TEN THESES ABOUT SOFTWARE ART

FLORIAN CRAMER

WHAT THIS IS NOT ABOUT

"Software art" as it is defined in the free Internet encyclopaedia Wikipedia (as of September 2003):

> "Software art is a term for the graphic design of visual elements contained in software, eg. GUI (Graphic User Interface), Icons etc."[1]

WHAT THIS IS ABOUT

Artists who use digital software to produce works which themselves are digital data create—as only writers have done before them—works made up of symbols using a set of instruments that is also comprised entirely of symbols. No literary writer can use language merely as a stopgap device with which to compose an artwork that is not in itself language—so, like in a recursive loop, literature writes its own instrumentation. In the same way, the zeros and ones of digital art are closely related to the zeros and ones of the instruments with which they are not only created, displayed and reproduced.

THERE IS NO DIGITAL ART WITHOUT SOFTWARE

It is always naive to assume that there is type, images, sound or networking in computers provided for themselves or in "multimedia" combinations, since these data forms do not exist without the computer programs that produce them. This applies not only to their design and processing (through, for example, text, graphics or music authoring software), but already to their mere display (in software browsers, viewers and players) and reproduction (through network and operating system software). Every digital artwork that is not itself a computer program exists only within the framework that

Date: 9/23/2003.

[1] `http://www.wikipedia.org/wiki/Software_art`

prefabricated software has defined for it. All digital art is therefore "software art" at least to the degree that it is software-aided art. It becomes software art in the narrower sense, I would suggest, when it does not regard software as an external aid, but as part of its own aesthetics.

SOFTWARE ART NEED NOT BE DIGITAL OR ELECTRONIC

A computer program is a series of formal (algorithmic) instructions which can, but must not necessarily be executed by a machine. Like this example:

```
// Classic.walk

 Repeat

 {

 1 st street left
 2 nd street right
 2 nd street left

 }
```

This is an example program[2] of ".walk" by http://www.socialfiction.org. ".walk" has been labelled by its inventors a "psychogeographical computer" because it is made up of the streets of big cities rather than transistor grids and executes its programs by having pedestrians rather than electrons run through them. .walk therefore reflects two historical precursors: firstly Fluxus and Concept Art with their para-algorithmic, minimalist action scores (like those composed by George Brecht, La Monte Young and Sol LeWitt following a paradigm set by John Cage), and secondly the modern computer in its earliest incarnation of only an imaginary, theoretical apparatus in the shape of the Turing Machine.

[2] socialfiction.org, .walk for dummies, http://www.socialfiction.org/dotwalk/dummies.html

SOFTWARE ART IS NOT SYNONYMOUS WITH CONCEPT ART

.walk differs from action scores such as George Brecht's first "Lamp Event" of 1961 and its binary instruction "on.off"[3] inasmuch as it reflects a tested cultural practice; the use of computers, software and their programming. While the "Lamp Event" could be read as an anticipation of artistic software programming through formalism, .walk's title—which is a play on Microsoft's ".NET"—already identifies itself as part of a software culture. In this work, therefore, it is not Concept Art that points to software, but the opposite; software points back to the conceptual actionism of the 1960s—which also included the psychogeography of the Situationist International—, rereading it as computer software. However, this look back is no longer conceptual in itself, but historical, ironic, a work of collage.

It is precisely in this respect that today's software art contradicts the equation of art and software as it was established in 1970 both in Jack Burnham's 1970 Concept Art exhibition "Software" in the Jewish Museum New York and in the first issue of the video art magazine "Radical Software."[4] Thirty years later, software is no longer a laboratory construct and a paradigm of conceptualist purification, but is—since the wide distribution of PCs and Internet—faulty code to a large extent, the cause of crashes, incompatibilities, viruses and thus of the contingency rather than the stringency of symbols.

As the Net.art by jodi, Alexei Shulgin, Vuk Cosic, I/O/D and others aestheticised precisely these contingencies and so liberated digital art from its apparent academic and industrial sleekness, it is no coincidence that we encounter familiar names in recent software art, which has a discursive continuity with the net art of the 1990s. Looking at the development of jodi's artistic work from 1996 to the present day, we gain an exemplary view of how Net.art experiments with screen graphics and network communication first became work rebelling against the limitations of its software context (for example in the browser manipulation "OSS" `http://oss.jodi.org`), then developed into the reprogramming of software (as in the "Untitled Game" based on the computer game "Quake" `http://www.untitled-game.org`) and finally a reduction of the visible object to simple BASIC source-code (in the most recent work "10 Programs written in BASIC ©

[3] Score-cards in [**?**]

[4] On the exhibition, see [**?**], "Radical Software" may now be found in facsimile at `http://www.radicalsoftware.org`.

