Andreas Broeckmann

Runtime Art: Software, Art, Aesthetics
http://runtimeart.mi2.hr/TextAndreasBroeckmann

I. Software as Art

For a long time, computer software has been understood as a purely functional element of digital technology. Only since the late 1990s, it has come under the scrutiny of media theoretical and cultural research. After the age of garage-based computer bricoleurs, followed by the massive distribution of personal computers that came with standard proprietary software but without the necessary media competency, the last years have been characterised by a growing 'do it yourself' culture where programmers cooperate internationally on writing free software, and where musicians, visual artists, literary critics and architects are increasingly acquiring programming know-how.

At the same time, we see the rise of a generation of media critics who are equally familiar with the Internet's technology, standards and politics, and with the marketing and techno-politics of computer hard- and software. The debates of these media critics show that software is a medium and a cultural artefact that is being design in a specific way which carries a particular socio-cultural meaning. In analogy to the cultural analysis of technology as it has emerged from social historiography over the past decades, we can observe the exploration of a technical medium being investigated with regard to the social and economic conditions of its development and application.

Since 2001, the Berlin-based international media art festival, transmediale, has been committed in its programme to the cultural and artistic dimensions of software. The transmediale was the first festival to award a special prize for software art, and it has hosted a whole series of discussions and lectures in order to foster the dialogue between programmers, artists, sociologists and media researchers. This initiative is not aimed at installing software art as yet another independent art category, but is meant as a heuristic intervention that seeks to stimulate the discourse in this important socio-cultural field. It raises the question in how far software art can be described as a form of 'autonomous' artistic practice, autonomous that is of the normal functional and utilitarian requirements of software, and what might be the aesthetic potential of the creative practice of programming.

In a text developed from their original jury statement, Florian Cramer and Ulrike Gabriel, who were members of the first Software-Jury of transmediale.01, put it like this: 'Coding is a highly personal activity. Code can be diaries, poetic, obscure, ironic or disruptive, defunct or impossible, it can simulate and disguise, it has rhetoric and style, it can be an attitude. Such attributes might seem to contradict the fact that artistic control over generative iterations of machine code is limited, whether or not the code was self-written. But unlike the Cagean artists of the 1960s, the software artists we reviewed seem to conceive of generative systems not as negation of intentionality, but as balancing of randomness and control. Program code thus becomes a material with which artists work self-consciously. Far from being simply art for machines, software art is highly concerned with artistic subjectivity and its reflection and extension into generative systems.'

Most of what constitutes software art today, belongs in the field of media art, in so far as it is concerned with the formulation of aesthetic modes of expression, with the expansion of the artistic field, and with the articulation of the shifting relationship of human and medium, or human and machine. Software programming will, not dissimilar to photography, video and the Internet, move from a status of novelty, on to being one more medium which artists can make use of according to their individual taste, expressive needs and technical faculties. A software art exhibition will then make as much and as little sense as an exhibition dedicated to photographs, or to video art.

II. Software as Culture

Software has by now come into view as a cultural technique whose social and political impact ought to be studied carefully. To the extent that social processes rely on software for their execution - from systems of e-government and net-based education, online banking and shopping, to the organisation of social groups and movements -, it is necessary to understand the procedural specificities of the computer programmes employed, and the cultural and political 'rules' coded into them. The ?killer apps? of tomorrow may, as Howard Rheingold claims, not be 'hardware devices or software programs but social practices'. Yet, these social practices will increasingly be determined by software configurations of the available infrastructure and the degrees and types of latitude that they offer. Aspects of software culture - a terrain that encompasses software development as well as the wide and multi-facetted field of software application - are being articulated by speculative and artistic software projects.

The term 'social software' has been used by Matthew Fuller, Graham Harwood, and others, to describe a type of software that consciously engages the social aspects of its application. Whereas a programme like MS Word, which Fuller has carefully disected in an extensive analysis, tends to conceal the rules and assumptions that served to constitute its structure, social software addresses the more or less specific social context of its application, whether in the form of the Linker software by Mongrel that offers an easy-to-use functionality for multimedia production, or in the online communication platforms that support, for instance, collaborative software and media development and that can easily be tweaked to meet the requirements of a certain co-producer community.

