SURFACE, INTERFACE, SUBFACE

THREE CASES OF INTERACTION AND ONE CONCEPT

Frieder Nake

Jürgen Habermas' essay "Arbeit und Interaktion" first appeared in an anthology in 1967. One year later, it was put together with four more of his texts and re-published under the title "'Technik und Wissenschaft als 'Ideologie'".[1]

When I first read it in 1972, it was because of the word "labor" in its title. I don't remember a thing from the fist reading, and the habitual underlinings and marginal remarks on the pages of the book don't indicate from when they are. I guess I put the book away on my bookshelf where it remained for more than a decade, perhaps untouched. When I read it the second time, probably in the 1980s, it was more or less by accident. I was browsing my bookshelves in search of something by Marcuse, when out of some intuitive reason I grabbed the thin Habermas volume and only then became aware of the other word, "interaction", in the title "labor and interaction". At the time (in the 1980s) I was teaching a four semester cycle in computer graphics whose specific topics were foundations, geometric modeling, graphic rendering, and techniques of interaction. Clearly, my bookshelf discovery forced me to immediately sit down and study what Habermas had to say on interaction. Why was he using the term?

Imagine the timely context of the situation! Charles P. Snow had delivered his "The two cultures and the scientific revolution" in Cambridge in 1959.[2] I had then been a student of mathematics at the University of Stuttgart. A year before, during a two month internship at IBM in their Böblingen Computing Centre, I had had my first encounter with a computer. Max Bense, the provocative philosopher at Stuttgart, had surely referred to Snow's talk and concept of two cultures. I found myself caught in a great melting pot: studying mathematics, the queen of all mental efforts, experiencing the grandeur and joy of strict axiomatics, formal concepts, theorems, and proves. From this comfortable centre to the left were Bense's thrilling lectures about aesthetics, ontology and, particularly, semiotics; to the right was theoretical and experimental physics, or the theory of electrical engineering, and more.

What a time, what a storm! We were a group of friends, trying to understand what the engineering types told us as well as what came from those in the humanities. We felt more and more at home in mathematics and, soon enough, in its rather trivial offspring, computing. But now we were confronted with Snow's claim that no communication was possible across the boundaries between the two types of disciplines we liked so much, because they were both exciting in their own way: the scientific and the literary cultures. It

1 The German original I am working with is Habermas 1969. It contains on pp. 9-46 the essay "Arbeit und Interaktion. Bemerkungen zu Hegels Jenenser 'Philosophie des Geistes'". It is available in English as "Labor and interaction: comment on Hegel's Jena 'Philosophy of Mind'", which appeared on pp. 142-169 in Habermas 1973.

2 The text is easily available, e.g. Snow 1993, which contains the original lecture of 1959 plus *The two cultures: a second look*, written five years later. I use Kreuzer 1987.

must have been puzzling to the young student who in the early morning was listening to a great electrical engineer, did his mathematics around noon, a bit of programming after lunch, and went to hear about Peircean semiotics in the late afternoon, just before rushing away to the opening of some artist's exhibition.

But now he had to swallow the fact that there was no communication, no interaction!? How could that be possible? Wasn't he himself doing exactly this: moving back and forth between those areas that – to be sure – had their own ways of expressing their findings, certainly, but wasn't communication happening through his own activities, even if restricted to himself and his friends? A decade later, in 1968, the great outburst of the youth's revolution against the father's generation shook the country. First readings about work and exploitation and suppression followed – a lot of Marx some years after this, and then, a strange sort of revelation.

Against the largely mathematical background of computer graphics, the open and heated discussions about human-computer communication were pure excitement. I had published a paper in 1984 on the impossibility that humans could ever communicate with computers, if the term "communication" was to be taken seriously.[3] Unix was ruling, Silicon Graphics machines were great, but the Apple Macintosh had appeared on the market. We were beginning to interact with the computer as never before. The researchers at Xerox PARC had done tremendous things that made beautiful surprises in the classroom. C. P. Snow's verdict was still present in my thinking. But now came the discovery of that second word in the title of Habermas' essay: interaction!

Reading, on page 9 of this collection of supposedly critical texts by Habermas, the word *Interaktion* not only meant that you were, perhaps, no longer restricted to saying "HCI" when you were talking about humans and computers and their intricate relations. It meant that you could, perhaps, add a totally different perspective than the one usually offered in computing circles. So what did I learn, how did I read Habermas?

