– Will be published soon (a bit different) in: Fishwick Paul (ed.): Aesthetic Computing. MIT-Press –

The Interface as Sign and as Aesthetic Event

Frieder Nake & Susanne Grabowski (2004)

What kind of artifacts would *Aesthetic Computing* be concerned with? How would they differ from artifacts that ordinary computing is concerned with?

As Paul Fishwick tells us in his introduction to this volume, computing deals with models, programs, data, and interfaces. Software systems developers invent abstract *models*, which are rendered in computable form, i.e. in *programs* and *data*. Users apply such programs to their context, which must be represented in the form of data. The use situation is called human-computer interaction. The *interface* is where the two interacting systems, human and computer, meet. It will be the topic of this chapter.

Using software to change some state in the environment has almost exclusively become an affair of using interactive systems. Communication plays a central role in it. Interfaces to interactive software systems must be concerned with successful communication. They must therefore be functionally effective and aesthetically attractive. This is what visual design is about (Mullet & Sano 1995). No contradiction seems to exist between function and aesthetics when communication is the goal. Interaction design, which is gradually replacing interface ergonomics, appears as the trivial case of aesthetic computing. We intend to analyze it a bit deeper.

There is reason to assume that the days of the ever-present graphic user interface are counted. You may see the tendency of going beyond the desktop when you skim the yearly CHI proceedings. Not many have realized that a cross-over, of computing and aesthetics, is happening. Computer users are triggering computational processes on semiotic machines, which they gain access to through layers of signs. When signs are involved, our perceptive capacities are required. Therefore, the situation is determined as an aesthetic one. Aesthetic computing is viewed only narrowly if we take it as the application, or addition, of some vaguely understood aesthetic rules to the usual computing situation. Rather, *aesthetic* computing emphasizes an *inherent*, but until recently largely hidden, aspect of computing, and pulls it into the open for investigation, interpretation, and construction.

Whatever the term "aesthetics" may stand for when defined narrowly, we all have some pre-conception of it. Its first, and most important, aspect is sensual perception; its second is beauty. "Aesthetic computing" is, therefore, deliberately introducing subjectivism into computing with all the consequences this may have. The most important consequence is that much of what computing science and software development are concerned with, must be reconsidered and, quite likely, be transformed into less precise terms than an engineering discipline would usually accept. A number of authors have in recent years pointed to the necessity of such a turn away from quantity and measure to quality and judgement. To single out only four from the vast amounts of wonderful books and papers, we refer to Winograd (1996), Ehn (1998), Raskin (2000), and Löwgren (2004). The ACM magazine, *interactions*, carries a lot of discussion on our topic.

Engineering comprises both, construction and evaluation. In the arts, the two are separate activities of expert artists and critics. Höök et al. (2003) have raised the question

of how user testing and art criticism could meet in new ways of sense making sensibility. Aesthetic computing will involve criticism of a kind yet unknown in computing.

To put it quite frankly: quality instead of quantity, style instead of truth (Wiesing 1991) is the concern of aesthetic computing. We expect that not all of our readers will readily accept such a formula. We hope to show why it is to the heart of this endeavor even.

A bit more on Aesthetic Computing

Aesthetic Computing is certainly not about art. And if it were, its proponents would not want it to be. Despite current attempts to create some cross-fertilization between two fields of design so far apart, and, at the same time, so close to each other as art and computing are, it remains to be shown what is hidden behind the term, *aesthetic computing*. Conspicuously, the terms *aesthetics* and *art* are being used in the first few publications as if they could easily be interchanged. To many artists, that would appear as a terrifying idea.

Syntactically, "aesthetic computing" must emerge as a special kind of application of aesthetics, much the same way as "electrical engineering" could be considered an application of James Clark Maxwell's theory of electromagnetism to engineering. Putting it this way reveals the unsymmetry in the combination of the two words: could we not also say that "electrical engineering" was the application of engineering principles to the realm of electricity? If we did, by analogy, aesthetic computing would appear as an application of computing principles to the realm of aesthetics (as, e.g., in Georg Nees' very early doctoral dissertation 1969).

Though this would bring us closer to computer art, aesthetic computing would still not become a matter of art. But it would definitely border upon the activity called "art". Interesting as the general relation of art and computing is, and complicated as aesthetics and art relate to each other, our immediated concern in this article is more specific. We want to take a look at the interface of human being and machine artifact. Our look will be largely determined by semiotics, the theory of signs.

