

# A Re-declaration of Dependence

## software art in a cultural context it can't get out of

When the term software art surfaced some years ago it quickly gained an enormous popularity in the digital art community. Even though, such a thing as art made by software had existed since the 60s, many viewed it as the next big thing after net art and thus it opened a new field of aesthetic discussions enriched by a large number of works, articles, and festivals like Read\_me. From the very beginning this field has been somewhat divided: On the one side are those artists and critics who emphasize the literary and mathematical sources of software art and argue in favor of aesthetics centered around the formal, linguistic, and expressive qualities of programming (languages) and code; and on the other side are those who inspired by disciplines such as media studies, critical philosophy, cognitive science, social, and political theory turn towards aesthetics that focuses on software art's conceptual, and discursive involvement with the culture of software. The jury statement of the first Read\_me festival in Moscow, which unofficially stands as the original definition of software art (after net 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 often acts as a dividing line. Software art is either treated as an art of formalism or of culturalism, to use a deliberately polemical distinction by Florian Cramer. Of course, the formal and cultural aspects of software art overlap to a great extent (Cramer who is both a literary scholar and a free software activist is a clear example), but as a tendency in the aesthetic discourse surrounding software the distinction is nevertheless an issue.

Of the two views, the former seems to predominate current theoretical discussions, criticism, and curatorial practices, as the "CODeDOC" exhibition at the Withney in 2002 and last year's Ars Electronica show. Certainly, this predominantly formalistic view prompts reflections, which are highly 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, I believe, that the so-called cultural view accentuates equally important and so far rarely treated aspects of software art, aspects that grasp software art not in itself, but as part of wider contexts, historically as well as theoretically.

### Concep/textual art: If you are in, you're out there

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 continues "the de-materialization of art" that conceptual art began, to cite the title of Lucy Lippard and John Chandler's seminal text from 1967. Through this so-called dematerialization, conceptual art initiated aesthetics of text, language, reading/writing, and interpretation; aesthetics based on the perceptual and cognitive challenges of processes, systems, and structures, instead of on the specific media, i.e. a physical definition of art, and the visually convincing art-object that modernism had championed. However, the dematerialization that Lippard and Chandler detected did not designate a literal dematerialization, but a conceptually charged approach to aesthetics and the making of art. The aesthetics of dematerialization was involved with objective, non-individual, forms of recording, documentation, mapping, and organizing different material (sic!), and not with the subjectivity of the artist in a psychological or mythological sense; and it 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 I mentioned before to some extent arises from how conceptual art is interpreted and which of these trends are understood as the forerunners of software art. To make this important aspect clear I would like to briefly sketch a number of different trends in conceptual art. I should mention though, that the various trends that I mention in the following are to a great extent overlapping, and the distinction I make between them here is more a matter of different perspectives pointing towards software art than of opposing aesthetics. The current aesthetics of code and programming (languages) is often related to two different, but parallel trends of conceptual art. The leading figures of the first trend, referred to by Alexander Alberro as "linguistic conceptualism", were mainly Joseph Kosuth, Sol Le Witt, and members of the Art & Language group. These artists and writers were engaged in ontological, genealogical, and epistemological reflections on the nature of "art as idea", as a self-defined and self-reflexive logistic system composed by writing (instructions, definitions, informative texts, etc.) and ideas, and as 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, comprehended 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 of software art as the basis of all computer art.

Nevertheless, when it comes to the cultural dimension of software art two other trends of conceptual art seem to be more relevant: One is represented for instance by Hans Haacke, Dan Graham, Victor Burgin, and Gordon Matta-Clark, which was political in an interventive and analytical sense; while another is represented by artists like Vito Acconci, Bruce Nauman, and Chris Burden, which was involved with performativity and stagings. These groups of artists saw conceptual art as a contextual activity that dismantled and completely rejected the notion of the transcendental and autonomous work of art that modernism had praised. For them conceptual art was as an experimental and critical practice 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.

In a text from 1975 published in the American conceptual art magazine *The Fox* and carrying the programmatic title "A Declaration of Dependence" co-founder and artist Sarah Charlesworth describes this change in the status of the work of art towards the contextual. The backdrop for her declaration was the disillusioned, retrospective realization that conceptual art could not, as it had originally set out to, exist outside institutions in its own dematerialized aesthetic sphere: "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." Charlesworth's description of the contextual nature of conceptual art points towards 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 going at the context**

This contextual definition of the work of art experienced a revival in the 1990s when a new generation of artists more or less explicitly reinterpreted the heritage of conceptual art's social politics. A number of critics and curators quickly picked up on this revival, and among them were Peter Weibel. In 1993 Weibel curated the show "Contextual Art. Art of the 90s", which presented a group of upcoming artists of the new decade and a few of their influential predecessors. (Before I go on I would like to note that while Weibel is certainly not the only one to theorize the art of the 90s or contextual art in general his exhibition nevertheless stands as a landmark that had a seminal influence on the discussions of contextual aesthetics at the time. I also want to note that his term contextual art, which he developed in connection with the exhibition is somewhat ambivalent, because, as Jan Avgikos points out, "the context was always already there"; meaning that whether art is pre-modern, modern or post-modern, it exists within a context. In this sense Weibel's term contextual art is a tautology, which does not specify that what makes contextual art contextual is its thematization of the relation between the work of art and its context. Not surprisingly, other terms have been introduced, for example by the German scholar of conceptual art, Thomas Dreher, who uses the more specific term Context Reflexive Art, or the American art historian Anne Rorimer, who talks about "Context as Content." However, I will not go further into the principles involved in this terminological discussion and simply stick with Weibel's term.)

