Software Art Panel - Media Arts Lab at Künstlerhaus Bethanien
In cooperation with transmediale.03
February 4th, 2003, Studio II

We gratefully acknowledge the support by Initiatief beeldende kunsten vzw / Digitaal Platform, Brussels, for this transcription.
Transcript by Vali Djordjevic
Edited by Inke Arns
Published on http://www.softwareart.net/

Inke Arns: [...] The somewhat polemic title of our panel today is "software art - a curatorial fiction or an artistic tendency". 2003 marks the third anniversary of the transmediale software award which since its first installment 2001 has instigated an ongoing debate about software art and its legitimacy. On the one hand the idea that artists may not just be users of prefabricated software to produce something else but engage with programming and software and code itself in playful inventive and critical ways has caught on and spread over to other festivals, exhibitions and artistic projects. On the other hand there is no comprehensive identity or network of software artists. Jury work for software art competitions often turns to be difficult because of little and mixed quality input and because much if not most of artistically interesting software - hacker code, conceptual art, algorithmic musical composition for example - is written either outside the art system or is not being thought of as software art by its creators.
This international panel gathers artists and critics from Russia, the United States, France, Great Britain and Germany: Olga Goriunova, Amy Alexander, Florian Cramer, Antoine Schmitt and Alex McLean - most of whom happened to be involved with the transmediale software prize either as winners or as jury members. Perhaps here I should mention, in the program it says that Margarete Jahrmann would participate, she could not come, so we have to do without her. The idea of this panel is to critically discuss software art and its curatorship in the last three years and draw a preliminary conclusion, if this is possible, of its viability. So before presenting the participants, let me give a brief outline of the topic of our discussion.

Software art, which I would consider as a heuristic term, in the broadest sense describes activities which use software or code as an artistic material. I will not elaborate on the notion of software art any further because I'm sure that each of us has a slightly different perspective which will be addressed during the discussion. I'd rather direct your attention on

the question of why software or code could be at all an interesting material for artistic production.

So why software? Software has become ubiquitous which means that even unknowingly you cannot avoid it anymore today. Software nowadays can be found not only in computers but in almost everything ranging from communication devices, telephones, all sorts of media machines, even washing machines and other household devices. But ubiquitousness in this case does not only mean that software is everywhere, that it is all pervasive but it also means that most of the time it remains invisible. This general invisibility of software applies on two different levels. First in most cases so called "raw software" or the code is covered by glossy surfaces where you are not manipulating the code directly but where you are working with a graphic user interface. The second level of the invisibility of software applies at the interface itself. It is what computer scientists call the "transparency of the interface". In everyday language transparency normally means visibility or clearness, or controllability through visibility. Transparency in the case of computer science rather stands for information hiding, which means that the user does not notice the software working in the background.

Even if information hiding in the case of interface design can be useful, it can be said that at the same time it suggests to the user or to the viewer a direct or even natural view or access to the data. This is of course not the case as Lev Manovich notes in his book "The Language of New Media". A short quote: "Far from being a transparent window into the data inside the computer the interface brings with it strong messages of its own." Today, in a time when our environment gets increasingly mediatized and digitized and thus can be said to be based increasingly on software, it becomes more and more important to be aware that code or software directly affects the virtual and actual spaces in which we are moving, communicating and living. It has the capabilities to directly mobilize or immobilize its users.

This is why Lawrence Lessig in his book "Code and other laws of cyberspace" claims that program code increasingly tends towards becoming law. This is his by now almost famous short motto: "Code is Law." Today control functions are being build directly into that very architecture of, for example, the net, which means into its code. Taking as an example the online service America Online (AOL) Lessig poignantly makes clear how code directly enables or disables freedom of movement, of speech and of behavior. Code should - even if it remains largely invisible - not be accepted as something natural or as god given fact. It is rather written by humans and can therefore be changed or conceived differently.

Code works or software art deals with this code and software structures underlying and generating visible surfaces. Software art focuses our

attention on the all pervasive raw program code which our increasingly digitized working and living environment is based upon and uses this code or this software as its artistic material.

Perhaps so much for an introduction to this broad field we are supposed to discuss today. I would now like to briefly introduce the panelists. Before I do this I should say that we decided two days ago that everyone of us will present a piece of software art - very briefly in five minutes - to give you a very broad overview of this field that could be called software art. We are trying to cover a broad range of practices from political activism, interface manipulation, simulation, programmers' humor and also music. This what we are going to start with.

Florian Cramer: Just to mention briefly why this panel came together: There was a mailing list discussion between Antoine, me and Amy, and then it somehow broadened. We discussed recent developments of software art, decided to meet in Berlin during transmediale and continue our discussion in person. Then somehow it became a public panel.
That we are sitting here and you are sitting there doesn't say much because I recognize so many faces and see so many people here in the audience who could sit and speak here in front as well. So we hope to make it as open as possible and involve you all. But as we also don't want to make it an insider discourse, we will first show you what we think of as software art to give you a better impression and to ground our discussion.

Inke Arns: This will be of course one of the questions: How much do you have to be an expert to enjoy software art?
I will start with Olga Goriunova who is sitting second left from me. She is from Moscow. She is the co-organizer of the Moscow based Read_me 1.2 and Helsinki based Read_me 2.3 festivals and of the software art repository Runme.org. I think we will have the chance to talk about this project later on because several people on this panel are involved in this project. She is the author of the suicide letter wizard for microsoft word which she will talk about and she also published several articles on digital culture. Currently she is a Ph.D. student in the media lab at the University of Industrial Arts and Design in Helsinki. She is based in Moscow and Helsinki.

Olga Goriunova: Hello, I'm starting, I'm the first one to present two projects. I'm the first one because the projects I'm presenting entertaining software art pieces with an intellectual touch or maybe pop software art. Both projects are easy but at the same time deeply critical. The first one was recently entered to runme.org. As Inke has mentioned almost all people who are sitting here and also some in the audience are involved in creating this big software art repository. The project is called SPS and was made by Karl-Robert Ek. It is a screensaver and it plays a "paper rock scissors" game with the user.

A gesturing hand is projected onto the wall. It plays the poses of "paper - rock - scissors". Olga Goriunova interacts with the projection and plays the game.

One - two - three. One - two - three. One - two - three.

Laughing from the audience

So as you see it is a very simple piece made in Flash but I think it is very conceptual and carries a lot of theories within it. It brings the game outside of the computer, in the physical space. It excludes any physical interaction of the user with the computer interface but it is a computer game anyway. I won't give you any further theories about it as they can go on for too long.

The second project is called "Suicide letter wizard for Microsoft Word". It is a wizard that generates templates for suicide letters. Basically it is an add-on to Microsoft Word. When it launches itself it also opens Microsoft Word. It is similar to many existent templates that are made to cover all possible areas of human life.

So here we have the wizard where we can choose a layout which is of course very important when writing a suicide letter … then we can use a sample salutation. Then there are a few categories why …

She demonstrates the suicide letter wizard for the audience. A pop-up box where the user can choose from menus similar to other "wizards" in Microsoft programs. Laughing from the audience during the demonstration.

Each category has its own text. Then there is a conclusion and also a closing. We enter the address. Then we can use style. I will choose modernistic now and then we finish.

Here we have the letter with a date and an address. Here you can put your signature. And some information. And it has a nice layout.

When I was making this piece I wanted to make a critical comment to mainstream culture, the mainstream software culture of proprietary programs especially from Microsoft who pretends to satisfy all needs of every individual user. But of course like any other part of the culture industry it aims at satisfying the needs of capitalist information production. Topics that refer to other parts of human life are never covered and they are always taboo.
Thank you, I think I got my five minutes.

Inke Arns: I would like to briefly present Amy Alexander now who is sitting to my far right. She has been working with film, video, music,

computer animation and new media. She is currently assistant professor at the University of California, San Diego. Since 1996 she has been working primarily in net and software art exploring net culture as well as dynamic processes and structures. Her recent work has been primarily in live net performances and software projects. Her internet projects include plagiarist.org, theBot and The Multi-Cultural Recycler and her latest performance work is called b0timati0n. She is on the software jury of the transmediale.03 - we are all looking forward for the award ceremony tonight where she will present the prize for software art - and she also has been on the jury of the read_me 1.2 in Moscow. And as I mentioned earlier she is a member of the runme.org software art repository group. OK, it's your turn.

Amy Alexander: My projects and my interest in software sort of run all test over from pop software to political to algorithmics so I thought I'll show you two extreme but related projects.
We have all seen something like the Windows screensaver [shows flying text screensaver] - this delightful monstrosity - so before we go into seizures we are going to look at a project that was one of the winners of the read_me 1.2 festival last year. It is called "Screen Saver" and it's by Eldar Karhalev and Ivan Khimin. This is the project right here.

Opens a text document.

It's basically instructions to modify the screensaver to do something else. So we go back in, select screensaver, select 3D-Text, select settings, in section display select *text* - it is a technically very difficult piece … in section size select *large*, in section resolution we select *max*, in selection surface style we select *solid color*, for speed we select *slow*, at spin we select *none*, and most importantly in the text area put in a full stop. Now move on to the choice of font and we should select *Verdana Regular*. Okay, okay and apply …

She presses the buttons.

… and now we've got something much better.

Laughing from the audience which sees now an oscillating square moving across the screen as the new screensaver.

