Digital Code and Literary Text

Florian Cramer

Freie Universität Berlin, Department of Comparative Literature
Sept. 2001, revised in 2002 and 2003

ABSTRACT

This paper is based on the (perhaps questionable) assumption that theoretical debates of digital literature have shifted, just as the poetic practices they refer to, from a perception of computed data as an extension and transgression of textuality (as manifest in such notions as ‘‘hypertext’’, ‘‘hyperfiction’’, ‘‘hyper-/ multimedia’’) towards paying attention to the basic digital textuality of computer 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 poetry in the Internet;
- the close discursive 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
- a convergence of the three cultures mentioned above - Net.art, net activism and hacker culture;
- Free/Open Source Software and/or 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. (As in the work of mez, Alan Sondheim, Talan Memmott, Ted Warnell and others.)

The question is how ‘‘codeworks’’ (Alan Sondheim) still fit notions of text that were conceived without reflecting digital code in general and digital program code in particular. Is it a pure coincidence that codeworks ended up aesthetically resembling concrete poetry when they started to mimick the aesthetics of low-level program and network codes? And, apart from aesthetic resemblances, how do computer programs relate to literature, is software art literary?

2 November 2003

Digital Code and Literary Text

Florian Cramer

Freie Universität Berlin, Department of Comparative Literature
Sept. 2001, revised in 2002 and 2003

1. Code

Since computers, the Internet and all digital technologies operate 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 is, in my point of view, pointless to limit the definition of ''alphabet'' in general to that of the Roman alphabet in particular given that the same textual information can be coded as letters, 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 be digitally stored only after they have been translated from analog visuals and sounds into a numerical code, which - unlike the translation of conventional text into digital bits - is a lossy, i.e. 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 that can be typeset in a printing press 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'' while 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 (word descriptions).

The reverse is true as well: Digital data and algorithms can be losslessy stored in non-digital media like print books, as long as they are translated into signs coded according to the logic of an alphabet, as it is done, for example, with programming handbooks and technical specification manuals for Internet standards. To date, there are two famous examples of a forth-and-back translation between print and electronic computers:

1 the sourcecode of Phil Zimmerman's cryptography program ''Pretty Good Privacy'' (PGP).

PGP's algorithms were legally considered a weapon and therefore became subject to U.S. export restrictions. To circumvent the 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 machine 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.

The opinion that code is ''speech'' (in the sense of writing) is common amongby programmers, and is also at the heart of Lawrence Lessig's legal theory of the Internet.[1] It is, strictly speaking, sloppy terminology to speak of ''digital media'', since there is no such thing as digital media, but only digital information. Digital information becomes ''media'' only by the virtue of analog technology; computer screens, loudspeakers, printers for example are analog output devices linked to the computer via digital-to-analog converters like video and sound cards.[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

[1] Lessig, Lawrence, *Code and Other Laws of Cyberspace,* Basic Books, New York (2000).

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

which uses books and alphabetic text as its storage media. Alan Turing showed that no electronics are needed to build a computer; the Boston Computer Museum features a mechanical computer built from broomsticks.

Juxtapositions of ‘‘the book’’ and ‘‘the computer’’ are misleading because they confuse the storage and output media (paper versus a variety of optical, magnetical and electronical technologies) with the information (alphabetical text versus binary code). They also ignore 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; precisely because literature is coded, it is (unlike sculpture and painting) not bound to specific material and can be transmitted via virtually any new medium.

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 is referred 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.[2]

Of course it matters whether a sequence of zeros and ones translates, into, for example, an image because that defines its semantics and hence interpretation and usage. 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 proper, 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 computing meet where alphabets and code, human language and machine language intersect, secondly in the interfacing of analog devices through digital control code. While of course code does not exist for human readers without media which make it perceivable, the computer does not extend the media of literature. All those output media - electricity, electrical sound and image transmission etc. - existed and continue to exist without computers and digital information processing.

To revise a position I took previously,[3] the notion of digital poetry, or computer network poetry, doesn’t imply specific media, and not even specific machines. If computers can be built from broomsticks - and computer networks via shoestrings or bongo drums (as realized <http://eagle.auc.ca/~dreid/> -; if digital data, including executable algorithms, can be printed in books and read back from them 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.

The term of digital ‘‘multimedia’’ - or better: ‘‘intermedia’’ - would be more helpful if redefined as the *possibility to losslessly translate information from one notation 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.

2. Literature

[2] 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 operating system -, mediating the program output to the output machine, whether a screen or a printer. Yet these three code layers are nothing but conventions. Theoretically, the ‘‘digital image’’ file could in itself contain 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.

