Deautomatization of Breakfast Perceptions

Renate Wieser

The article describes an art project consisting of an 'acoustic breakfast'. Sounds of breakfasting were recorded and transformed in real-time to make communal interactive music. The author identifies the conflict between artists explaining algorithmic processes to participants, and the desire for an informal social situation focused on eating and drinking. Ideas from literary theorist Shklovsky are used to discuss automatized perceptions and the interest in defamiliarisation. Was the acoustic breakfast deautomatizing the computational processes? The article also discusses the same issue in relation to live coding.

(edited preprint of Chapter 8 in: Alex McLean and Roger T. Dean, (eds.): The Oxford Handbook of Algorithmic Music, Oxford University Press, 2018).

A while ago, a friend asked me to participate in an art project she organized. It was concerned with shared space and individual perception. My part was to think up one in a sequel of events, which were dedicated to different sense modalities. I was responsible for 'hearing' and decided to organize an 'acoustic breakfast'. There was an open invitation and anyone could come and join, the table was set, one could eat, drink coffee or tea, and at the same time, one could pick up the sounds which were created by all these actions with a self-made pickup. Two game pads controlled sound algorithms programmed in the programming language SuperCollider, allowing the guests, while having their breakfast, to record audio snippets, change, replay, and spatialize them across six speakers placed in various positions of the room.

The concept of a long breakfast was chosen, because it epitomizes an event of leisure, where there is no aim and no time pressure. Since many years there had been existing a broadcasting sequel in a free radio called "Sunday breakfast on Monday mornings", a title which I always loved for its great symbolic value. Much less subversive, the breakfast I organized was on a Sunday, and I also wanted to incite the participants to make their own music while having breakfast, to work in their time of leisure.

Of course I faced the problems usually faced in interactive art projects. My role as an organizer/artist involved far too much explaining of how the technology is meant to be used and this 'didactic' relationship didn't fit too well with the breakfast situation. In my experience, audio-based interactive works are sometimes difficult if there is no major visual layer, which helps to understand what is going on. Many of the breakfast guests were completely happy to only amplify sounds and remained unexcited about all the other possibilities provided. The ideas of the project would have worked out better perhaps in a space that made it easy to listen carefully. But of course: chatting is an element of the breakfast atmosphere. So it was a nice event indeed, which I remember with pleasure. But I could imagine more interesting sounds to be produced. My idea was to create an environment where the participants explore the possibility by listening to the changes they are able to obtain.

My practical and more theoretical work is focused since several years on algorithmic processes. It is an exploration of possibilities and an investigation in related discourses. With a background in fine art, philosophy and media theory rather than in music, I have been using programming languages for my art projects. I perform musical pieces and exhibit installations, but the ideas behind this practice are often connected to the mindset of art theory. As a participant and visitor of events of both music and art, I sometimes have the impression of a curious difference between these two worlds, despite the fact that their borders have become blurred over many years. The connection between music and contemporary art helps to break with cultural conventions.

For me, art is a particular way of thinking: it implies the exploring of social and political conditions, not with the means provided by academic disciplines, but by finding out new ways to explore, conceptualize and change them.

The concept of change and novelty here is not based on the idea of a world perfecting itself as time passes, an idea which one could typically find in idealistic early modern aesthetic. It is rather based on the renewal of our understanding of the world we are familiar with. An art practice of that kind is explored very compendiously by the Russian literary theorist Viktor Shklovsky in his famous 1917 paper 'Art as Technique'. Here, he coins the term *defamiliarization* (ostranenie) for a strategy that artists use in their work in order to bring awareness to automatized perceptions and behaviors:

"*If we start to examine the general laws of perception, we see that as perception becomes habitual, it becomes automatic. Thus for example, all of our habits retreat into the area of the unconsciously automatic; if one remembers the sensation of holding a pen or of speaking in a foreign language for the first time and compares that with the feeling at performing the action for the ten thousandth time, he*

will agree with us."[1]

And some pages further:

"*After we see an object several times, we begin to recognize it. The object is in front of us and we know about it, but we do not see it hence we can not say anything significant about it. Art removes objects from the automatism of perception in several ways.*"[2]

Contrary to common intuition, Shklovsky uses the term *automatism* to define routines which are neither planned nor programmed, and which largely elude conscious control.[3] The examples he refers to come from literature, but his approach is also very valuable for the understanding of art practices that involve programming. In Tolstoi's writing, Shklovsky observed techniques of bringing blanketed perceptions to awareness again. Within art he found ways not only of exploring automatisms, but also of exposing them to the readers' understanding. As a central example Shklovsky describes how Tolstoi writes about torture, describing the procedure as if there was no common word for it, as if he were describing something he knows nothing about. Like this, many of his examples deal with the defamiliarization of positions of power and the ways they interweave with habitualized behavior. So how does this apply to art forms that work with algorithmic procedures?