1984").[5] It is true that recent software art has surface similarity to older Concept Art when it makes use of minimalist form. But this resemblance is contradictory, because it does not reflect the spirit of what Lucy Lippard in her book "Six Years" called the dematerialisation of the art work from 1966 to 1971. On the contrary, in today's software art software is certainly understood as material. This understanding is also a precondition to the written "codeworks" of artists including jodi, antiorp, mez, Alan Sondheim, Johan Meskens and Lanny Quarles,[6] which combine syntactic elements from programming languages, network protocols, system messages, and computer-cultural slang. The following email by the French artist Pascale Gustin is an example of this:

```
L'_eN(g)Rage \ment politi][~isch][K et l' _art is T(od)

][ref lex][1/O.ns 10verses NOT es][

--------\B(L)ien-sUr 2 que/S\tions f.Ond(ent)

---------------------------------------A:

------------------------------][menta les_sel][l] a tenement) T nem T

-tout d_abord--------1/O(f.ne

1 of 1 deletions
1 deletion done
apply: Command attempted to use minibuffer while in minibuffer
```

SOFTWARE ART IS NOT SYNONYMOUS WITH ALGORITHMIC ART

If software, generally defined, is algorithms—does that mean software art is the same thing as algorithmic or generative art? The following, helpful definition of generative art was given by Philip Galanter: "Generative art refers to any art practice where the artist creates a process, such as a set of natural language rules, a computer program, a machine, or other mechanism, which is then set into motion with some degree of autonomy contributing to or resulting in

[5] Exhibited at Electrohype in Malmö.

[6] In this respect, see sources including [**?**] and [**?**]

a completed work of art."[7] It is true that software art may involve autonomy in a sequence of events as it had also been described in Jack Burnham's essays, strongly influenced by cybernetics and general systems theory, from the 1960s:[8] for example as running code in the guise of classic PC user software, or also as unambiguous formal instructions as in ".walk." But if one looks at popular sub-genres of software art like game modifications[9] and experimental browsers,[10] these are not concerned with the aesthetic autonomy of algorithmic processes, but with interrupting these by means of irritative couplings of software, humans and network data. In generative art, according to Galanter's definition, software is only one of several possible means which, rather than being an artwork in itself, may only "contribute" to it, in the same way that many computer-aided arts (including electronic music) do not see software as part of their aesthetics, but permit it to work in the background.

For its part, software art fails to meet the criterion of the generative, or it only fulfils this in the metaphorical, rather than the technical sense when it writes—as in "codeworks" for example—dysfunctional and imaginary software.

SOFTWARE ART IS NOT BEING MADE IN A VACUUM, BUT AS PART OF A SOFTWARE CULTURE

If recent software art does not understand software as generative process control, but as material for play, it no longer reads it—as in classic conceptual and generative art—as pure syntax, but as something semantic, something that is aesthetically, culturally and politically charged.[11] While software culture in 1970—as is documented by Burnham's "Software" exhibition with its confrontation of concept

[7] Quoted for example at http://www.philipgalanter.com/pages/acad/idx_top.html and http://www.generative.net

[8] See also the German edition of Burnham's "Structure of Art", [?] rather unfortunately translated as "Kunst und Strukturalismus".

[9] jodis "Untitled Game", Joan Leandres "retroyou" http://www.retroyou.org

[10] I/O/D's "Web Stalker" http://www.backspace.org/iod/, Netochka Nezvanovas "Nebula M.81", Jodi's "wrongbrowsers" http://www.wrongbrowser.org, Mark Napier's "Shredder" http://www.potatoland.org/shredder/, Kensuke Sembo's and Yae Akaiva's "Discoder" http://www.exonemo.com/DISCODER/indexE.html, Peter Luining's "ZNC Browser" http://znc.ctrlaltdel.org/pc_znc2.0.htm

[11] The "Injunction Generator" by ubermorgen.com http://www.ipnic.org/intro.html, which automatically generates legal injunctions and the content-censoring web proxy server "insert coin" by Alvar Freude and Dragan Espenschied

art and research laboratory software development—was an academic matter, and even hacker culture was limited to elite institutes such as MIT and Berkeley, today there is not only a mass culture and everyday aesthetics of software. As is indicated, for example, by the debates on Free Software, software monopolies, software patents, adware and spyware, software has become an increasingly political matter. However, cultural criticism of software only exists in scattered efforts, for example in essays by Wolfgang Hagen and Matthew Fuller and on the mailing list "softwareandculture" initiated by Jeremy Hunsinger.[12]

SOFTWARE ART IS NOT PROGRAMMER'S ART

Historically, the gap between the "using" and "programming" computers results from the iconic user interface and its commercialization by Apple and Microsoft, which for the first time assigned the two methods of operation different media: iconic images to "usage" and alphanumerical text to "programmation." It was only in this way that the programming of computers became a black art, mystified as a supposedly elitist, specialist knowledge.[13] Programmers have of course cultivated this myth, taking over the ideological heritage of the late 18th century by creating, in the hacker, a reincarnation of the romantic genius.