The situation and context of our contribution will be determined as a relation of three human activities, those of aesthetics, computing, and semiotics. We use the case of human-computer interface to raise, under a semiotic perspective, questions like: "What has aesthetics to offer to the design of good interfaces? How can we better understand the design and use of an interface if we approach it aesthetically? Is pleasingness of an interface more relevant than usefulness? Should joy of use replace ease of use? Should we play with the interface rather than the interface function for us?"

We will subdivide the remainder of this treatise into six sections. First, we will describe a particular application of computing to art, and the interface to that program. The example will serve as reference for the more general discussion. We will then take three views of the example: the objective and formal view (from computing), the subjective and emotional view (from aesthetics), and the connecting and medial view (from semiotics). The next two sections will generalize our case to the situation of interactive use of software. We will offer semiotic fundamentals of interactivity and point out how important an aesthetic perspective is for understanding and designing interaction.

Before we do so, we would like to add one more introductory comment. It is clear, and often deplored, that works of art, as they are presented in galleries and museums, in most

cases reach only tiny percentages of the population. True, there are shows nowadays that generate a tremendous impact nationally and internationally. Examples could be the Venice Biennual, the olympics of the Documenta at Kassel, Germany, or the Ars Electronica Festival at Linz, Austria. But even so, they are by far inferior to global commercial enterprises of aesthetics like certain movie shows (*Jurassic Park*, or the *Matrix* series), or book events (*Harry Potter*). Sports events carry a lot of mass culture aesthetics with them, and by far outperform the aesthetics of fine art.

The computer has become so popular and ubiquitous that it is on a par with the telephone, radio, or television. One important aspect of this tremendous spreading is the Graphic User Interface (GUI): a mass culture aesthetic phenomenon of, perhaps, unparalleled significance for culture on a global scale. Think of work situations as well as computer games and their reliance on graphic interfaces.

Digital media increasingly determine the current state of society. It is impossible to use them without being familiar with their interface. The development has reached a point where the actual computing processes almost disappear behind the interface. Take the beautifully designed little colored Apple *iMac* computer as it appeared in shopping centers in 1998. It constituted a case where the computer as a *machine* had almost entirely been swallowed by its interface. Even if no total commercial breakthrough, the *iMac* (which has, in its original form, already been taken from the market) marked an aesthetic event. As everybody knows, one of its successors from Apple, the far-out design of the *Cube* (again no commercial success), became the first computer in history to be acquired by the New York Museum of Modern Art Design section.

Figure 1: iMac

Much more could be said about the relation of aesthetics and computing in general. Much of this has recently started to be published in the context of digital media, their theory, history, and practice. At times, it may appear as if more attention was paid to the origins of digital media than used to be the case in respect to the origins of computing. To again name but a few, we refer the reader to Bolter and Grusin (2000), Lunenfeld (1999), Manovich (2003), and Packer and Jordan (2001).

In our humble attempt to justify aesthetic computing we have to look deeper into the essence of computing and interaction. Interfaces are signs. Signs may gain such power that they replace, not only represent, what they stand for. We will now embark on our journey whose end will be this message.

An example of computing and art

As mentioned before, aesthetic computing is not to be mistaken as computer art, which we could describe as the application of software to generate aesthetic objects in hope of their relevance for art.

On the other hand, Paul Fishwick says, "*Aesthetic Computing* is the application of the theory and practice of art to computing." (Fishwick, this volume) Both these views apply aspects, techniques, or results of one kind of human activity (computing or art) to another one (art or computing). Logically, both are of the same type. Limiting our view this way, one or the other, will not get us very far. We want to adopt the position that aesthetics and computing have entered a state of concrete dialectics.

Any two facts or processes may be studied dialectically. This helps to understand contradictions and discrepancies between the two, and their development. Dialectics is really about mutual change, influence, and evolution. The *abstract* dialectics of aesthetics and computing would only mentally consider differences between the two. Two more or less opposing forces of reality have entered a relationship of *concrete* dialectics when we observe actual change in the environment that appears to be influenced by the two forces. We claim that something like this is happening in the case of using software.

We want to put weight on this claim by studying an example. Inspite of the demarcation against computer art, we chose the example from an artist working with computers.

