# WEB STALKER SEEK AARON

## Reflections on Digital Arts, Codes and Coders

**Erkki Huhtamo**

"[A]ny notion of software leads us to reconsider our historical notions of art."
*Jack Burnham, "Notes on art and information processing", in the catalogue to the Software exhibition (1970)*

### 1.

The mainstream of computing has evolved towards hiding the code. While producing and reading code was an everyday activity for early computer users—scientists, engineers, operators and even artists—the presence of the code has become more and more obscure, hidden behind the facade of the interface. Interfacing humans and computers in a "user-friendly" fashion has been one of the guidelines of the digital culture since the late 60s. For Nicholas Negroponte, writing in 1969, the real task was to teach the computer to understand humans, not vice versa: "A designer, when addressing a machine, must not be forced to resort to machine-oriented codes. And in spite of computational efficiency, a paradigm for fruitful conversations must be machines that can speak and respond to a natural language".[1] The effort of creating "seamless" and "intimate" human-computer user interfaces became almost a definition of progressive computer culture, expanding from specialists to general users. From Xerox Star to the Apple Macintosh desktop and on to its bastard son, the Microsoft Windows, generations of users were taught to ignore the inner workings of the "box". They dealt with software packages that opened their metaphor-filled offerings on the desktop by a mouseclick. The users saw icons of tools and trash cans, but no strings of zeros and ones—the program codes that actually power the whole system.

While the use of the computer has spread to all imaginable fields, direct access to the code and coding has remained a domain for specialists, from computer scientists and professional programmers to hackers and builders of game engines. Mac users and even most Windows users rarely see even a glimpse of code. Even the brief revival of coding as a normal occupation for the general computer user as a result of the introduction of HTML was soon obliterated by an avalanche of easy-to-use automated webpage authoring software. While this development can be justified by claiming that the computer is just an intermediary, a tool or a medium, and not a goal in itself, there is ample room for counter-arguments. The computer's constantly expanding role as a universal machine powering innumerable applications and systems worldwide makes the invisibility of its workings alarming. Corporate and governmental coders (not to forget code breakers), as well as pranksters, cybercriminals and terrorists are monitoring and interfering with "innocent" computer use through the Internet. Lacking knowledge of programming, the users can only respond by subscribing to more corporate services or installing commercially marketed software packages. Yet both the core of the problem and the nature of the counter-measures remain vague. It has also been argued that merely using a pre-existing commercial software package restricts the user's freedom of expression, forcing him/her unknowingly to adapt to a role envisaged by the corporate planners.[2] John Berger's famous slogan, "every image embodies a way of seeing", could perhaps be modified to "every software

embodies a way of using".[3] Gaining access to computer code, understanding its "message" and being able to use it for one's own means are political, as well as social and economic issues. They are deeply intertwined with the dynamics of power and knowledge in contemporary society.

Against this background it is extremely interesting to note the recent emphasis on coding within the field of the digital arts. This interest has manifested itself in various forms, particularly in the emergence of the discourse on "software art".[4] In recent times we have witnessed the appearance of artworks that modify the look and functioning of commercial software applications from web browsers to game worlds, introducing features that may be interpreted by users as formal interventions, disturbing pranks, ideological subversions or simply as technical bugs. The famous *Web Stalker* (I/O/D, 1997-) is a web browser that displays control codes and link structures instead of the usual graphical interface. *Life_Sharing* by 0100101110101101.ORG attacked the false openness of the web browser by giving anybody permission to access its own hard disk with all its private files and programs through the Internet.[5] Other works use elements of computer code openly as building blocks of their aesthetics, without disguising them as graphics, images and sounds. Common to all these forms is the urge to question the prevailing conventions of computer use. Tearing down the veil of "user-friendliness", seen as a deception, the artists are giving the users a glimpse behind the scenes, to the engine room (to use an anachronistic metaphor). They themselves feel at ease in this engine room, using it as a laboratory, hangout, toolkit and venue for art. What are the reasons behind this interest? Is it enough to label it as an "inevitable" avantgarde of the digital culture? How will it relate to the wider, perhaps even non-digital, cultural context? This essay provides some preliminary answers to such questions by delving into the relationship between art, codes and coding from a media-archaeological point of view.

