Purloined Letters and Distributed Persons

Julian Rohrhuber, Alberto de Campo, Renate Wieser,
Jan-Kees van Kampen, Echo Ho, Hannes Hölzl.

Sound reveals some other sides of things. Curious about the material body of something, we are used to knock or shake, and listen for a clue. This gives a direct impression of something close by but inaccessible, an intimate access to hidden properties. At the same time, a sound may be a distant attempt to communicate - someone knocking on the door, or the neighbour knocking on the ceiling. One may have to listen for a while until one can tell such a sign from the sound of someone driving a nail into the wall, and it may simply impossible to tell whether a door was slammed accidentally or in anger.

Why is it enjoyable to watch an acoustic music ensemble on stage? Certainly one of the reasons is that one can watch persons and objects in an interaction where it is not always certain whether a sound is a musical sign or a trace of the instruments' peculiar materiality — or both; while all that is audible can be traced back to some material thing, the same time it leads back to some human action that directed the energy to cause it vibrate. By making such a situation public, the stage condenses the unity of sound and action, and at the same time gives space for the whole field of uncertainty of the origin of each sound event.

When Michel Chion describes the *punch* in the film sound track as the paradigmatic moment of clear causality (*synchresis*) in the film,[1] he mentions it only as a culmination point of a general synthetic but consistent causality that film has to construct, while in everyday life this is part of the perceptual background: A sudden silent footstep would be a surprise, without doubt. Similarly, in the rendering of a composition the audience may be surprised, but the musicians are usually expected to be in a state of prediction and control. In improvisation or more experimental compositions, on the other hand, the stage becomes a place where the effects of actions are not the result of a global perspective; they are rather local and heterogeneous, and generally they are expected to be unexpected. Audience and players move toward a more similar standpoint - the action is less a rendering of an immaterially existing structure by means of the instruments, than the exploration of unknown territory or the interested waiting[2] for the response of someone or something.

Algorithmic composition takes a peculiar place in this landscape. Instead of writing down

---

[1] M. Chion. *Audio-Vision*. Columbia University Press, 1994, pp. 58-65.

[2] J. Rohrhuber and A. de Campo. Waiting and Uncertainty in Computer Music Networks. Proceedings of International Computer Music Conference, 2004.

a score of a piece, the composer sets up general rules for sound generation — and instead of passing them to instrumentalists to perform them, she hands them over to a computer for interpretation. Although such rules may be entirely deterministic, so that the performing process can derive every step and every action from the prescription, the outcome may nevertheless be surprising - very pedantic and even formal rules turn out to be a way to move beyond the expected; from a mathematical point of view the uncertainty about the outcome of a given calculation procedure is known as *decision problem*, which is why experimental methods are used by mathematicians to explore areas that cannot be reached inductively yet. Without a formal specification (how do we formally specify an acoustic impression?), experimentation is even less avoidable — compositional process becomes a navigation in which the algorithmic description is repeatedly modified and related to its resulting sound. Just like shaking a wrapped parcel, experimental programming gives access to unknown properties, which are now only partly material, but rather the properties of abstract chains of consequences. Step by step, the experiment explores modifications and new insights. Such algorithmic sound is a part of thought; it rather is the sonification of musical reasoning than its product.

From this point of view, a public improvisation with algorithms is no less plausible than experimenting with sounding objects on stage, and the numerous *live coding* approaches have led to interesting variety of performances.[3] Here, it is an ever changing dynamics of reprogrammed microcompositions that make up the improvisational situation, playing with the double time structure of processual change and change of the process.[4] In such an improvisation, the intervention of a performer is not carried by physical contact, but rather by a symbolic one: each programlet[5] can be thought of either as a new sounding object, or as a modification of ongoing algorithmic processes. Just like in the case of an ensemble playing with physical sound objects, algorithmic sounds are ambiguous: they may take the role of a musical sign,[6] or that of a peculiar property of the algorithm (or, both at the same time of course). Because algorithms are so complicated and their effect often difficult to predict entirely, the changes caused by the algorithmic process and those caused by the interventions of the performers can be hard to tell apart — even for the performers themselves. For an unhappy audience who is inclined to believe that a performer with the movement patterns of a clerk is probably playing back a cd while

[3] See *TOPLAP*: http://toplap.org/. Also good overviews are provided by: N. Collins, A. McLean, J. Rohrhuber, and A. Ward. *Live coding in laptop performance*. Organized Sound, 2004., or: Brown, Andrew R. *Code Jamming*. M/C Journal 9.6 (2006). 13 Aug. 2007.

