# ENTERING THE MACHINE AND LEAVING IT AGAIN: POETICS OF SOFTWARE IN CONTEMPORARY ART

FLORIAN CRAMER

## ABRAHAM MOLES AND THE SITUATIONIST INTERNATIONAL

The history of algorithmic programming in art is much older than that of electronics: It includes, for example, word permutation poetry like that of the 3rd century Latin poet Optatianus Porfyrius and automatic composition formulas like Athanasius Kircher musical automata of the 17th century (both created in Italy). However, today I would like to speak about the poetics of software in recent and contemporary digital art. It is, of course, inseparable tied to modern computing.

In 1962, physicist and philosopher Abraham M. Moles wrote a seminal programmatic and theoretical outline of computational art, the *first manifesto of permutational art* (*erstes manifest der permutationellen kunst*) [slide]. The booklet combines structuralist and cybernetic theory with examples of mathematics, contemporary experimental poetry, music, visual art, and even mysticism and erotic art. Moles' demanded to refound both the poetics and the aesthetics of art on the grounds of computation: As composition, the new art would "narrow down and exhaust the field of possibilities accessible through a set of rules."

Moles even speaks of the new art as a "fundamentally anti-semantic activity." In his conclusion, he writes that artists would turn into "programmers" and, quote, "from now on, artworks will be realized either by machines or through their own consumers".

With this statement, Moles pretty much set the agenda of the new computer arts, and today, after almost half a century, it still phrases a virulent point. To my knowledge, his "manifesto of permutational art" is the earliest and most concise program of what later would be called generative art.

However, Moles' implication that computer-generated art would be only formal and eliminate all cultural semantics, was controversial.

*Date*: Feb. 7th, 2006.

Already in 1963, one year after the manifesto had appeared, it made him subject of a fierce polemical attack by an other group of contemporary artists and theorists, the Situationist International. On the surface, the programs of both Moles and the Situationists shared many common points. Drawing both from the sociology of Henri Lefebvre, they conceived of industrial automation as the root of a society of surplus and leisure. In the early 1960s, painter Guiseppe Pinot-Gallizio had even promoted a machine-generated "industrial painting" within the Situationist International. However, the Situationists were not fighting against semantics, but – in their indebtedness to romanticist and surrealist programs – on the contrary advocating a revolutionary imagination. On these grounds, Guy Debord attacked Moles as a "petite tête" ("small head") technocrat and told him "tu es un robot".

I would like to argue that this schism between a rigidly formalist and a rigidly "imaginist" (to use a word by Situationist Asger Jorn) poetics obstructed computer arts for almost three decades until the advent of the personal computer and the Internet.

## SYNTHETIC COMPUTER ART

Before the personal computer and the Internet, computer art was thinkable only as synthetic creation, i.e. the construction of algorithms in clean-room laboratories. Of course, this was the inevitable condition of computer-based generative art and computer science in general in the 1960s and 1970s when almost all software had to be written from scratch. But it is also true from computational art that did not actually work with electronic computers, and probably not even think of itself as computational art at all.

**Proto- and Para-Computer Art.** In 1960, the composer La Monte Young who's know today mainly as a pioneer of minimal music wrote a piece that consisted solely of the following instruction [slide]:

> "Draw a straight line and follow it."

First of all, it's a performance score. But its instruction is unambiguous and formal enough to be also executed by a machine and adapted as a computer program. It is, in other words, an algorithm and a source code. However, it is an impossible algorithm at the same time. If either the performer or the machine would radically carry out the instruction, this seemingly simple piece mutates in the most monstruous art work of all time. One cannot consequently draw a straight

line and follow it without going beyond physical limits and writing a circular inscription into the whole earth. So the piece implies a philosophical defiance of space and time constraints, and leaves the piece in a non-resolvable gap between its physical execution and its mental, conceptual imagination. Doing so, this score is not only the founding document of minimal music, but it also creates a paradoxical union of minimalism and late romanticist Wagnerian total artwork (Gesamtkunstwerk), by the virtue of a source code that condenses an abundance into one line of instruction. The piece reverses subject and object: ultimately, the performers turns into its object, and the line becomes its subject.

In other words, the conflict articulated in the controversy between Moles and the Situationist exists within the piece. It is not resolved, but sustained as a paradox.