This obviously makes the high-tech 3D, obnoxious icon of the Windows screensaver do something its programmers didn't intend it to do. So you could call this “software art for users” or “user generated software art,” as opposed to the traditional programmer-generated software art. It shows the users' resistance to the restrictions imposed by software and technology but also points out that software art is not just for programmers.

But what about programmers? *She opens a text on a website which documents a piece of software.* This project is called "Acme-Handwave" by Simon Batistoni (sometimes also called Simon Kent for some reason) and it's a software library. Libraries are pieces of software that programmers use to build other software out of. The Acme collection is a bunch of software that subverts the usual utilitarian uses of libraries to make things that are clearly not utilitarian. This is a something for programmers who want a certain result without tedious algorithms or real programming. As any programmer knows sometimes you are writing something, you are writing this algorithm, you are expecting a certain answer but that's not the answer that's coming out. So what Acme-Handwave does: it asks you what kind of data you are putting in, what the expected result is, and it will sort of wave its hands and give you the result that you want. So you can just give it what you want; it does some hand waving magic and the result you were looking for comes out.

This is a goofy little simple thing, but it also reminds us that it is not only users who are restricted by and trying to resist technology, but programmers are in this position as well. More importantly it reminds us that the programmer is not a mechanical part of the software or the interface but also a human struggling with the technology. Finally, it reminds us that programmers and algorithms are not neutral, objective, mechanical and interchangeable, but in fact they are highly subjective. So maybe when you punch in something in a piece of software and it gives you the answer it’s not the answer, the neutral godlike sterile, infallible technological answer, but rather it’s the programmer's voice expressed through an algorithm.
Next victim, please!

Florian Cramer: Since Inke now is going to present something, I will introduce her briefly. Inke is very well known in net culture: She's the co-founder of the Syndicate - now Spectre - mailing list together with Andreas Broeckmann and a founding member of the Berlin-based mikro e.V. net culture collective. Apart from that, she studied art history and Slavic philology and just completed her Ph.D. thesis on the artistic reception of the historical avant-garde in Eastern Europe in the 1980s and 1990s. She has also worked as an exhibition curator and a critic of net art. I'm sure I forgot something but I hope my introduction did her justice.

Inke Arns: It is difficult to show anything because it is a piece that is strongly process oriented. It is a piece by Dragan Espenschied and Alvar Freude entitled “insert_coin”. They were both students at the Merz Akademie in Stuttgart studying with Olia Lialina. They started the project in 2000 and it went on until 2001. It was a project they realized within the Merz Akademie as a diploma work.

Espenschied and Freude manipulated the proxy server of the school. When the people in the network are surfing on the internet their computers are

not directly connected to the other computers on the internet but mostly for certain technical reasons they are connecting through a proxy server. Proxy in German means "Vertreter". It is sort of caching websites so that your own computer does not have to contact websites directly the whole time. So you can imagine that such a proxy server is quite important.

What they did was to install a manipulated version of a proxy server on their school computer through which they could manipulate the content of websites. Everything the students and the professors of the Merz Akademie saw when they were surfing was completely manipulated web content. I tried this out the day before yesterday on my own website because there I easily can tell what I have done and what has been manipulated.

Florian Cramer: This is unmanipulated right now. Now I will change it to the manipulated.

Inke Arns: Wait a second. This is an unmanipulated website. Here for example you see *Neue Slowenische Kunst* - this a title of a book that I published. Now we change it to the manipulated server where we get the manipulated version. [...] What you see through this proxy server is not *Neue Slowenische Kunst* [New Slovenian Art] but *Neue Slowenische Schrotthandel* [New Slovenian scrap trade]. *Neue Slowenische Schrotthandel* is of course not equal to *Neue Slowenische Kunst* what I put on the website. Or it changes … you see the book *Netzkulturen* - net cultures. It was not published in Munich, it was published in Hamburg. Here you see Regensburg, *Showroom Ostdeutsche Galerie*. But on my real website it says *Museum Ostdeutsche Galerie*. What I am confronted here is my own personal website whose content has been changed heavily. Normally only I would have access to this website; no other person would have the possibility to change this.

You can find a very good documentation on their website. You can find a list of all the words that were exchanged by the manipulated proxy. For example the names of the former and the actual German chancellor - Kohl and Schröder, so that when the students of the Merz Akademie would surf the internet on major German news portals like Spiegel.de or Focus.de they would suddenly get very strange information. Even when they checked their email through web interfaces the content of their email was altered.

Question from the audience: What does *Schrotthandel* mean?

Florian Cramer: Trash trader.

Inke Arns: It changes all the words into the opposite. It changes *chancellor* into *emperor*, it changes *chief*, or *head of department* into *Obersturmbannführer* [a military rank in nazi SS-troops] and stuff like

this. Words like *violence* are changed into very positive words and so on. It is funny, when you experience this you get confused at a certain point what is real and what is been manipulated.

Actually this project was set in this whole discussion about filtering net content. There has been a very strong discussion in Germany about sort of *cleaning the internet*, making the internet free from pornography, child pornography, nazi propaganda and so on. The government [of Nordrhein-Westfalen, a province of Germany] in Düsseldorf is still trying to impose filters on the internet. What this project shows is that these filtering mechanisms can be used not only for filtering "bad" content but also for really effectively manipulating content of internet websites.

OK, I think that's it.

Just one remark: if you have any technical questions about the projects I think it would be okay to ask them immediately during or directly after the presentation of the project.

Question from the audience: When did the students in the school notice that their internet was changed?

Inke Arns: Oh, I forgot to mention. Nobody noticed. The concept was two people control 250 and nobody noticed. Even when they made it public and gave out a very simple instruction for the students how to change the proxy server to normal again nobody cared. It is quite shocking.

Question from the audience: They noticed when the proxy server broke down.

Inke Arns: It was working for several months. Then there was a technical break down and then the system administrator of the school noticed and thought they were intercepting secret information …

Question from the audience: Did they get their diploma?

Inke Arns: Yes, they did. I think so.

[Question unintelligible]

Florian Cramer: It actually was their diploma work. They entered it as their diploma work and it was the class of Olia Lialina, so they got it.

[Unintelligible remarks from the audience]

Remark from the audience: They even changed the logo of the school that was displayed during the diploma ceremony and nobody said anything.

Inke Arns: We should move on to Antoine Schmitt. Antoine Schmitt is sitting at the far right seen from your perspective. He is an artist and a programmer. In his works he is trying to address mostly fundamental problems like free will or happiness these kind of very emotional things. He is producing art works, programs and installations in which he is mostly creating abstract beings. One of his works I saw was quite irritating because it never really did what you were expecting it to do. So the work he does is quite irritating.

Laughing from Inke Arns and the audience.

Sorry, I'm talking about "Vexation I" which got a honorary mention at the first transmediale software art award in 2001. He is mostly getting his material from video games and artificial intelligence, artificial life. He is dealing with the algorithms of these works. He has shown work in many festivals, he received the first price of the net art competition in "Medi@terra" in Athens 1999.

Antoine Schmitt: Hello, I'm going to talk about an aspect of software art that has not been talked about yet. We have seen conceptual art, we have seen lots of social art, I mean art dealing with social issues. I wanted to say that art can also deal with sensations. That is what is called aesthetics in art history. I'm going to show one work that I think is representative of this aspect of software art that deals with sensations. It is a piece called "Etude organique", that means "organic study", from an artist called Alexandre Gherban who is Romanian but lives in France. He is an artist who has been involved in video and audio poetry for a long time. Now he is interested in programming as an artistic medium and material. That is what he states. He makes programs that are stand-alone and slightly interactive and that are designed to be experienced by a single user in front of his computer.

It’s a very low tech visual work graphically - programmatically actually also. It is designed in scenes like this. This is one scene. If I click on an element I go to another scene. If I do some actions it is not always clear what triggers what. I just clicked here and nothing happened, something else happens now. I clicked - something changed.

We see that the visual elements are very minimal. It's squares, lines and letters. There are various movements like going slow, accelerating, grouping, gathering, being attracted, repulsed. You have minimal actions: If I move the mouse something happens sometimes, sometimes not; if I click something happens sometimes, sometimes not.

All this is dealing with what the artist calls "a-semantics". That is everything that is happening is just below the level of meaning. It looks

like it means something, that it has some kind of reason for happening but it is just below. That is the important aspect what he says about his work.

I just play around some more. As you see sometimes things happen, you don't exactly know why. You feel that you have some kind of influence on what is happening. The movements and not the graphics are the thing that attracts our attention almost despite our will. At first you say it's just a cheap graphic thing, but then you wonder. It is quite strange what is happening and you try to find the reason for what is happening.

That is an example for what I feel is important in software art. Programs are mostly actions. That is one of the main and a very specific aspect of programming material. Actions can have some relationship to sensation. This is mostly done through what is called simulation in computer science - that is simulating some kind of behavior and it has also to do with real time. It is important to know that what is happening is happening here and now and has not been prerecorded. So simulation and real-time for me are the two key words for the aesthetics of software art. Thank you!