Back to Weibel's exhibition. A monumental catalogue, which included Weibel's curatorial text, entitled "Contextual Art. On the Social Construction of Art" accompanied the exhibition. The text was an updated version of his 1971 text "A Contextual Theory of Art", in which he had criticized the "syntactical" poetics of modernism to emphasize the social aspects of language (use), i.e. communicative relations and the structural conditions behind the linguistics, to conclude that "contexts [were] more important than texts." In the 1993 text Weibel expanded the theoretical horizon of his contextual aesthetics from the perspective of contemporary art that had emerged in the early 90s. 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 to 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 1990s stressed different transformations of the contexts, reworkings of specific and actual realities. The relation between the work of art and its context characterized a complex dynamic of constant negotiations and exchanges, an "open field of signs and actions", as Weibel some years later described it in his text, "Art as Open Fields of Action" written for the Austrian Pavilion at the 1999 Venice Biennale. It is thus important to notice that Weibel's contextual aesthetics do not dissolve the work of art in a contextual determination; rather it emphasizes 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, which emerged in the late 70s and 'lasted' close to a decade, was involved with a critique of the representations of the social field within the art institution. The third generation, which is 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 or intervention in the social field; it replaced symbolic actions with real actions, and became "partisans of the real", to use one of Weibel's formulations. Instead of acting politically within the art institution, this generation acted aesthetically within the social field. And 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 aesthetic based on the idea that the recognition of art as a social construction, represented a potential for the construction of the social. The last paragraph in his text thus reads: "The goal of the social construction of art is to take part in the social construction of reality."

### **The arrival of a fourth generation: software art**

I believe that the different trends of conceptual art outlined above, plus Charlesworth's and Weibel's theories respectively, form a relevant historical and theoretical perspective from where an understanding of the conceptual dimension in the cultural trend of software art can be developed. It is a perspective that allows us to see important works of software art as part and innovators of an aesthetic tradition, which approaches art's conceptual dimension from a context of social engagement.

The perspective points back to Jack Burnham's 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. In his thorough and enlightening writings on the exhibition Edward Shanken explains how Burnham saw "[t]he notion that art can be separated from its everyday environment [as] a cultural fixation." Burnham used software as a metaphor for conceptual art, which aesthetically computed the politics of the technological developments that influenced post-modern culture at large. And from Burnham, the perspective points beyond modern aesthetics based on formalistic taste and ideas of absolute artistic autonomy, beyond the black box of Donald Knuth's "art of programming", to the 'expanded' aesthetics based on conceptual criticism and interdependence between software art and its cultural, social, economical, political, and technological contexts; to a re-declaration of dependence.

More specifically, I think that this perspective allows us to pursue Weibel's historical thesis and suggest that a number of significant artists working within (or associated with) the cultural trend of software art, constitute a fourth generation of contextual artists. It is a generation represented by a diversity of artists and works like LAN (TraceNoizer), I/O/D (The Web Stalker), Mongrel (Linker), 01.org (life\_sharing, VOPOS), Knowbotic Research (10\_dencies, Connective Force Attack), EDT (Floodnet), Carbon Defense League/Institute for Applied Autonomy (re-code.com), ubermorgen (The Injunction Generator), and RSG (Carnivore), to mention just a few. This latest addition to the 'contextual family' is closely related to the third generation, but the artists of the fourth generation are digital by birth and act as partisans (freedom fighters as well as construction workers) of the culture of software, not of the real as such. They conceive software art as a 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 in *Empire*. This society they write, echoing Gilles Deleuze in his book on Michel Foucault, is 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." However, these contextual software artists do not limit their criticism to claims of aesthetic independence and exclusivity; rather they apply it to the culture of software. Instead of taking up the dialectic position of the negative outsider, they conceive their criticism to be positively dialogical in the sense that they work from the inside, both with and on the culture of software. 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, its material structures and collective imagination. It sees this culture as a context, which is not pre-formatted but "up for grabs," 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" of the culture of software, on 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 construction workers 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. These alternative tools are not tools to be used for specific purposes in the construction of concrete objects and specific 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", not stabilizers. Their powers are constituent rather than constitutional, to borrow a distinction from Hardt and Negri. They bring about change, difference, and liberation, sometimes even of a revolutionary nature (this is the rise of a new kind of working class). 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, the creative modes of operation of these tools 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. To quote the French art historian Nicolas Bourriaud from his important text *Relational Aesthetics* written in 1998, these tools produce "formations rather than forms," in the sense that they are concerned with dynamic and differential relationships rather than with enclosed objects.

