RENATE WIESER

THE WORK OF ART IN THE AGE OF THE UNIVERSAL MACHINE. INTRODUCING AMY ALEXANDER AND CARMIN KARASIC

This section serves as an introduction to the two articles which follow, both contextualizing an art installation. They are each written by an artist involved in their making. It is my concern to show why these artworks are particularly interesting for the study of automatisms. Just like a scientific paper, an artwork, understood as an investigation, questions an interrelation between perception and technique. With their art, their statements and their writings, the artists contribute to research, and thus bring awareness to their tools and methods as well as to the social and political context of their work.

Amy Alexander and Carmin Karasic combine their programming skills with the specific methods and modes of expression of the art context. In their contribution they each give insight into an interactive installation which alludes to political discourse with a sense of humor. Amy Alexander reinvestigates her project SVEN[1] (Surveillance Video Entertainment Network), which contextualizes software for automated surveillance in new ways. Carmin Karasic introduces SjansMachine, an installation that relocates social software from the world wide web into a local network which operates in an actual room with people able to meet in flesh and blood.

The code employed creates a familiar environment for the visitors of the installation. They recognize it on account of the ubiquity of computer based technology in everyday life. But it also differs from the familiarity we find in other spheres of daily routines. Unlike hardware, software is immaterial and based on a complex set of rules, normally hidden behind a user interface, a cover that creates the impression of a self-contained commodity – especially if we look at market-leading products, predominating in certain sectors. Consequently, what customers buy are not goods, representing solidified labor and concepts, but a set of digital instructions still open to improvement and mutation.[2] One could say that this mutational character is camouflaged by the ap-

---

[1] The article develops further and reconsiders former trains of thought. See Amy Alexander, "About... Software, Surveillance, Scariness, Subjectivity (and SVEN)", in: *Transdisciplinary Digital Art. Sound, Vision and the New Screen*, ed. by Randy Adams, Steve Gibson and Stefan Müller Arisona, Berlin, 2008, pp. 467-475.

[2] Still we deal here with commodities and it is important not to confuse de-materialization with de-commodification, as Rancière argues in: id., "Von der Aktualität des Kommunismus zu

parent solidity of the graphical user interface, where every change is labeled as software update, preventing the users from knowing in detail what changes are going to happen and how deeply they will change the products in use. The process of software development is far more complex than that of most of our daily goods, in the sense that the developers design and regulate the interactive potential of the device step by step. All these regulations influence the usage of software we own, or have access to in all kinds of situations, and which determines our way of life. Thus it is a plausible diagnosis that programming as an activity is something like a blind spot in the everyday use of software and computers.

Daily routines of individuals as well as those of social and political relations are structured by software. The research group *Automatisms* at the University of Paderborn is mainly interested in those routines which are neither planned nor programmed, and which are not the result of a willful action. It explores processes that largely elude conscious control. Behavior that grows into a habit, rooted not so much in the compliance with regulations but in repetition, with an arbitrary side to it. Automatisms can be experienced as something mechanical, but they are, by definition, not describable as technical automata.[3] No matter how complex technical automata are, there has to be a set of rules which the processes and their outcomes can be reduced to. Automatisms, even though they result in schemata, can't be reduced to such a constitutive policy. In differentiating automatisms from automation, a difference is made between two very similar concepts – using the notion of automatisms in conjunction with computer technology therefore calls for further distinctions.

Even though programs are sets of instructions, their effective character in a given situation can neither be completely described by making those instructions explicit, nor by referring to the conscious act on the side of the programmer. Certainly any considerable deviation from the set of rules of a given computer language will cause an error, which is one of the reasons why a program remains within the bounds of computability. This by no means implies, however, that programming is an entirely transparent and determinable procedure. Habits of programmers, their state of knowledge, their reuse of existing code (or even copy and paste programming) and the influence of conventions determine the social influence of software to a considerable degree, just as much as the habits and comprehension of its users do. The concept of automatisms is particularly beneficial for understanding this situation.