Manfred Mohr is a New York artist who has written programs for the generation of his works since the end of the 1960s. For about thirty years, he has used the cube in three and more dimensions as the basis of his paintings. Up until the end of the 20th century, he had created his very personal style of concrete, constructivist art. His canvasses stand out with strong black lines on white ground (occasionally, he allowed for some shade of grey or silver). He never accepted the term "computer art" for his work. If any identification other than just *art* were in need, he would use "algorithmic art".

Recently, Mohr has re-introduced color again. He has totally changed the appeal of his paintings except for one fact: they still possess clearly algorithmic foundation. Mohr's paintings are now in bright colored areas of geometric shape. Their reason is some process happening to the six dimensional hypercube. The complexity of that process needs color to express visually. Figure 2 is a black-and-white reproduction of one of his recent results (Museum für Konkrete Kunst 2001).

Figure2: Manfred Mohr 2001: P-707-F

Different to what we might expect, Manfred Mohr is not interested in another case of "making the invisible visible". He knows that there is no way to visually gain in*sight* into six dimensions. His interest is not didactics but art. His mental jump into the sixth dimension is to create an algorithmic background of high complexity, which he uses to generate surprising two-dimensional events of color and form.

Up to here, the case is not interesting for aesthetic computing. It is not different from similar models that are formally described and transformed into graphic appearance on the display screen.

We now take the algorithmic situation as the starting point for a different consideration. Have a look at Figure 3. It shows the interface of a program, which Matthias Krauss, then a graduate student, wrote overnight after we had discussed Manfred Mohr's art. The focus of the discussion had been how to use inherent features of digital media to create new approaches to works of art. We felt that the dual existence of the work on screen and in computer memory must become the xource for new encounters with the work. We started to look for presentations between canvass and Internet. We wanted to literally *enter* into the picture.

Figure 3, to the lower left, shows a picture typical for Mohr's new genre. Although reproduced here in black and white only, you easily identify polygonal shapes of different levels of grey. You also see black lines, some wider than others.
The screen shot on top repeats a small version of the same picture. To the right, you see a regularly sized pattern of small squares. They are separated horizontally by strong lines, vertically by thin lines.

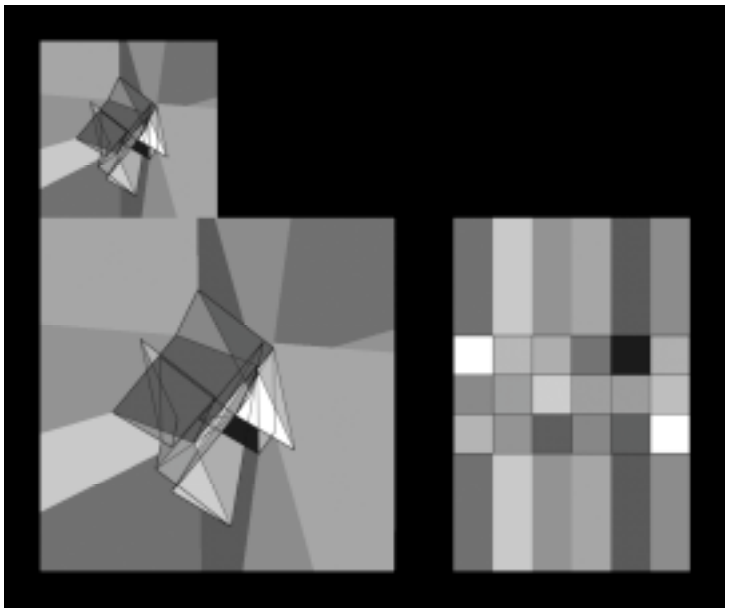

Figure 3: DeviceX

Users of the program get hold of the mouse, move the cursor about, and find out what they can do. They soon discover (no hint given!) that they may grab the small picture on top and push it across the screen, from left to right, and back again. As they move it, its content changes (Figure 4). The sliding image coincides with the large left picture when it reaches its left-most position, and shows the large picture on the right, when moved there.

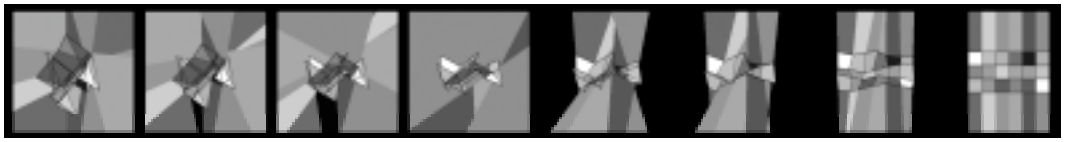

Figure 4: Transformation DeviceX

We have observed students playing with this device (we call it *DeviceX*). They soon discover that the slider continuously transforms the left into the right picture. There must

be a rule governing their relation. The right-hand side displays each one of the colored areas in the same shape and size. Some of them, not necessarily all, are also visible to the left. The transformation suggests that the areas get reduced to standard size and shape without losing their color nor the way they are neighbored to others. Although left and right look totally different, they appear as twins.