Examining the commonly used devices and artifacts helps to understand how they mediate social structures and power relations. Computers are ubiquitous in many places in the world, and most people in the richer countries are very familiar with the use of hard- and software. The multiple layers of algorithms that make them function is mostly hidden, however. Shklovsky developed his ideas with respect to literature, differentiating unambiguously between poetic and everyday language. Here, language plays an important role in the formation of automatized structures, as much as it provides the means to deautomatize them. Against this background, we can ask new questions about the relation between natural and machine language. Because a programming language is, by itself, entirely rule-based, including the source code, and taking into consideration automatisms and the artistic techniques that make them visible, we may elucidate the contrast between the purposive and the poetic, which are both equally implied in programming. Most people use software without being aware of its rules and language-like character. Because they are used to work with computers, they intuitively know the workflows and functionalities involved, but they don't mind its grammar. Here, technical necessities and wilful decision merge into an unknown

[1] V. Shklovsky, "Art as Technique," in Russian Formalist Criticism. Four Essays (L. T. Lemon and M. Reis, eds.), pp. 5–24:11, Lincoln: University of Nebraska Press, 1965.

[2] Ibid., p. 13.

[3] This definition is emphasized in: A. Brauerhoch, N. O. Eke, R. Wieser, and A. Zechner, Entautomatisierung. Schriftenreihe "Automatismen", Willhelm Fink Verlag, 2014.

terrain. Programming as a tool for artistic research, as well as for research about art, helps to illuminate and differentiate this terrain. By distinguishing between automatisms, automats and artistic working methods, this type of critical programming becomes crucial for the understanding of changing techno-social developments. Illustrating its automatized character, as well as the impossibility to fully automate it, code shares many properties with natural language. Like literature, an algorithmic artwork can be an investigation into an ambiguous relation.

In the digital world, rules are not only blinded out by habitual behavior but also through technical standards. In IT, the word *transparency* is used in an unexpected way: it here describes the relation between "user-friendly" graphical user interface and the hidden program code.[4] Transparent is the program if it is like a clear window, which is not to be noticed, a notion which is as common as it is misleading. This strange disappearance of the work of the programmers might be an important reason for the widespread anxieties with regards to programming. Of course, many people share strong disinclinations against such strategies of disempowerment.

During the algorithmic breakfast, the computer program between pick-up device and amplified output wasn't questioned much at all. Somehow in this commonplace atmosphere of having breakfast, the computer and the set of rules governing the interactive possibilities were hardly noticeable. To me, it seems an interesting requirement, that the visitor of an art space is challenged to learn about the technological aspects of an artwork. This requirement differs from the more spectacular pretension of installations which try to create illusion. I'm more inspired by art works which reduce transparency (understood in the IT sense of the term), by making the program code traceable. The rule structure behind an audible or visible output is understandable at least in principle by reading the code, even if the reader doesn't know any programming language. Evidence from live coding is very revealing here. As described by many people who witnessed a live coding concert, this practice is very often alienating and explanatory at the same time – a combination that reminds of the process of deautomatization. It is interesting that the mere presence of code can often cause strong reactions and become an important aesthetic component.

In my installations, the code was hidden in the black box. Thinking back, I start to suspect that hearing is a sense which tends to operate subconsciously. It seems to be really hard to wrest it from its automatized condition. It was part of the concept of the installations that the visitor tries to find out more about the otherwise hidden functionality through listening to the algorithmic sound. Often people were able to use the pro-

[4]See: I. Arns, "Read_me, run_me, execute_me. Code as Executable Text: Software Art and its Focus on Program Code as Performative Text," 2004.

vided tools, but verbalizing or even relating to the experienced processes proved difficult. This makes explicit what can also be seen as the diagnosis of a contemporary situation. Algorithmic processes have become ubiquitous in everyday life, not only because they are unavoidable, but because their appearance is increasingly simplified and streamlined. At the same time, paradoxically, many people experience this proliferation of modes of simplicity as an overburdening with alienating routines which provokes much anxiety. This development may be an explanation for the surprising relevance of Shklovsky's work, which may inspire new possibilities of how to face algorithmic procedures.