Inke Arns: The next panelist is sitting next to me. It is Florian Cramer who tries to connect his laptop computer which he build himself. It is quite heavy for a laptop but it is great and I admire him for building a computer all by himself. He is a lecturer in Comparative Literature at Freie Universität Berlin. He does research on literature, art and executable code since 1997. He is a free software activist with - as he writes - his brain attached to Unix and Perl. He has been a member of the jury for the first transmediale software award in 2001. He was also a member of the read_me software art jury 1.2 and - as was already mentioned - he is also a member of the runme.org expert group. He is co-administrating the rohrpost mailing list which is a German speaking mailing list for net culture in the broadest sense and editing the nettime unstable digest. He has published a lot of interesting things about code and code art and software art.

Florian Cramer: Thanks very much. I'm going to present what I already consider to be a classic of newer software art. The artist who created it is Adrian Ward who's sitting right here in the first row. I feel very humbled and am afraid to screw up! Briefly after that I will show you something which you might not think about as software art.

This... *He shows the first piece by Adrian Ward on the beamer.* ...is an example of a software art piece that relates very closely to what you experience and what you know as computer software. It's called "Auto-Illustrator" and it is a regular computer program for Macintosh and Windows. You can start it up and it looks exactly like a graphics program. If you are a professional graphics designer you probably know Adobe Illustrator or Photoshop - it's very close to that.

He demonstrates Auto-Illustrator.

Now I will try to draw a circle in this program. First it looks like a circle but it doesn't quite end up being a circle. *The circle executed by the programme isn't exactly circular.* I could even change the style and when I do that - oh sorry, I made a mistake - it ends up being like this. *The software has now rendered the circle as a smilie.* I could also, like in a normal graphics program, put some text into the graphics - here is the text tool. It also looks like a normal graphics text tool, but when I use it, it generates its own text randomly. So I have no control over the text. The software also has bugs - you know there are bugs, errors, in every software. But in this program, the bugs are graphical. If I use the bug tool and click here on the button, then the bug runs through the image. I also squish the bugs - debug - kill them by using the bug killing tool and then trying to catch them… Yes, I have stopped one bug, but the other one is still running, I guess.

OK, this is the graphics that not really I have generated but rather the program has generated. And it even has some fancy tools to manipulate it. If it looks to childish right now, I can use a filter like, for example, "Inspirational Instant Bauhaus-Style". If I use it, it looks like this. *The smilie becomes rectangular.* There are many other filters, like: "stupid", "pointless"… Let's use "stupid"… I could go on for hours. It's a complex program - it is as complex as Photoshop or Illustrator, but it doesn't behave as you expect such a program to behave.

This I think is the point, and this also is why we gave this program the transmediale software prize in 2001. If you use a piece of software, it is usually modelled after something you know. It has paintbrushes, it has buttons that represent something that you expect from a normal, non-digital tool. But who says that the software tool behaves the way it presents itself? If you have any piece of software and there is a button that says "save", you expect that pushing "save" actually saves your work and does not erase it. But nobody can give you the certainty that it doesn't erase it instead. So it is a pure cultural convention to rely on software as something that does what it pretends to do, and which acts like a tool you control and that doesn't control you by generating things on its own. So this piece of software shows you what software is about and who is actually in control of what you do with software.

Maybe it's enough to state that. For me this piece is a point of departure for software art. You take a given cultural notion of software and you criticize it and turn it upside down. This is where the creative imagination of what artistic software could be may depart from.

I would like to show you another art work totally different from that.

Cramer presents La Monte Young's "Composition 1960 #10" (October 1960) which instructs the performer to "draw a straight line straight line and follow it".

This is a Fluxus conceptual art piece from 1960 by the American composer La Monte Young, who is also one of the founders of minimal music. It simply consists of what you see here. It is a simple paper card and it says: "Draw a straight line and follow it."
Why do I consider this software art? It's a formal instruction. It is an algorithm which you can execute. It could be executed by a machine. You could build a machine that draws a straight line and follows it. But on the other hand it's something I would call imaginary software because a radical execution is physically almost impossible. You would need a machine that breaks through walls, that breaks the entire physics of the earth to draw straight line once around the globe to complete the piece. It is something that you can physically perform in a limited way, but actually only can perform in your mind. It is a code executing on your brain and not on a machine. That shows for me that computer software, or software in general, is not limited to machine execution, a certain hardware or the whole concept of computing as something involving a CPU and electricity, but that it also can be something fantastic, utopian, something that is almost literary, writing a code which runs in your imagination.

Inke Arns: Thanks a lot. I think we should move straight on to the last panelist. Next to me is sitting Alex McLean [http://slab.org/]. He is from London and works in the field of new media. He is - together with Adrian Ward, who is sitting here and who is the author of "Auto-Illustrator" that we just have seen - the founder of the "eu-gene" mailing list that started 2000. It is a mailing list for software art and generative art. He is also involved in the setting up/organizing of "dorkbotlondon". It is a meeting focusing on electronic art especially software art. I don't know, do you have monthly meetings or is it irregular?

Alex McLean: Highly irregular.

Inke Arns: Irregular. OK.
He is also involved in the runme.org software art repository. He did the design and the web programming. What I also want to mention - because he is going to present a piece of software art in the field of music - he is performing music, mostly together with Adrian Ward. They perform together as "slub" and also write generative software applications and perform them in real-time. I don't know, have you performed in Berlin already?

Alex McLean: Yes, last year at transmediale.

Inke Arns: Oh, okay. I think it's your turn.

Alex McLean: Thanks. Hello. I wanted to explore the idea that programming is an expressive medium. I wasn't sure how to do it [? unintelligible] talking to Adrian Ward he said I should write some software in the five minutes that I have to present some software. So I'm going to try and do that and see how far I go. Hopefully it will only take a couple of minutes. We'll see.

He opens a xterm on his computer and starts to write a Perl script.

Inke Arns *while Alex is still typing*: Sorry I forgot to mention he developed forkbomb.pl, that won the transmediale software award last year 2002. After that he developed a new program called animal.pl. Perhaps we can talk about it later on.

Alex McLean: See what it does now. *He is typing, executing, getting error messages, changing the script.* I made a bit of a mistake there. Got confused. *More typing, then a drum rhythm can be heard from his computer.* What I wanted to do is to draw a similarity between playing the guitar - strumming in your bedroom - and writing software where you can also play around not have a clear idea what you are going to start with. Just start with something and play around with it. So I could try now with a different set of samples.
He types and changes the script.
Maybe some gabba sample and some different parameters. *A new, more complex rhythm is heard with two different tracks.* I'll add a bit of *randomness. He is typing, changes the script. After a while new music with three different tracks can be heard.* Then I introduce a couple of other programs. *More Music is performed.*

Applause from the audience for the impromptu performance.

Thank you! It's probably more than five minutes.

Inke Arns: What an amazing presentation. I think we should start now with the discussion, but if you have any questions please make a sign and we will pass the microphone to you.

Question from the audience: In the last presentation where did the sounds come from? Was it from the program or was it sample-based?

Alex McLean: It was sample based, using software that Adrian and I use and have written. The idea is that we perform just using software that we have written ourselves. Adrian actually wrote the software sampler and synthesizer and I was choosing from different sets of samples that I was using to render the sound.

Question from the audience: [unintelligible]

Alex McLean: Yeah, kind of.

Dragan Miletic (from the audience, addressing Inke): You mentioned in the introduction also household appliances when people try to categorize what software art is. So I'm wondering how come that you chose only computer artists to present their software art. Also the panel looks very eurocentric. You know that in Asia - in Korea and in Japan - the web thing is hysterical especially at this point. How do you chose to forget about it at this panel [unintelligible] To me it is quite strange not mention and give credit to something that is so influential at this point in time in regard to software art.

Olga Goriunova: I'm not a computer artist.

Florian Cramer: To answer the first part of your question: I fully agree with you. Software art - which we define as work involving both executable code and cultural reflections of software - shouldn't be limited to computer art. A good example is the piece by La Monte Young we presented. "Draw a straight line and follow it" is not computer-based. In fact, I picked it just because of that. Still, what we presented today typically comes from the digital media arts camp. Much of it was made by artists who have worked with computers since a longer time, many of them have gone from working with software as a tool to create other material else to working with software as material. But even with a broad definition of software that goes beyond computer software, there will always be a strong coherence between software and computing, simply because today this is where most software runs.
At one point I acoustically didn't understand your question. You said something along the lines that our focus was a bit eurocentric or Western? … [answer from Dragan Miletic unintelligible] I don't rule out that we simply overlooked things outside our cultural scope. What we presented today were our suggestions; it is what we know and like at the moment. We may have our blind spots.

Olga Goriunova: Also about this geographical thing. The software art as a term is very doubtful - a lot of people don't like it. […]
If we think about creating the field of software art, because it still kind of doesn't exist yet or just starts to exist, we should think in terms of building it by ourselves. A lot of people here in the hall work on a software art repository runme.org. One of the ideas behind it is that not only people who made the projects can put them there, but also people who found projects that are interesting can submit these 'found works', too. So there is an open structure that can unite various practices from different locations and even times
The main problem is how to distribute information and get to know people who are interested in these things in Asia, in China or in Argentina. If their

projects appear in Runme.org I will have a chance to know them also. At the moment I just don't know them.
I found this first thing [the "stone-rock-scissors"-screensaver] that I showed through runme.org. I don't know where the guy comes from. […] I have no idea. So we didn't try to be eurocentric at all. I'm not from Europe.
[comment from the audience unintelligible]
Russia has a huge Asian part. I don't feel European.