The idea behind this arrangement is, to invite people to play with the slider and, in doing so, derive some hypothesis about neighborhoods. The painting in its digital form reveals part of its algorithmic background. Object and interpretation are united. In the usual art situation, the two are canvass and catalogue. Here, work and interpretation, both exist as software and graphics. The graphic appearance is the face of the software.

Before leaving the example, we mention briefly that the software is a bit richer than described. Clicks on lines between areas, or on areas themselves, cause the object to flicker in each of the three presentations. By now moving the device, users may observe what is happening to an area or a line.

Incidentally, the device may be used with (hyper)cubes of any dimension from 3 upwards. Playing with it may, by way of analogy, add further insight. Also available is a wireframe projection of the rotating hypercube. These renditions will at some later time be brought closer to the *DeviceX* presentation, thereby enhancing functionality and interface features simultaneously. – In the following sections, we interpret the example from three different perspectives: the computational, the aesthetic, and the semiotic perspective.

The computational view of the example

In a first step, we look at our case from the perspective of computing. Computers appear as machines to evaluate computable functions. Anything in the world, an object or a process, for which a formal model is given in terms of computable functions may be subdued to computer treatment. Processor and memory swallow computable objects and processes. These exist and reside inside the computer. There, they may be transformed, generate all sorts of off-springs, and generally lead a "life" unnoticed but rich and rapid.

In our example, the external object or process is the idea of Manfred Mohr's to generate some specific and well-defined event in the world of the six-dimensional hypercube and project it down to the two dimensions of the image plane.

Once this process has been described in general terms, it gets turned into a program, complete with input and output routines. The input is needed to tell the operating system at which point in parameter space the computable function (program) is to be evaluated. The output is needed to give us news of the result.

In the particular example, the program possesses a model of the 6D hypercube. It randomly selects one vertex (identified by a 6 bit binary code), and determines its opposing vertex (as the end of a spatial diagonal). The idea is to move from the first vertex to its opposite one along edges. Such a "diagonal path" consists of a sequence of six connected edges. Four diagonal paths are selected. Vertices along each one of them are numbered, and vertices with the same number are connected. This procedure generates quadrilaterals. They are projected onto the image plane.

Quadrilaterals are colored by random choice from a palette. Provision is taken to wrap areas around between the first and fourth diagonal path. Colorization in the image plane is done in the sequence of projection. Therefore, edges of quadrilaterals may cut through color areas, and shapes may turn out to look different.

The case allows pointing out a remarkable fact distinguishing computing from other manipulations of subject matter. What we have described is an *idea*. It exists mentally. Its description may become so complex and complicated that we must use pen and paper to exactly fix its details. From experience we know that we often present an idea fully convinced of the absolute precision of all its details. Only later, when we again work on it, we realize that it was not all that clear. As a mental construct, the idea is clear. When it is externalized, vague parts or even flaws may show up.

The ultimate test for precision and clarity is the explicitness required of a program text. Relative to the expressive power of the programming language, we are forced to spell out everything, or otherwise the computer will do different things or nothing. Programming turns mental constructs into executable descriptions. Programming is the ultimate answer to Peirce's question, "How to make our ideas clear" (Peirce 1940).

The description as such is of linguistic nature. As such it is a sign. Since it is executable on the computer, it itself turns into a machine. "The program runs", we say. It runs invisibly, perhaps filling up memory space with its results. To receive news of its performance, we need a perceivable presentation of the result.

Aesthetic aspects of the example

Aesthetics is concerned with the sensual dimension of perception. The question of whether we like what we perceive, or not, i.e. the question of beauty, is a different one and second only. Sensual perception of the program in the example is bound to the appearance of output on the screen.

The program becomes invisible once it is entered into computer storage. Equally invisible are the objects it is applied to. Output routines transform *internal representations* of computable objects and processes into *external presentations*. Internal representations are invisible to us, but manipulable by the computer. External presentations are visible to us, but not manipulable by the computer.