Daily routines, like communicating via email, doing a bank transfer at an ATM or creating an invitation card, are fundamentally determined by soft-

---

seiner Inaktualität", in: *Indeterminate Kommunismus! Texte zu Ökonomie, Politik und Kultur*, ed. by DemoPunK | Kritik und Praxis Berlin, Münster, 2005, pp. 23-30: 25.

3 Hannelore Bublitz/Roman Marek/Christina Louise Steinmann/Hartmut Winkler, "Einleitung", in: id., eds., *Automatismen*, Paderborn, 2010, pp. 9-16: 11.

ware. How could one discriminate which stages of such routines originate in the repetition of habituated usage and which are determined through the functionality of the software itself? In order to understand how the world has changed through the proliferation of personal computers, it is crucial, but also very difficult, to discriminate between two aspects: habitual behavior in the development and use of software (automatisms) on the one hand, and the processes determined by the automatization through technology on the other. This task is further complicated by the fact that to many people computer programs are completely familiar, but their source code remains alien.

The study of automatisms involves making visible what normally eludes conscious control. It is useful here to discuss an early concept of dealing with this task that can be found in art theory. When Viktor Shklovsky coined the term "*ostranenie*"[4], or defamiliarization, he discovered strategies that some artists use to bring awareness to automatisms through their work. In this context, he also defined automatisms:

> 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.[5]

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.[6]

The examples used by Shklovsky come from literature, but the concept of *ostranenie* is discussed in a wide range of art contexts. He discovered an aspect addressed by art which at that time couldn't be subsumed under existing aesthetic theories. If art doesn't serve mimetic or educational purposes, if it alienates rather than trying to improve humanity, what is its raison d'être? The fact that it expressed an aspect which didn't coincide with the humanist belief in progress was an important reason why *ostranenie* became one of the central concepts in 20th century art theory. In Tolstoi's writing, Shklovsky saw techniques of bringing blanketed perceptions again to awareness. Within art he found ways of not only exploring automatisms, but also of exposing them to the readers' understanding. In his view, art had an essential function in creat-

---

4 Was first coined in 1917 in Victor Shklovsky, "Art as Technique", in: *Russian Formalist Criticism. Four Essays*, ed. by Lee T. Lemon and Marion Reis, Lincoln, NE, 1965, pp. 5-24: 12.

5 Ibid., p. 11.

6 Ibid., p. 13.

ing a new perception of something known by defamiliarizing it, for instance by treating an everyday act or object as if it were something alien.[7]

Shklovsky developed his ideas with respect to literature, differentiating unambiguously between poetic and everyday language.[8] All the hidden troubles of differentiating between purposive and poetic language became a central concern for 20th century philosophy, with authors like Derrida, Lyotard, or Kristeva.[9] 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 questions about the relation between natural and machine language. Because a programming language is, by itself, entirely rule-based, including 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.

Soon after the developments of the first computers, the universal machine was discovered as a tool for artistic research as well as for research about art. Well known artworks, for example by Mondrian or Klee, were recreated using programming languages.[10] From the early days on, it has been claimed that a computer program is capable of producing a poem or even a whole novel, a claim that has been emphatically discussed ever since.[11] As an aspect of artificial intelligence, the dreams and fantasies of machines creating art mushroomed with the proliferation of the computer. In the context of such developments, to differentiate between automatisms, automats and artistic working methods becomes crucial for the understanding of changing techno-social de-

---

7 Ibid., p. 21.

8 Ibid., p. 10. The first quote continues: "Such habituation explains the principles by which, in ordinary speech, we leave phrases unfinished and words half expressed." (Ibid., p. 11.)