Florian Cramer: A question that might be asked is why did we present all these works as software art? Of course, one could categorize what Alex showed as music or musical performance; what Antoine presented as interactive screen-based art; what Inke presented - "insert_coin", the censoring proxy server - as net art because it is related to the Internet, and so on. So what's the point of calling it software art?
Software is involved in any digital, computer-based art, but it traditionally wasn't paid much attention to and was put into black boxes. Take the whole genre of so-called interactive art, which is dominated by artworks consisting of video projections and feedback loops through motion-tracking video cameras. Such works are called "interactive video installations", without paying attention to the fact that they always involve computers hidden from the installation itself, with a software running on it that actually controls the human-machine interaction.
The same holds true for any other digital art, electronic music for example. Electronic music today is produced with software. But everybody talks about the music, nobody talks about the software and how the software also makes certain aesthetics possible - or impossible. I guess the common sentiment among us here on the panel is that we want to pull the software out of the black boxes and pay attention to art which does so, too.
Software doesn't tell anything about media representation: Software can interface visually, it can be auditory, it can work through any medium. But it always tells something about control structures and, if we talk about software art, about the reflection of control structures. While all the works we presented can be perceived as music, net art, generative video on their own, they have in common that they don't use software as a black box, but
reflect their programming and make statement about how software influences human perception and actions. Antoine, I'm not sure whether you agree with me because you took a different approach…

Question from the audience: I will try to ask something in English. My question is, what you were explaining now is […] is the idea of author. […] Of course there was software design, engineering and so on but it was not recognized, embodied by someone. It had not this recognition as [a work by] an author. But now there is the award, there is software art. I just read the other day a text by Anna Bosma on female voices in electronic vocal music. There was this discussion, I mean this issue on a

conversation with Berlioz who didn't recognize Berberian as a co-author and other examples where the singer was not recognized as co-author. To finish my question … do we have to go back to the notion of art, of author […] to present software as a painting or a sculpture in a place. I have the impression that all the [unintelligible] from being a fine artist to be what ever I don't know go back to this old secular [unintelligible] being recognized with your name, being in a museum, […] being published in a catalogue and so on.

Olga Goriunova: I can answer your question from my point of view. I see two parts in your question. Answer for the first one is that programmers for a long time were not recognized as creative people and they were never given any credits and in this sense, yes, software art is an attempt to make a revenge. Yes. But on the other hand it yields a specific result. The code statement is very subjective. I understand it as subjective because it is culturally shaped, socially shaped. But for many it is also an individual statement: good code demands high expertise and talent, which are individual qualities.
There is the danger that we come back to the idea of self-expression, of genius and of all that stuff. I see it as a danger. I don't like the movement towards this.

Antoine Schmitt: Well, there are many things I want to say. Yes, it is true that nowadays programmers are becoming more and more recognized as authors. But an author is not an artist, that is a bit different. I'm not afraid by people using programming as a material and saying it's art. I think that since Marcel Duchamp everyone who says "this is art", then it is art and that's all right. You can like it or not, you can agree or not, but you have to consider it. Somebody uses programming and says: "I made it with programming and it is art"? Then you have to accept it and say: "Well, why not?" And then ask yourself "Do I like it or not?"

Amy Alexander: I'm going to show something briefly so I'll just speak from here. And I apologize because I couldn't hear all that just transpired so I could be slightly off topic.
I think there is two parts to this. One is "Is it art?" and the other one is "Why do we care about software?" The "Is it art?" question is, I think, sort of the repetition of "Is photography art?" You know that question came up in the early twentieth century with painters and so on. Every time there is a new medium that uses a new technology there is the question of authorship and "Is it art?"
The part I am more interested in responding to is why is software relevant? Why do we care if we have something that is already interactive art, why should we care about focusing on this software? The thing that came up in our jury discussions at transmediale this year was the idea of software coding being a subjective text. We keep hitting on this issue. The

one thing that I will bring up is not really software art at all, but it is my favorite example so those of you who have seen it I apologize. Something like Google technology - Google is of course the biggest […] search engine in the world that tells us all what to think … of course it is the universal propaganda tool. One of their marketing gimmicks is that they are very fair, they don't take advertising, they don't take pay for their top results. [shows "About Google" site] They present this as … - the red highlights are mine - "the way our ranking works is just technology." PageRank is simply the algorithm that they use to put some results higher and than others so that you can see some results and not others. For example they say that their PageRank technology relies on "the uniquely democratic nature of the web": sites that have more links to them from other important sites get higher rankings. And "Google's complex, automated methods make human tampering with our results extremely difficult." Okay, [joking] so there are no humans doing the programming at Google, it's just some machines that do it themselves and that's how it stays objective, I guess.

Brings up on-screen graphical diagrams representing the linking structure between various popular websites.

There is some diagramming software that you can use to create diagrams of links between highly ranked websites. If you look at Microsoft, they own a bunch of different domains that all link to each other. And lo and behold if you do a Google search for Microsoft, all of their domains are highly ranked and link to each other. I apologize if this is not completely accurate, these two diagrams were done at different times. But this basically shows you that they have this little internal network and it keeps their results high. If you look all through the Google results for Microsoft you won't see any of this stuff at all [shows a page that is critical of Microsoft] even though it uses the word "Microsoft" and it's about Microsoft.
An algorithm is used in an entirely biased way but according to Google it's just technology and doesn't involve humans. So one of the things in which we were interested, in the jury, was this idea of bias, what is the text, let's look at the algorithm. We thought software art is really useful because software is so invisible. It's a black box, not just in software art, but it is a black box all over the world whether it's Google or the way your toaster works. By giving us a discourse to examine the bias in software, we can then start to think about software in the larger culture. For example the Suicide Letter Wizard gives us something to talk about, not in terms of how can we talk about software art that make suicide letters, but what are the implications of Microsoft Word, what kinds of text does it encourage, what sort of assumptions does it make? It makes these "you are a corporate drone"-kind of assumptions. This is something the jury was interested in, how can these examinations of software art lead to examinations of software, its biases, its seeming transparency which is a farce.

Sally Jane Norman (from the audience): I have a question that actually you mentioned earlier, Florian, in your introduction but you being very democratic and giving us the floor but it would be good to hear your viewpoint on literacy. How, what kind of publics, audiences, fellow artists, peers are being addressed by this different kinds of works.
And - this is about the other question that was posed about eurocentrism - I didn't feel at all that you were claiming to present anything exhaustive. It is really good that you presented very specific things that we can bounce of. But for exactly those reasons something like the "Suicide letter" requires a certain kind of literacy with a certain kind of net in our practice. Somebody is going to get off on Alex's music just [on getting the] rhythms just like I do if there are [unintelligible] and I don't know anything about programming except that the code looks good coming up on screen. Somebody else with very good code literacy is obviously going to get a very different level of enjoyment.
I did a test on a few student a few weeks ago with Antoine Schmitt's "Avec determination" and I had three students with three totally different levels of expertise. Their analyses of your piece were completely, schizophrenically different. I just like to hear your viewpoints on this.

Florian Cramer: This might be the time to disclose our own source code of this panel which says: presentation first; then the question "why is software interesting as an artistic material?"; third, "what is the audience for software art?" - which you just pronounced - and finally, "who can make software art?", which is also important to ask.
I totally agree that most of what we presented as software art may be difficult to understand for somebody not socialized in computer culture. One can appreciate the "Suicide Letter Wizard" better at least if one knows Microsoft Word and the semantics of its user interface - to give just one example. On the other hand, this applies to all art. If you look at the "Man with the Golden Helmet" not far from here in the Berlin Gallery of Painting (which we know today is not by Rembrandt), of course you can look at it as simply as a man with a golden helmet. But if you are an art historian, you see the painting differently because you know the cultural background, the iconography and subtle references encoded into it. The pieces we showed work, it seems to me, in a similar way. Somebody who doesn't know software at all can appreciate the "Suicide letter wizard" as something that generates funny suicide letters, somebody who is deeper into software culture can appreciate it as a reflection of how software makes you write and think in a particular way, and the anthropological presumptions Microsoft makes about "users".
Therefore there seems no simple solution to the problem you describe. But I'm optimistic in one respect. As Inke said in her introductory statement, software is pervasive and what we see on our PC screens is just a small part of it. My favorite example are bank accounts. We all have a maximum credit line on our checking accounts and that balance is calculated by a piece of software. So everybody who has a banking

account is dependent on software in a nontrivial way. Not to speak of the software which calculates your income, your tax, your social security, your health plan, or even your medication if you're sick in a hospital. So being a software user is not a matter of owning a PC or not. So I expect software to get increasingly more cultural and political attention. The more pervasive it becomes the more people will hopefully reflect on it and the more important it will also be as an artistic material. - You don't think so?

Sally Jane Norman: I wonder if you can't see the wood for the trees, you know? You know this expression?

Florian Cramer: Yes.

Sally Jane Norman: How many people can describe the principle of a combustion engine because they drive a car?

Inke Arns: This is also something I would also strongly doubt the assumption that, you know, the the more there is software gets employed everywhere, the more people will reflect on software … This is what I tried to say in my introductory statement. A more pervasive use of software will not result in an increased reflection on software. Exactly because software is so invisible .... people in most of the cases don't know what is behind that kind of black box.

