A RE-DECLARATION OF DEPENDENCE—

SOFTWARE ART IN A CULTURAL

CONTEXT IT CAN'T

GET OUT

OF

JACOB LILLEMOSE

When the term *software art* surfaced some years ago it quickly gained an enormous popularity in the digital art community (as the next thing after net art) and opened a new field of aesthetic discussions. From the very beginning this field has been somewhat divided: On the one side, those artists and critics who emphasize the literary and mathematical sources of software art and argue in favor of an aesthetics centered around the formal and expressive qualities of programming and generative code; and on the other side, those who inspired by disciplines such as cognitive science, philosophy, social, and political theory turn towards an aesthetics that focuses on software art's conceptual, and discursive involvement with the culture of software. The jury statement of the *Read_me* festival in Moscow, which unofficially stands as the original definition of software art, anticipates this division: "We consider software art to be art whose material is algorithmic instruction code and/or which addresses cultural concepts of software." Although the "/" in the middle symbolically seeks to bridge any categorical differences between these two views on software art, it most often acts as a dividing line. Of the two views, the former seems to predominates current theoretical discussions, criticism, and curatorial practices. Certainly, this view prompts reflections, which are important for the understanding and analysis of the technological specificities of software as an aesthetic potential. With some reason, it can even be argued that this view marks a more genuine and clear-cut form of software art. Nevertheless,

the other view accentuates equally important and hitherto rarely treated aspects of software art as a contemporary art form, aspects that grasp software art not in itself but as part of wider contexts.

Conceptual art/contextual art

Software art is often treated as a digitally updated version of the conceptual art that emerged in the mid 60s. Generally, the connection is founded in the apprehension that software art continue "the de-materialization of art"[1] that conceptual art began. Through this dematerialization conceptual art initiated an aesthetics of text, language, reading/writing and interpretation; an aesthetics based the perceptual and cognitive challenges of processes, systems, and structures instead of on the specific media, on the visually convincing art-object, that modernism championed;[2] an aesthetics involved with objective, non-individual, forms of recording, documenting, mapping, and organizing and not with the subjectivity of the artist in a psychological and mythological sense; and an aesthetic that anticipated the more or less direct participation of the viewer as an integrated part of the work of art. In this way, the advent of conceptual art marked a significant aesthetic turn, which coined a broad spectrum of different artistic trends ranging from minimalistic sketch-like drawings and elaborate installations of text files over experiments with video and photography to physical performances and political activities in the public space. Connecting software art to conceptual art is thus not an unequivocal task, and it seems that the division mentioned above to some extent arises from how conceptual art is interpreted and which of these trends are understood as the forerunners of software art.

The aesthetics of code and programming is often related to two different but parallel trends of conceptual art. The leading figures of the first trend, refered to by Alexander Alberro as "linguistic conceptualism",[3] are mainly by Sol Le Witt, the Art & Language group, and Joseph Kosuth. These artists and writers were engaged in ontological, genealogical, and epistemological reflections on the nature of art as a concept, a self-defined (tautological) and self-reflexive

logistic system composed by writing and ideas, and a language in which form and content tended to merge. The second trend, represented for instance by John Cage and La Monte Young's conceptual takes on compositional music, comprehends the work of art as a set of instructions and art in general as a purely mental, non-physical, phenomenon. These two trends of conceptual art are unquestionably relevant for the understanding as well as the development of the complex formal dimensions software art. However, when it comes to the so-called cultural dimension of software art two other trends of conceptual art seem more relevant: The one, represented by Hans Haacke, Dan Graham, Victor Burgin, and Gordon Matta-Clark, which was political in an interventionistic and analytical sense, and the one, represented by Vito Acconci, Bruce Nauman, and Chris Burden, which was involved with performativity and stagings.[4] These artists saw conceptual art a contextual activity that dismantled and completely rejected the notion of the transcendental and autonomous work of art that modernism praised. For them conceptual art was as an experimental and critical praxis fundamentally connected to the communication processes, information economies, representational systems, ideological power structures, and codes of the surrounding social and cultural reality, including the art institution.[5]