[3] like in the paper ‘‘Warum es zuwenig interessante Computerdichtung gibt. Neun Thesen’’, <http://user-page.fu-berlin.de/~cantsin/homepage/writings/net_literature/general/karlsruhe_2000//karlsruher_thesen.pdf>

2.1. 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 a language which lacks semantics and is syntactically far less complex than common human languages. 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. After all, computer code, and computer programs, are not machine creations and machines talking to themselves, but writings by humans.[4] 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.''[5]

One also could call it composing scores, and it is not coincidental 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. Code-unjokes 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 the experimental score drawings of John Cage show that, beyond a merely serving the artwork, formal instruction code has an aesthetic 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 is La Monte Young's ''Composition No.1 1961'' which simply consists of the instruction ''Draw a straight line and follow it.''[62] Most Fluxus performance pieces were written in the same notation style. Later in 1969, the American composer Alvin Lucier wrote his famous composition ''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,[7]
which in turn recur to the tradition of letter combinatorics in Kabbalah and magical spells. Also 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.

The grammar of computing however differs from the grammar of natural language in that, to use computer science terminology, the namespace of instruction and execution is the same: 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 and in the same medium of, predominantly, flows of electricity, whereas in spoken language, the genotext of grammar is implicit and not explicit in the writing. One cannot tell from a snippet of digital code whether it is machine-executable or not, a phenotext or a genotext. Every digital code, even a ''Project Gutenberg'' text of, for example, Homer's ''Odyssee'', is

[4] No computer can reprogram itself; self-programming is only possible within a predefined 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 the rules of its game.

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

[2] Galerie und Edition Hundertmark, *George Maciunas und Fluxus-Editionen*, Galerie und Edition Hundertmark, Köln (1990).

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

potentially executable depending on whether other code - a compiler, runtime interpreter or the embedded logic of a microprocessor - is capable of processing it as machine instructions. Computer code is highly recursive and highly architectural, building upon layers of layers of itself.

2.2. Analysis: taking things apart

The fact that one cannot tell from any piece of digital code whether it is a machine-executable program or not provides the principle condition of all E-Mail viruses on the one hand, and of the codeworks of jodi, antiorp/Netochka Nezvanova, mez, Ted Warnell, Alan Sondheim, Kenji Siratori - to name only a few pioneers of the genre - on the other; work that, unlike the actual viri, is fictional in that it aesthetically pretends to be viral machine instructions.[8]

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[93] , 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, codeworks differ in one significant aspect: The Internet code poets and artists do not construct or synthesize code, but use prexisting code or code grammars to 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 an historical 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 instructions in the classical 20th century avant-garde, an exception being the ALGOL computer programming language poetry written by the Oulipo poets François le Lionnais and Noël Arnaud in the early 1970s.[4] 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 onto and put out via analog media, is perfectly mirrored in codework poetry. Unlike hypertext and multimedia poets, most of the code artists mentioned write plain ASCII text. To see complex techno-poetical reflections and low-tech media as contradictory would miss the point. Quite on the contrary, low-tech is crucial as a critical element of codework poetics.

hyperfiction and multimedia poetry was, as a matter of fact, invented in anticipation and in parallel to the World Wide Web; hyperfiction authors rightfully considered themselves 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.[10]
Whether intentional or not, digital art thus 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 code (like 3d simulations) that pretended not to be code back to code as code, codeworks have clear aesthetic and political affinities to hacker cultures. While hacker cultures are far more diverse than the singular term ''hacker'' suggests[11] , hackers could as

---

[7] with ''biennale.py''of the net artists <http://www.0100101110111001.org> being the exception of a code that is a computer virus in fact.

[3] Queneau, Raymond, *Cent mille milliards de poèmes,* p. 3, Gallimard, Paris (1961).

[4] *Oulipo Compendium,* p. 47, Atlas Press, London (1998).

[8] like Shockwave, QuickTime and Flash

[9] 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>

well be distinguished between those who put things together, like Free Software and demo programmers, [5] 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 artists 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[12] and poetry in programming languages (like Perl poetry), or they even belong to both the ‘‘hacker’’ and the ‘‘art’’ camp.[13]

From its beginning on, conceptualist net.art engaged in a critical politics of the Internet, 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 apart from these practices? 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 -, both can be considered closely related to each other. This does not imply, as John Cayley suggested in his abstract to the German ‘‘p0es1s’’ conference[14] , subscribing to Friedrich Kittler’s technocentric media theory; a theory which, despite its provocation of humanities and philosophical idealism seem to fall into a metaphysical trap which Jacques Derrida described in ‘‘Écriture et différence’’: By replacing one metaphysical 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 continues 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, by the its codedness, should help readers understand the language and textuality of computers and digital poetry, computers and digital poetry might teach to pay more attention to codes and control structures coded not only into machines, but into all language, by contaminating within itself two antagonistic linguistic concepts, the structural and performative.

---

[5] Levy, Steven, *Hackers*, Project Gutenberg, Champaign, IL (1986 (1984)).

[10] like ‘‘7331 wAr3z d00d’’ for ‘‘leet [=elite] wares dood’’

[11] like Walter van der Cruijsen from the ASCII Art Ensemble or the Italian programmer-artist and -activist jaromil.

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