Can notions of text which were developed without electronic texts in mind be applied to digital code, and how does literature come into play here?

DIGITAL CODE AND LITERARY TEXT

FLORIAN CRAMER

ABSTRACT. This paper is based on the general (yet disputable) assumption that the theoretical debate of literature in digital networks has shifted, just as the poetic practices it is shaped after, from perceiving computer data as an extension and transgression of textuality (as manifest in such notions as "hypertext", "hyperfiction", "hyper-/ multimedia") towards paying attention to the very codedness – i.e. textuality – of digital systems themselves. Several phenomena may serve as empirical evidence:

- The early focus of conceptualist Net.art on the aesthetics and politics of code;
- in turn, the impact of Net.art aesthetics on experimental literature / poetry in the Internet;
- the close discooursive affinity of Net.art to political activism in the Internet;
- the close aesthetic affinity of Net.art to a the languages and codes of an older, technically oriented "hacker" culture (of Chaos Computer Club, 2600, and others);
- a convergence of the three cultures mentioned above – Net.art, net activism and hacker culture;
- (a) Free/Open Source Software and/or (b) open network protocols as key discursive, political and aesthetical issues in all these camps;
- finally, the impact of hacker aesthetics, Net.art aesthetics, code aesthetics and network protocol aesthetics on contemporary writing in the Internet. (See the work of mez, Alan Sondheim, Talan Memmott, Ted Warnell and others.)

The question is how "Codeworks" (Alan Sondheim) fit notions of text that were crafted without digital code – most importantly: machine-executable digital code – in mind, and vice versa. Is it a coincidence that, in their poetical appropriation of low-level Internet codes, codeworks ended up aesthetically resembling concrete poetry? And, apart from aesthetic resemblances, how do computer programs relate to literature? Is that what is currently being discussed as "Software Art" a literary genre?

Date: Sept. 27, 2001.

CODE

Since computers, the Internet and all digital technologies are based on zeros and ones, they are based on code. Zeros and ones are an alphabet which can be translated forth and back between other alphabets without information loss. It does, in my point of view, make no sense to limit the definition of the alphabet in general to that of the Roman alphabet in particular when we can the same textual information in this very alphabet, as Morse code, flag signs or transliterated into zeros and ones. The Internet and computers run on alphabetic code, whereas, for examples, images and sound can only be digitally stored when translating them into code, which – unlike the translation of conventional text into digital bits – is a lossy, that is, not fully reversible and symmetric translation. Sounds and images are not code by themselves, but have to be turned into code in order to be computed; where as any written text already is code. Literature therefore is a privileged symbolic form in digital information systems. It is possible to automatically search a collection of text files for all occurences of the word "bird", but doing the same with birds in a collection of image files or bird songs in a collection of audio files is incomparably tricky and error-prone, relying on either artificial intelligence algorithms or manual indexing, both of which are methods to translate non-semantic writing (pixel code) into semantic writing (descriptions).

The reverse is true as well: We can perfectly translate digital data and algorithms into non-digital media like print books, as long as we translate them into signs coded according to the logic of an alphabet. This is what is done, for example, in programming handbooks or in technical specification manuals for Internet standards. Today there are two notorious examples of a forth-and-back translation between print and computers:

(1) The sourcecode of Phil Zimmerman's cryptography program "Pretty Good Privacy" (PGP). The PGP algorithms were legally considered a weapon and therefore became subject to U.S. export restrictions. To circumvent this ban, Zimmerman published the PGP sourcecode in a book. Unlike algorithms, literature is covered by the U.S. First Amendment of free speech. So the book could be exported outside the United States and, by scanning and retyping, translated back into an executable program;
(2) the sourcecode of DeCSS, a small program which breaks the cryptography scheme of DVD movies. Since U.S. jurisdiction declared DeCSS an "illegal circumvention device" according to the new Digital Millennium Copyright Act (DMCA), the ban equally affected

booklets, flyposters and t-shirts on which the DeCSS sourcecode was printed.

That code is speech is a fact stressed again and again by programmers and is also at the heart of Lawrence Lessig's legal theory of the Internet ([Les00]). It is, strictly speaking, sloppy terminology to speak of "digital media". There actually is no such thing as digital media, but only digital information. Digital information becomes "media" only by the virtue of analog output; computer screens, loudspeakers, printers are analog output devices interfaced to the computer via digital-to-analog conversion hardware like video and sound cards or serial interfaces.[1]

An average contemporary personal computer uses magnetic disks (floppy and hard disks), optical disks (CD-ROM and DVD-ROM) and chip memory (RAM) as its storage media, and electricity or fiber optics as its transmission media. Theoretically, one could build a computer with a printer and a scanner which uses books and alphabetic text as its storage media.[2] Alan Turing showed that no electronics are needed to build a computer; the Boston Computer Museum even features a mechanical computer built from wooden sticks.