[4] J. Rohrhuber, A. de Campo, and R. Wieser. *Algorithms today - notes on language design for just in time programming*. In Proceedings of International Computer Music Conference, pages 455–458. ICMC, 2005.

[5] Small program scripts that each may represent an independent cause of actions, or may affect other programs. In his book on artificial intelligence, Douglas Hofstaedter calls them *codelets*. (D. Hofstadter. *Fluid Concepts and Creative Analogies*. Basic Books, 1996)

[6] For a detailed discussion of algorithmic signs, see P. B. Andersen. *Semiotic models of algorithmic signs*. In *Algorithmik - Kunst - Computer*, pages 165–193. Synchron Publishers, 2003.

checking his mail, laptop performances are lacking appeal. Not only for this reason, in live coding, the code is usually projected on stage — this projection is the place where the negotiations of agency between performer and algorithm are made public. Again, the stage takes its role to compact the relations between performers, instruments and sound, only now on a level where the program text is in the position of the instrument.

Delocalised persons[7]

If the situation between musicians, sounds and algorithmic processes is already pleasantly or unpleasantly ambiguous, networked computers and communication media remove even more pieces of the background on which we are used to integrate the physical world. Action is not only passed on to go its own way by uncertain consequences, but also the location of action is neutralised away from its initial point.[8] In a network, the sound origin may differ radically from its source, and in this situation, inference of sound origin can become a quite absorbing activity. Here, the computer music stage may become again a display of unusual causality, as it used to be in the era of magicians' shows: There is always something the audience does not know. On the other hand, we find ourselves in the interesting situation where nothing needs to be hidden, because the interconnections of a sound network is so hard to reason about that the performers take on the roles as listeners. Rather than to completely give way to an ideal of an uncontrollable nonlinear system in which the performers are immersed, it is more about following the edge of understanding though. Listeners and players scan the 'sound surface' (the audible properties, or cues) to make inferences about the actions that 'caused' to them, and further away, the intentions that 'caused' the actions; even in 'uncaused' actions, the game of inference is essential.

In live coding performances, some members of the audience get upset about the fact that the code is projected on stage, while others, even non-progammers, find it an interesting insight into something that is usually hidden. These reactions could be explained by the fact that something that normally happens behind closed doors, or in silent preparation, is moved into a public space. Such a displacement of thought, leading to a situation of *public reasoning*, certainly is one of the central motivations of live coding. The double displacement — a shift from internal thought to external thought and from private studio to public space, integrates the algorithmic description, the program texts, and even commentary in the improvisation. Especially in network music, this form of algorithmic conversation becomes a second level of communication in addition to the acoustic. From a more structural point of view, the ambiguity between *sound-as-sign* and *sound-as-materiality* is paralleled by the ambiguity of algorithmic action — instead of trying to

---

[7] At this occasion, we would like to thank James McCartney for writing SuperCollider and making it open source; all the generous contributors to this language; the Academy for Media and Arts Cologne sponsoring the Warteraum series; and the Academy for Fine Arts Hamburg for sponsoring the symposium *changing grammars*, where *toplap* was founded.

[8] For a more detailed discussion with examples, see J. Rohrhuber. *Network music*. In N. Collins and J. d'Escrivan, editors, *The Cambridge Companion to Electronic Music (Cambridge Companions to Music)*. Cambridge University Press, 2008.

find out what person or object caused a sound event on stage, the curious listener is drawn to the traces of program text.

Combining live coding with network music, code is not only a public display of a performer's reasoning, but it becomes a constitutive element of conversation between the musicians. The ironic staging of problems of ownership, artisic skill and privacy, that is typical for participative and appropriation art, has been a part of network music concepts since its beginning, for instance the piece *borrowing and stealing* by *the hub*, where parts of the musical score is accessible to all participants, leading to divergent modifications of the privately edited sound material that may not have been intended at first. The collaborating players use each other's score as material, taking it apart, interpreting pitch as rhythm or rhythm as timbre and transforming it beyond recognizable connection to the original.[9] It is in this spirit, that takes up aspects of network art of the fluxus era,[10] that networked live coding makes code a means of communication and of collective musical thought. More literally, the program becomes a letter, a letter that circulates amongst the participants; interpreted by humans — as an expression of someone's thought, and by machines — as a recipe for synthesising sound.