Let us call objects and processes that become subject matter of computations "algorithmic objects". The above remark indicates that algorithmic objects exist as pairs. The algorithmic object is a pair for an internal representation and an external presentation. Our human interest is focused on the external presentation since we want to see it, point to it, talk about, or communicate it etc. If computation did not enter the game, this would be the end.

But *with* computation things change. They change insofar as the visible aspect of things is pealed off their manipulable aspect. The computer does not operate upon the pixels on the screen. It operates on representations of them in the display buffer, and even further down, there are representations of other entities that form the real stuff of programs. They must be discretized before being pushed into the display buffer. In a very real sense we can say that what is important for us as humans with our contextualized and

situational interest is but a side effect for the computer as a machine with de-contextualized and de-situated mechanics, and vice versa.

The human's interest in having a machine do the computation is to get rid of something that we are, in the long run, not particularly good at. To be sure, we could do all that ourselves. But this is only a statement of principle. We will never do it exactly because we have a machine. The difference here is a decisive and telling one.

Most of the organization of the computable processes rests upon our ability to divide complex tasks up into groups, networks, hierarchies of simpler tasks down to the point of triviality. Whenever this is reached, the machine may take over.

However, there always remain parts that cannot be turned into algorithms, or that cannot even be made explicit. Not only do such parts "remain" as left-overs of mechanization. They may, and actually do, emerge as new implicit tasks and skills. They are all ours to carry out. The machine parts of a joint activity are local, free of context, and independent of situation. The human parts are global, rich of context, and dependent on situation.

The latter is where aesthetics enter. We need a bodily determined sensual access to whatever the machine does. Otherwise, object, processes, or relations that exist inside the computer, don't exist for us.

Without sensual perception of the situation, we remain blind and deaf, literally untouched. We may persuade ourselves into believing that we could think it all up. If you have ever tried it, you know that this is false belief.

So back to the results of the program's operations in the example! The program has determined an internal representation as indicated. It continues to produce the two (or even three) views on the screen (cf. Figure 3).

We call the left part of the screen image a presentation of the geometry, whereas the right part presents, in graphic form, the topology of the 6D situation. To be sure, neither geometry nor topology can be *seen*. They are abstract mathematics. Interestingly, the program now appears closer to our mental efforts (though still infinitely away). It is closer insofar as it, too, cannot see anything. The algorithmic events behind Manfred Mohr's canvass are, however, so complex that only *thinking* of them does not help much.

The aesthetics of the interface enable us to gain access to the algorithmic side of the program. Don't expect the visual quality to already be breath-taking. Figure 3 shows two important aspects. One is: we need the difference of two distinct aspects, we need the possibility of comparison in order to gain insight. We try this with the presentation of geometry and topology. The second aspect is: we need continuous change or movement, best if caused by some bodily movement. We try this with the slider of *DeviceX*.

The coupling of a bodily operation with a sensual perception is so important because it creates a non-symbolic level of experience, some kind of immediacy, which appears to be important for many processes of cognition and insight.

The importance, as we see it, of the aesthetic dimension of computing is not primarily some sort of visualization as such. Of course, the visual presentation is needed if the algorithmic object is to be seen instead of listened to. Visualization, therefore, is but a

trivial aspect of aesthetic computing. Mathematicians, medical doctors, all kinds of scientists have always known about the advantages of visualizations. If visualization is prepared algorithmically, that is fine and much needed. But it is what you would expect.

But if aesthetic computing is to make use of the tremendous depth of the aesthetic dimension, the external presentations of algorithmic objects must themselves become the focus of art and research. As representation of algorithmic objects they may not be allowed to lose touch with their internal representations. Aesthetic computing would need to take up sensual facets of mental constructs.

The introductory example suggests appearance of difference, and continuous transformation caused by manual operation, as two components of the dialectics of aesthetics and computing.

Semiotically looking at the example

Our third, and last, look at Manfred Mohr's art under the grip of DeviceX is the most abstract. It is the semiotic perspective. We propose that the nature of objects and processes in computing is semiotic, and that their semiotic nature shows even more when we look at computing under an aesthetic perspective.

The internal representations of algorithmic objects are semiotic in nature, and so are the external presentations. This claim must be substantiated. If we succeed in doing this, semiotics would offer common ground to aesthetics and computing alike.

Observe what is happening when a user grabs the slider (with the mouse) and drags it along the top row of the display, from left to right and back. As she is doing this, the image "in" the slider changes instantaneously even when the dragging is done quickly. This means that an interpolation is taking place between the left-most and right-most picture. It depends on the current position of the slider.