In a nutshell, Habermas, in my reading and restricted conclusion, said the following. The mediation of subject and object is what constitutes "Mind" (*Geist*). There are three ways how the subject and the object may be related. Three categories mediate between subject and object. The three categories are language, tool, and family. These terms stand for three patterns of dialectic relations, the patterns of symbolic representation, labour process, and reciprocal interaction.

3 Nake 1984, pp. 109-118

The tool and labour process stand for that mediation of subject and object, where the subject changes the state of the object. The tool-relation transforms the object into a state better fit for the subject's needs.

The language or symbolic representation stands for that mediation of subject and object, where the subject observes and describes the state of the object. The language relation creates a semiotic layer that stands for the object.

The family or reciprocal interaction stands for that mediation of subject and object, where the subject accepts the object as of the same kind and capacity as the subject itself. The interaction relation leads to cooperative, communicative exchange between subjects that are equal.

Mind you, this is the simple, naïve, and immediate interpretation of a probably difficult reflection. The interpretation was by a computer scientist who was quite happy to find something in the other faculty which he thought could help him. The writing was by a philosopher and social scientist who, by the time of his writing, was developing his theory of communicative action, which played an important role in his attempt to reconstruct, as he said, dialectical materialism. Undoubtedly, dogmatists must have fiercely attacked him for working on such a project, whereas more liberal representatives of the left may have accepted the premise that Marxism also has the right to evolve when social reality changes.[4] My naïve view, however, quite happily suggested to me that a systematic approach could be applied in order to introduce an interactive mode of using computers. I may have, of course, grossly misunderstood, nevertheless, the threefold distinction may serve a purpose here.

In the rest of this essay, I will describe three cases of using computers. Each case involves two persons in varying positions relative to the machine. In each case, we will see the open surface of the computer periphery, which will be commented on. We will also see the hidden and more or less inaccessible subface, and will gain some insight into how, and why, the two are necessarily related to each other. We will discover what the reader may already have been aware of all the time: computer things come in pairs. We will briefly give a semiotic interpretation of this claim by introducing the concept of an algorithmic sign. In conclusion, we will point to the current importance of digital media as those media that explore the dialectics of algorithmics and aesthetics.

4 Keane 1975, pp. 82-100

Scene 1: Automaton.
Two persons interacting, waiting for the computer

The year is 1963 or 1964. Two men are sitting in front of a computer. The size of the room may be 15 by 6 meters. No-one else is in there. They are not really sitting in front of the computer. The racks and cabinets that make up the computer create a system way too large to be sitting in front of. Besides, the two are sitting inside a separated space surrounded by glass walls. They see the metal cases, lined up in a long row of several meters length, and the tape drives next to the cases. Inside the glass room, temperatures are a bit more agreeable for humans. Outside, the climate must be closely monitored to stay within a small margin to prevent frequent failure of operation. The peripherals of the machine share the glass-walled space with the two men; a paper tape reader and a tape printer next to a large computer console.

Fig. 1. The operator's interface. Computer SEL ER56 at Computing Centre, University of Stuttgart 1965. We see dials (lower right) and push buttons to get at specified data. Through the glass in the background some of the hardware

One of the men has recently graduated in mathematics. The other one is a researcher in mechanical engineering. His theoretical investigations have led him to describe the behaviour of a metal sheet as a non-linear fourth order differential equation with boundary conditions. The equation can only be solved numerically.

When the researcher approached the computing centre for help, the young mathematician was given the job of cooperating with the senior person. At their first meeting the engineer explained what he was doing, why he was interested in getting the solution, even an approximate one, and that he hoped the younger man would do all the programming since he himself had no clue of what might be involved and could see no chance of changing this. Why should he learn to program? He would rather agree on meetings and the admittedly arduous work of cooperation.

A great chance for the young mathematician. He did not take much interest in the details of the mechanics. For him, it was enough to accept the differential equation as his starting point. To which extent it described the vibrations of the sheet was the engineer's responsibility. As a mathematician, he was responsible for the selection of a modern numerical method for numerical integration (he chose a Fehlberg algorithm). Fine tuning for efficiency in those old days was an important and creative task. You could come

up with improvements week after week. You did the improvements yourself, no compiler could take the burden from you (in fact, there was no compiler on this machine). But sooner or later the program worked and production could begin.