Various systems have been written to bridge the anyhow rather small abyss between online chat and *multi-user-dungeon* on the one side, and *just-in-time-programming* on the other. Progamming languages that integrate network models and sound synthesis, such as *SuperCollider*, or experimental environments like Craig Latta's *Quoth*,[11] allow to create musical pieces which gain their character by the specific way sound and conversation interlock. While an important element of network music is the structural recombination and dislocation of agency (both of human and algorithmic nature) the reflection and negotiation of this structure itself can be made part of the game. The networking implementation of the *Just-In-Time-Programming Library,[12]* for example, may be used to dynamically restructure the synthesis graphs of each participant and at the same time create rules for synthesis combining the graphs of various groups of participants. Applying the semantics of chat channels as logic for interconnecting and distributing

---

[9] *"Technically, this was accomplished by providing each player with a separate area in the Hub's memory to which the melody that he was playing had to be published. Essentially, it was a shared database marked off into territories over which each individual player owned write privileges, but where reading (and copying) was free. (Of course, this was only a protocol that each player was individually responsible for programming on their own computer, there was no "server" that enforced the concept of territorial rights!)"* Brown, C., Bischoff, J., 2002. *Indigenous to the Net: Early network music bands in the San Francisco Bay area.* http://crossfade.walkerart.org/brownbischoff/

[10] A. Chandler and N. Neumark. *At a Distance: Precursors to Art and Activism on the Internet (Leonardo Books)*. The MIT Press, 2005.

[11] http://www.netjam.org/projects/

[12] *JITLib* is part of the SuperCollider sources. It has been written by Julian Rohrhuber, with considerable contributions by Alberto de Campo, and many others. For a discussion of the library, see J. Rohrhuber, A. de Campo, and R. Wieser. 2005. More extended functionality is available as an external library (*NetLib*), which will be published shortly.

processes, both program text and the resulting processes are selectively delocalised in the network. The ensemble *powerbooks unplugged*[13] (consisting of the authors of this paper) have developed a setup that uses a subset of this implementation — concentrating on the premise that the identity of each single person is delocalised as much as possible, every bit of program text is public, and everyone plays on everyone else's computer. To be more specific, we will now go into some details about this setup, and discuss some of the underlying ideas.

Powerbooks unplugged

The name of this ensemble may be read as an allusion toward the power games of pop bands and orchestras: Who writes the song, who plays the quietest instrument or the longest solo, who cannot adjust their tuning to the others? And the order of the stage: who stands back? Realising that as algorithmic performers, we are as much listeners as the audience, and we know as little as the audience, we decided to rid ourselves from some obstacles that seem often enough just be the historical remnants of display of power: the stage, amplification and artistic individuality. Having abandoned the stage, sitting amongst the audience instead, with laptops[14] connected by wireless network, we write and modify sound synthesis routines that can be heard through the pleasantly narrow band of built-in speakers. Using text as sound source only, every program becomes an open letter to the others, who may (or may not) read it, copy it and modify it further. A single, often random or algorithmically generated number in the code specifies for each sound event on which of the computers it will sound. In this way, a double delocalisation takes place: anyone's algorithm may become active on anyone's location, and, at the same time, anyone's algorithm may be copied by anyone else: Both sound and text stray about.

In acoustics, the delimitation between continuous and discrete is itself rather continuous. A steady sound may be thought of (and listened to) as a single individual, stretching over the whole duration, or as a series of a great number of very short individual grains.[15] Part and whole have a peculiar relationship in hearing. The fairly excessive use of microsound techniques by *powerbooks unplugged* fits well in the paradigm of a delocalised individual: A single sound may well turn out to be a group, as well as a sequence of

---

[13] http://pbup.goto10.org/.

[14] Yes, currently, some of them are actually still so called *"PowerBooks"*.