For almost a decade, the Nettime mailing list has been an active, international forum for the discussion of software-related cultural and political issues. In a seminal essay posted on Nettime, Behind the Blip, Fuller talks about key aspects of social software and also refers to the Californian researcher Ellen Ullman who has worked on software development as a distinctly social practice for several years. Important practical and theoretical work in this field has also been done by the Amsterdam-based Society for Old and New Media, De Waag, whose software development projects have engaged the needs and possibilities of different user groups by way of models for a 'participatory software design'. In cooperation with De Waag, the New Delhi-based media and communication centre Sarai has also worked on both the practical issues of social software development, and on the critical reflection of software culture on their online Reader-List and in the Reader print publications. While Nettime has often carried postings articulating differences between European and US media cultures, Sarai has, importantly, helped to raise awareness for the differences in software cultures, esp. with regard to developments in South Asia.

In his essay, Behind the Blip (now also available in a book of the same title, published by Autonomedia), Fuller distinguishes social software from 'critical' and 'speculative' software, critical software being 'software designed explicitly to pull the rug from underneath normalised understandings of software'. It critically engages with existing software programmes and mutates or critically analyses them. In contrast, 'speculative software' comes closest to what can be understood as an artistic approach to software: it is, as Fuller writes, 'software that explores the potentiality of all possible programming. It creates transversal connections between data, machines and networks. Software, part of whose work is to reflexively investigate itself as software. Software as science fiction, as mutant epistemology. Speculative software can be understood as opening up a space for the reinvention of software by its own means.'

In comparison, the notion of 'software art' is an attempt to describe a practice that is artistic, non-functionalist, reflexive and speculative about the aesthetics and politics of software, and that takes computer programming as the material proper of the artistic practice. The term is especially used for works of generative art whose main artistic material is program code, or which deal with the cultural understanding of software. Thus, software is not understood as a functional tool serving the 'real' artistic work, but as a generative means for the creation of machinic and social processes. Software art, in the understanding of researcher, software activist and co-editor of the Nettime Unstable Digest, Florian Cramer, can be the result of an autonomous and formal creative practice, but it can also refer the cultural and social meaning of software, or reflect on existing software through strategies like collage or critique.

Like transmediale, other exhibition and curatorial projects (Generator in the UK, the ars electronica's CODE festival, the exhibition 'I Love You' on computer viruses, a.o.) have sought to circumscribe a field of artistic work that deals with the aesthetic potential of software. Most notably, the festival Read_Me (Moscow, Helsinki and Arhus) has been exclusively devoted to software art and has led to the establishment of the Runme.Org collaborative online database for software art projects. The CODeDOC project has presented software developed by artists and has included comments and documentation of the programming process and has thus attempted to introduce an aspect of transparency and the idea of Open Sources into the discourse on software by and for artists, an issue which is also being addressed in discussions about 'open content' and the 'creative commons' licenses for artistic productions. In contrast, Free Software developers like Jaromil, who is pursuing a.o. the MuSE project for a free audio streaming software, insist on the necessity to resist proprietary licensing models altogether.

It is worth noticing that the Free Software and open source models have increasingly also influenced art-related software productions in independent labs like the V2_Lab, the Ars Electronica Center or the MIT Media Lab. The copyright issue, which Georg Greve, president of the Free Software Foundation Europe, suggests should not be referred to as 'Intellectual Property Rights' but as 'the question of industrial control of information', will become crucial for the information and knowledge society and must be addressed experimentally in the arts and culture sector, like in the recent exhibition Illegal Art which presented some of the ridiculous results of tight copyright laws.

The issues of interface design and interaction have been among the prime concerns of digital art production, yet, while software has mostly been treated as a tool towards realism in virtual environments, software art projects like I/O/D's Webstalker, Jodi's Wrong Browers or Joan Leandre's retroYou R/C have offered irritating and enlightening insights into the construction of digital realism by means of software.