In a text from 1975 carrying the programmatic title "A Declaration of Dependence" the artist Sarah Charlesworth describes this change in the status of the work of art. The backdrop for her declaration was the disillusioned, retrospective realization that conceptual art could not exist outside institutions in its own aesthetic sphere of dematerialization: "When we discuss a work of art or an art tradition, we are discussing a phenomenon in which exists in an integral relationship with the entire complex of human and social and historical forces defining the development of that work or tradition. This same complex of social and historical forces in turn inevitably defines the context in which the work or tradition claims significance, and ultimately functions as a force or an agent in the ongoing evolution of that culture. Thus we are at once the products and

the producers of the culture in which we participate."[6] Charlesworth's description of the contextual nature of conceptual art points towards an aesthetics based on the relationship between the internal structure of the work of art and external non-artistic structures. The dependence she declares represents both a condition and a potential. Art is both "framing and being framed," to quote the title of Hans Haacke's 1975 monograph.

Three generations

This contextual definition of the work of art experienced a revival in the 90s when a new generation of artists explicitly reinterpreted the heritage of conceptual art. In 1993 Peter Weibel curated the show "Kontext Kunst. Kunst der 90 Jahre", which presented a group of upcoming artists of the new decade and a few of their influential predecessors.[7] A monumental catalogue, which included Weibel's curatorial text entitled "Kontextkunst. Zur sozialen Konstruktion von Kunst" accompanied the exhibition. The text was an updated version of his 1971 text "Kontext-Theorie der Kunst", which criticized the "syntactical" poetics of modernism to emphasize the social aspects of language (use), i.e. communicative relations and the structures behind the linguistics concluding that "contexts became more important than texts." In his 1993 version of contextual aesthetics Weibel expanded the theoretical horizon of this contextual aesthetics from the perspective of contemporary art. He was no longer concerned with the structural textualization of art as such but with the relations between the symbolic language of art and social spaces and institutions. He emphasized art's potential to criticize the false consciousness of the constituted powers and create social fields of art characterized by critical consciousness and transparent structures. Instead of repeating the avantgardistic conception of an absolute other reality, of a transcendence of the context, the contextual aesthetics of the 90s stressed different reworkings of specific and actual realities, transformations of the context. The relation between the work of art

and the context characterized a complex dynamic of constant negotiations and exchanges, an "open field of signs and actions," as Weibel later described it.[8] It is thus important to notice that Weibel's contextual aesthetics did not dissolve the work of art in a contextual determination; rather it emphasized the works of art's ability to act actively in relation to its context.

Weibel added a historical dimension to his contextual aesthetics by pointing out three generations of contextual artists. The first generation was the conceptual art of the 60s and 70s, which criticized the art institution, a.k.a. the white cube, as an oppressive and restrictive space that only accepted as certain type of art and a certain type of aesthetics; the second generation emerging in the late 70s was involved with a critique of the representations of the social field within the art institution. The thrid generation, which seems to be the most interesting generation for this discussion of the contextual aesthetics of software art, took the stage in the early 90s. This generation represented a direct involvement in the social field, replaced symbolic actions with real actions, and become "partisans of the real." Instead of acting politically within art institution, it acted aesthetically within the social field. In relation to this generation, and echoing Charlesworth's idea of artists being both produced by and producers of the culture, Weibel formulated a contextual aesthetics based on the idea that the recognition of art as a social construction represents a potential for construction of the social: "The goal of the social construction of art is to take part in the social construction of reality."

A fourth generation: software art

The different trends of conceptual art outlined above plus Charlesworth's and Weibel's theories respectively form a historical and theoretical perspective from where the understanding of the conceptual dimension of cultural trend of software art can be developed. It points back to Jack Burnhams 1970 exhibition "Software, Information Technology: Its New Meaning for Art", which is a hot topic in current discussions about the relationship

between conceptual art and software art. Burnham saw "[t]he notion that art can be separated from its everyday environment [as] a cultural fixation"[9] and used "software as a metaphor for [conceptual] art"[10] which aesthetically computed the politics of the technological developments that influenced post-modern culture at large. And from Burnham, it points beyond a romantic/modern aesthetics based on formalistic taste (contemplative appreciation for the beautiful as well as for the ugly) and absolute artistic autonomy (of the artists as well as the work of art), which is prevailing in some trends of software art/criticism, to an expanded aesthetics based on conceptual criticism and a interdependence between software art and its cultural, social, economical, political, and technological contexts.