9 As these pages only serve as a short introduction, there is clearly not more space for more than a hint at some theoretical threads one should mention in this context. Certainly, one should think of the so called semiotic or linguistic turn. This discourse was extremely influential, just as much as it has been claimed dead by some recent authors (e.g. cf. Karen Barad, *Meeting the Universe Halfway: Quantum Physics and the Entanglement of Matter and Meaning*, Durham, NC, 2007, pp. 132 et seqq.) Also one could trace some motives back to the roots of philosophical aesthetics, as Lyotard tries to do (Jean-François Lyotard, *The Inhuman: Reflections on Time*, Cambridge et al., 1991. pp. 71 et seqq.). Kristeva discussed the term *poetic language* comparing it with formalized, mathematized concepts of language (Julia Kristeva, *The Revolution in Poetic Language*, New York, NY, 1984, p. 21). Dosse describes Kristeva's first and influential encounter with Barthes. She introduced her concepts as "drawn from Russian postformalism and based on the work of Mikhail Bakhtin." (François Dosse, *History of Structuralism: The Sign Sets, Volume 2*, Minneapolis, MI, 1997, p. 54.) Those sources may serve as an entry point for analyzing the conceptual development in terms of a discourse circling around the question of the relation between poetic and purposive language.

10 E.g. Noll recreated Mondrians "Composition With Lines" and showed it, among other exhibitions, at "Cybernetic Serendipity". A. Michael Noll, "A Subjective Comparison of Piet Mondrian's 'Composition with Lines' 1917", in: Jasia Reichardt, ed., *Cybernetic Serendipity: The Computer and the Arts*, London, New York, NY, 1969, p. 74.

11 E.g. Douglas Hofstadter, *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*, New York, NY, 1996, pp. 155-168.

velopments. Code shares many properties with natural language and illustrates both its automatized character, as well as the impossibility to fully automate it. An artwork can be an investigation into this ambiguous relation.

In this context, a very direct parallel can be found between the technique Shklovsky finds in Tolstoi's writings and the so called *codeworks*. This genre came into being at the turn of the century and involves a mixing of literary writing with writing code of computer languages. Various artists experimented with this kind of poetry. It is plausible that such a literary work is able to change the usual perception of code. In such texts, one cannot focus on functionality in the usual way, as the poem doesn't function as computer program. As a poem, it wants the reader to examine the character of programming language apart from its computational functionality.[12]

Another parallel that is worth mentioning here can be found in *live coding*[13], an art form that dispenses code from its hiding place and exposes it to the public. While an artist is programming in real time, creating the music for an audience, which may or may not dance to it[14], the content of the computer screen is projected onto a wall (instead of the more commonly used animated and animating graphic displays).[15] This may merely serve as a means to convey information between the performers. But it also functions in a way that resembles Shklovsky's concept of art as a method to create a renewed perception of its object. In the present case however, the artistic use of programming language does not only serve to question commercial or governmental computer technology. It rather takes this technology out of its habitual context in order to create a new one for it.

In these two art forms, the public display of texts in a computer language can be considered an investigation into the role and influence of computer technology in general. But programming may also be used to create an artwork which reminds us of very common and familiar software. In such cases, the focus is more on software as a designed and preconfigured commodity, which changes or even determines individual, social, and political processes. The contributions by Amy Alexander and Carmin Karasic which follow, can be read in this light. Accordingly, I shall briefly mention the central concepts concerning automatisms which the two installations address.

---

[12] There is a short extract of a codework in *Speaking Code*. Cox writes: "Harwood's codework *Class Library* (2008) plays on these inherent antagonisms (involved in working with code, R. W.), along with the double coding of the term 'class'." Geoff Cox, *Speaking Code: Coding as Aesthetic and Political Expression*, Cambridge, MA, et al., 2012, p. 40.

[13] http://toplap.org/, last downloaded 2014-01-01.

[14] Alex Mclean, "Hacking Perl in Nightclubs", on: *perl.com*, 2004: http://www.perl.com/pub/2004/08/31/livecode.html, last downloaded 2014-01-01.