Juxtapositions of "the book" and "the computer" are quite misleading, because they confuse the storage and analog output media (paper versus a variety of optical, magnetical and electronical technologies) with the information (alphabetical text versus binary code). It further ignores, by the way, the richness of storage and transmission media in traditional literature which, aside from the book, include oral transmission and mental storage, audio records and tapes, the radio and television, to name only a few.

If there is, strictly speaking, no such thing as digital media, there also is, strictly speaking, no such thing as digital images or digital sound. What we refer to as a "digital image" is a piece of code containing the machine instructions to produce the flow of electricity with which an analog screen or an analog printer is made to display an image.[3]

[1]On the reverse end of the chain, keyboards, mice, scanners and cameras are analog-to-digital converters.

[2]While such a machine would operate slower than with magnetical or optical media, it would provide more robust and durable information storage on the other hand.

[3]Normally, this code is divided into three pieces, one – the so-called sound or image file – containing the machine-independent and program-independent abstract information, the second – the so-called display program – containing the instructions to mediate the abstracted information in a machine-independent, yet not program-independent format to the operating system, the third – the so-called operating system –, mediating the program output to the output machine, whether a screen or a printer. These three code layers however are nothing but conventions. Theoretically, the "digital image" file could in itself contain

Of course it is important whether a sequence of zeros and ones translates, into, say, an image because that defines its interpretation and semantics. The point of my (admittedly) formalistic argumentation is not to deny this, but to underline that

(1) when we speak of "multimedia" or "intermedia" in conjunction with computers, digital art and literature, we actually don't speak of digital systems as themselves, but about translations of digital information into analog output and vice versa;
(2) text and literature are highly privileged symbolic systems in these translation processes because (a) they are already coded and (b) computers run on a code.

Literature and computers meet first of all where alphabets and code, human language and machine language intersect, secondly in the interfacing of analog devices through digital control code. While of course we cannot think of code without media because we can't read it without them, the computer does not really extend literary media themselves. All those output media – electricity, electrical sound and image transmission etc. – existed before and without computers and digital information processing.

I therefore have to revise the position I took in several my previous writings[4]: If we speak of digital poetry, or computer network poetry, we don't have to speak of certain media, and we don't even have to speak of specific machines. If computers can be built from broomsticks – and networked via shoestrings; if any digital data, including executable algorithms, can be printed in books and from them read back into machines or, alternatively, executed in the mind of the reader, there is no reason why computer network poetry couldn't or shouldn't be printed as well in books.

Perhaps the term of digital "multimedia" – or better: "intermedia" – would be more helpful if we redefine it as the *the possibility to losslessly translate information from one sign system to the other, forth and back, so that the visible, audible or tacticle representation of the information becomes arbitrary*. A state not be achieved unless the information is not coded in some kind of alphabet, whether alphanumerical, binary, hexadecimal or, if you like, Morse code.

---

all the code necessary to make itself display on analog end devices, including the code that is conventionally identified as a boot loader and core operating system.

[4]like in the paper "Warum es zuwenig interessante Computerdichtung gibt. Neun Thesen" (" Nine Points on Why there is so little interesting computer net literature"), `http://userpage.fu-berlin.de/~cantsin/homepage/writings/net_literature/general/karlsruhe_2000//karlsruher_thesen.pdf`

LITERATURE

Synthesis: putting things together. To observe the textual codedness of digital systems of course implies the danger of generalizing and projecting one's observations of digital code onto literature as a whole. Computers operate on machine language, which is syntactically far less complex than human language. The alphabet of both machine and human language is interchangeable, so that "text" – if defined as a countable mass of alphabetical signifiers – remains a valid descriptor for both machine code sequences and human writing. In syntax and semantics however, machine code and human writing are not interchangeable. Computer algorithms are, like logical statements, a formal language and thus only a restrained subset of language as a whole.

However, I believe it is a common mistake to claim that machine language would be only readable to machines and hence irrelevant for human art and literature and, vice versa, literature and art would be unrelated to formal languages.