More specifically, this perspective allows us to pursue Weibel's historical thesis and suggest that the so-called cultural trend within software art represented a diversity of artists like The Yes Men, Institute for Applied Autonomy, TWCDC, LAN, I/O/D, 01.org, Knowbotic Research, EDT, übermorgen, Carbon Defense League, etoy, and RSG constitute a fourth generation of contextual artists, to name just a few. This latest addition is closely related to the third generation but the artists, digital by birth, are partisans of the culture of software, not of the real. They conceive software art as a conceptual criticism of the culture of software as a highly politicized field, as a "society of control" like the one described by Michael Hardt and Antonio Negri: A society where power is "exercised through machines that directly organize the brains (in communication systems, information networks etc.) (...) toward a state of autonomous alienation from the sense of life and the desire for creativity."[11] However, they do not limit this criticism to claims of aesthetic independence and exclusivity but applies it to the culture of software. Instead of taking up the dialectic position of the negative outsider they conceive its criticism to be positively dialogical and to be working from the inside. It is a generation that believes in software art as a wide-ranging activity directly involved in the shaping and construction of the culture of software. It sees

this culture as a context, which is not pre-formatted but "up for grabs, for remaking,"[12] for de- as well as re-coding. In the name of free information flows, open source, creative commons and tactical media use it wants to get "behind the blip"[13] of the culture of software, to its ideological and conceptual politics, and revitalize the desire for digital creativity and the sense of digital life.

Tools for re-contextualization by way of conceptuality

Like all other constructors these artists need tools, but instead of using the standard tools available at the software supermarket they produce their own alternative tools produced in alternative ways and, not least, for alternative uses.[14] These alternative tools are not tools to be used for specific purposes in the construction of concrete objects and products. Rather, their qualities lie in the ability to generate a diversity of open-ended and abstract, yet real processes of production, on every scale and in all directions, within the culture of software. They are "means of mutation"[15], not stabilizers. Their powers are constituent and not a constitutional, to borrow a distinction from Hardt and Negri, in the sense that they bring about change, difference, and liberation, sometimes even of a revolutionary nature. Instead of building the same over and over again, reproducing the familiar or the identical, they are productive, creative, and innovative in a radical and fundamental sense. Working in real-time, their creative modes of operation are set on "becoming" and "actualization" in the Deleuzian sense, which means the bringing into existence of that which can neither be pre-scribed or terminated.[16]

As such, it is not only the conceptual production of the tools themselves, or the tools *in* themselves, as much as the potential use(r)s that the tools hold in relation to the different contexts of the culture of software, which interests the fourth generation of contextual artists. They are, as Thomas Dreher puts it, "programmers of programming possibilities"[17] who conceive their tools for a multitude of users to make them work and work with them on their own through more or less direct interpretive interaction (which is not

to be confused with simple choose-and-click interactivity). Their aesthetics of conceptual production thus implies aesthetics of contextual use, which is in continuation of Fuller's notion that software "participate in "conceptuality.""[18] This notion refers to software's construction of concepts or more precisely to software's conceptualization of ways of thinking, knowing, seeing, and doing. Through its design, use of metaphors, representational structures, and practical functionalities, basically HCI, software not only determines possible uses but also constructs a user. To rephrase Fuller: Software engineers 'humans' through HCI.[19] As his exhaustive analysis of *Microsoft Word*, the archtypical example of conventional software, shows, this conceptuality is very often regulative, manipulative, and one-dimensional in the sense that it reduces the use(r) to a question of usability and simple effectiveness. But as his insightful text on *The Web Stalker* shows with equally conviction, this participation in conceptuality also represents a significant aesthetic potential. As the classical I/O/D slogan says, if "[s]oftware is mind control" then "[c]ome and get some." And it is this aesthetic potential that the fourth generation of contextual artists works with and develops further. By reappropriating and reconceptualizing the politics of HCI on the level of programming, design, and functionalities it turns software art into a contextual activity, mental as well as practical, involved with the forms of social reproduction, communicative relations, power structures, and technological developments in the culture of software. It rejects an isolationistic use of software as well as the rationales of universal homogeneity expressed by the three-clicks-and-you-are-where-you-want-to-be logic. Instead, it embraces experiments, subversion, tactics, criticism, dialogue, fantasy, science fictions, affections, antagonisms, paradoxes, irrationalities, complexities, and heterogeneity as its principles of engineering. Thus by constructing alternative tools it engineers alternative 'humans' and consequently, this is the utopia, creates an alternative culture of software.