As such, it is not only the conceptual production of the tools themselves, or the tools in themselves, as much as the potential uses and users 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 in his text "Networking Artists", "programmers of programming possibilities". These programmers of programming possibilities conceive their tools for a multitude of users, including themselves, 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 Matthew Fuller's notion that software "participate in "conceptuality."" This notion, which is formulated as a critique of Deleuze and Guattari's technological skepticism, 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, software not only determines possible uses, but also constructs a user. To paraphrase Fuller: Software engineers 'humans' through HCI. Human here does not refer to human beings of flesh and blood, but to "conceptual persons" like Descartes' cogito, Nietzsche's Dionysus, and Plato's Socrates; persons which 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 a conceptual visuality referring to the reflective and productive mind, not solely to the contemplative eye. (In parenthesis, I would like to add that I think 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.) As Fuller's exhaustive analysis of Microsoft Word, the archetypical example of conventional software, shows, this conceptual engineering of humans is very often regulative, manipulative, and one-dimensional, in the sense that it reduces the use or user 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 for

software. To paraphrase the classical I/O/D slogan: If software is mind control then come 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 the use of software, which isolates the user and his application from the rest of the world, 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 singularity and heterogeneity in toto. Subversion, distortion, criticism, dialogue, fantasy, science fictions, remixes, affections, antagonisms, paradoxes, irrationalities, and complexities are its principles of engineering; connectivity and experiments its working ethos. In other words, by constructing alternative tools, this generation seeks to engineer alternative humans and consequently, this is the hands-on utopia, create an alternative culture of software.

### **Towards an aesthetic of contextual software not-just-art**

My suggestion that software art represents a fourth generation of contextual artists 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.

Therefore, it seems important to me to turn attention to which new conceptualizations of art in general, and conceptual and contextual art in particular, software art introduces through these specificities. By this I do not intend to assimilate software art and reduce it to just another new kid on the block, on the contrary. Even though software art, like every art form, has its particular qualities, these qualities need 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; and I believe that software art can benefit from these connections in a number of ways. They emphasize the relational, inclusive, and synthetic qualities of software art, which, to me, seem to be some of the most relevant for a discussion of software art in relation to the contextual trend in contemporary art. They make it possible, and plausible, to discuss the aesthetics of software art as part of the "material tradition" that Bourriaud places his relational aesthetics in. This tradition is concerned with how materials (in the broadest sense) are fundamentally coded by culture and how they can be de-coded and re-coded by art. Fuller anticipates this discussion with his look "behind the blip", "digital objects" and interest in "materialist energies in art and technoculture", as does Alex Galloway with his neo-Marxist readings of the Internet in his recent book Protocol. Hopefully, more will follow.

Another 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 something like "a pure technicity without purpose", 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 categorical. It leaves any notion of formalistic purity behind to define a new kind purposefulness that recognizes an essential potential in software art's ability to reconceptualize art through the tool, and the tool through art, which was, I believe, exactly what Burnham, among others of his time, tried to do. Fuller also anticipates such a reconceptualization by introducing the term not-just-art in relation to *The Web Stalker*. This canonical work of software art, he writes, "can only come into concurrence by not just being itself. It has to be used." 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." I find this term very instructive for a further development of software aesthetics that avoids the avantgardistic 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. In digital, virtual, or, maybe more appropriately, soft terms, it updates what Bourriaud, talking about the art of the 90s, has called "operational realism." In his *Relational Aesthetics* he defines this realism, which he implicitly opposes to pictorial realism, as "the presentation of the functional sphere in an aesthetic arrangement", in which the artist "aims to set up a certain ambiguity, within the space of his activity, between the utilitarian function of the objects he is presenting, and their aesthetic function." I believe this ambiguity is also present in the term not-just-art. Not as hindrance or hesitation to call a piece of software 'art'; on the contrary, this ambiguity represents an aesthetic flexibility and elasticity that allows software art to move in Marxian interstices of the culture of software and thus create spaces that differs from "the "communication zones" that are imposed on us" as Bourriaud puts it; hitherto unknown aesthetic spaces for reflection and production, conceptual as well as contextual.

As a closing remark I want to emphasize that the contextual approach to software art that I have presented here, is, of course, just one of many ways to give this art form of today and 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 theory should not be objective sciences, but experimental methods and models for the production of aesthetic knowledge and experience. Instead of trying to distil the pure form of software art, they should contaminate and pollute software art in multiple ways leading to complex, rich, and visionary understandings of software art, as well as of the culture of software.

*Jacob Lillemose, Copenhagen 2004*