Since this was a boundary value problem, another difficulty had to be tackled: how to satisfy the boundary conditions? Such conditions require that a valid solution starts and ends at specified locations. It must leave and arrive there under specified directions (making a total of four conditions).

The rather simple approach was trial-and-error: Start from the prescribed position at the left end; solve an "initial-value" problem by arbitrarily assuming two more initial conditions; compare the calculated endpoint to the right with the prescribed goal. If there is a difference (which will, most likely, be the case), adjust the arbitrarily chosen initial condition such that, hopefully, the discrepancy is diminished, and repeat. This shoot-and-run approach will, under not too heavy circumstances and after some systematic attempts, come close enough to a realistic solution.

In those days, computers were incredibly slow when you compare them with their performance now, fourty years later. The difference in efficiency must be 6 to 10 degrees of magnitude, if not more. The two men, after having started the next shot, had to wait for several minutes (up to five), before the machine presented the few numbers they needed to judge how close the shot had come to the goal. While waiting, they had plenty of time to talk about mechanics, mathematics, programs, artificial intelligence, philosophy, politics, university gossip, the latest jazz concert, and a dozen more topics.

The situation thus indicated is heavily interactive! The two intellectuals were cooperating to solve a tough problem (they actually spent about two or three months before enough data had been generated from the virtual experiments in the algorithmic laboratory). Enough data was achieved when the engineer decided that his theoretical model was now backed up by enough empirical evidence. The cooperation between the two men allowed them to interact in the most complex and interesting ways. No procedures, no methods prevented them from journeying down any of the myriads of alleys open to the mind. The two actually became friends.

Embedded into the human-human interaction were short moments of very low-level human-computer interaction. When the result of the last calculation had become visible, the two discussed it and decided how to proceed. Proceeding was defined by a choice of new initial values. The process of choosing could have been automated since the goal was well-defined. But the amount of extra calculations would then, very likely, increase much more rapidly than with personal inspection. Inspection and discussion took advantage of the human capacity to detect patterns, consider context, and be aware of the situation and its changes.

The machine they were using, an early transistorised decimal (!) computer was good enough for the purpose. Its interface displayed current numerical values of data stored in memory cells. The two friends, in order to read the coordinates of the final destination, had to dial knobs to get at those memory locations (given by their absolute addresses), and then lock up the numerical contents of that location.

At the machine level, only a tiny bit of interaction was happening. Really, this wasn't more than looking up and reading some signals – ridiculously low-level when compared to what became standard twenty years later. Slow times those days, you would say. Two men engaged in watching the computer come up with the result of a calculation. They had to wait because the machine was used as an *automaton*. They fed the automaton with data, hit the start button for the automaton to do its work, and were thrown back to talking, becoming bored, or listening to the radio.

Scene 2: Tool. One person using the computer, the programmer far away

The year is 2002, nearly forty years later. In terms of technology, the world has turned upside down. The technical infrastructure of all processes, in the private, economic, administrative, or political realm, is determined by data processing. In parts of the world, there is virtually no room that does not contain at least one computer. The art historian is sitting at her desk at home. Her current field of interest is an area not well known, certainly not mainstream, but slowly and steadily gaining interest. Already in her Ph.D. thesis a few years ago she started to seriously study the phenomenon that in the mid 1960s was called "computer art". She now generally prefers to use the term "digital art".

She is preparing for a meeting with her students. Recent work by Manfred Mohr will be the topic. He is the German artist who first got access to a computer controlled drawing machine (often called a "plotter") in 1969 in Paris. He gave up painting in favour of programming, stopped using colour in favour of black and white and, later, some grey and silver. He gradually became a recognised artist who could make a living from selling his art. He had discovered his topic in the early 1970s: the cube and its symmetries. In order to gain complexity, the cube had become the hypercube of four or five or even more dimensions. Like other artists before him – say, e.g., Paul Klee, Josef Albers, the concrete artists – he had become a researcher as well.

