Transmediale.01 Festival (Berlin, 8th February 2001)

Is software art a genuine artistic material?

Andreas Broeckmann
abroeck@transmediale.de
Artistic Director of Transmediale

Florian Cramer
cantsin@zedat.fu-berlin.de
Lecturer in Comparative Literature at the Freie Universität Berlin

Ulrike Gabriel
gabriel@em.uni-frankfurt.de
Director of Codelab Berlin

Golan Levin
golan@flong.com
Lecturer in Interaction Design

Rafael Lozano-Hemmer
rafael@lozano-hemmer.com
Digital Artist

Anne Nigten
anne@v2.nl
Artistic Manager

Daniela Plewe
plewe@is.in-berlin.de
Digital Artist

Gerfried Stocker
gerfried.stocker@aec.at
Managing Co-Director of the Ars Electronica Center

Abstract: The *transmediale.01* organised the first competition that includes an art award for software. This competition acknowledges the artistic work done by artist-programmers who are neither "interactive media artists" nor "net artists", but whose aesthetic material is code and whose form of expression is software programming. One definition suggested for software art is that it encompasses projects in which self-written algorithmic computer software-stand-alone programs or script-based applications-is not merely a functional tool but in itself an artistic creation. Does software serve a merely instrumental function, or does it offer new, creative cultural perspectives? Is computer code a genuine artistic material like paint or digital images?

Andreas Broeckmann: The *transmediale.01* has organised the first competition that includes an art award for software. This competition acknowledges the artistic work done by artist-programmers who are neither "interactive media artists" nor "net artists", but whose

ISSN 1695-5951
Legal deposit B-52019-2002

aesthetic material is code and whose form of expression is software programming. One definition suggested for software art is that it encompasses projects in which self-written algorithmic computer software-stand-alone programs or script-based applications-is not merely a functional tool but in itself an artistic creation. Does software serve a merely instrumental function, or does it offer new, creative cultural perspectives? Is computer code a genuine artistic material like paint or digital images?

Gerfried Stocker: The interesting thing for me is that even at this very advanced stage of dealing with software art there is always the strange need to justify this kind of art outside the traditional context of art. And I think that is not radical enough... Not as radical as you wanted it...

Andreas Broeckmann: I think that you underestimate the development of art in the last 30 years. I think what you call radical and new in software art is the kind of radicality that every avant-garde lays claim to. I'm sure that there are three or four other avant-gardes going on at the moment where people claim that same radicality. And it's still art.

Gerfried Stocker: Of course. It's still art, but I mean a lot of things are art. We know that playing Mozart at the Vienna Staatsoper is wonderful art. No doubt about it. We know and we call it art when we see the movies at the Berlinale. Art is just too broad as a definition and actually not really interesting enough for this process of development. We have to be confident enough to realise software art is a reaction to the general development of society and culture driven by new technologies, by digitisation, by the conversion of everything into code. There are only very few, limited possibilities for artists really to work with it if they want to stay within this process and its dynamics of transformation. Otherwise they step outside and make interesting works of art, like for example Jeffrey Shaw or Bill Viola. Wonderful artists, but examples of what I mean...

Ulrike Gabriel: We won't talk about them here, because they don't code...

Gerfried Stocker: Probably everybody here knows the work that Bill Viola did for the opening of the ZKM. You have a screen in front of you and when you walk towards the screen there is a tree growing and ageing and dying depending on the distance that you are from the screen. A few months ago I was at a conference in Stanford and Bill Viola was explaining this work and he was claiming that this is interactive art. And he was referring to himself as an artist and to a programming technician at ZKM, Bernd Lintermann, who did some programming for it. In my understanding, the real art in this project is done by Bernd Lintermann because he wrote the code that makes this tree grow, transform and die. And what Bill Viola did was that he completely gave up his role as an artist and diminished himself into a kind of stage designer. Because the only thing he did was to decide how the thing should look inside the museum.