## 2.

The earliest computer artists in the 1960s were all coders. There was no alternative. Each work, whether graphics, animation or music, was necessarily the result of an unique act of writing computer code. As if foreshadowing the recent claims for "software art", some pioneers, like Michael L. Noll and the members of the Japanese Computer Technique Group (CTG), openly stated that the true work of art was the generating program itself rather than the computer-produced output.[6] Most of the early work on computer graphics and music took a formalist path, exploring issues like the possibilities of generative grammars and the relationship between rule-based behaviour and randomness. This state of things is aptly mirrored by the diagram-dotted pages of Jasia Reichardt's early overview, *The Computer in Art* (1971).[7] Pioneers of computer graphics like Frieder Nake described their activity as formal "visual research", deliberately segregated from any social or political concerns. They found inspiration from cybernetics, Claude Shannon's information theory and the exact mathematical aesthetics of theorists like Max Bense.[8] Bense excluded the role of subjective perception from his aesthetic system and based it on mathematical equations that emphasized the universal rather than the particular, the rational rather than the irrational (identified with the subjective art impulse), the abstract rather than the representational. Although random operations were often used by the early artist-programmers, the computer was considered mainly a "tool": it was expected to execute a program written beforehand.

This situation can be explained both by the state of the technology and the institutional context. In the 1960s computers were mostly available at governmental and corporate institutions. They were used primarily for statistical calculations, which emphasized the

Harold Cohen: *Clarissa*

Harold Cohen: *Machine*

act of programming in the form of batch-processing. Although the early artists and composers began to "bend" the technology to other kinds of uses, they were forced to accommodate themselves to the roles that had been established in the non-artistic "mainframe" culture: working within a dedicated institution, the artists created programs and then waited until they had been executed by the computer. They were members of a small elite, living on the fringe of a larger technical and institutional elite, exploring a powerful means of expression in its infancy. However, the nature of computing was constantly changing as a result of innovations like new interfaces, the idea of time-sharing, and the creation of the first programming languages meant for creative purposes. BEFLIX, written by computer scientist Kenneth Knowlton at Bell Labs, was applied to groundbreaking computer graphics and animation by artists Stan Vanderbeek and Lillian Schwartz, working in collaboration with Knowlton. Equally interesting was the appearance of the program ART 1 created by university professors Katherine Nash and Richard H. Williams and meant as a tool for artists interested in using the computer without having the theoretical and technical skills to program. Criticized at the time for considering the artist "as a specialist with pre-defined professional needs", it was, however, an early pointer towards the parting of ways between the creators of "soft" digital art and those involved in writing algorithms.[9]

The proliferation of art using pre-existing software tools such as Photoshop or Maya has not meant the disappearance of the activity of creating original algorithms. Many of the most rigorous digital art projects have been created by artists who either write software for themselves or work in close collaboration with a programmer. Prominent examples of the first are Myron Krueger, Harold Cohen and David Rokeby, while figures like Jeffrey Shaw (in collaboration with Gideon May and Bernt Lintermann) and Rafael Lozano-Hemmer (with Will Bauer) represent the second type. In some cases, like that of Christa Sommerer and Laurent Mignonneau, separating programming from other aspects of creation is almost impossible.[10] Pioneers like Krueger, Cohen and Rokeby have spent years, even decades, writing program code to refine their increasingly sophisticated systems. The artworks these artists have exhibited over the years may have had own identities, but they can also be characterized as "materializations" of these systems, documenting their state of development. The code provides the core of Krueger's *Videoplace* as well as Rokeby's *Very Nervous System* and *Giver of Names.*[11] Arguably the most rigorous and long-term effort to use original programming as a means of creating an evolving art project has been Harold Cohen's *AARON,* continuously under development since the early 1970s.[12] *AARON* is an AI-based computer program, an expert system that creates paintings and drawings. Over the past thirty years, different output devices have been used, from a drawing "turtle" moving on paper placed on the floor to complex painting machines and

more recently to a software application automatically creating pictures on the desktop. *AARON* can be characterized as a semi-autonomous creature. Its works are based on the complex rules defined by Cohen, but it also has a considerable amount of autonomy from composition to coloring. It is significant that Cohen has not made *AARON's* code public. The different stages of the program have not even been systematically preserved, which is why the code's development can only be deciphered indirectly, as reflected in *AARON's* numerous drawings and paintings.[13] Although it is a major creative effort, Cohen never considered the code as the artwork proper. Rather, it can be likened to the skills and techniques accumulated by a human artist during his lifespan. What will remain are the paintings, the material traces of a lifework. In this sense Cohen, painter by education, resorts to a conventional model of the creation and preservation art. However, *AARON's* most recent manifestation, the software application available as freeware on the Internet, could point in a more radical direction.[14] Instead of remaining Cohen's own cybernetic extension, *AARON* has been given a degree of independence from its creator. However, while becoming a software application on anybody's desktop, *AARON's* development has also stopped. Although it can keep on producing different paintings endlessly, it does not have a facility to learn other routines. Moved to the desktop *AARON* certainly immortalizes Cohen's achievement, but it also turns into a cybernetic zombie. Releasing its source code would give it a chance to develop further by means of collective programming efforts through the Internet.[15]