The sensation happened in 2001 when he exhibited large canvasses in bright colours.[5] Our art historian was trying to understand those pictures. They had again been exhibited under the title space.color in October 2001 at the Museum für Konkrete Kunst in Ingolstadt, Germany. Ever since, Mohr had been using coloured fields in his paintings (and, later, computer installations). He needed six- or eleven-dimensional space and colour to increase complexity by orders of magnitude.

The art historian at her home desk, not being acquainted with geometric spaces of higher dimensions, tried to understand the algorithmic process that generated some of the pictures in front of her. The algorithm behind the canvas had been Mohr's secret for all his productions. In his catalogues he had been friendly enough to publish brisk and sober definitions of the algorithmic behaviours of the generative processes. But now, with colour reappearing in Mohr's works, she felt lost. She had been studying the catalogue of the Ingolstadt exhibition: reading statements, analysing pictures. She was unable to grasp how the remark that something was going on in six dimensions could be helpful. She turned on her laptop computer. Someone had given her a program called deviceX. It was supposed to help develop some kind of understanding of the space.color period of Mohr's art.

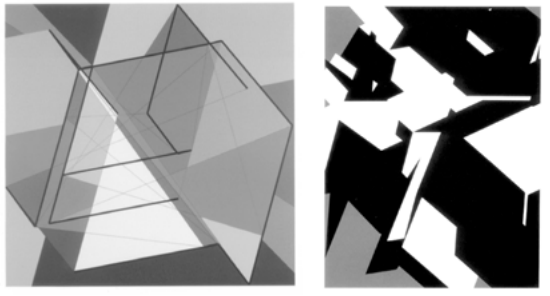

Fig. 2. *Left:* ***Manfred Mohr: P-107-f (1999).*** *Right:* ***P-1011-z2 (2004). Six dimensions behind the left, eleven dimensions behind the right image***

The name, deviceX, rang two bells in her mind: tools and X-rays. Tools are instruments we use to more easily change the state of some material; X-rays are dangerous but helpful in looking into the human body. DeviceX could, perhaps, be a tool to look into the structure of those paintings.

Mohr's paintings of the space.color variety appear – just like any other painting does – as a configuration of coloured forms. The configuration of the forms corresponds to the geometry of the painting. We may derive from the geometry of a painting a more abstract rendition of the same content. This abstract rendition may be called the painting's topology. The abstraction gives

5 For the first time, Mohr showed these pictures in June 2001 at Galerie Wack in Kaiserslautern, Germany.

up the particular form of an area, its size and, to a large extent, its location. Of the geometry, the topology keeps only one relation: that of neighborhood. If two areas are neighbors in the geometry, they must also be neighbors in the topology. So if they have a common edge in geometry, their topological neighbor-relation is an edge-neighborhood.

Topology is abstract. It gets described in formal symbolism. We can, however, use a minimal visualisation by using squares as the only form features. DeviceX makes use of this. The art historian started the program. Soon she found out how to use it. Among some other features not of prime interest here, the most prominent one is the following (Fig. 3). There is a small replica of one of the possible Mohr paintings. It can be grabbed with the mouse device and shifted horizontally, left and right. As it is moved further to the right, the intricate geometric forms untangle more and more into an arrangement of coloured squares.

We detect, as squares, the same colours as in the original picture. We also detect some that were hidden before. DeviceX is like a slider, i.e. an instrument we use to set one parameter to a certain value. The slider's relative position usually indicates the parameter's value along a scale from 0 to 1.

DeviceX also functions according to this scheme, with one important difference, however: it does not indirectly indicate the parameter's value. It rather shows it directly. The device is loaded with the contents it controls. By looking at the slider we look into the picture. This is its X-property: the property of looking *into* (or even through) an invisible material.