We mention in passing that we will use a bit of the terminology of semiotics as founded by Charles Sanders Peirce. An easily available source is (Peirce 1992, 1998). What does the slider's position stand for? The data provided by that position is taken as the representamen (Peirce) of a sign. Call that data x and assume, x is a value between 0 and 1. The program then calculates a new external presentation $P(x) = (1 - x)\ P(0) + x\ P(1)$, where P(0) and P(1) are the left and right extremes.

The semiotics of the situation are such that movement of the mouse/slider-pair creates, in every moment, a number x which becomes the *representamen* of a sign. This sign is created by the program's interpolating computation. The result of this computation becomes the *object* of the sign. Since this operation is determined without any freedom of interpretation, the sign's *interpretant* (roughly, its meaning) coincides with its object.

The newly constructed object gets displayed, i.e. externalized. Technically speaking, the new appearance of the little image gets stored in the display buffer, partially replacing the old buffer contents. The display processor, of course, immediately creates the new screen image.

For the human observer, the internal sign *object* becomes a new external *representamen*. It is what she perceives. She is immediately – in fact: permanently – re-interpreting what

she sees. Chances are that the object of the sign, as she constitutes it, remains almost invariant, at least after some first interaction. But her construction of an *interpretant*, the most important component of the sign, changes under the influence of the newly appearing image.

We thus come to the conclusion that the process of observing *DeviceX* (and, through it, the work in Manfred Mohr) is a complex relation between the human's interpretation and the program's determination of a semiotic process. The human continually observes the screen image. She must gain the impression that it is her who is changing the image. The visible appearance on the screen is largely determining her perception.

The computer, in the meantime, is constantly active. Without its activities, nothing would happen. It permanently produces internal representations and displays them immediately. This immediacy, enhanced by the coupling of manual operation and mental cognition, convinces the user that it is her, who generates all the changes.

Interactive use of software: the interface as sign

We now leave the example for some generalization. We first restate the situation as described in the preceding section. We observe two autonomous systems. One system is the human user. The other one is the computer under control of the currently active program. The two systems are autonomous only in our analysis. But in a way, we may have good reason to assume some sort of autonomy even on behalf of the machine. We do not further go into this argument here but want to emphasize that, from the typical non-expert user's point of view, such a perspective seems well justified.

The two autonomous systems are engaged in their respective activities. In the case of the human, these activities may become arbitrarily complex. On the computer's behalf, they may be as restricted as running the operating system in order to control all current user operation.

Since, obviously, the two systems coordinate their different operations, there must be some coupling between them. This coupling is of semiotic nature. Insofar as the two systems coordinate their operation, they are engaged in semiotic processes. The user is observing the screen and taking new measures according to her interpretation and interest. The screen appears to her as a complex sign (the *representamen* of a sign, to be more precise). Figure 5 shows a typical example. That complex sign (fully created omly in the user's head) causes her to execute certain operations that all end in signals as inputs to the program. The program (or some part of it) reacts by updating the screen according to the user's external moves (feedback), and by executing an internal operation triggered by the input signal. We have described this in more detail elsewhere (Nake 1994).

Figure 5: Desktop

For the current purpose it should suffice to say that a genuine and a restricted sign process get coupled in the interactive use of the computer. The genuine sign process is the user's. Let us simply call it, the *sign process*. The restricted sign process is the computer's. It is restricted insofar as no full signs, but only signals, are treated here (signals are signs with a trivial interpretant: the interpretant is equal to the sign's object). Let us call this the *signal process*.

So human-computer interaction turns out to be the coupling of a sign process and a signal process. What is called the interface is the place and location of their coupling. It appears to the user as a complex compound sign that is easy to interpret and to change. At least this is what should be the case. The pliability Jonas Löwgren emphasizes so much as an important use quality, here appears as the ease of semiotic change.

The interface appears as the representamen of a sign of great dynamics. Simple operations on behalf of the user – as, e.g., moving the cursor sign by moving the mouse, or selecting an item from a menue list by clicking on it – may be carried out "inside" that interface sign. Any such operation results in an immediate change of the representamen of the sign. This means that the interface acts somehow like a permanently changing billboard onto which both systems draw and write.