Olga Goriunova: I disagree. I think it depends on the amount of time people spend with software and how widely it is used. Of course, if it is just everywhere there are higher chances that it will be reflected and criticized.
I can speak about myself as an example.
When I bought this computer it came with this cheap version of Microsoft Office - Microsoft Works, which actually contains maybe 100 different templates for all kinds of things that you might need in life like a marriage planer, sports calendar, letter to a friend who is ill, whatever, just 100 templates. They were so completely stupid that I got really mad - and made SLW. It's the result of direct impact of this pervasive software on me. I didn't need Microsoft Works, I didn't look for it, the software just came to me packaged with hardware.
Another question is critical attitude. There is a whole trend in sociology called sociology of technology, which studies how society shapes technology and how technology shapes society. It's a critical discourse that raised once grows bigger and bigger. If now everybody knows what culture industry is and what advertisement does to us, it's due to things that were done in the 40s and 50s.
At last, about art literacy. It would be a bit hypocritical to pretend that there is art that is easy to perceive for everyone. People in many, many countries will never ever see a piece of video art. And why should they? It's maybe easier to perceive than software art, but the cultural, economic and whatever circumstances are so different that we can hardly talk of

presentation or production of video art in …, let's say, Togo. Even in western countries, just as Bourdieu wrote, perception of art is a question of education and social origin.
…of course software art is very closed in a way but it is also very open in another way.

Alex McLean: Software art for me is not a very new thing, because programmers hang together in communities. Especially since the internet has become popular more programmers have been hanging together in communities and they started making art for each other. I guess that kind of art you'll never be able to comprehend unless you are a programmer. I guess the first software art was made in MIT […] … some model railway club hackers managed to get access to a computer and started playing with it, making it make music and all these kinds of things. Then there is software art like the "Game of Life"-people who make very simple models of life and play around turning them into computers within software [I don't really understand that] and so on.
Now software art is still emerging - it might not be called software art - on websites that form galleries like freshmeat.net when it first came out or there is [unintelligible] and sweetcode.org which is a nice website for open source software that is innovative. You get some really interesting pieces there. The Acme Namespace for the Perl modules that Amy mentioned earlier is another kind of art gallery. It's where Perl programmers put all their code, that is mostly completely useless, but it is interesting in very varied ways - sometimes funny sometimes just quite mind bending.
So I think this kind of art is always existing. It will be one starting point to try and find this software art and document it rather than try to invent software art. There are other starting points like experimental musicians making processes very similar to software and this kind of thing.

Question from the audience: My checking account is a disaster in economic terms but maybe if I try to look at it as a software art piece it maybe helps.

Florian Cramer: Well, I didn't refer to it as software art but simply as software! Software that controls your account balance and your maximum credit - an example of how software determines your life even if you don't own a computer.
But I disagree with Alex about what he just said, at least partially. Alex, I think you're right that software and anything that is connected to artistic reflection of software may be strongly conceptualistic. We historically know art which approaches formal languages, mathematics and formal music composition, a history which neither started with the concept art of the 1970s, nor with the constructivism of the 1920s, but is at least as old as Pythagorean aesthetics. But I think conceptualism is not enough. Today, we experience software as a cultural phenomenon. If we see a web browser window and identify it as software, then software has a certain aesthetic. Software has thus acquired an everyday meaning or

connotation which is not formal. Therefore, if somebody would make an oil painting of the web browser window with the Google site - which Amy just fired up -, as a reflection of our time and what happens when you enter the search "Microsoft", then I would have no problems to think of it as software art.
One and half year ago, the "Browser Day" festival took place here in Berlin, where somebody presented a "browser" that was simply a wooden window frame. The idea to think of not of the browser window as a metaphor of the conventional window, but of the conventional window as a metapher of the browser window. You should carry this frame around and browse the world looking through it. For me, this is software art. Software becomes an aesthetic paradigm here through which things can be differently perceived. It doesn't have to be coded as computer software or run on computers. But, as an algorithm or script for perception, it even has both aspects of software art we defined in our Moscow read_me 2.1 jury work; it is formal instruction code and it is a cultural reflection of software, done either in instruction code itself or using a completely different material.

Amy Alexander: […] Back to the question of audience. I like software art because you don't have to go to a gallery, you don't have to know about art history. And sometimes they write about it in Wired magazine and even I can understand it.

Antoine Schmitt: Well, sometimes I'm wondering if we are not in the history of software art just before the renaissance or something like that. The moment where Gutenberg invented the printing machine. Before Gutenberg there was this great technology which was writing and reading and who had it, it was the church. They also had the power. It's a strange analogy with computers that the religion of today is business - money. Who has the computers and the power of the computers and the mastering of the computers? It's the business people. Now with free software and people interested in programming it's getting out of this corporate business machine. People are using it now. Lots of what is called software art or what we see in software art festivals, is actually people getting a different use of software, realizing that software has been here for 50 years now and that it has been controlled by big corporations in hidden ways. Everybody is just looking at that and saying "Let's fight against it!" I think we are at a moment before software is used in a creative way. Today it is used in a critical way to look at society. But maybe it is going to change soon, because the audience is going to change also. Right now it is very often puns and little jokes about actual software as it is today or criticism of how it is used today that is in a way not very con… it is constructive because construction is constructive …

Laughing from the audience.

Olga Goriunova: Sorry, I won't answer your statement, because I'd like to support Alex. I think - and we wrote it in the runme.org site - that software gets its lifeblood from programmers' cultures. That is why it is interesting, because it is connected to the living cultures, the folklore cultures in a way. At the moment it is almost the only art practice that is connected to some live practices that exist outside of the domain of art and there are many people engaged in these activities . Though I don't understand much of it as I am not very well acquainted with programming languages, I can also be quite happy on the borderlines or margins of software art where code is not that important and where you can play around with different things - concepts, cultural implementations.

Florian Cramer: Then you both refer to software art as a particular social practice, something with close affinities to Free Software culture, to hacker culture and the like. For me, it's not only that. I rather see the term software art as a particular perspective from which to look at art. Just as, for example, looking at La Monte Young's "Composition 1960" as software art seems be feasible and justified. Let me put it this way: there might be an artistic software culture connected to a certain self-referential and ironical programmers' culture, as it is for example described in the Hacker Jargon File maintained by Eric S. Raymond. On the other hand, I see software art as a generic term and not as the name of a movement or network of artists. I would never suggest that software art historically replaces net art, or interactive art, or generative music. As said, you can look at a piece like "insert_coin" as net art and software art at once. The terms are not mutually exclusive. Just as, since the 1960s, people have called La Monte Young's composition minimal music, a Fluxus piece and concept art, without these categories being exclusive either.

Ryszard Kluszczynski (from the audience): It's not a very important question, but I'm curious. What do you think about the relationship between software art and conceptual art especially when we think of software art as something belonging to the field of computer art?

Florian Cramer: I will try to be brief! Yes, the correspondence is also historically evident, because the very first concept art exhibition, curated by Jack Burnham in New York in 1970, had the title "Software". It juxtaposed concept art installations to computer software. So the link has been made in conceptual art as well. But both fields do not fully overlap. For example, I wouldn't call the "Suicide letter wizard" and Alex's music concept art.

Alex McLean: I wasn't trying to say that all software art should be about programmers, but I was just saying that it was the starting point of many. We should take all this starting points in consideration and see where it takes us.

Inke Arns: Perhaps to get back to the initial questions. Florian already mentioned that we actually found it quite interesting to try to focus on the audience and the producers, the people who do this kind of software art. When we had a meeting two days ago we repeatedly talked about […] the exclusivity of this kind of practices. I would like to ask the panelists here, who participated in juries over the last few years, who has been doing software art in say 2001, 2002 and now, how did it change and also perhaps how the audience is changing. Is there any change?

Amy Alexander: The first one: who is making software art? It seems like there's two camps and that's a lot of what we've hit on: people coming from the programming culture [and] people coming from the art culture, interactive art. One thing we commented on in our jury statement for transmediale this year [was, that] since it got genes from both DNA's - software and art - maybe the parents are trying both too hard to mold the child onto their image. [But] it seems to be starting to develop its own identity. Maybe art festivals like transmediale mostly get entries from the interactive art side of things and maybe not so many from the programmers' side of things. That is one of the things with the runme-project: that we are trying to bring together these two cultures not in a way of "Look, we've got programmers and artists" but so that they both blend and combine a little more.

Alex McLean: I'm not sure if anyone is really making software art apart from people who are directly trying to make things that win festivals. I think the people making software art are the audience here looking at software and thinking "Hey, this is art. This is software art." I don't think there are any programmers making it.

Amy Alexander: You think they're not making it or they're not calling it that?

Alex McLean: Okay, they are not calling it that. They are making nice software.

Statement from the audience: That's the problem perhaps that art is just a label whereas software has always just been software.

Diana McCarty (from the audience): I missed Sally's panel about the crisis in interactive art, but actually I wondered if it is not more a crisis of the jury. You know when you speak about this work? I know that in the first software jury award at the transmediale the projects weren't submitted that the organizers had in mind, that would fulfill the requirements of the festival. That's one of the things that happen. So in spite of this gigantic networks - 4000 people can submit projects to a jury - what you have is a … there is kind of this idea that it is very open and you avoid an exclusive situation but I think you go back to the situation that the names that are known are the ones who get selected. Nobody has

the time to go through 50000 or however many applications within any kind of time. Sally, maybe you could also join the panel for 30 seconds?