The art historian, when applying deviceX to some data content of which she doesn't really know where it exists, considers herself a "user" of the software. The mode of use is usually called "interactive". The interaction is between her and the software, deviceX. It is quite clear to her that she is *not* shifting deviceX but the mouse in her hand. But it appears to her as if she was directly (and not indirectly) shifting the graphic rendition on her laptop's screen. The interaction between her and the computer has become so fast

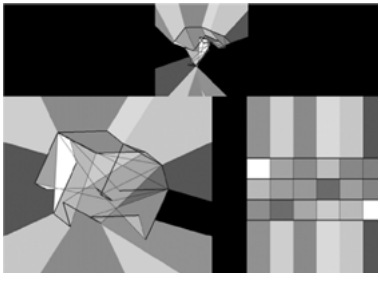

Fig. 3.* deviceX.** *Top:* ***Geometry of an image (lower left), topology (right) and intermediate state of slider (above). *Bottom:* ***a series of states of an image in transformation from geometry (left) to topology (right)***

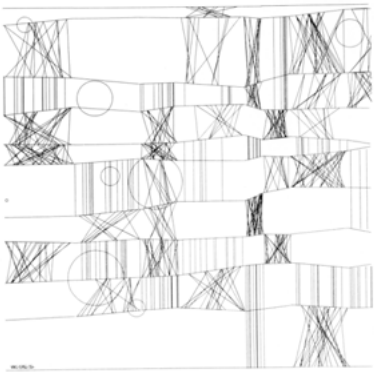

Fig. 4. *Left:* ***Frieder Nake, Homage à Paul Klee. Computer drawing, 1965.*** *Right:* ***Frieder Nake, Susanne Grabowski, Matthias Krauß: Spannung. Interactive installation. Picture taken during the opening of my exhibition at ZKM Karlsruhe: to the far right, Peter Weibel in discussion***

that she can ignore the tremendous amount of calculations going on at every single moment.

The art historian probably also knows that the immediacy on the screen is caused by a programmer. He may now be living far away, working on something new.[6] But once in the past, he wrote that program that now effectly appears like a tool in the art historian's hand. In some metaphorical way, it *is* a tool.

Scene 3: Medium. Two persons having fun, others watching, the computer: where?

It is now 2004. We enter the Korbakow room at Kunsthalle Bremen on our tour of *Die präzisen Vergnügen*[7] ("precise delights"). We observe a couple having fun at a bistro table (Fig. 4 right). A screen is mounted into the tabletop. Two small graphics input panels can be operated by using a pressure sensitive pen. The screen in the middle displays a line drawing belonging to the well known "Homage à Paul Klee" program (Fig. 4 left). There is a tremendous difference between the old algorithmic drawing of 1965 and the interactive installation of 2004.

6 The programmer to be credited here is Matthias Krauß; see Nake/Krauß/Grabowski 2007, pp. 137-144.

7 The retrospective show of algorithmic works of Frieder Nake from the 1960s was put up under the condition that he could also present four new interactive installations. They were the result of collaborations with a group of students.

The algorithmic drawing displays a complex structure of straight lines. Some run horizontally across the entire format. Others build bundles of parallel verticals. They are much shorter, and run from one of the horizontals to the next or next-but-one. There are also substructures of oblique lines building rhizome-like groups. A number of circles seem to be floating on or above the straight-line structure.

Each line and each structure of the static, algorithmic drawing is kept in its place by some mechanic force. All the hundreds of lines stick together keeping a graphical balance. This observation became the starting point for a dynamic and, in fact, interactive adaptation: each line was interpreted as a force, as a spring.

A system of springs is kept together by attachment points. These springs are not visible. They are a metaphor for an invisible property of each of the lines. The visible lines are a graphic interpretation of the spring system.

When a visitor puts the tip of one of the two available pens onto one of the graphics input panels, some line or point or cell is highlighted in colour. This feedback tells the visitor, which one of the objects he picked. Depending on the kind of objects, the visitor can perform a number of possible actions.

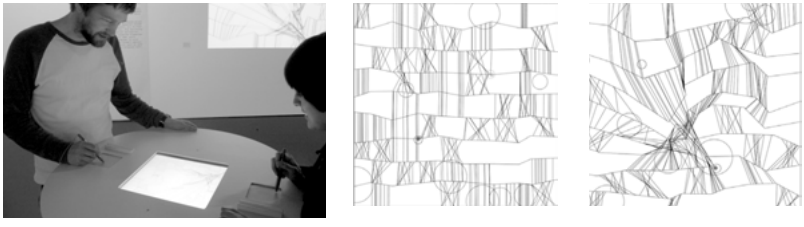

Fig. 5. The interactive installation* Spannung.** *Left:* ***Two visitors operating (display, two panels, pens; projection visible in the background) *Centre:* ***Display image with visitors' identified vertices (coloured dots).*** *Right:* ***Distorted image***

He may move the contents of one of the horizontal bands of lines. By their spring property, they remain attached to the horizontals at all times (Fig. 5 for two visitors in action).