## 3.

The advocates of "software art" emphasize the primacy of the code as the main creative achievement and demand an un-obstructed presence and role for it in the artwork. According to the statement of the first software art competition jury at the Transmediale 01, "software art is opposed to the notion of software as a tool".[16] For the jury, software art has

Harold Cohen: *Computer*

many faces: “it could be algorithms as an end to themselves, it could subvert perceived paradigms of computer software or create new ones, it could do something interesting or disrupting with your computer, it could be creative writing, it could be science.” In another text Florian Cramer and Ulrike Gabriel (who were among the jury members) write: “Software art means a shift of the artist’s view from displays to the creation of systems and processes themselves; this is not covered by the concept of ‘media’”.[17] For these writers the principal “sin” media art has committed seems to be its excessive attention to interface design. For the software purist, the creation of detailed immersive environments and elaborate multisensory interfaces is in itself an act of mystification. Works that involve the participants both bodily and emotionally seduce them, instead of making them aware of the true nature of the system, hidden “behind the facade”.

It is most interesting that interactive systems that in the not-so-distant past were seen as an empowering and critical alternative to “passivating” media experiences like watching television, have now become the target for criticism. This has to do with the rapid acceptance of interactivity and its adoption into the mainstream of digital culture. If interactivity was once seen as a gesture that questioned the prevailing logic of media, it has now been normalized, added to the internal cultural repertory. However, the recent criticism does not seem to be addressed to interactivity per se; rather, it addresses the way interactivity has been packaged and marketed, institutionalized and commodified. In fields like electronic gaming the near realtime interaction between the user and the game software/hardware complex has been claimed to lead to the “automation” of the action/reaction mechanism. Immersion into rapidly changing and emotionally engaging gameworlds leaves little time for reflecting on the algorithmic basis of the experience.[18] For the gamer it is as if the system leaves only the phenomenological experience of the gameworld. The appearance of a phenomenon like game patch art is interesting in that it uses programming to directly address such mechanisms of identification. As a recent example, Anne-Marie Schleiner’s *Velvet-Strike* project (2002) invites people to create digital graffiti on the walls of *Counter-Strike,* the popular network shooter game about terrorism.[19] Situated somewhere between “textual poaching” gaming subcultures and critical software art, game patch art functions as an ambiguous counter-discourse to commercial game culture, true to the legacy of hackerism.

It should also be pointed out that media art from recent years has shown signs of self-reflectivity and a growing critical awareness of its own position vis a vis commercial and institutional applications. Works like Maurice Benayoun’s and Jean-Baptiste Barrière’s *World Skin* (1998) or Ken Feingold’s *That Sinking Feeling* (2002) are far from naive cele-


Anne-Marie Schleiner: Velvet Strike
From *http://www.opensorcery.net/velvet-strike/*

Anne-Marie Schleiner: Velvet Strike
From *http://www.opensorcery.net/velvet-strike/*

brations of the pleasures of the interface. While still offering memorable interactive experiences they at the same time disturb the bond with the user.[20] Both works create ambiguous situations where the seductive potential is constantly undermined by discrepant (*World Skin)* or deliberately "malfunctioning" *(That Sinking Feeling)* elements. Although neither work deals primarily with computer code, its role has by no means ignored. The functioning of the algorithmic base of these works may not have been highlighted, but its role is interconnected with the issues these works raise, from the role of media and the politics of simulation to social-psychology and the construction of identity. In their own ways they demonstrate that singling out the code for exclusive scrutiny at the expense of everything else may not be the right way to go in an integrated digital culture, where technical, ideological and cultural codes are no longer separable from each other.