**Software as Metaphor of Dematerialization.** In the immediate context of American Fluxus and conceptual art, the notion of "software" got introduced in the early 1970s, however in a semantics that was strangely detached from both Moles' theoretical and La Monte Young's practical anticipation of software art. In 1973, Lucy Lippard published her famous book *Six years* with the subtitle "The Dematerialization of the Art Object from 1966 to 1972". The keyword "dematerialization" also sums up how the term "software" had been introduced and understood in contemporary art since 1970. "Software" had been the title of an art show curated by critic Jack Burnham in New York in 1970. It mostly consisted of concept art works, partly juxtaposed with experimental computer software development projects such as Ted Nelson's first prototype of a hypertext system. However, the emphasis of the exhibition was not algorithms in art, but immaterial "software" as opposed to material hardware. As Edward A. Shanken puts it in an essay on Burnham's exhibition, the exhibition used the term software as a "metaphorical premise" for the dematerialization of art, not as a reflection of computation.

In the same year, Sidney Youngblood published his book "Expanded Cinema", a reference work on the extension of experimental film into cinematic performances and installations, including video, "cybernetic cinema" and "computer films". Elsewhere, Youngblood uses the same broad metaphorical notion of software as immaterialization as Burnham when he writes:

"Just as every fact is also metaphysical, every piece of hardware implies software: information about its existence. Television is the software of the earth. Television is invisible. It's not an object." [slide]

Perhaps this phrase was the inspiration for *Radical Software*, an underground magazine for video artists and activists that first appeared in the same year, 1970. Despite its name, and under the same metaphorical premise as Burnharm's exhibition and Youngblood's paragraph, it was not concerned with computing at all, but propagated an "Alternate Television Movement." The issues combined aesthetic reflection with political debates about free media and publicly accessible radio spectrum, much like the contemporary free wireless network movement. Otherwise, the journal conceived of "software" purely as dematerialized art, and did not cover computer programming.

**Software-aided art.** Abraham Moles' idea that artists should become programmers therefore remained restricted to the specialized field, the ghetto, of electronic art or "media art" as it is still exists today – although I think it's outmoded as a category and likely to be given up soon. There is, first of all no computer art without software, unless the hardware is being used as purely non-computational sculptural objects – as bricks. In that respect, all computer art could be called software art. However, in only rare cases, it is an artistic play with the software as a medium, but something that should correctly be called software-aided art. In most computer-generative art, both the software and the hardware acted as mere catalysts. They functioned as black boxes. Neither the hardware, nor the code or its processing was considered the artwork, but only the output: i.e. a computer-generated image, animation, installation or audiovisual piece. Often, this is linked to the concept of an autonomous machine creation, in other words the idea that an artwork is no longer a human product, but a creation by the computer. If we take the original Greek term poiesis, which literally means "making", we could say that in such artworks, poiesis turns into poetics, the making of making. But when making turns into meta-making, human subjectivity is not abandoned. Instead, it just shifts to a second order position, expressing itself in the design of the formula rather than the design of the product. When critics and viewers, fixated on the material product, conclude that technology has done away with human agency behind a work, this is a cognitive fallacy reminiscent of Plato's cave. It is yet another fallacy to believe that conversely on the aesthetic side, i.e.

that of perception of the work, viewers would be liberated through the mechanical variations of the work permitted by the formula.

*Jeffrey Shaw, The Legible City*. To illustrate my point, I would like to fast-forward to the years 1989-1991 and Jeffrey Shaw's computer installation *The Legible City* at the ZKM media arts center in Karlsruhe, Germany [slide]. It is a contemporary classic in the genre of interactive installation art and consists of a video-projected 3D simulation coupled with a stationary bicycle. The projection shows abstract cubic 3D representations of cities of New York, Amsterdam and Karlsruhe. The spectator, or player, of the work sits on the bicycle and cycles, in a "virtual reality" simulation, through the cities. The cityscapes are made up of letters and words written by Shaw's artistic collaborator Dirk Groeneveld. The work was realized on Silicon Graphics workstations, and completed two years before the computer game *Doom* came out and established immersive first-person 3D navigation games on commodity PCs.

The *Legible City* could be called an alternative interface to reading texts on a computer. The conventional flat two-dimensional emulation of print and text pages on the screen is being replaced with an immersive three-dimensional text-scape. The navigation seems to be intuitive thanks to (a) the simulation of anthropomorphic, euclidian space and (b) the emulation of the bicycle as a familiar technology of moving through spaces. So the piece is a perfect example of a concept of digital art as "interactive" simulation and "virtual reality", through anthropomorphist interfaces created with complex, high tech hardware and software, realized, because of that complexity, as an installation in a dedicated high tech art space.