The visitor may also change the contents of a cell (from obliques to verticals to empty). Or he may grab one of the vertices, and pull it to a new location. This creates the most exciting effect because the entire structure must follow exactly, keeping its attachments as they were before.

The effect of applying an outside force to test the tensions inside the drawing becomes even more dramatic when two visitors grab vertices simultaneously and pull in different directions. This is an illustration of the basic

characteristic of computing: objects are always double. We will describe this in different ways in the next section.

Here we should add how the field of forces influences the slow motion of the circles mentioned above. They have displayed a very calm and slow motion at all times even when the lines were static. The circles thus invited the visitor to think about the picture in dynamic terms.

The circles possess another unique property. Two visitors may grab them simultaneously. With one circle attached to the interactive tools of the visitors, they can move the circle around or change its size by pulling in different directions.

When no one is operating the installation, it displays a static minimal picture: a black square and a black circle, slightly overlapping, somewhat reminiscent of Malevich's suprematism. As soon as one of the pens is applied, one of the Homage à Paul Klee drawings slowly unfolds out of the square-and-circle. At the same time the image is projected onto the wall.

Pictures in a museum or gallery are silent witnesses of their artist's work, of their epoch, and of systematic and historic contexts. When we walk the halls of the museum, we may not get anything from the rich context. When we read a book, listen to a guide, or engage in conversation with friends, the situation changes, and the paintings also.

At first sight the interactive installation seems to be similar. But it is ready to tell us more about itself if we interact not only mentally, but manually as well. Interactive use of a program – today the ubiquitous mode of use – belongs to the characteristic view of the computer as medium. The exhibition in 2004 had installations that were to be used without tools. A camera detected passers-by and a minimalistic static projection was set into motion.

The message is: I am waiting here for you to do something; you don't have to be instructed, just do what you see fit, and I react. The interchange between you and me may tell you something, but what that could be, is totally up to you. My aesthetics is the *unfinish*. I am finished as far as I am a technical product. But I am obeying the law of *unfinish*[8] (a funny kind of activity, isn't it?) by going on and on with no end, no goal.

This is exactly *the* identity of digital media. The computer has disappeared. Where is it, we may wonder? Hidden somewhere, having become a medium. Media are ubiquitous and unobtrusive.

8 The term is introduced in Lunenfeld 1999, see his introductory essay *Unfinished business*, p. 7.

Surface & Subface *or* the Algorithmic Sign

The three scenes described above stand for stages in interaction related to computing machinery. Its acknowledged history tells us that, without doubt, the computer started as an automaton. This does not only apply to the mathematical term automaton as used in formal theory. It applies much more to the historic evolution of the work of machines that had no choice, under given economic and political circumstances, but to give rise to automatic machinery. The concept of algorithm and the paradigm of computability stand for this.

The computability feature got wrapped into a tremendously beautiful and successful adorning layer of pseudo-tools. They are programs which no one must think of as programs. You start and stop them, specify their input parameters, and observe their outputs in the most joyful way. Never has machinery made use of its own capabilities so inventively, humbly, progressively, and aesthetically as computing technology. From the first bold steps by Alan Kay and his group at Xerox PARC in 1971 to Apple's first Macintosh and its software in 1984, only a dozen years passed. Ever since no other use had any chance, no other mode of existence had any relevance, but the tool perspective.

The economy of the time required the transformation of the computer into a market commodity. The concept for this was the invention of algorithmic tools; the mental paradigm was interactivity. As it evolved, and as miniaturisation continued at a breath-taking speed, algorithmic tools disappeared into the general environment. The third phase of interactivity brought back the original situation. In the first scene, we had seen two men interacting with each other – and, in the true sense of the word, there is no interaction other than between humans. The two, in their deliberations, occasionally turned to the computer to get answers to certain well-defined tasks.

The tool phase pushed groups of humans into the background in order to generate the false ideology of human-computer interaction. Human interaction had to be brought back artificially by inventing CSCW – computer-supported cooperative work. But the tool phase was necessary and of the utmost importance. Without mercy, it caused everybody to use computers as an absolute given. Nobody can now work or live anymore without a computing machine.

