accessed 12 August 2013.