Daniela Plewe: I agree and I think that there are several ways to specify the code. I think I agree that saying "Here we need a tree or there we need something that makes decisions", is not enough. For example, in my case I was always really concerned about learning the very abstract formalisms. And it took much longer than I intended. Maybe we should also talk about this delegation of the code and the disadvantages which this always brings with it. Of course you lose the quality feedback from the artistic material. I would always admit that-and it was a painful experience. If you communicate with the programmer of course these people get much more involved in the aesthetic concept by the time everything is ready. The other question is whether we want the same structure as in film production, where division of labour is fully accepted-and how software art could be done in the future...?

Golan Levin: My own background was originally as a painter and composer. It was only after many bad experiences trying to convince engineers to help me build my work that in the end, out of frustration, I had to learn how to do it myself. I would ask them to help me with things and they would say, "Oh, I'm a hundred dollars an hour," or "I help you out of pity," or I'd lean over their shoulders and would say "It's not right aesthetically," and they would say "I don't understand what you mean." In the end it was just necessary to dig in. It's not only a craft, but the craft is important. One has to be in touch with it and then the concept is also important of course.

ISSN 1695-5951
Legal deposit B-52019-2002

Ulrike Gabriel: I'd like to return to something Gerfried Stocker said. As an artist, you have a certain material you use. In Antoine Schmitt's project Vexation 1, you saw this black screen with a ball bouncing around-there was a certain material used, a certain space opened up. The space was very minimal. And the artistic material was used inside this space, with this definition of the parameters. This was adequate. In Gerfried's example, Bill Viola didn't even know what he was using. He did a job he couldn't do as an artist. That's why I have this very basic position: let's stay artists. Just go there, touch the code, use it, use it for your artwork, consider yourself an artist just in the classical sense, because then you keep this process.

Gerfried Stocker: I'm absolutely in favour of any type of software art. We're not at a trade fair here, selling software art, we're in a very intellectual community discussing problems and issues around software art. And we have to become more radical and more demanding. It should not be enough that somebody is writing code himself. That is not really enough to be a software artist. Because some of the works are very similar to this still very important approach of low tech artists: "We get control of our tools and this is important." This is vitally important, but it is a different direction. I'm very interested in any type of work that is really using software in a way that reflects a new way of producing and making art in terms of the shift from document to event. Any digital picture, any digital sound, any sample is a stream of code, but it is not really software. It is still a document. And then we come to the event in a form of description of something that is really the algorithm which is reproduced every time. You have this process of converting text, coded into silicon memory and then it becomes something.

Audience/Tim Druckrey: There are two things we should distinguish. One of them is that software is not just code, it's the algorithm. If I can code something in C, but I want to put it on the web and I can't program in Java and I get someone to code it in Java for me, does this mean that I handed over my role as an artist? If I formulate the algorithm to get someone else to code it, are they the artists or am I the artist?

Daniela Plewe: I've written down what you just said more or less literally in my notes. I'd like to bring into the discussion the notion of algorithmic art, instead of artistic software.

Golan Levin: I use algorithms in my work and I can't deny that sometimes the output of these algorithms can be lovely but I suspect that if you were to have a competition like this of algorithms themselves in the pure state I think it would be rather boring. You can go out and look at Donald Knuth's five volume set of software algorithms and see for yourself. It's really just a tool that we can use. And you could have a competition, but I think you need a really well-qualified audience to appreciate why "quick sort" is better than "retex sort" or "bubble sort". And why should we give an art prize to those people?

Audience/Tomiko Thiel: The works that we have seen tonight are mostly the sort of tools to generate sensory experiences. Maybe Daniela Plewe's came the closest to making the data and decision structures visible, which the jury suggested was what they were looking for or hoping to find through this competition.

Florian Cramer: Perhaps the sensory experience of what we just saw as demonstrations of software art works sometimes gets in the way of actually seeing the generative processes going on inside. If you look at the presentations tonight you may have the impression it's media art as we know it. But imagine we would, for example, have included a self replicating algorithm. We could have shown you a sample of C++ code or something like that. But you wouldn't have grasped anything from that. The entries were at the threshold between conventional media art, which is very strongly focused on visual and sensory representation, and a software art that is concerned about concepts, about algorithms. How can this be translated into a sensory experience at all? I'd be very glad if we could some day do without data beams. Data beams are an aspect of media art which I really hate since they exist. And everything boils down to presentations on data beams. I think we're aiming towards the invisible. This is really exciting. We're aiming towards processes which are no longer focused on interface representations. The fact that we now have many examples of software art which take the interface experience but subvert it or turn it upside down, marks something like the threshold, the end of one discourse and perhaps the beginning of a different discourse in electronic arts.