I see Shaw's "Legible city" as hardly anything more than a technology gimmick and a glorified interface design study. *Its subject of the city inscribed with texts reminds of Tommaso Campanella's "Città del sole", the utopian city whose walls are covered with educational explanations of all knowledge and sciences. Just as Campanella's utopia is naive and even problematic, so is Shaw's if it was intended as such. The Legible City is not, as was written, liberating the letter like concrete poetry. While concrete poetry and Marinetti's "parole in libertà" were about freeing type and language from their conventional typographic and grammatical constraints and freeing them, as much as possible, from anthropomorphisms and spatial dimensions, Shaw's system puts them just under a different restraint – the anthropomorphic Euclidian space of the city. It does not take apart writing and reinvents it from*

*scratch, but puts letters into a pseudo-interactive human kitsch world.* One could compare this to the treatment of letters in 19th century children's books or alphabetic toys, only that the latter are interactive in a much more comprehensive sense than the *Legible City*. First of all, Shaw's installation suffers from the fact that it does not think of itself a toy, but takes itself overly serious as an "interactive" and experimental art work. On his web page for the project, Jeffrey Shaw's writes:

> Travelling through these cities of words is consequently a journey of reading; choosing the path one takes is a choice of texts as well as their spontaneous juxtapositions and conjunctions of meaning.

The text misses to reflect that these allegedly "spontaneous juxtapositions and conjunctions" are not spontaneous at all. They only exist within the set of possible combinations encoded into the software that controls the installation. There is no possibility, for example, that a word appears on the screen that has been inscribed into the software before, and no conjunction can be made (a) outside the predetermined possibilities in the program and (b) outside the Euclidian space constraints of the visual simulation. It is, in other words, an illusion of interactivity, spontaneity and intuitivity which the piece sells. Nothing of this could be criticized if the work would actually reflect and critically engage with this illusion. But this lack of reflection, and cognitive fallacy of "interaction" and "spontaneity", is not only characteristic of Shaw's work, but the whole field of generative and so-called interactive art. It is struck with dangerously simplified notion of interactivity – a reductive understanding of interaction as pointing, clicking and other Pavlovian stimulus-response-reactions within the constraints of a programmed box.

## Analytic Computer Art

**Net.art.** In the mid-1990s, net.art embodied a paradigm shift in so-called media art whose nature was institutional, poetic and aesthetic at the same time. In institutional terms, it was the first computer art outside research labs and highly funded institutional environments. In poetic terms, it was low tech computer art. In aesthetic terms, it borrowed from the older low tech artisanship of hacker cultures by adopting its aesthetics of disruption and digital humorism: network collaboration and subversion, ASCII art, code poetry, viruses, computer game modification. While all computer art before had used

a synthetical approach, creating its works from scratch, net.art used an analytical approach of taking digital information and code as material. It was computer art under the new conditions of cheap personal computing. Unlike in earlier computer arts, artists could use ready-made digital information and code "out there" and treat it like Dadaist and Pop art painters treated found objects in their collage work.

One could call it an informal, playful and performative approach to digital art. With the example of the work of jodi and other net.artist, I would like to show how this art developed from experimentation with network information to experimentation with software, and from experimentation with software to performances and interventions.

*jodi*. I would like to start with OSS, an early work from `http://www.jodi.org` [slide oss.jodi.org]. Jodi stands for Joan Heemskerk and Dirk Paesmans, a Dutch-Belgian artist couple. Their early work *OSS* [slide] makes small browser windows pop up and fly around that evade manual control. If one opens the site, it performs a hostile takeover of one's web browser. It is a hack, a punk-like aesthetic and technological hijacking. It involves no simulation, no anthropomorphism, no virtual reality, but is the technology itself read against the grain. It does not simulate an anthropomorphic space in order to be perceived and experienced, but simply uses everyday experience with personal computer operating systems and the Internet as its frame of reference. It is not a high tech installation in a white cube, but low tech running on any home or office computer. The whole source code of the pages takes up less than 10 Kilobyte, i.e. has the average size of a short E-Mail, as opposed to a complex software application with several 100,000 lines of original source code. It uses ready-made, industrial software – a web browser in this case –, however not in an affirmative way, but in an attempt to hack it and subvert its cultural interface paradigms. It is ironical and melancholic to the degree that it promises no computing utopias, and is not futurist human-machine-interface research, but ultimately depicts "interactivity" inside the computer as a scam and sad hoax on the users. By forcing the user to hack the computer in order to regain control – by killing the browser, shutting down the machine or perhaps even throw it out of the window – it however creates a genuine interactivity outside the box and outside preempted behavioral patterns in the software.