Interactive use of software: an aesthetic event

The perception of art in the form of one person in front of a painting in a gallery is clearly no mass event. It requires a delicate response. The response of an admirer of works of art in a gallery will usually be totally different from that of an addict to Internet surfing or gaming in some private home. And yet, there is common ground to their experiences.

Different as the sensations and emotions may be, it is aesthetics that catches or repulses our two friends. Film, video, or GUI may be reacted to in very elegant and delicate intellectual manners. Contrary, however, to much of the aesthetics of art, the aesthetics of interface shows in mass reaction. It may be compared to the aesthetics of advertising and packaging consumer products of mediocre quality.

Interface aesthetics is, at the same time, different from packaging aesthetics. The interface to software *belongs* to the software. Software never appears without interface. The human-computer *inter*face is, first of all, the *face* of its software. In fact, the semiotic analysis has exactly emphasized the point that the interface, considered as something *between* two systems, largely disappears. If we assume the interface as an important, but in some way separate component of computing potentials, we render software faceless. But software cannot exist without face. We take notice of the existence of a piece of software only when it is allowed to show its face. This currently takes place on the display screen, mostly visual, and a bit acoustic.

It has not always been that way, nor will it necessarily remain so. In the days, when computers filled rooms, you carried a stack of punched cards to the machine, and waited for some other machine to print out results. You were not totally wrong when you said that the stack *was* your program. By that time, the program was still a tangible, local, and individual *thing*. Only gradually it revealed its *media* qualities. They now appear as its face. "Interface", we tend to believe, is a concept better akin to the world of machines than to that of media.

We gain a different approach to the design of software if we recognize software as always possessing a face. Design of software artifacts then includes design of the face of

software. The interface between human beings and software artifacts disappears as a material thing. It re-appears as a semiotic process. As such it is deeply entangled in aesthetics. The aesthetics of computing appears as *part of*, not as an addition to, the design of software.

Like any other human activity extending beyond pure survival, computing allows for two socio-technical relations: (i) designing and producing artifacts for others to use; (ii) using artifacts designed by others.

Computing relies on two kinds of artifact, hardware and software. We adopt the position here that the hardware is given, and software is the matter of our discussion. From the point of view of a user, her situation is not all that different from that of the visitor to a gallery. The gallery visitor enters one of the rooms. She casually walks to one of several paintings on the walls. She puts herself into a position favorable for what she wants to do: stare at the painting.

Now compare, on this superficial level, the software user's case. She casually moves up to her desk and positions herself favorably for what she is about to do. Sitting down, she generates a few mouse clicks, waits for a moment or two, and then: stares at the screen.

Once the staring positions have been attained, similarities between the two use-of-an-artifact situations end. The art lover may be thrilled by what she stares at. She approaches the canvas for a closer look, reads a sign next to the picture, talks to someone, steps

back again, scans the canvas with her eyes, engages in a lot of physical and cognitive actions before she moves on – but never touches the artifact.

The software user, quite to the contrary, remains relatively stable in her chair. She grabs the mouse, moves it around, clicks, hits some of the keys readily available in front of her and, most likely, neither touches the artifact (except for keys and mouse). Her distance to the stared-at object, the screen, is much shorter than in the gallery. Her eye movements are fewer, perhaps, and over shorter distances. But her involvement is likewise great and challenging. Let us identify the difference.

The painting in the gallery remains invariant, no matter how often it gets stared at. The mental images, or ideas, which visitors may develop and carry away, however, change and are different in all cases. On the other hand, the screen image on the display changes almost permanently. Its changes influence the mental images of users. However, an advanced or even expert user will have quite a stable idea of the software, its appearance, and its functions. Even though she is usually not physically touching the screen, metaphorically she is doing just this. She "opens" some icon (standing for a "folder", standing, in turn, for a data file, standing for a well organized portion of physical memory). She "pulls down" a menue (again, standing for something). She redefines the value of some parameter, and does all sorts of other things. Each one of these operations has a double effect: the state of the hardware is changed, and the visual (or acoustic) appearance of the display is also changed, if only temporarily.

Both of the spectator situations, which we so negatively called „staring at", are blatant cases of aesthetics. Our subjects are confronted with a framed image of something and if

they don't make sense of it, they get lost. Not many will refuse to think of the gallery situation as one of aesthetic relevance. Perhaps not many will immediately accept the idea that the software situation is also one of aesthetic relevance. But it is. It is aesthetics insofar as we cannot even take notice of what we are interested in, unless we concentrate on the aesthetics of its appearance.