Every discourse on software art, therefore, is in danger of continuing the cult of the programming genius. This is countered by imaginary, simulated and dysfunctional software as well as by manipulations of existing software which require no programmer expertise at all.[14] If software can be not only the material of software art, but also the object of its reflection, this reflection can also be set into completely different material to software itself, as was demonstrated, for example, by the work "n:info" by Julia Guther and Jakob Lehr presented at the "browserday" Festival 2001. This was a browser in the form of a portable window frame, a work that turns the rhetoric of

---

`http://odem.org/insert_coin/` are two convincing examples of politically activist software art.

[12]Wolfgang Hagen, Der Stil der Sourcen, [**?**], Matthew Fuller, Behind the Blip [**?**], softwareandculture homepage and archive at `http://listserv.cddc.vt.edu/mailman/listinfo/softwareandculture`

[13]Although in order to be able to program a computer in one of the common languages, all that is needed is a knowledge of variables, loops and if-then-conditions.

[14]Like for example the "SCREEN SAVER" by Ivan Khimin and Eldar Karhalev `http://runme.org/project/+screensaver/`, a configuration of the Windows screen saver into a suprematist-hypnotic, floating square.

iconic PC software on its head by presenting an analog device as a metaphor for digital software, and thus exposing the software application "Web browsing" as a cultural technique, a mode of perception and of thought.[15] There is nothing, therefore, to be said against software art in the form of a painted picture.

GENRE CLICHÉS COULD MAKE SOFTWARE ART BORING

Of course, the danger of becoming paralyzed in stereotypes also exists in art forms which, like Fluxus, do not define themselves through specific materials. Nevertheless: software art would become boring if—in the perception of critics, curators and juries—its repertoire were to be narrowed down to experimental web browsers, data visualizations, modified computer games and cracker codes (like computer viruses and fork bombs). Another problem is the association of software art with the "media art" system, with the side-effect that artistically interesting computer programs—like those which emerge in the field of GNU/Linux and Free Software, for example—do not reach software art competitions, festivals and exhibitions.

THE DISCUSSION WHETHER SOFTWARE ART CAN BE CALLED ART AT ALL IS NOT ACTUALLY CONCERNED WITH SOFTWARE ART

Over and over again, the question is raised whether software art should be given the suffix "art" at all. The naïve version of the question views software as simply engineering, and therefore doubts its artistic value; a more complex variation complains that yet again a multifaceted culture has had the unnecessary criterion, the attribute of "art" stuck onto it. And indeed, just as, for example, traditional Japanese culture existed without a concept of the liberal arts as opposed to the applied arts, an understanding of "art" in the old sense of "ars", of artifice, is widespread both in free and corporate software developer culture. Thanks to the hacker imagination of Free Software programmers, it is certainly possible to combine the works of declared artists and declared non-artists in the field of software art, as a festival exhibition curated by artist Alexei Shulgin has demonstrated.[16] Nonetheless, ultimately, objections to the "art" suffix as it is

[15] http://myhd.org/ninfo

[16] Examples of this are the award-winning hacker program "WinGluk Builder" at the readme-Festival 2002 http://www.macros-center.ru/read_me/art_work/27/readme27.zip and the program "Tempest for Eliza" exhibited in the following

applied to software art are only a vehicle with which to question the concept of "art" itself.

In his review "Don't Call it Art: Ars Electronica 2003,"[17] Lev Manovich comes up with a third, refined variant of the objection when he called software art "not art" because, due to its focus on a specific material, it did not belong to the system of "contemporary art." However, the contemporary art that can be seen in galleries, on fairs and in museum exhibitions is made up of subdisciplines which display anything but a neutral attitude to their material: on the one hand there is large-format painting and photo art for private collectors, on the other hand academic (often video-aided) installation art, which is typically exhibited in state- subsidised buildings and produced by curators and artists trained in cultural studies. Quite apart from that, software art is simply a generic term no different to painting, sound, script or video art—nor was it defined by the artists themselves, but by critics and curators, who observed a trend towards work using software as its material in contemporary digital art.[18]

The term "software art" is therefore easy to legitimate, because it results quite simply from the fact that remarkable contemporary art (like the works mentioned in this text) is being produced in the form of software, therefore demanding a theory and criticism of software art.

year http://www.erikyyy.de/tempest/, which implemented a short-wave radio broadcast by means of screen graphics on tube monitors.

[17]Published on the mailing lists "Rhizome" and "Nettime", [**?**]

[18]For example Saul Albert in his 1999 essay "Artware" [**?**], Alex Galloway in "Year in Review: State of net.art 99" http://switch.sjsu.edu/web/v5n3/D-1.html, Andreas Broeckmann, who added a software section to the Transmediale-Festival in the year 2000 and, in 2001, Tilman Baumgärtel with his article "Experimentelle Software" [**?**]