Towards an aesthetic of contextual software not-just-art

The suggestion that software art represents a fourth generation of contextual aesthetics is an attempt to escape the idea that software art represents yet another avantgardistic break in the history of art. Software art, like multi-media art and net.art before it, is part of the historical continuum of post-modern art. Of course, software art is also part of a technological development that introduces (what seems like) radically new forms of art. But as the post-modern, or post-media, aesthetics of conceptual art point out: Art is primarily conceptual and only secondarily formalistic, and because it is conceptual it is also contextual. Thus it seems important to turn attention to which new conceptualizations of art in general, software art introduces through these specificities. Even though software art, like every art form, has its particular qualities these qualities needs to be qualified through the establishment and discussion of connections, of differences as well as of similarities, to other art forms, for instance to contextual art. These connections emphasize the relational, inclusive, and synthetic qualities of software art, which seem to some of the most relevant for a discussion of software art within the context of contemporary art.

An important aspect of these connections is that they make it possible to bridge a contradiction between software as art and software as a tool, which is based on the traditional separation of art and technology, the purposeless and the functional, the sensuous and the rational, found in both classical and modern philosophy. Some critics talk about somehting like "a pure technicity without purpose,"[20] implicitly saying that what defines software, as art, is exactly that it is not a tool. According to the contextual aesthetics of software art suggested here, the definition is not that catagorical. It leaves any notion of formalistic purity behind to define a new kind purposefulness that recognizes software art's ability to reconceptualize art through the tool/technology, and the tool/technology through art, as an essential potential, which was exactly what Burnham tried to do. Fuller also anticipates such a reconceptualization by introducing the term not-just-art in rela-

tion to *The Web Stalker*. This canonical work of software art, he wrties, "can only come into concurrence by not just being itself. It has to be used."[21] However, the fact that *The Web Stalker* has to be used does not disqualify it as a work of art: "Alongside the categories of art, anti-art and non-art something else spills over: not-just-art."[22] The term seems very instructive for a further development of software aesthetics that avoids the avantgardstic idealism of either-or and is able to balance the vital dynamics between the aesthetic and the instrumental, the art institution, and the culture of software, and embrace their interdependence.

Of course, the contextual approach to software
art is just one of many ways to give this art form
of tomorrow a historical and theoretical perspective.
Other ways exist and more will be discovered in the future.
That is how it should be. Software art history and art
theory are not objective sciences but experimental
forms for production of aesthetic knowledge
and experience. Instead of trying to distil the
pure form of software art they contaminate
and pollute software art in multiple
ways leading to complex, rich, and
visionary understandings of
software art, as well as
the culture of
software

.

1 "The Dematerialization of Art" is the title of Lucy Lippard and John Chandler's classical 1967 essay. In her thesis on net art Josephine Berry explicates this connection by talking about "the re-dematerialization of art".

2 Symptomatically for the anti-visuality inherent in the conceptual aesthetics of software art Florian Cramer contrasts the poetics of code with what he calls "Neo-Pythagorean digital kitsch."

3 Alberro, Alexander: "Reconsidering Conceptual Art, 1966–1977", *Conceptual Art: A Critical Anthology*, MIT Press, London 1999, p. xvii.

4 Many other artists can be mentioned as representatives of these trends of conceptual art. It should also be noted that the various trends of conceptual art are overlapping to a great extent, and the distinction made between them here is more a matter of different perspectives pointing towards software art than opposing aesthetics.