ISSN 1695-5951
Legal deposit B-52019-2002

Gerfried Stocker: I'll take up the discussion about aesthetics in software art because it's a very problematic term: it's so strongly related to traditional art but it's a very interesting point really to look at what we're thinking of when we're talking about aesthetics in programming. I'd like to refer to Golan Levin's work, which I admire very much. And the aesthetics of his code is not in the beautiful pictures or the nice sounds that he's making. What he achieved is to create something like a clay. The whole interface becomes completely integrated, an integral part of the software itself, of the code and its expression. And I think this is really one of the greatest achievements that software art can do no matter what it produces, whether in the end it's nice pictures, nice sounds or maybe some virus that is so well done that it's able to bring down the whole of the Internet.

Andreas Broeckmann: You'll have to explain, though, why this full integration as an aesthetic model is superior to, for instance, friction.

Gerfried Stocker: Friction might also be very interesting. But it's a question of these different levels of writing and inventing the code, creating an interface as technology or the interface in its sense as the surface where communication can happen with the project and the result. This integration is very important. In the case of Golan Levin of course it's not really creating friction, it's creating nice pictures. But I said the pictures are not important, and nor is the friction really important. And that is why I don't accept a definition of software art that only refers to viruses and codes and this subversive potential of software art—and I would also not accept any description of software art that only referred to nice pictures and nice sounds. The quality itself lies in a completely different area. There is a potential danger that we are looking at the wrong thing.

Rafael Lozano-Hemmer: These things are not mutually exclusive. On the one hand when we're talking about software in a very fetishistic way, like Gerfried Stocker is doing, I share that kind of formal attraction to the fetish of the code. At the same time we have to be fully aware that in no way is the code not socialised. In other words the code, as we all know if you program in different coding languages, is in itself a kind of collaboration with the people who wrote the code. So to the degree that there is no purity in that kind of expression, I'm interested in the fact that code itself is social. And you can say the opposite thing for instance with respect to other works that like mine are more assemblages of other kinds of media or other kinds of social patterns. In that we are working on a code as well, except that it is not a software code; it may be a language code or a political, social, or economic code. I think Golan Levin's work is extremely social and the way in which my body reacts to his code in terms of the choices he made for the effect of the images and sound is extremely social. To me, that's where the value lies; whether he did it in Director or otherwise, I think it's meritorious and interesting.

Gerfried Stocker: I would agree. That's the very important point of the interface which is one part of this change. And I think it would be wrong to divide these things. Just to make it clear again, I was specifically referring to the aspect of aesthetics. What could be the aesthetics in this type of art? There are fortunately criteria in looking at art other than only aesthetics. And software art also has to refer to these other criteria.

Anne Nigten: I think there is a difference between artistic code and software art or artware. I think that's something not to be confused. Personally, I'm not that interested in the aesthetics of the code, but what is created through this code or what this code will bring about. Then I'm not referring to the common notion of art again, but really to a much broader experience caused by the code.

Daniela Plewe: I still wonder about the transparency issue. How can we communicate to the audience something of the complexity we are all interested in? Do we want that or are we happy with a very specialised audience? Like Gellernter says: machine aesthetics and the pure optimised algorithm. So if the mathematician finally says what is great and what is art, then we open up the whole spectrum of what we are talking about to mathematics and the beauty of mathematics, too. Since there is a lack of definition of art, I think I would even agree if mathematics was included in the future.

ISSN 1695-5951
Legal deposit B-52019-2002

Related links:

- Transmediale Festival:
  http://www.transmediale.de
- Transmediale Jury Statement on Software Art:
  http://www.transmediale.de/en/02/softjurystate.php

Partners:

http://www.n5m.org

transmediale

http://www.transmediale.de

Published on: December 2002

Recommended citation
BROECKMANN, Andreas (2003). "Is software art a genuine artistic material?". *Artnodes*, issue 2 [article online].
DOI: http://dx.doi.org/10.7238/a.v0i2.683

ISSN 1695-5951
Legal deposit B-52019-2002