The Internet, while accelerating the demise of utopian hopes once invested in its liberatory potential, has also become the site of a multiplicity of collaborative forums, whether on mailing lists, Wikis, in weblog communities, etc. For the Net in general, software developments around Java, the Linux system, and online publishing forums like Slashdot or Freshmeat, have all had shares in a complex and vibrant cultural development. For software art in particular, a.o., the eu-gene and linart mailing lists, are continuing to play an important role. The social and theoretical implications of these kinds of online cooperation have been investigated by projects of the interdisciplinary artists group Knowbotic Research for over ten years, most notably in the IO_dencies series in the mid-90s, but also in the more recent collaborative hacking projects. Similarly, the Italian EpidemiC collective explores new forms of software based online activism in their Anti-Mafia project.

Collaborative and activist projects like these frequently also involve debates about network security, ironically referenced by Technology To The People's Phoney(TM), and about privacy issues which were tackled by LAN's Tracenoizer project and, more recently, by Franz Alken's Machines Will Eat Itself, both of which instigate a deliberate erosion of relations between human individuals and their online data bodies.

If anything, software art projects like these should indicate the necessity to delve more deeply into the cultural specificities of software development and application. Software needs to be understood as a set of digital media which need to be explored regarding their specificity, their political and cultural dimensions. An immense amount of knowhow already exists in the open source and free software development communities, as well as in hacker and art coder circles. It will be crucial to devise ways how this knowhow can be interwoven, at times pooled, and exploded across the entire field of software development and usage.

III. Notes about the Aesthetics of Software Art

When talking about software and art, we have to speak about aesthetics, that is engage the value systems that inform our experience of art, and our perceptions in general. References have been made to the traditions of Fluxus, Conceptual Art, or Net Art, each of which implies a set of assumptions about the ways in which to judge the artistic quality of artworks. Over the last 200 years, European culture has seen aesthetics of beauty, aesthetics of the sublime, aesthetics of ugliness, and aesthetics of formal order. But this history teaches us, that there are alternative ways of approaching software-based artworks than Max Bense's extremely formalistic "Generative Aesthetik" which he formulated in the 1960s. For instance, it would also be interesting to revisit the debates about Realism vs Formalism between Lukacs and Brecht in the 1930s in this respect, if only to sharpen our perception for the level of critique that can be brought to significant artworks.

My own idea of art practice, which I also bring to this field of software-based work, is opposed to bland visualisations and translations from one formal system to another. I believe that we need a strong notion of what constitutes art. For me, art is about the transgression of boundaries, about making familiar experiences strange, about dramatising what pretends to be innocent, and about exploring the virtualities, the potentialities of technologies and human relationships.

In many cases, art projects relate to or express their cultural environment in very restrained or benign, at times even banalising ways. This is not only an issue in software-based art, but of digital art practice in general - it often tends to be affirmative of the technology, uncritical of its corporate politics and superficial in its formulations and expressions. Where is the desire for excess in software-based art? Where do we find the surplus, the surprise, that which we do not know yet and that is not already legible in the software code or the technical dispositif that artists prepare so ardently?

IV. Runtime Art

For the exhibition 'Runtime Art', we have selected a number of projects that engage specifically with one aspect of software, i.e. its execution in the 'runtime' of the computer processor, and thus the close connection between code execution and aesthetics.

A classic piece in this respect is Every Icon by John F. Simon, which exhibits both the precise clock-speed of the computer that runs through the iterations of black and white pixels in a binary grid, and the impossibility of a comprehensive representation of reality by a computer which, quite apparently, already fails in the simple task of offering all possible computer icons in a reasonable amount of time. While 'Every Icon' dramatises the execution of code by the sheer tedium of its almost endless process, Vexation1 by Antoine Schmitt achieves a more singular tension by giving the impression of a hesitant, self-conscious computer program that 'drags its feet' as it has to decide which path a white dot should take from one pre-determined side of the rectangle to the next. Despite the clearly given determinacy, the program still appears to be caught in a subjective decision-making process.