5 A canonical text on the critique of the art institution is of course Brian O'Doherty's *Inside the White Cube. The Ideology of the Gallery Space* (1981). Readings of the anti-institutionalism of net.art could no doubt profit from this text's exemplary analysis of the politics of the exhibition space.

6 Charlesworth, Sarah: "A Declaration of Dependence", *The Fox*, 1:1 (1975), p. 1–7.

7 Weibel is certainly not the only one to theorize the art of the 90s or contextual art in general, but his exhibition and text nevertheless stand as important landmarks. His term *Kontext Kunst* is somewhat ambivalent though, because, as Jan Avgikos points out, "the context was always already there": Whether pre-modern, modern or post-modern, art exists within a context. In this sense *Kontext Kunst* is a tautology, which does

not specify that what makes contextual art contextual is the thematization of the relation between the work of art and its context. Thus Thomas Dreher uses the more specific term *Kontextreflexive Kunst*, while Anne Rorimer talks about "Context as Content."

[8] See his text "Kunst als offenes Handlungsfeld" for the Austrian Pavilion at the 1999 Venice Biennale, at http://www.geckomultimedia.net/venedig1999/biennale/Ausstellungstext/body_text.html.

[9] Shanken, Edward: "Art in the Information Age: Technology and Conceptual Art", at http://www.duke.edu/~giftwrap/InfoAge.html.

[10] Burnham, Jack: "Notes on Art and Information Processing", *Software Information Technology: Its New Meaning for Art*, Jewish Museum 1970.

[11] Hardt, Michael and Antonio Negri: *Empire*, London, Harvard University Press 2000, p. 23. This description is strongly indebted to Gilles Deleuze's text "Postscript on the Societies of Control".

[12] Fuller, Matthew: "A Means of Mutation", *Behind the Blip. Essays on the Culture of Software*, New York, autonomedia 2003, p. 65

[13] The title of Matthew Fuller's collection of essays published in 2003. Behind this notion of a *behind the blip* is the assumption that artifacts have politics, to paraphrase Langdon Winner, or that technology is the product of culture and not the other way around.

[14] Fuller lists three more or less specific models of alternative software, critical, social and speculative software, which form the basis of the software art toolbox. Although not directly mentioned and included in the argumentation these models are implicit points of references in this text.

[15] "A Means of Mutation" is the title of Fuller's text on *The Web Stalker*.

[16] Deleuze's concept of becoming is developed in most of his books, whereas his concept of actualization derives from his book on Henri Bergson.

[17] Dreher, Thomas: "Vertnetzungskünst(l)e(r)", *Ars Electronica 95*, Springer Verlag, Wien 1995.

[18] The quotation marks around conceptuality refer to Deleuze and Guattari's use of the term in their last book *What is Philosophy?* Whereas Deleuze and Guattari dismiss electronic media's participation in the production of concepts (that is for philosophy only) Fuller argues that software is indeed conceptual.

[19] 'Human' here does not refer to human beings of flesh and blood but to the "conceptual person" like Decartes' cogito, Nietzche's Dionysos, and Platon's Socrates that Deleuze and Guattari see as the origin, courier, and subject of a philosophy. Fuller's emphasis on the conceptuality of HCI differs from those critics and artists who see the back-end as the decisive level of conceptuality in software art. Significantly, his emphasis makes it possible to re-include visuality in the aesthetics of software art, although not in the form of abstract imagery. Rather, inspired by Lakoff and Johnson's theories on "metaphors we live by" and phenomenological theory on the structural nature of perception it is visuality refering to the reflective and productive mind, not solely to the contemplative eye. This is an important re-definition of the visual dimension not just in software art but in conceptual art in general that hopefully will be discussed in detail on future occasions.

[20] Schultz, Pit: "Jodi as a Software Culture", *INSTALL.EXE—JODI*, Cristroph Merian Verlag, Berlin 2003, p.83

[21] Fuller, Matthew: "A Means of Mutation", *Behind the Blip. Essays on the Culture of Software*, New York, autonomedia 2003, p.62

[22] Fuller, ibid., p.61