## 4.

It is interesting to note that the software art purists have already begun to trace their own genealogy. This proves the well-known fact that history is mainly written to justify the present. A major formative event in the pre-history of software art has been located in the Software exhibition, curated by Jack Burnham for the Jewish Museum in New York in 1970. This technically catastrophic and until now largely ignored event has been identified as the nexus at which the efforts to explore the creative potential of information technology met conceptual art and Burnham's own interest in structuralist analysis.[21] In his introduction to the exhibition catalogue Burnham made it clear that Software was not a normal art exhibition. Rather, it displayed exhibits that dealt with "conceptual and process relationships". One of the purposes was "to undermine normal perceptual expectations and habits which viewers bring to an art exhibition".[22] The visitors were supposed to interact with various technological devices, without being asked to consider them as artworks. The most noted exhibit was *SEEK,* created by Nicholas Negroponte and his colleagues at MIT's Architectural Machine Group. It was another AI-inspired programming effort that hardly reached its goals, except on a metaphorical level: living gerbils had been placed in a glass-caged arena with aluminium building blocks, and a computer-controlled robot arm operating from above. The system, engaged in arranging the blocks according to pre-programmed schemes, was supposed to respond "intelligently" to the "noise" created by the gerbils, the sounds of their paws on the blocks, etc.[23]
For the software art advocates, the most inspiring exhibit seems to be *Labyrinth,* an early version of Ted Nelson's hypertext displayed as a bare branching structure. Several other exhibits interfaced visitors with technological apparatus making few efforts to weave fictions or elaborate "stage-sets" around them. In *Software,* Burnham drew a connection between the use of computing technology and conceptual art, which made him include non-technological pieces from artists like John Baldessari, Lawrence Weiner and Douglas Huebler. In Burnham's view all these forms shared the tendency to move away from art objects and to investigate linguistic structures and forms of information exchange underlying other forms of expression. This was also in unison with Burnham's simultaneous effort to reveal the mythical structures underlying the traditions of Western art.[25] Conceptual art, interpreted broadly to include also John Cage's method of composition, the series of instructions for imaginary events composed by the Fluxus artists and some forms of lettrist poetry, certainly provide one possible background against which to assess the role of code in software art. However, one might also refer to the field of structural and materialist film and the ideology of "anti-illusionism" in the film culture of the late 1960s and early 70s. Attacking the illusionism of conventional narrative cinema, filmmakers began to deconstruct the cinematic apparatus in its constituent elements. They emphasized the materiality of film, includ-

SEEK by Nicholas Negroponte and the Architecture Machine Group at MIT, 1969–70. Shown at "Software", Jewish Museum, New York, 1970.

ing sprocket holes, frames, emulsions, scratches and dirt. Some filmmakers, including Hollis Frampton, Michael Snow and the Croatian Ladislav Galeta, used (quasi-)generative principles to structure their films. In its most extreme form, filmmakers abandoned film altogether, staging events that merely highlighted the basic elements of the cinematic apparatus, the light beam from the projector, the screen, the darkness of the auditorium. Writing in the seminal *Structural Film Anthology* (1976), Peter Gidal defined Structural / Materialist films as "at once object and procedure".[25] In another text he stated: "A film is materialist if it does not cover its apparatus of illusionism. Thus it is not a matter of anti-illusionism pure and simple, uncovered truth, but rather, a constant procedural work against the attempts at producing an illusionist continuum's hegemony."[26] In other words, materialist film was not as much a purification ritual as a constant struggle against the hegemonic forces resorting to illusionism.

## 5.

The struggles of the structural and materialist film movement could perhaps be compared with the efforts the software artists are currently making to scratch the slick corporate facades of cyberculture, metaphorically manifested in the deceptive openness and the pretended democracy of the graphical user interface.[27] Yet comparisons across time are risky. Drawing a parallel between a 1960s computer generated picture consisting of dense arrays of ASCII characters and a piece of "ASCII art" from the late 1990s is only valid in a limited sense. The student hackers that created *Spacewar,* considered the first computer game, have little in common with today's commercial game developers and even with game patch artists. Similarly, the programming efforts of the computer graphics pioneers of the 1960s cannot be directly compared with the software art of the early 21st century, because of the widely different cultural contexts. The discourse on software art has emerged in a situation in which digital culture has already had time to create a history and a memory. During its first half century, digital computing has gone through a number of different stages that have progressively re-defined the meanings of computing in warfare, administration, society, economy and culture. Issues like artificial intelligence, virtual reality, agents and avatars, A-Life, GUI, physical computing and digital networking are all elements in an evolving fabric that changes shape depending on time and place and the position and identity of the observer. Last but not least, software art is in a position to profit from the experiences of the net art pioneers.