The art world has widely accepted interactive applications of computability. It has provided another layer of wrapping paper: the transformation of the automaton into a medium. A great story of only forty or fifty years: The *algorithmic revolution* (Peter Weibel)! It depends on a very simple technical circumstance. I want to mention it at least briefly, giving it two names. Whatever

thing you may address, choose, pick, apply, use on or in a computer, it comes as a sign of a special character. I call it the *algorithmic sign*.

The second name I want to attach to computer things is the pair of *surface and subface*. Let me explain this use of language before I take up the algorithmic sign, which is the more theoretical twist.

When we use a computer, we use a program running on the computer. A program – called software – is running on a computer, when a machine – called hardware – is executing a code. The code is a sign for the program. When the program is running, it generates images on the computer display monitor, its main output device. Those images change in extremely rapid sequences. Each movement of the mouse or hit of a button changes the current image.

Screen images are visible to us. The program exists for us by its name and the world of images it generates. We anthropomorphise the operation of the program. We tell each other things like: "you should see what the new version of X is doing! It can now take your pictures and organise them such that it becomes much easier for you to …" Thinking twice about such talk reveals a false conception. The program is really behaving just like any other machine: it is carrying out exactly what we want it to do, or at least, what our parameter settings force it to do.

A metaphorical way of talking about the program's behaviour is, nevertheless, justified. It is justified because the program is *manipulating* in its innermost organs what we only *see* as the current state of affairs. From an outside perspective, we may collapse this into one observation. Whatever is to become an utterance of the program for us to perceive, must first be stored in the display buffer to which the image on screen is tied in a one-to-one correspondence.

The screen is the *surface*, the display buffer is the *subface* of the algorithmic thing that the two of us – we ourselves and the program – are engaged in. The algorithmic thing comes as a visible appearance for us. At the same time, it comes as a computable appearance to the program. Without both being present and being tied to each other, nothing would work the way we want it to work. It does not make any sense to talk about the computer image without keeping in mind its visibility *and* computability, i.e. its computable visibility *and* its visible computability.

The computer thing is a double insofar as it is not *only* visible, nor is it *only* computable. It is visible in a new meaning of the word, and it is computable in a new meaning. Our thinking needs an understanding of the old meanings of those two concepts. Computer images are *more* than visualisations of a computation, and they are *more* than computations of an image.

The world of the surface and the subface that cannot but appear together is apt to catch exactly this: the inherent duplicity of anything happening on

the semiotic machine. "Semiotic" is the correct word to characterise this. The algorithmic sign, which I am now going to introduce, is the theoretical concept for this semiotic perspective.

To recall from Charles S, Peirce, a "sign, or *representamen*, is something which stands to somebody for something in some respect or capacity. It addresses somebody, that is, creates in the mind of that person an equivalent sign, or perhaps a more developed sign. That sign which it creates I call the *interpretant* of the first sign. The sign stands for something, its *object*."[9]

Of the many definitions of the sign by Peirce, this may be the most lucent one. To repeat in my own words, Peirce introduces the sign as a triadic relation, which wraps up the old dyad of something standing for some other. The sign, as a relation, cannot be perceived. But it must possess the feature of being perceivable. This feature is called the representamen. To be perceivable by our senses, it must be material. In some way, the representamen carries the sign. It carries it insofar as it gives rise to the relation that the perceiver is creating upon her perception. She is creating the object and the interpretant. The two together constitute what traditionally is called the meaning of the sign.

In Peirce's great analysis, the meaning of a sign depends on a culture, on a context, on a community. That community makes it possible by all of its conventions, history, habits, etc. for us to use the sign in the interests of communication. This general or public component of the meaning of the sign is the *object* of the sign.

Each subject perceiving a representamen and trying to make sense of it, also creates a particular or private component of the meaning of a sign: the *interpretant*. If the object is the long-lasting and generally accepted meaning of the sign, the interpretant is its short-term, situational and individually generated meaning.

Peirce thus gives us a dual way of talking about the sign: perceiving the red light of the traffic sign with the general and enforced interpretation of "Stop!" as well as the particular and deviating interpretation of "Proceed with great care!". The interpretant, by the way, is itself a sign. This introduces the sign as a recursive concept, as a process without end. Only the pragmatics of a given situation force us to interrupt the infinite sign process of interpretation.