[15] Ward, Adrian/Rohrhuber, Julian/Olofsson, Fredrik/McLean, Alex/Griffiths, Dave/Collins, Nick and Alexander, Amy, "Live Algorithm Programming and a Temporary Organisation for Its Promotion", in: *Proceedings of the README Software Art Conference*, 2004: http://toplap.org/wiki/Read_me_paper, last downloaded 2014-01-01.

Carmin Karasic describes an installation that can be understood as a defamiliarization of social software. She brings concepts into play that allow us to reason about automatized behaviors and schemata of perceiving others. Social media brought along many new ways of getting to know people and keeping in contact, social habits have been transformed, along with the terms and expressions used in social contexts. It is not always easy to decide whether the way of using these tools is due to personal preferences, social formations, or just the preconfigured options of the software.

Amy Alexander writes about an installation that deals with surveillance technology. The software described in the article was created by a group of artists she was part of. It operates by means of face recognition and the tracing of movements. To create such a software, one has to formalize the knowledge about the perception one considers to be common to different social groups. In such an application, stereotyping is probably the only possible form to do so. Finally, two other important aspects are addressed in this work as well as in Alexander's paper: computer literacy and the conventions of popular culture. Both raise questions about automatisms and the unequal access to the technical background which determines the user's behavior as well as her interest.

Both installations explore habitualized and automatized use of software. They are interactive not so much in order to empower the visitor to take an active part in the installation, but to alienate or defamiliarize daily routines. Routines that have accumulated and developed quickly in the last decades. The following articles make a significant contribution to the understanding of automatisms: discussing the installations after the lecture the two artists gave in Paderborn helped us to defamiliarize our own research objects.

Literature

Alexander, Amy, "About... Software, Surveillance, Scariness, Subjectivity (and SVEN)", in: *Transdisciplinary Digital Art. Sound, Vision and the New Screen*, ed. by Randy Adams, Steve Gibson and Stefan Müller Arisona, Berlin, 2008, pp. 467-475.

Barad, Karen, *Meeting the Universe Halfway: Quantum Physics and the Entanglement of Matter and Meaning*, Durham, NC, 2007.

Bublitz, Hannelore/Marek, Roman/Steinmann, Christina Louise/Winkler, Hartmut, "Einleitung", in: id., eds., *Automatismen*, Paderborn, 2010, pp. 9-16.

Cox, Geoff, *Speaking Code: Coding as Aesthetic and Political Expression*, Cambridge, MA, et al., 2012.

Dosse, François, *History of Structuralism: The Sign Sets, Volume 2*, Minneapolis, MI, 1997.

Hofstadter, Douglas, *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*, New York, NY, 1996.
Kristeva, Julia, *The Revolution in Poetic Language*, New York, NY, 1984.
Lyotard, Jean-François, *The Inhuman: Reflections on Time*, Cambridge et al., 1991.
McLean, Alex, "Hacking Perl in Nightclubs", on: *perl.com*, 2004: http://www.perl.com/pub/2004/08/31/livecode.html, last downloaded 2014-01-01.
Noll, A. Michael, "A Subjective Comparison of Piet Mondrian's 'Composition with Lines' 1917", in: Jasia Reichardt, ed., *Cybernetic Serendipity: The Computer and the Arts*, London, New York, NY, 1969, p. 74.
Rancière, Jacques, "Von der Aktualität des Kommunismus zu seiner Inaktualität", in: *Indeterminate Kommunismus! Texte zu Ökonomie, Politik und Kultur*, ed. by DemoPunK | Kritik und Praxis Berlin, Münster, 2005, pp. 23-30.
Shklovsky, Victor, "Art as Technique", in: *Russian Formalist Criticism. Four Essays*, ed. by Lee T. Lemon and Marion Reis, Lincoln, NE, 1965, pp. 5-24.
Ward, Adrian et al., "Live Algorithm Programming and a Temporary Organisation for Its Promotion", in: *Proceedings of the README Software Art Conference*, 2004: http://toplap.org/wiki/Read_me_paper, last downloaded 2014-01-01.

Internet Sources

http://toplap.org