Digital art has for some time shown signs of a growing self-consciousness in relation to the history of digital technology. Yet the recent interest in AI among artists does not mean an artistic revival of the classic artificial intelligence research. It is not a simple homage either. Projects like David Rokeby's *A Giver of Names,* Kenneth Rinaldo's *Autopoiesis* and Ken Feingold's talking and responding puppet heads are examples of meta-art that engages in a dialogue with the cultural representations of AI (including, in Feingold's case, Joseph Weizenbaum's quasi-intelligent conversation program ELIZA), while pursu-

ing at the same time other intellectual and ideological goals. Perhaps it is not wrong to say that there has been a cultural demand for a phenomenon like software art, just like there was for the appearance of structural / materialist film, which was "called for" by a conglomeration of conflicting cultural forces, from the increasing uniformity and untouchability of commercial film production to the impact of counter-cultural movements, the emergence of decontructionist philosophy and the "dematerialization of the art object". The structural / materialist film movement emerged as a critical position that questioned the prevailing audiovisual hegemony, demanding an approach that highlighted the "primitives" of the filmic medium in dynamic interplay with its application to narrative and metamorphic purposes. In its anti-illusionistic fervor it evoked the idea of a modernist reaction, but not in a simple revivalist fashion.

The claims and prognoses made by the software art advocates also have a certain neo-modernist flavor. Emphasizing the centrality of the code and the algorithmic approach means positing a "hard core", often felt to have been lost in the postmodern world. Indeed, there are "small but uninfluential" (to brutally appropriate an expression from Vuc Cosic) groups, such as the one dedicated to exploring the aesthetics of the generative code (Geoff Cox, Alec McLean, Adrian Ward, etc.), that fulfill many of the criteria for classical avantgarde movements.[28] Florian Cramer has identified the group's activities, which include poetry readings in Perl program code, as "software formalism".[29] On the other hand there are approaches that emphasize the cultural and ideological underpinnings of computer programming. For groups like the British Mongrel and I/O/D (creator of *Web Stalker),* digital code cannot be separated from the operations of ideology manifested on the Internet and elsewhere. Their actions and projects are much more difficult to fit into a modernist straitjacket. The situation becomes even more difficult in the case of the independent artist-activists operating on the no-(wo)man's land between popular cultural forms like gaming, various forms of net activism (including cyber-feminism) and theoretical approaches. Appropriation, pastiche, bricolage and other postmodernist tricks are still among their favourite tools.

If the digital arts are going to make a difference in the media culture of the 21st century, it is clear that they have to shed their veil of innocence. They have to face the problematic, conflicting realities of cyberculture. While doing so, they need to scrutinize and make public their own internal workings, as well as their relationship to the systems of power, control and commerce that envelop them, infiltrate them, co-opt them and influence their public image, whether welcome or not. Just as science and technology can never be free from the constraints imposed upon them by economy, culture and politics, the digital arts cannot be "pure" and "free", even when purporting to focus on the quest for formal, mathematical, algorithmic beauty. Researching the aesthetics of digital grammars and exploring the workings of the code are important goals; yet getting the findings out from the isolated engine room and into the consciousness of the cyber citizen—also in the role of the cyberart lover—is quite another matter.

---

1 Nicholas Negroponte: *The Architecture Machine.* Cambridge, Mass.: The MIT Press, 1970, 9.

2 See Matthew Fuller's penetrating analysis of Microsoft Word, "It looks like you're writing a letter: Microsoft Word", available on-line at *www.axia.demon.co.uk/wordtext.html.*

3 John Berger et. al.: *Ways of Seeing.* London and Harmondsworth, Middlesex: BBC and Penguin Books, 1972, 10.

4 "Nodes" for the discourse on "software art" have been the software art award competition organized by the Transmediale in Berlin since 2001, the website *www.runme.org* and the *www.readme.org* events. The protagonists active through these and other channels come from different countries, but mainly from Europe.