Amy Alexander: We really only had 43 people enter this year at transmediale. Could you guys send in some work, please? No, really only 43 people, so we looked at the projects. I think you're right, it is a crisis of jurying. Again one of the things with runme.org is making it easier to enter, making it so you can find projects and put them in. Because who has time to fill out these forms. Two people on the jury, if not all of us, heard this from other people [who were] asked about entering festivals: "I'm too busy programming, I don't have time to enter." Yeah, it's true it is a pain in the ass. You have to write a statement, they want pictures, if you are not in Germany you have to mail it internationally. Who wants to do this? And if you are not from the side of things that likes [to call] what you're doing art, it's not even that enticing. I think it is a crisis of jurying. It doesn't particularly interest me too much because there are other ways of bringing software art to the surface.

Alex McLean: If the audience is making software art then I guess it has to be more of a culture that is centered around this idea of software art.

Florian Cramer: My experience is the same as Amy's. In the first transmediale software jury we also had about 40 or 50 entries, which was only a fraction of what - for example - the video got for review. A considerable amount of the software entries was institutionally funded so-called 3D interactive virtual spaces which weren't even entered as computer software, but as video tapes. In comparison to that, I think the situation has improved today. But still: We called this panel "Software art - An artistic perspective or a curatorial fiction". I have no problem if it is a curatorial fiction. As I said it's a perspective, a way of looking at things that might not always match the self-perception of the artist. As critics or as the audience, we might perceive something as software art while the artist him- or herself never thought of the work in this term. I don't have the slightest problem with that.
Take a term like "impressionism". Impressionism was a coinage of art critics in the 19th century. Impressionist painting didn't exist as a context that artists themselves created, but it was a creation of critics. Nevertheless this made sense. I have no problem bluntly saying: Why not create something which is interesting as a perception of an audience or of critics although artists do not equally identify with it.

Diana McCarty (from the audience): Maybe I will just go back. What I mean with there's a crisis of the jury, is that people that are doing the work that would fulfill whatever the show is about are not always the ones who submit even if the work exists and that is maybe the work that's in the mind of the organizers. So maybe … one of the funny things about net art, maybe you remember discussions about it here - not in this room I guess but [in Bethanien] from the other net art discussions - there was

this early euphoric moment when net art attempted to get away from the curator but maybe it's time to escape from the juries.

Laughing.

Florian Cramer: Yeah.

Olga Goriunova: Yeah, but we are doing that. In the second read_me [festival] there will be no jury, no prizes, just people who will be featuring works that will be freely accessible in an online database.

Florian Cramer: I think there is one important difference to net art, because net art, if you spell it "net.art", in my perception was a movement. It was a movement of certain people in the mid 1990s who then had a group identity. The catalogues of Jodi and Heath Bunting layed out here in his room are one distillation of that identity. That identity doesn't exist in software art...

Olga Goriunova, *interrupting*: I disagree, whoa, come on. *Laughing.* I don't want to go deeply into this isms-quarrels and actually I don't care if software art can qualify as a movement or can't. But I think it's more about personal connections, about subjective values - it doesn't have to do strictly with formal aspects of particular art projects that make us capable of uniting. I would like to identify myself with software art for a while. I think the community exists, and at the moment I feel myself a part of this community.

Pit Schultz (from the audience): I think we can agree that software art is one of many genres of media art and insofar suffers from the same problem of definition. One could possibly agree that every art is media art and every art form uses some media, but defining arts through the medium you go into a classification of art which is rather premodern almost. Like using Kupferstich [copper engraving, etching] or … defining a work of art through a medium is a very reduced form of describing art. So if you look today for example at the history of video art which went from all the stages from being a movement to going to the museum we see now a whole generation of artist who are called artists who use video but they are not called video artists. And they might even do different things, mixed media, they use also video art. And they use software. Liam Gillick for example uses software but he is not a software artist. In his [artistic] process software is a very important reference or tool, it's part of the work. But it's not part of the presentation possibly.
Software is insofar, as we all use it, as you said with the ATM, Geldautomat, it's part of our whole culture. Then we enter the field of software culture, the cultural phenomenon of using software, we go into the Perl communities, the different tribes so to say. Then you have another phenomenon where media art is very known for, which is mediating this kind of tribal cultures. It is this kind of almost

anthropological view of the otherness of the original cultures who speak in tongues and have strange rituals, and [media art] mediates it to the general audience through festivals like transmediale. Then you have another kind of translation error, as you know from the discourse of anthropology, the constructiveness of the view to the tribe creates the media art itself. So media art becomes an effect of the need of mediation between the new technologies and the … [?] on the society to explain the cultural field what's going on.
So we have media art as a product of mediation rather than any kind of artistic practice. There are basically a lot of discursive problem which are circumvented I would say through very pragmatic responses like runme.org. By experiment, by basically empiricist factors you can show or you can falsify there is software art. So after a while you might see, okay, the work is substantial or not, but the discourse is from there like a bottom up process and it uses, speaking again of runme.org, strategies from many sites on the net, which are working, e.g. download.com. It's not an anthropological approach, runme.org uses categories more in the history of other download.com projects.
So it's more like a statement not a question.

Florian Cramer: But it was completely intentional that runme.org should be made like freshmeat.net or download.com, as a reference to existing software culture. The idea was to create a download.com of software art. You might consider this absurd, but I find it quite funny. Why not make it? Otherwise, concerning what you said about defining art through a technology, yes, of course you're right. But again: Why not? I don't see your position and the position of looking at art by its technology as contradictory. As I said, it's a perspective and a particularly interesting perspective, because software is a technology with a higher political impact than copper prints.

Pit Schultz: I would really disagree. The use of copper prints in the 18th century had a high political value because of its use for pamphlets or all kinds of communication for people who were not able to read. So it had a very political use. And of course in the 19th century there was no software so you have to look … it's a bad comparison of course but …
But reacting to that, I think, software is something that needs social theory so you have to go to Luhmann and to Kittler and so on. [Maybe] then, from that perspective, art, artistic practice, might be relevant in some ways. But to say just because some people use a certain medium we already have [a new genre] … of course there is a mechanism in media art which is always progressing like that. We have the internet and we get net art. There has been net art but net artists used that kind of need in a very reflected way to play with the market of the demand of net art. And created an artist movement that was basically auto-destructive in the history of these strategies by predicting their own death and saying, we know about our problem, we don't try to escape the museum, we know about all the avant-garde. We use these strategies in a reflective way. I

don't see it yet in the software art movement. There are lots of pockets of different cultures I would say, what Alex also said, and it is interesting to see more from a cultural perspective. But I don't see the art aspect of it, because what you said with the art aspect, we have the whole history of software art and computer graphics [unintelligible] and we have any interactive art piece had a hidden software piece in the back, Peter Weibel and so on, and so all this things, as you said is a perspective on things … as a kind of procedural approach I would fully agree but as a whole own art category I don't see that yet.

Florian Cramer: Just two objections from my side. One is that software is not a medium. Why is software not a medium? If you define "medium" in a reasonable way, then a medium is something between a sender and a receiver. Software is not only between a sender and a receiver, but it can be itself a sender and receiver. It's a process.

[Peter C. Krell's objections are made without microphone and remain unintelligible.]

Yes, I know there is a totally inflated notion of "medium" today in media theory which deliberately avoids to clarify what it thinks a medium is. Today, medium has become a substitute for what has previously been called "sign". But a medium is not a sign. If one reasonably uses the term "medium", then it means a communication channel, something which transmits or stores information. But then software is not only a channel because it is generative and processual; it does not only store or transmit signs, but it generates and interprets signs within the constraints of a formal grammar.
That's the one thing. Now I forgot the other - Pit, what did you say last? … Yes, interactive art and process-based art always involving software. Yes, maybe it involved software but it didn't reflect that it involved software. And that's the crucial difference. That's the difference, and that's why I'm not interested in Peter Weibel or Jeffrey Shaw because their art doesn't reflect the fact that it is driven by software. For me, this is also the qualitative difference between the art we presented today and a piece by Peter Weibel or Jeffrey Shaw. If a category "software art" helps us to describe such a difference, then this alone justifies it.

Inke Arns: I would like to jump in, just a short remark. I would also agree with Pit that you should always ask yourself if you are not sort of creating the phenomenon by finding a name for it. But in this case I would agree with Florian, I consider this term of software art as a kind of heuristic category. It's not an ontological term or anything like that. It can be a helpful construction. Of course it's another question how it's used and employed by people and how it is used to construct something.

Olga Goriunova: May I also make a little comment on the problem of software art as a new art movement? I don't think there will be software

art as we know it in two years. I mean at the moment it's kind of zone that allows people to freely and constructively exist - and that's enough. I don't see why it should create a lot of critical discourses about its own death. Of course it will die. So?

Antoine Schmitt: I don't agree at all. I think software art is going to be, maybe not as a category but maybe as a category also, alive for a long time.