For his project Micro Images, Casey Reas has developed a complex set of algorithms that send the generative graphics into unpredictable and hugely complex configurations. Here, 'runtime' is the medium of excessive machine-based differentiation. A similar task is approached very differently in the project Electric Sheep by Scott Draves, a screen-saver program that takes its calculating power from a distributed network of computers running the software generating the graphics which can be individually designed and viewed. The 'excess of runtime' is also explored in Mandl & Krautgasser's Pedigree which uses the actual written code to translate the story of Oedipus into an algorithmic language which, when executed, offers a dynamic representation of the Oedipal drama of Love and Death in multiplying triangulations of relationships.

The question of the interface through which human users can interact with software-based systems has been approached very elegantly by Golan Levin. His Audio-Visual Environment Suite (AVES) is a series of tools for the gestural creation and manipulation of sonic and visual structures. More analytical in its approach is the Webstalker by I/O/D, one of the first projects to deconstruct the notion of the webbrowser: the Webstalker offers several ways of looking at the code and link structure of a website, giving access to all the information that a regular browser would also show, and more, but in a way that is formal and purely structural. While the Webstalker was designed in opposition to existing representational models, Minitasking by Schoenerwissen/OfCD had to invent its own representational paradigms for a structure that had not been visualised before, i.e. the Gnutella file-sharing network. Minitasking combines the function of a crawler with that of a network scanner, offering a complex image and a functional interface to an ongoing, internet-based communication system.

Such representational models are called into questions by Auto-Illustrator, a software package developed by Signwave and modelled on existing vector graphics programs. Unlike those, however, 'Auto-Illustrator' exhibits a distinct autonomy in the visualisation of commands and movements of the user, which it frequently translates into crazy and uncontrolled results, thus highlighting the fundamental difference between intention, user-interaction, and result. A similar deconstructive approach is taken by Joan Leandre in the retroYou R/C project which, on its different racing game levels, employs increasingly corrupted rules for the representation of the virtual environment in which the race is taking place. True to its slogan, "F*ck the gravity code!", it pinpoints both the constructedness of such virtual 3D-spaces, and the limitations of interaction in non-rational spaces.

Finally, a project by Robert Luxemburg exemplifies the political dimension of digital data and their execution. The Conceptual Crisis of Private Property as a Crisis in Practice is a screen-shot, i.e. a digital image, whose code can also be executed as a program - when executed, the file produces the novel 'Cryptonomicon' by Neal Stephenson. The project is a riddle about the different representational and juridical layers at which coded, and thus also encoded intellectual property, can exist. Here, 'runtime' is not only the condition, but the problematic medium, interface and gateway to the borders of representation and legality.

This selection of works covers the particular field of generative art and is in no way exhaustive, and certainly not for all the ways in which software is currently being explored as a medium and theme of artistic practice. These include a whole range of projects, from the exploration of code as poetry, through the psychogeographic programming of cities by socialfiction.org, to the real-time programming environments developed for live performance and interaction tasks. In this entire field, artists are no longer just working within existing technological paradigms, but through their creative programming efforts, they are actually trying to push the boundaries of what software means as a social and cultural technique. The cooperation between artists and programmers is often very close, and in many of the examples presented here, the artists have in fact done the programming themselves.

The exhibition 'Runtime Art' is an exploration into the generative aesthetics of software. Rather than being conclusive, it hopes to open up further debate about the artistic potentials of computer programming and digital code as cultural artifacts, and techniques.

Selected projects and exhibitions:

transmediale Festival, Berlin - http://www.transmediale.de

Kontrollfelder Exhibition, Dortmund - http://art.net.dortmund.de
I Love You Exhibition, Frankfurt/M. - http://www.digitalcraft.org
Digital is not Analogue Festival, Bologna - http://www.d-i-n-a.net
Read_Me Festival, Moskau/Helsinki/Arhus - http://www.runme.org
Generator Exhibition, Liverpool - http://www.generative.net/generator
Art Bit Exhibition, Tokio - http://www.art-bit.jp
Electrohype Festival (2002), Malmö - http://www.electrohype.org
Ars Electronica Festival (2003) - http://www.aec.at
CODeDOC Exhibition - http://www.aec.at/en/festival/programm/codedoc.asp