Let us now take a look at what the computer does when it receives an input signal. The signal corresponds to the representamen of a possible sign. Of course we expect the computer to function well and to do exactly what the input signal "tells" it to do. With the rare exception of a malfunction, our expectation comes true. Nevertheless, the computer performs an act that

9 Peirce 1955, p. 99

formally is of an interpretive nature even if the computer is not capable of any interpretation. It is programmed in a definite, and precise way. The program "interprets" the input signal, i.e. it determines the one, and only one, interpretation of the line of code, or the command that it is then forced to execute.

The formal act of interpreting a unit of code, in the case of the program, reduces to a determination. *Determination* is the limit case of interpretation: finding out the one and only meaning. Thus algorithmic signs are signs in the usual (Peircean) sense of the word, but with an extra interpretant. We call this the *causal interpretant* to distinguish it from the *intentional interpretant.* The latter one is what the human creates.[10]

We now close the ring. The surface of any object on the computer corresponds to the intentional interpretant of the computer sign. The subface corresponds to the causal interpretant. I am not saying that the subface *is* the causal interpretant. For my intention here is to point at a correspondence between two perspectives.

The components of digital media, of semiotic entities on a computer, or of things we are interested in when using a computer, can only be understood in the world of relations, not in the world of things. This is my message in this essay. What is usually called the interface between human and machine, appears as the coupling of surface and subface. Both are machine-bound. Both are faces at which one process ends, and another process starts. The human places rather trivial components onto the surface (like mouse positions, or menu selections). He interprets what the program delivers in a rich way, influenced by his intentions, interests, situation, and context. Once the surface is transformed into the subface, the program starts its signal processes, which consist of chains of determinations like any other process on a machine.

The miracle of human-computer interaction is that it is *impossible* as interaction in a true sense of the word. It is happening nevertheless. This is possible because human acts of interpretation correspond in a rich (but computable) way to machine operations of determination. The miracle is that humans were bold and intelligent enough to establish this. The miracle is not that machines were so intelligent to do it.

10 This concept is further elaborated in a chapter of the book by Peter Bøgh Andersen and the author (forthcoming).

Algorithmics & Aesthetics

A very brief remark in conclusion. Since about the mid 1990s, some aspects of computing science have been collected under new programmes of study. They are often called Digital Media, or something similar. These programmes carry the heavy burden of striking a balance between a serious and appropriate study of algorithmics up to the point of script programming, and of aesthetics of 20th century art. They need a bit of art history as well as the history of computing. Their questions should be directed towards an understanding of the algorithmic sign in as many ways as possible – aesthetic, educational, and cultural.

References

Andersen, Peter Bøgh/Nake, Frieder (in preparation): Computers and Signs. Prolegomena to a Semiotic Foundation of Computing Science, Heidelberg: Synchron Wissenschaftsverlag der Autoren.

Habermas, Jürgen (1969): »Arbeit und Interaktion. Bemerkungen zu Hegels Jenenser ›Philosophie des Geistes‹«. In: Jürgen Habermas, Technik und Wissenschaft als »Ideologie«, Frankfurt a.M.: Suhrkamp, 9-46.

Keane, John (1975): »On tools and language: Habermas on work and interaction«. New German Critique 6, 82-100.

Kreuzer, Helmut (Ed.) (1987): Die zwei Kulturen. C. P. Snows These in der Diskussion, Stuttgart: Klett-Cotta.

Lunenfeld, Peter (1999): The digital dialectic. New essays on new media, Cambridge: MIT Press.

Nake, Frieder (1984): »Schnittstelle Mensch – Computer«. Kursbuch 75, 109-118.

Nake, Frieder/Krauß, Matthias/Grabowski, Susanne (2007): »The sound of silence in spaces of many dimensions«. In: Biennale of Electronic Arts Perth: CADE: Computers in Art and Design Education Conference, 12-14 September 2007. Conference Proceedings, 137-144.

Peirce, Charles Sanders (1955): Philosophical Writings of Peirce. Selected and Edited with an Introduction by Justus Buchler, New York: Dover Publications.

Snow, Charles Percy (1993): The Two Cultures, Cambridge: Cambridge University Press.