It is important to keep in mind that computer code, and computer programs, are not machine creations and machines talking to themselves, but writings by humans.[5] The programmer-artist Adrian Ward suggests that we put the assumption of the machine controlling the language upside down:

> "I would rather suggest we should be thinking about embedding our own creative subjectivity into automated systems, rather than naively trying to get a robot to have its 'own' creative agenda. A lot of us do this day in, day out. We call it programming."[6]

Perhaps one also could call it composing scores, and it does not seem accidental to me that musical artists have picked up and grasped computers much more thoroughly than literary writers. Western music is an outstandig example of an art which relies upon written formal instruction code. Self-reflexive injokes such as "B-A-C-H" in Johann Sebastian Bach's music, the visual figurations in the score of Erik Satie's "Sports et divertissements" and finally the experimental score drawings of John Cage shows that, beyond a merely serving the artwork, formal instruction code has an aesthetic

[5]No computer can reprogram itself; self-programming is only possible within a limited framework of game rules written by a human programmer. A machine can behave differently than expected, because the rules didn't foresee all situations they could create, but no machine can overwrite its own rules by itself.

[6]quoted from an E-Mail message to the "Rhizome" mailing list, May 7, 2001

dimension and intellectual complexity of its own. In many works, musical composers have shifted instruction code from classical score notation to natural human language. A seminal piece, in my opinion, is La Monte Young's "Composition No.1 1961" which simply consists of the instruction "Draw a straight line and follow it."[7] Most Fluxus performance pieces were written in the same notation style. Later in 1969, the American composer Alvin Lucier wrote his famous "I am sitting in a room" as a brief spoken instruction which very precisely tells to perform the piece by playing itself back and modulating the speech through the room echoes.

In literature, formal instructions is the necessary prerequisite of all permutational and combinatory poetry.[8] Kabbalah and magical spells are important examples as well. But even in a conventional narrative, there is an implict formal instruction of how – i.e. in which sequence – to read the text (which may be followed or not, as opposed to hypertext which offers alternative sequences on the one hand, but enforces its implicit instruction on the other). Grammar itself is an implicit, and very pervasive formal instruction code.

Although formal instruction code is only a subset of language, it is still at work in all speech and writing.

It is particularly remarkable about computing that the namespace of executable instruction code and nonexecutable code is flat: If, like Inke Arns proposed (using structuralist terminology), we speak of instruction code as a "genotext" and non-instruction code as a "phenotext", then computed language differs from spoken language in that both genotext and phenotext are coded in the same alphabet of bits and bytes, whereas in spoken language, the genotext of grammar is an implicit, mental code. One cannot tell from a snippet of digital code whether it is machine-executable or not, a phenotext or a genotext. In fact every digital code, even a "Project Gutenberg" text of, say, Homer's "Odyssee", is potentially executable depending on whether there's other code – a compiler, runtime interpreter or the embedded logic of a microprocessor – capable to process it as machine instructions. Computer code is highly recursive and highly architectural, building upon layers of layers of code.

Analysis: taking things apart. The fact that one cannot tell from any piece of code whether it is machine-executable or not provides the principle condition of all E-Mail viruses on the one hand, and of the codeworks of jodi,

[7][uEH90], no page numbering

[8]Some historical examples have been adapted online on my website `http://userpage.fu-berlin.de/~cantsin/permutations`

antiorp/Netochka Nezvanova, mez, Ted Warnell, Alan Sondheim, Kenji Siratori – to name only a few – on the other; work that, unlike the actual viri, is fictional in that it aesthetically pretends to be potentially viral machine code.[9]

The codeworks, to use a term coined by Alan Sondheim, of these writers and programmer-artists are prime examples for a digital poetry which reflects the intrisic textuality of the computer. But they do so not by being, to quote Alan Turing via Raymond Queneau, computer poetry to be read by computers[10], but by playing with the confusions and thresholds of machine language and human language, and by reflecting the cultural implications of these overlaps. The "mezangelle" poetry of mez/Mary Ann Breeze, which mixes programming/network protocol code and non-computer language to a portmanteau-word hybrid, is an outstanding example of such a poetics.

Compared to earlier poetics of formal instruction, like in La Monte Young's *Composition 1961*, in Fluxus pieces and in permutational poetry, an important difference can be observed in the codeworks: The Internet code poets and artists do not construct or synthesize code, but they use code or code grammars they found and take them apart. I agree with Friedrich Block and his "Eight Digits of Digital Poetry" `http://www.dichtung-digital.de/2001/10/17-Block/index-engl.htm` that digital poetry should be read in the history and context of experimental poetry. A poetics of synthesis was characteristic of combinatory and instruction-based poetry, a poetics of analysis characterized Dada and its successors. But one hardly finds poetry with an analytical approach to formal instruction code in the classical 20th century avant-garde.[11] Internet code poetry is being written in a new – if one likes, post-modernist – condition of machine code abundance and overload.