Diana McCarty: I think that's very brave of you to say, Olga. But I wanted to say, Florian, you made this argument earlier that software art is something that can … also I liked what Amy said very much about reasons for making software art but if you look at the trend it's actually about, I don't want to say disappearing software, but if you look at the direction that Microsoft is going and that's the main operating system and software developer for most of the computer users on the planet so it's kind of important - and basically it's about making very fuzzy lines between the software and the system.
So for the general public that line is not very clear. The computer, the operating system, the software - those things are not very clear. What the difference is, where the lines are between those things, where the lines are between the software and the hardware. The direction is more towards convergence for the general user. It was funny when we had this discussion with Lev Manovich within the Kittler seminar because I think on the one hand we are talking about theory and praxis but I think there is a difference between the theory - their kind of academic theory -and the praxis of say bootlab and the developers and there's a completely different praxis which is the general user or the main public. For them these divisions are not there so much.
Florian, I think you argued against yourself, because you said that on the one hand what is important about software art is that it brings the software to attention but the trend within the market is going exactly the opposite way. But at the same time in order to access the work that's produced by software the software itself is the medium. You can't access the work without the software and therefore it is a medium.
[…]

[A short passage here is missing due to recording problems.]

Statement from the audience: […] the essence of it is the interaction and what's that really meaning? Is it really meaning still to work with the material or isn't the material something really immaterial? How we are thinking? I just want to open the discussion on what comes next, so we don't circle always around this categories. In my view they are wrong they don't fit anymore.

Florian Cramer: I think it's easy to agree that categories are always crap but you need them. That would be my pragmatic answer to that.

But you both now have come up with an interesting contradiction whether software is something that is bound to material or whether it can be immaterial. This is for example what Friedrich Kittler says: "There is no software without hardware". I disagree and my example would be the piece by La Monte Young "Draw a straight line and follow it". Software can be something that executes in your mind purely.

Peter C. Krell (from the audience): Your mind is the hardware.

Florian Cramer: My mind is not a hardware. Excuse me, excuse me, no! Then - that's mechanism, that's a caricature of seventeenth century thinking. My brain is not a hardware, excuse me but I can't take this seriously.

[Unintelligible remark from the audience]

Florian Cramer: But hardware is not identical with material. Sorry.

Peter C. Krell (from the audience) [without microphone]: […] extending also the text from Friedrich Kittler ready to think of it as being an approach to capture a being, a physical entity as a whole. I think this would be exactly where the critique of you would be too narrow to say that the category hardware would only describe the technological. I think when we talk about material we are also talking about... [unintelligible, without microphone]

Florian Cramer, *interrupting*: Okay. But then we don't talk about hardware in the sense of the English term. A "hardware store" is a tool store. Then you use a notion of hardware which would more generally mean "material". If you talk about material then I would agree. I think every English speaker would say that you are crazy if you say that your brain is hardware.

Inke Arns: Sorry, but I think Henning Ziegler wanted to make a statement.

Henning Ziegler: You took the discussion somewhere else now but I was going to point to Dieter Daniels thesis or his example of the wire that was laid from Europe to the United States which is just a wire probably to you, Florian, but it seems to be inlaid with cultural and political meaning. The intention to wire Europe to the States first of all was a financial interest and then the wire is not a cable anymore. So I was trying to point this out. The social and political decor [?] I found that very interestingly illustrated in this wire example. I totally did not know about this till yesterday.

Antoine Schmitt: Just to talk about the metaphysics of … but if you mean metaphysics in the sense of Aristotle, "beyond physics", that is

something that is not hardware, maybe that is that what you mean, I don't know … But I think programming programs - that's what I'm advocating for a long time now - programs are a really new medium for creation in general because unlike other media a program exists here and now and it makes decisions and it has internal states. It has a before and an after and this is very much like beings in general. I think programs are a really good way to reflect on the nature of reality, of nature itself, of human beings in a new way. For me the metaphysics of the program lies here and it's really interesting.

Golo Föllmer (from the audience): I would like to ask something about what Antoine just said. Bob [O'Kane] last year in the jury put it that way and said: "You can never look into a piece of software, that's the point. Even if it's very simple, in fact what is happening in there you can only guess." This is the point where people last year started to think of software as something metaphysical and I think you could put it parallel to what Sarat Maharaj in the keynote lecture called the "indecidables". That art is staging and representation of indecidables - it wants to state that and it focuses on that. I think this is the idea - maybe only in the curatorial view - that there are people working with this point in software.

Statement from the audience: I think to think about the metaphysics of software art - I'm not saying there is no software art - might adapt to the entire thrive to render an intellectual property which might be a paradigm which we might also call software whatsoever art. I think art draws its juice from also undecidable questions, for instance like a metaphysical one. And this is all I wanted to really add to the discussion. Not to talk against it but to be ready to also address these kind of questions if we want to have the formal debate of a phenomenon we want to describe somehow maybe with a captured term like software art, which I think is fairly nice.

Florian Cramer: Regarding my own internal metaphysics, I have been thinking about Diana's point all the time. Even if it might be problematic to see software as a medium, it translates into media as soon as you can perceive what it does, in the effect it creates. To use the example of your checking account again: the medium you perceive the software in is the money you have on your banking account, and not the software itself. That's one of the problems of dealing with software. It's an operation that generates effects you can perceive, and this operation can be translated into the medium of sourcecode. But the operation itself is not a medium.

Sally Jane Norman: I think this where as Diana said maybe you are saying some really useful things but shooting yourself in the foot, Florian, with some things you said earlier.
Because one means [?] the question about literacy I quite agree that there is no reason for something that we are calling software art to be suddenly a big revelation about the universe and everybody should

understand it instantly. As Olga said, we have strong cultural coding that enables us to access all different kinds of human artifacts and software is a part of this history.
But you're saying that because there is such pervasive use of software, of programming, then literacy levels are going to be rising and I'm saying that's probably precisely why they're not rising. I think one of the big problems that we're having, one of the paradoxes and interesting contradictions of this thing that we are calling software art is that it is a merging in an unconsecrated space in a very pervasive manner, so there is a ubiquitousness about it that is making it look as though it's everywhere. What we're doing is boding this category called software art is to suddenly make explicit […] some very interesting specific features, cultural features. We are using this category to try to recognize, identify and talk about what this may be capable of expressing, that other forms of communication and other media haven't yet been able to express. And this is why I liked the question this woman just asked about how it's making us think. It's one of the processes, what are the new ways of thinking that this programs are inducing in us, once we wake up to the fact that they have aesthetic and communicational specificities.

Florian Cramer: I think you're right.

Sally Jane Norman: And I'm going to shoot my hardware in the head.

Florian Cramer: No, you're totally right criticizing me on this initial assumption I made on literacy.

Pit Schultz: I just have a question for Alex. Do you think you could do these things with php?

Alex McLean: What things?

Pit Schultz: Those things … like deep programming … I mean, could you do forkbomb with php or with basic or another programming language?

Alex McLean: I guess it's possible but I only use Perl.

Pit Schultz: Is there some kind of inherent - it's a kind of rhetorical question - [quality of programming in the different languages?] You know about the culture of Perl and the freedom it gives to the programmer on the level of coding. It seems to me if you look at the culture of php or Visual Basic programmers that it seems to be more possible to use Perl to create a kind of aesthetic value out of it than with Visual Basic or even with Java. I mean every programming language has its own in-build culture so to say and this is also something that we learn through its uses that software doesn't have to be always like this completely industrialized, completely pragmatic product. It has cultural vectors or potentials. But

they're getting explored and discovered and then presented in this software art [?].
It's more like adding to what was being said before, that if you go in a precise local mode then it can become meaningful but the more you get into the view from above, the generalized view, it means everything or nothing. That we have very often in this media art discourses I think.

Alex McLean: There's a lot of computer languages that are very similar to Perl. I think php is very similar to Perl. I think, it's maybe the first to draw a lot of influences from a lot of different languages, there's lots of different ways of doing different things. Perhaps it's easier to reach the state where you're not thinking about the language at all you're just thinking about what you're doing with Perl. But I'm sure that's true of any point that you reach where you don't think about what you're doing in specific and you can just do it. I'm sure there's C-programmers who know their craft so well that they don't feel any boundaries in what they're doing.
Then I guess there are always boundaries that you might not actually be aware of because as a programmer you do think in terms of the language and to some extend you are ruled by the language. The first time that a program is run is, I guess, in the mind of the programmer so in that way you are thinking Perl if you're a Perl programmer. So it has a very strong influence on your thoughts. Maybe that has also to do with the metaphysical question, although I don't really understand it too well, in that if you're running this code in your head before you're programming it and then you're introducing it to an operating system where it's run then - an operating system is also software which someone thought - then you're looking at a system which is just a collection of people's thoughts. Is it a medium or is it an environment? I don't know. But I think it's an interesting place to work.

Question from the audience: I would like to ask Antoine, when you showed us the black boxes and the A-letter you said that the action is below meaning. This really struck me but how do I get this information?

Antoine Schmitt: What's the last part of the question? I mean …

Question from the audience: You said the action is below meaning … and this is really, I mean that's a bomb. But if you only see these movements, they are so random. So how do I get this information that the action you see is below meaning?

Antoine Schmitt: Well, I talked to the artist and he told me. That's his point of view.

Question from the audience: So only you know this?

Antoine Schmitt: Well, I first saw this piece when I looked for pieces on the internet. I looked at it and I really liked it. Actually that was how I defined it, that it is just below language, I thought. All the actions I saw, all the movements looked like they had some meaning but it was just the beginning of the meaning but not the end. Just below the meaning. Like if you see two dots going to one another and stopping, this is some kind of finished action. It has a meaning: they were attracted and they stick together and you can understand it. But if they seem to do it and at the last moment they change or they do it for one second and they split, it looks like meaning, it looks like … meaning something, but it does not really. He plays with this on many different levels both at the graphical level, where you have letters but not words, and on the movement level, where you have semi-movements, and also at the level of interactivity, where what you do sometimes has an effect sometimes does not and it is not always understandable. But it is not random, because randomness is white noise. I think it is very important to think about the shape of randomness. You can have totally random actions or movements and you can have semi-random. Life is semi-random. Lots of things that are random happen and still things have a shape. This shape is interesting and it's an interesting field, the shape of randomness.