[15] C. Roads. *Microsound*. The MIT Press, 2004.

events spilled out all over the room may turn out to be the result of a single keystroke. In such clouds of sound grains, many spatially interesting effects result from the coincidence of events and of network delays: A cartoon version of special relativity.[16] While the origin of a sudden noise may be anywhere, its location is always *someone's* personal computer. Compared to a situation where the sound is amplified and redirected to a network of multichannel speakers, the individual agency is maintained — even if it is usually just the image of someone (or something) else's action.

Usually, a microsound algorithm is written in two interwoven parts: At first, a description of each sound grain itself, with parameters like frequency, duration, or other specific properties; On the other hand, a description of an arrangement of a number of such sound events (usually ranging from 1–10^4 per second). Both parts can be independently rewritten so that a procedure that created a series of sounds of one kind, may end up with a different kind exchanged underneath. In the current setup of *powerbooks unplugged*, the sound grain definitions are common to everyone, but may be globally, or locally modified over time. To give a simple example: initially, a routine that plays such a grain on each computer at a time, may make the impression of an individual object wandering around. At any time, anyone may modify anyone's sound definitions, so that at some later point in time, the very same routine may sound like different individuals making their special sound one after the other.

For keeping a little bit of orientation, the algorithms that arrange the sound events are local to each computer and (apart from some restricted exceptions) cannot be directly influenced by anyone else. This island of certainty turns out only to be the base of a whole range of irritainments,[17] that follow from the fact that such an algorithm, once applied, may be copied by anyone else. A sequence that I have just written could shortly after already be in someone else's hands. This distribution of codelets is maintained by a small code chat window, which displays the history of all participants, and that one may even use for

---
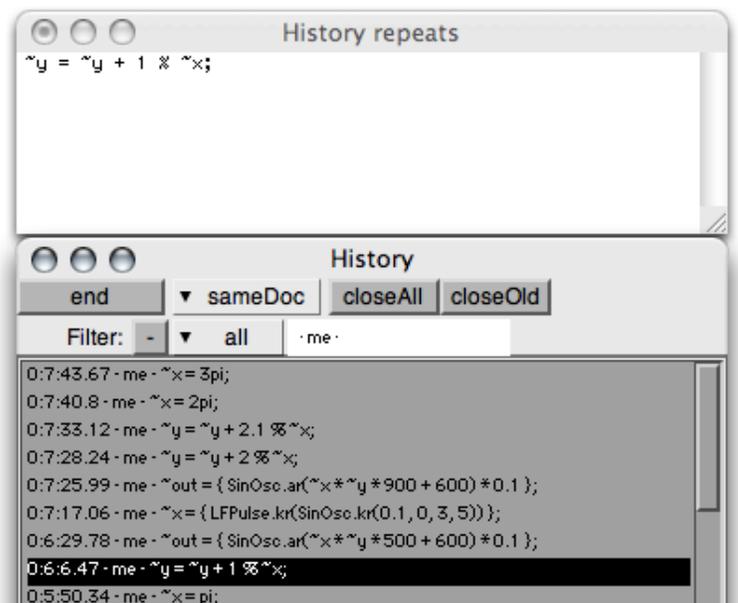
[16] For the musical effects of delay and waiting in netwo A discussion of delay in remote network performances *Supported Cooperative Work for Music Applications* (F latency on ensemble performance, May 2002. For thos network latency (the network-harp) may be an interesting example: C. Chafe, S. Wilson, and D. Walling. *Physical model synthesis with application to internet acoustics*. http://www-ccrma.stanford.edu/~cc/

[17] This very appropriate term was coined by Gordon Monahan, who is probably most famous for his "speaker swinging", and has conceived impressive and highly amusing network sound installations.

searching with keywords (see image).[18] It is supplemented with some other 'throw-away user interfaces'[19] that display status information and help with a general orientation about oneself. Apart from this, the interface is not much different from the of a poet's typewriter who writes strange little constructivist aphorisms that may or may not become proverbs for a short time.

---

[18] This class, called History, was written by Alberto de Campo.

[19] *'Throw-away user interfaces', or 'throw-away-GUIs'* are graphical windows that can be closed at any time without changing the system state, which is a relief from the representation-centeredness of usual computer applications.