The hypothesis that there is no such thing as digital media, but only digital code which can be stored in and put out on any analog medium, is perfectly verified by codework poetry. Unlike hypertext and multimedia poets, most of the artists mentioned here write plain ASCII text. The contradiction between a complex techno-poetical reflection and low-tech communication is only a seeming one; quite on the contrary, the low-tech is crucial to the critical implication of the codework poetics.

[9]with the "biennale.py" computer virus of the net art groups `http://www.0100101110111001.org` being the only exception to date.

[10][Que61], p.3

[11]An exception being the the ALGOL computer programming language poetry written by the Oulipo poets François le Lionnais and Noël Arnaud in the early 1970s, see [MB98], p.47

The development of hyperfiction and multimedia poetry practically paralleled the construction of the World Wide Web; hyperfiction authors rightfully saw themselves as its pioneers. In the course of nineties, they continued to push the technical limits of both the Internet and multimedia computer technology. But since much digital art and literature became testbed applications for new browser features and multimedia plugins, it simultaneously locked itself into non-open, industry-controlled code formats.[12] Whether intentional or not, digital art thus strongly participated in the reformatting of the World Wide Web from an open, operating system- and browser-agnostic information network to a platform dependent on propietary technology.

By readjusting the reader's attention from software surfaces which pretended not to be code back to the code itself, codeworks have apparent aesthetical and political affinities to hacker cultures. While hacker cultures are far more diverse than the singular term "hacker" suggests[13], hackers could as well be distinguished between those who put things together – like Free Software and demo programmers – and those who take things apart – like crackers of serial numbers and communication network hackers from YIPL/TAP, Phrack, 2600 and Chaos Computer Club schools. Code poets have factually adopted many poetical forms that were originally developed by various hacker subcultures from the 1970s to the early 1990s, including ASCII Art, code slang (like "7331 wAr3z d00d" for "leet [=elite] wares dood") and poetry in programming languages (such as Perl poetry), or they even belong to both the "hacker" and the "art" camp[14]

From its beginning on, conceptualist net.art engaged in a critical politics of the Internet and its code, and continues to be closely affiliated with critical discourse on net politics in such forums as the "Nettime" mailing list. In its aesthetics, poetics and politics, codework poetry departs from net.art rather than from hyperfiction and its historical roots in the Brown University literature program.

How does digital code relate to literary text? If one discusses the poetics of digital code in terms of the poetics of literary text – instead of discussing literary text in terms of digital code –, one may consider both of them interrelated without having to subscribe, as John Cayley suggested

[12]like Shockwave, QuickTime and Flash

[13]Boris Gröndahl's (German) Telepolis article "The Script Kiddies Are Not Alright" summarizes the multiple, sometimes even antagonistic camps associated with the term "hacker", `http://www.heise.de/tp/deutsch/html/result.xhtml?url=/tp/deutsch/inhalt/te/9266/1.html`

[14]like Walter van der Cruijsen from `http://www.desk.org`desk.org and the ASCII Art Ensemble.

in his abstract to the German “p0es1s” conference[15], to Friedrich Kittler’s techno-determinist media theory; a theory which, despite all of its intellectual freshness seem to fall into the metaphysical trap Derrida described in “Écriture et différence”: By replacing one metaphysicial center (in Kittler’s case: “Geist”/spirit, “Geistesgeschichte”/intellectual history and “Geisteswissenschaft”/humanities) with another one – technology, history of technology and technological discourse analysis – it writes on metaphysics under a different label, contrary to its own claim to have rid itself from it.

The subtitle of this text writes an open question: “Can notions of text which were developed without electronic texts in mind be applied to digital code, and how does literature come into play here?” For the time being, I would like to answer this question at best provisionally: While all literature should teach us to read and deal with the textuality of computers and digital poetry, computers and digital poetry might teach us to pay more attention to codes and control structures coded into all language. In more general terms, program code contaminates in itself two concepts which are traditionally juxtaposed and unresolved in modern linguistics: the structure, as conceived of in formalism and structuralism, and the performative, as developed by speech act theory.

References

[Les00] Lawrence Lessig. *Code and Other Laws of Cyberspace*. Basic Books, New York, 2000. 3

[MB98] Harry Mathews and Alastair Brotchie, editors. *Oulipo Compendium*. Atlas Press, London, 1998. 7

[Que61] Raymond Queneau. *Cent mille milliards de poèmes*. Gallimard, Paris, 1961. 7

[uEH90] Galerie und Edition Hundertmark. George Maciunas und Fluxus-Editionen, 1990. 6

[15]`http://www.p0es1s.net/poetics/symposion2001/a_cayley.html`