5 See *www.walkerart.org/gallery9/lifesharing/.* During the Venice Biennale 2001, invited to participate in the Slovenian Pavillion, 0100101110101101.ORG got attention by announcing the creation and distribution of a festival (computer) virus. With this conceptual act the group in a way confirmed the notion of the Transmediale 01 software art jury according to which "[c]omputer viruses might be seen as a critical form of software art ..." (transmediale 01, jury statement, *www.transmediale.de/01/en/s_juryStatement.htm).*
6 Cynthia Goodman: *Digital Visions. Computers and Art.* New York and Syracuse: Harry N. Abrams & Everson Museum of Art, 1987, 24.
7 Jasia Reichardt: *The Computer in Art,* London: Studio Vista, 1971. See page 81 for a discussion of the Computer Technique Group and its belief that [i]t is the program itself that is the work of art".
8 See Herbert W. Franke: *Computer Graphics Computer Art,* pp 107-108. Phaidon, New York, 1971 Another influential theorist was Abraham Moles.
9 Jonathan Benthall: "Science and Technology" in *Art Today,* p 52. Praeger Publishers, New York, 1972.
10 It is well known that Mignonneau is responsible for the programming in a technical sense, but their numerous installations are the product of a very close collaboration, which makes separating their creative inputs next to impossible.
11 Rokeby's *Very Nervous System* is also available as a commercial software package known as SVNS. It has been used by numerous other artists as well.
12 See Pamela McCorduck: *AARON's Code,* W.H. Freeman, New York,1990.
13 In a recent e-mail message to the author Cohen explained: "And, as with any other serious artist, what isn't carried forward is abandoned. So the answer to one of your questions is that I don't keep old versions of the program hanging around very long. Actually I wouldn't be able to use them even if I did; they were written for different machines and stored on media and devices that are no longer in use. It would probably be easier to reconstruct the early versions from memory than to try to resurrect the old media: not that I would bother to try." (private e-mail communication, May 6, 2003.)
14 Available from *www.kurzweilcyberart.com/.* The site also contain material about *AARON's* history, including a film clip demonstrating a painting machine in operation. It might also be claimed that becoming as desktop application, essentially a screensaver, has trivialized *AARON.* This impression may have to do with the speed of current computers. *AARON* somehow seems to create its paintings too quickly, too effortlessly. This is naturally just a subjective impression that has nothing to do with the sophistication of the code.
15 In discussions Cohen often refers to the complexity of the program, which has been in the making for three decades. Although it has been turned into freeware, Cohen obviously still sees *AARON* as his own creation as an artist. Releasing the source code would compromise this role, but it would also lead to much more radical collective creation.
16 *www.transmediale.de/01/en/s_juryStatement.htm*
17 Florian Cramer and Ulrike Gabriel: "Software Art", available on-line at *www.netzliteratur.net/cramer/software_art_-_transmediale.html.*
18 Eddo Stern has pointed out how artefacts, programming errors and network malfunctions in massively multiplayer roleplaying games can occasionally make the player aware of the digital architecture of the game. See Stern's article in *Mariosophy: The Culture of Electronic Games,* edited by Erkki Huhtamo and Sonja Kangas, The University Press of Finland, Helsinki, 2002 (in Finnish).
19 See Anne-Marie Schleiner: "Velvet-Strike: War Times and Reality Games", *www.noemalab.com/sections/ideas/ideas_articles/schleiner_velvet_strike.html.* Schleiner has also curated important game patch related art events.
20 In the case of *World Skin* this happens by using deliberately artificial frozen 2D images within a navigable 3D space, in the case of *That Sinking Feeling* by having an animatronic puppet head deliberately misunderstand the visitor's speech, showing traits of schizophrenic behaviour that could also be attributed to malfunctioning technology.
21 See Edward A. Shanken: *The House That Jack Built: Jack Burnham's Concept of 'Software' as a Metaphor for Art,* available online at *www.duke.edu/~giftwrap/House.html.*
22 Jack Burnham, "Notes on art and information processing", in *Software. Information technology: its new meaning for art* (catalogue) p 12, The Jewish Museum, New York,1970.
23 SEEK gained notoriety because several of the gerbils reportedly died during the exhibition. The gerbils, of course, were stand-ins for human beings operating in a technologically saturated environment.
24 See Jack Burnham: *The Structure of Art.* George Brasiller, New York,1971.
25 *Structural Film Anthology,* edited by Peter Gidal, London: BFI, 1976, 14.
26 Peter Gidal: *Materialist Film,* London: Routledge, 1989, 17.
27 A useful book for developing this connection further is Malcolm Le Grice: *Experimental Cinema in the Digital Age,* London: British Film Institute, 2001. Le Grice was an active protagonist in the structural film movement.
28 See Geoff Cox, Alex McLean and Adrian Ward: *The Aesthetics of Generative Code,* available on-line at *http://generative.net/papers/aesthetics/.*
29 Florian Cramer: "Concepts, Notations, Software, Art", available on-line at *www.netzliteratur.net/cramer/concepts_notations_software_art.html.*