This play involves simulations, too, but unlike the "Legible City" it is not simulation of anthropomorphic space, but simulation of machine functions.

*web stalker*. Analogous to jodi, net.artist Olia Lialina stated that many of her early works were based on bugs in the Netscape browser and therefore no longer work on contemporary computer setups. These plays with the web browser were not only a critical engagement with the Web and its aesthetics, but also an engagement with the software that shaped its access modes and interfaces. It was therefore a logical step from subverting standard browsers to developing alternative browsers. Most famous is the I/O/D, web.stalker [slide `http://www.backspace.org/iod/iod4.html`]. It turns web browsing upside down by not showing the smooth typographic rendering, but the otherwise concealed technical layers of the web, including HTML source code and http protocol communication, in separate windows and controls. It takes apart the separate components of web browsing – "takes apart" in the literal meaning of analysis. It thus achieves two things at once: It frees the cultural technique and the cultural imagination of web browsing from its conventional interface metaphors, including that of "browsing" itself. Secondly, it maps the World Wide Web as a controlled space, controlled by codes. This duality of freeing the user's imagination and revealing control structures paradigmatically expresses itself in I/O/D's slogan, "software is mind control, get some".

**Software art.** Within net.art itself, there was an increasing shift towards work with software, and as a result, software manipulated or written by artists. Critical observers described these works as "Artware" (Saul Albert in 1999), "experimental software" (Tilman Baumgärtel), "speculative software" (Matthew Fuller), "artistic software" (Andreas Broeckmann) and "software art" (Alexander Galloway, 1999). It was reflection on the fact that digital artists had first taken software as a transparent tool, and later began to reflect which influence that tool had on their own work and aesthetics. The more intensely artists worked with the computer, the more problematic the alleged tool became – not because of some "objective" limitation, but because of the culture, philosophy and subjectivity imposed by the creators of onto the users of the software.

*Signwave Auto-Illustrator*.

*Codework*. Subjectivity expressed in code is also characteristic of the whole genre of artistic codeworks whose chief medium are E-Mail messages written hybrids of English and code fragments from programming languages, character encodings, markup languages, emoticons and network protocols. Jodi were pioneers of this genre of digital art, along with Ted Warnell, Alan Sondheim, Netochka Nezvanova and the Australian female net artist mez (Mary Anne Breeze). One of Alan Sondheim's codeworks reads as follows:

```
From: Alan Sondheim <sondheim@panix.com>
To: _arc.hive_@lm.va.com.au
Date: Thu, 9 Jan 2003 17:17:20 -0500 (EST)

sleeping and running zombies through bodies

CPU states:  4.7% user, 5.8% system, 0.0% nice, 89.4% idle:36 processes:
35 sleeping, 1 running, 0 zombie, 0 stopped:1m 4:20pm up 8 min, 1 user,
load average: 0.54, 0.26, 0.11: :Mem:  38664K av, 35084K used, 3580K
free,

[\ldots]
```

The work is based on the output of the Unix system command "top" which displays a list of running processes, memory and central processor load. "Zombie" is a technical Unix term for a program process that can no longer be terminated with the "kill" command. Sondheim's text takes these descriptors—or "semantics," as computer science would call it—literally. He reads the output of the program as a physical inscription of bodies, as performance art and a subjective utterance in the medium of computer software. Yet it is not simply a poetic metaphorization because the technical apparatus of writing becomes a part of the text. There is a feedback of textual input, output and processing inside the text and within the medium of code. Subject and object, syntax and semantics, formalism and culture become inseparably entangled, crisscrossing and writing over each other. As such, the "codeworks" by kodi, mez, Alan Sondheim and other artists manifest a most radical understanding of formalisms as meaningful. They appropriate languages that were designed to be asemantic—programming languages, protocol code, shell commands—to unveil and elaborate their metaphorical and physical inscriptions, implications, and engendered meaning lurking between the lines. At this point, that is equally present in the works of I/O/D, for example, computational art has turned into a flat-out antithesis and refutation of Abraham Moles' claim that cybernetic art would be "fundamentally anti-semantic".

This also means, by implication, that there is no difference between "code" (or artificial language) on the one hand and "interface" on the other, because the code already is an interface, and the interface is a code.

