

On Software as Art

ANDREAS BROECKMANN

I. Parameters

For a long time, computer software has been understood as a purely functional element of digital technology. Only since the late 1990s has it 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 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 technopolitics of computer hard- and software. The debates of these media critics show that software is a medium, and a cultural artefact that is being designed in a specific way that carries a particular socio-cultural meaning. Analogous 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.

In cooperation with *transmediale*, the art organisation *hArtware* in Dortmund, Germany organised the exhibition "Control Panels" (Kontrollfelder, April 2002) which focussed



entirely on software art projects, and which pursued the question of 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. (It was soon to be followed by the "Generator" show in Liverpool, UK; links to other, related events are listed below.) The exhibition tried to show 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 diary, 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. Programme 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 – insofar

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 relationships of human and medium, or human and machine. Software programming will, not unlike photography, video and the Internet, move from a status of novelty to being one more medium which artists can make use of according to their individual tastes, 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.

Another area of the artistic work with software concentrates not so much on the aesthetic but on the social and technocultural dimensions of computer software. It is closely related to a critical tradition of the avant-garde, rather than to an artistic modernism striving for autonomy. In this field of practice we find works that deal with the social processes enabled by software, as well as ironic or politically motivated analyses of the functionality and economy of proprietary software products. For this kind of creative work, Matthew Fuller has coined the term "speculative software", which I would suggest can be used to describe this particular type of software art: "Speculative software is 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".

The reflection of the aesthetic and cultural aspects of software are crucial gestures of software art, and it remains to be discussed whether it would be necessary to differentiate between art, critique and speculation, or whether such differentiation is superfluous.

II. Suggestions towards an Analysis of Software Art

We are currently only beginning to develop a critical understanding of what software/art might be (a computer programme? An executable text? A generative process?), where its boundaries might be and what might be adequate aesthetic and critical criteria for its description and interpretation. The following is a first and incomplete attempt to develop something like a taxonomy according to which such an analysis might be pursued.

1. Tools

These are closest to the 'normal' function of software, i.e. to 'make', 'do' or 'show' something else. They can be:

- performative (as in Golan Levin's *Audiovisual Environment Suite*)
- reflexive (as in Adrian Ward's *Signwave Auto-Illustrator* or Frédéric Durieu's *Puppet Tool*)
- networked (as in Knowbotic Research's *Minds of Concern* or LAN's *TraceNoizer*)
- mapping and visualising (like IOD's *Webstalker*, Schoenerwissen's *minitasking* or RSG's *Carnivore*)
- browsers (as in NN's *Nebula.M81* or Jodi's *Wrong Browser*)

2. Interactive Environments

These follow the idea of an interactive engagement with the machine and are, in a mutated form, modelled after:

- games (as in Joan Leandre's *retroYou*)

3. Code as Text

These are closest to conceptual art and most strongly concerned with the textuality of (often experimental) computer code that is written and executed. It can be more:

- literary (as in Jaromil's *forkbomb.pl*)
- political (as in Projekt Gnutenberg's *textwarez*)
- technological (as in Alex McLean's *forkbomb.pl* and *animal.pl*)

4. Machinic Autopoiesis

These are generative dispositions which, in my view, are closest to a generic form of software art. They can be:

- autonomous (as in Antoine Schmitt's *Vexation1*)
- semi-autonomous (as in David Rokeby's *n-Chant* or Knowbotic Research's *Anonymous Muttering*).

III. Selected Projects and Exhibitions:

transmediale (Berlin, <http://www.transmediale.de>).

Kontrollfelder (Dortmund, <http://art.net.dortmund.de>).

I Love You, Frankfurt/m. (<http://www.digitalcraft.org>).

Digital is not Analogue (Bologna, <http://www.d-i-n-a.net>).

Read_Me, Moskau (http://www.macros-center.ru/read_me).

Generator (Liverpool, <http://www.generative.net/generator>).

Art Bit (Tokyo <http://www.art-bit.jp>).

Electrohype, Malmö (<http://www.electrohype.org>).