Florian Cramer: One point concerning Pit's and Alex's debate about the style of the programming language influencing what you do with it. I would say the question whether a programmer writes in C or Python or Perl is just as relevant as the question whether a poet writes in English, German or French.
What does this tell us? It tells us that software programming by no means is something objective. - [to Adrian Ward:] I know, Adrian, that's a point you have made several times, that you express subjectivity in code and that you see Auto-Illustrator as an expansion of your own subjectivity and imagination into a generative process. And this point is important when we speak about software art, and by this we also come back to the question of authorship posed right at the beginning of this panel. It means that there is no such thing as autonomous technology. This concept is bullshit in my opinion. I strongly disagree with any media theory which says that there technology is autonomous, creating itself and running itself. That's not the case.
If you use Microsoft Windows, you use a construct implemented by the programmers of Microsoft, and they did it with an intention. The intention is not in the machine, but in those who created it. It's political, it's cultural. This is what we wanted to stress by highlighting this notion of software art.
Because of that I've come to be critical of poststructuralist and postmodernist theories that talk about the vanishing of subjectivity. I don't buy that anymore. Software art, using a technology that supposedly is impersonal, shows how much subjectivity is in the technology. I agree there is a danger to return to notions of the genius or autonomous artist. In fact, the notion of genius is quite common in programmers' cultures

where people like Richard Stallman, Daniel Bernstein or Donald Knuth are considered geniuses just as artists were considered geniuses in the 18th and 19th century. So we have to be careful not to go into this extreme. But nevertheless, going into the other extreme of ruling out subjectivity and individual creation altogether would be just as wrong.

Adrian Ward (from the audience): I was just going to add an important point. At the moment I come to feel that software art is partially about discovering responsibility. I think that's what we're all realizing in different ways that people writing software have a lot of responsibility. Traditionally perhaps those have been in banks for calculating bank balances or controlling flight control data […] but it's also about how you express yourself and how you use language. I think that's really what's causing us to be here because we are realizing we have lots of responsibility and that needs to be explored before we can go any further.

Sally Jane Norman: Just very quickly on the language analogy. Somebody who chooses to write in German, chooses to address a German public, somebody who chooses to write in French, chooses to address a French public, somebody who chooses to write in C, or C++, who is he addressing?

Florian Cramer: In case of Perl if its open software she's addressing the Perl community. Definitely. Yes. And ACME [Perl] is the example of that. ACME [Perl], what Amy has shown, only works for the Perl community and for no one else.

Amy Alexander: What? Everything written in Perl is for the Perl community? *Teasing.* OK I'd better shut up then. But really, plenty of projects are written in Perl. Like this project from the Yes Men software programming group for example - the Reamweaver project. It's a lot like the project Inke described, and it's for users. It lets you make an automatic parody of a website. It's not for Perl people; it's just written in Perl for certain reasons. A lot of projects … Runme.org is written in Perl but it's not for Perl users. It's not important that you know it's in Perl. Something like that ACME module, yeah, that's at a different level and that's for Perl users. But not everything written in Perl is for a Perl audience.
But I would say, yes, language does influence what you do. One of the things that interests me about Adrian's Auto-Illustrator project is that when I start reading about Real Basic, the language which he wrote in, I can see a lot of connections. RealBasic facilitates doing things with the interface and playing with the interface. In something like my b0timati0n performance project I'm playing with "what is the aesthetic of Lingo and Director?" Now the audience might not know "Oh, that's the Lingo and Director aesthetic!" but they kind of realize "I've seen a lot of this cheesy effects on the web on some place." So it is certainly relevant what you write - software influences the user and influences the user's behavior.

But also, the programming language influences the programmer's behavior and therefore it influences the output.

Alex McLean: I was going to draw a comparison rather than with human language with languages like painting. So one could paint a picture which a non-painter can appreciate but if you then want to take the language or take expressions from that painting and put it into another painting you have to be a painter yourself. Then you could draw comparisons between painting and Perl and pastel drawings and C.

Olga Goriunova, *addressing Sally Jane Norman*: When a writer writes in a certain language he doesn't choose to address a certain public he just writes because he speaks that language, it's his mother tongue. Writers who write in foreign languages are quite rare (probably only Nabokov is famous for that among Russian writers).
This tendency to compare programming languages with human languages and code with text and literature … how should I put it? … I really dislike it, because it's really reductive. It inclines towards very formalistic understanding of literature. Literature and software, text and code are indeed very different things.

Amy Alexander: It's a complicated question: the comparison of text with … of programming languages with human written languages. It's a shame that Margarete Jahrmann couldn't be here cause on the software jury this was her interest in particular. It's not supposed to be this reductive thing. The assumption that it is a one to one reductive comparison often gets the point dismissed and gets "Code is a language" dismissed. But that's not what it's about.
It's this idea of not losing the idea of coding, remembering that software is both interface and something written behind it. Remembering that this something written behind it has a style, has a subjectivity, has all this sorts of personal aspects that matter, even the fact that it's written in Perl or it's written in Real Basic… If it's written in Real Basic, it might play with the interface more; if it's written in Perl, it might do more things that you can't see, it might do more text manipulation and so on. Realizing that there are differences depending on how it's written, what language it's written, who's writing it and whose style, that is important. That's not to say "Oh, it's just like a human language." I don't think that's ever been the point.
I'd also like to bring up - to tie this into another question that came up - cause I have to go really soon - what is the material, what is the medium? How does this relate to music and musical languages? Because in music we also don't have material, we have the language and it's an executable language. And that's what software is - it's an executable language. And we also have people writing in different musical languages as well. Western music and eastern music are written differently. How does this relate? Do we have the same problems thinking about software as people did at first thinking about music? I'm interested in how this relates.

Because there's also been people like John Cage who wrote compositions with instructions in the middle such as "everybody stand on your left foot and play the third thing that comes into your head" which is very much like Alex's animal.pl software project. So there's some precedent here: there are these compositions with very literal instruction sets, but also, we can think about the fact that there's been executable language long before software.

Inke Arns: I think we should slowly start to wrap up the session. Because I see slight signs of fatigue in the audience. Perhaps it would be a nice … I don't want to give you a summary, because most of you have been here for the whole session - but I think it would be a nice question to ask you [the audience]. It'd be interesting to know if you could find all these topics we talked about in our discussion in the pieces we presented at the beginning. Perhaps this was a discussion that totally carried us away and it had nothing to do with the pieces we showed in the beginning. Perhaps you have some other examples. I don't know. It would be nice to …

Lars Midbøe [organizer of the "ElectroHype" software art conference in Malmø, 2002] (from the audience): Last spring when I was working on an exhibition in Sweden we really noticed a change that a lot of artists started working with code and programming. I think, a lot of it has to do with that people are tired of that the colors from the companies doing the software are rubbing off on the final art product. I haven't seen that "Oh, that's done in Flash and blabla…".
I said the other day that the focus is going to the internal processes between software and hardware. I think it's a good sign that people start looking at things that have driven that computer based art thing and going away from the VR crap projects that somebody also mentioned. I would like to point out one artist, John F. Simon [http://www.numeral.com/], who talks about coding as creative writing, but he had a problem because he couldn't sell the programs. When he made objects running the programs the galleries really picked it up. Have a look at his website, he's a good guy.

Statement from the audience: Up until now I never heard the term software art. I've been doing something for about 20 years which might be described as software art. I want to give a short statement or point of view from someone outside of this area, which might give you something to think about.
I've been doing programming to express what's inside of me. Just like using a brush to express something. But then sometime [ago] I asked myself a question: Does the program program the programmer? Is there a circle, a feedback loop? So maybe software art is a way to talk to the machine. Thank you!

Inke Arns: Thanks a lot! Are there any more questions, statements? Ja, Henning?

Henning Ziegler: What you just said relates to a discussion we had at the bootlab in Berlin last weekend about the programmer being an outcome of the object oriented programming. Since that language came up all kind of programs had to be programmed again and again because nobody could read the code any more. So what you end up with is lots of programmers spending lots of their time doing things over and over again. So what we're talking about in the field of software art is actually only a very tiny little piece of what programmers actually do.
This is one thing I wanted to say and the other thing: What I found most important about Dragan Espenschied's [and Alvar Freude's] "Insert_coin" project was, that we have to throw in norms and standards as well, cause one of the interesting things was that people noticed that some things wrong with the websites when they saw the CSU [Christian Socialist Union] party label becoming the RAF logo, a German terrorist organization. So they saw that on Spiegel Online, and they checked Newsweek and then they checked another paper. And every paper used the same wording, so then they started thinking "Oh, okay, never mind. Probably it's right." The odd thing I won't forget. Just to add images were also exchanged and sound, because everything on the web is based on HTML […], so they exchanged the image of the German chancellor with an image of Hitler. They just redirected the image links.

Inke Arns: Then I would like to thank everybody for participating in this discussion. I found it very interesting. I like to thank all the panelist on this panel.