*Bifo.*

*jaromil, forkbomb.* This energy is also embbedded into the twelve characters of jaromil's forkbomb [slide]:

```
:({ :|:& };:
```

Most computer operating systems can be crashed or at least brought to a grinding halt when users, even those without superuser privileges, launch an abundant ever-growing amount of programs that eat up all memory and CPU time. The easiest way to achieve this is a "forkbomb", a little program which does nothing but launch two or more copies of itself upon startup. Since these copies do the same in turn, this sets off a chain reaction with an exponentially growing number of processes. Forkbombs have been popular entertainment among hackers since about the mid-1990s, but jaromil manages to condense them to a most terse, poetic syntax, arguably the most elegant forkbomb ever written.

Unwillingly, this example also reveals a problematic issue of the term "software art": That it is often misunderstood as high programm craftsmanship. In fact, this understanding has its roots in computer science itself. Donald Knuth's textbooks "The Art of Computer Programming" or Paul Graham's recent book "Hackers and Painters" are founded on a post-classicist notion of art as beauty and high craftsmanship, for example in the elegance of an algorithm.

*Negativeland, Squant.* A counter-example to this – software art that expends programming skills – is a rather unknown work of the American experimental music group Negativeland, the "Squant" browser plugin `http://www.negativland.com/squant/plugin.html`. Negativeland claim that

> Squant is a color that cannot be seen on traditional RGB monitors. This plug-in changes the spectral display capabilities of your system software. THE NEWHEW SQUANTVIEW PLUG-IN utilizes a new color model ("RGBS") to facilitate the visualization of the Squant

> color spectrum, in addition to the already-established RGB color model.

Negativeland's website offers downloadable software packages for Windows and Mac OS and a "Tech Support" forum. It is filled with actual help inquiries by people who tried to get the plugin running, failed at one step, were helped, and still failed. Of course, the plugin and the "Squant" color is a hoax and doesn't work at all. Yet it is a clever artistic reflection of software as culture that includes vaporware just as much as actually running code. The false promises, installation nightmares, support horrors and other frustrations with software, known to any PC user, become the material of the work and get turned into an social-artistic performance.

*ubermorgen.com/Alessandro Ludovico: Google will eat itself.* This tendency is even more pronounced in recent artistic work – work that has its origins in the realm of net.art and software art, but is developing into interventionist performance art both in the Internet and outside.

A very recent example is "Google will eat itself" `http://www.gwei.org` [slide] by ubermorgen.com and Alessandro Ludovico. ubermorgen.com consists of former etoy member Hans Bernhard and Liz Haas, Alessandro Ludovico is well known in Italy as the founder and editor of Neural magazine.

"Google will eat itself" is simple to explain: it is a website that runs ads via the Google "AdSense" program, i.e. embedded commercial text advertising provided by Google, but bought from other companies. Google pays website owners a small fee for every click on an ad link; "gwei.org" uses this money to buy Google shares. The idea is that Google will pay the site to get bought up by it. Ideally, gwei.org should make so much money from Google ad payments that it can buy up all Google shares. To accelerate this process, "Google will eat itself" imploys some hidden dirty programming hacks that trigger automatic clicks on the advertising so that any user who visits the site will click multiple Google ads at once.

It is not only one company eating up another, but also a piece of software eating up another software. Google is one of the first world companies that is a piece of online software, with search requests as its input, and a double output of search results and money to the shareholders. This collapsing of software program and corporation get turned against itself by gwei.org. It is the net.art of an Internet

that is no longer an open field of experimentation, but a corporate space. The dark-humorous actionism of the piece manifests yet another resolution of the conflict that had originally voiced by Moles and Debord, technical formalism versus agency.

*dot.walk, psychogeographic computing.* Computation and situationist urban drift ultimately converge in the "generative psychogeography" of the Dutch artistic project `http://www.socialfiction.org`. Its *.walk* is a "psychogeographic computer," operated by pedestrians who walk through street grids like electrons flow through the gates of computer chips. The *.walk* computer can execute simple program code like the following:

```
// Classic .walk
Repeat
{
1 st street left
2 nd street right
2 nd street left
}
```

Psychogeographic computing has a double effect: It demystifies computing and turns it into a radically simple and popular low-tech and low-cost operation. Secondly, it liberates the imagination of what a computer can be and which purposes it may serve. Socialfiction.org has expanded and systematized this idea into a broader concept of "speculative programming" in which computing becomes a figure of thought and reflection not only in theory, but also in artistic practice.

While the same could be said about Moles' manifesto from 1962, the implications are contrary. Where Moles models art, criticism and aesthetics *after* computing, superimposing the latter on the former, speculative programming does the opposite, modelling computation after the arts and and speculative imagination.