When the cultural impact of computers was still limited to complex or voluminous calculations in places like research laboratories, universities, and some first administrations, computers could still be taken as what their roots in the productive forces of society suggested: as machines of a special kind. Their aesthetics was absolutely irrelevant, superficially added at best, a separate, idle, and uninteresting consideration. This applied to all components of computer systems, hardware and software.

This situation has changed dramatically. With the advent of the personal computer in the early 1980s, the characterization of software as a *tool* gained a material basis beyond mere metaphor. The tool period did not last that long. It created a tremendous wealth of research results in the field of HCI ("software ergonomics" in parts of Europe). But it soon had to give in to the *media* period. When the globally connected computer raged around the world, it disclosed its hidden media quality. Being digital is not, in our view, the decisive feature of digital media. It is rather their dual quality of instrumental and medial properties, which single out computer systems. We have used the term *instrumental medium* to acknowledge this (Schelhowe 1997).

As instrumental media, hardware/software systems are driven by their own dialectic. They can no longer be understood nor designed as two (or even more) separate components:

function and interface. Digital media, the means of computing, have gained their *eigen*aesthetics. To acknowledge this in all consequence is what aesthetic computing may be all about.

It is hard to imagine that aesthetic computing could spread and thrive without taking notice of Walter Benjamin's seminal essay, *Art in the age of its mechanical reproduction*. Nadin (1997), referring to Benjamin's central notion of *aura*, draws our attention to the shift away from the artifact itself to the process of art. Something similar is true to signs and media: they are determined as processes more than as objects.

The artifacts themselves, which aesthetic computing may become concerned with, do not appear different from those computing is generally concerned with. The way they get treated will, we expect, be different. Aesthetic computing sounds like the call for a shift in attitude towards the distinction of hard and soft science. Truth would no longer, even here, be expected from strict formalism only but also from vague aspects of style.

Acknowledgement. We happily acknowledge the continued exchange with Matthias Krauss. He has contributed *DeviceX,* but much more to our joint project, compArt.

References

Bolter, Jay David, & Grusin, Richard (2000), Remediation. Understanding new media. Cambridge, MA: MIT Press

Ehn, Pelle (1998), Manifesto for a digital Bauhaus. Digital Creativity 9. 207-217

Höök, Kristina; Sengers, Phoebe, & Andersson, Gerd (2003), Sense and sensibility: evaluation and interactive art. CHI 2003 Conference Proc., ACM Press. 241-248

Löwgren, Jonas (2004), Articulating the use qualities of digital designs. In: Paul Fishwick, ed., this volume

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

Manovich, Lev (2003), New media from Borges to HTML. In: Noah Wardrip-Fruin & Nick Montfort, eds., The new media reader. Cambridge, MA: MIT Press. 13-25

Mullet, Kevin, & Sano, Darrell (1995), Designing visual interfaces. Englewood Cliffs, NJ: Prentice Hall

Museum für Konkrete Kunst (2001), ed., Manfred Mohr space.color. Catalogue to the exhibition. Texts, in German and English, by Manfred Mohr and Frieder Nake. Ingolstadt: Museum für Konkrete Kunst

Nadin, Mihai (1997), The civilization of illiteracy. Dresden: University Press

Nake, Frieder (1994), Human-computer interaction: sign and signals interfacing. Languages of Design 2. 193-205

Nees, Georg (1969), Generative Computergrafik. München: Siemens

Packer, Randall, & Jordan, Ken (2001), eds., Multimedia. From Wagner to Virtual Reality. New York: W.W. Norton

Peirce, Charles S. (1940), How to make our ideas clear. In: Justus Buchler, ed., Philosophical writings of Peirce. New York: Dover Publ. 23-41 (originally 1878)

Peirce, Charles S. (1992, 1998), The essential Peirce. Selected philosophical writings, 2 vols. Ed. By Nathan Houser & Christian Kloesel, and the Peirce Edition Project. Bloomington: Indiana University Press

Raskin, Jef (2000), The humane interface. New directions for designing interactive systems. Reading, MA: Addison Wesley

Schelhowe, Heidi (1997), Das Medium aus der Maschine. Zur Metamorphose des Computers. Frankfurt, New York: Campus

Wiesing, Lambert (1991), Stil statt Wahrheit. Kurt Schwitters und Ludwig Wittgenstein über ästhetische Lebensformen. München

Winograd, Terry (1996), ed., Bringing Design to Software. Reading, MA: Addison Wesley