Screen-Based Musical Interfaces as Semiotic Machines

Thor Magnusson

Creative Systems Lab
University of Sussex
BN1 9QN, East Sussex, UK
T.Magnusson@sussex.ac.uk

ABSTRACT

The ixi software project started in 2000 with the intention to explore new interactive patterns and virtual interfaces in computer music software. The aim of this paper is not to describe these programs, as they have been described elsewhere [14][15], but rather explicate the theoretical background that underlies the design of these screen-based instruments. After an analysis of the similarities and differences in the design of acoustic and screen-based instruments, the paper describes how the creation of an interface is essentially the creation of a semiotic system that affects and influences the musician and the composer. Finally the terminology of this semiotics is explained as an interaction model.

Keywords

Interfaces, interaction design, HCI, semiotics, actors, OSC, mapping, interaction models, creative tools.

1. INTRODUCTION

In our work with ixi software [14][15], we have concentrated on creating abstract screen-based interfaces for musical performance on computers. These are graphical user interfaces (GUIs) that do not necessarily relate to established conventions in interface design, such as using buttons, knobs and sliders, nor do they necessarily refer to musical metaphors such as the score (timeline), the keyboard (rational/discrete pitch organisation) or linear sequencing (such as in step sequencers or arpeggiators). Instead we represent musical structures using abstract objects that move, rotate, blink/bang or interact. The musician controls those objects as if they were parts of an acoustic instrument, using the mouse, the keyboard or other control devices. We have created over 15 of these instruments – each exploring new modes of interactivity where some of the unique qualities of the computer are utilised in fun, inspirational and innovative ways. Qualities such as remembering the musician's actions, following paths, interaction between agents, generativity, randomness, algorithmic calculations and artificial intelligence; all things that our beloved acoustic instruments are not very good at.

Over the course of our work, we have developed a loose and informal language for these instruments – a semiotics that suggest to the musician what the functionality of each interface element is, and what it signifies in a musical context. Human Computer Interface (HCI) research [2][3][17][1][6] is usually concentrated on the chain of meaning from the software designer to the software user. The user is the *receiver* of information and the aim of HCI is traditionally to make the interaction between the two systems (the human and the computer) intuitive, representational and task based (where the tasks are based on real world tasks). What is lacking is a stronger discussion of the situation where the computer is used as a tool for artistic creation – an expressive instrument – and not a device for preparing, organising or receiving information. In artistic tools we have an important addition, where the signifying chain has been reversed: the meaning is created by the user, deploying a software to achieve some end goals, but this very software is also a system of representational meanings, thus influencing and coercing the artist into certain work patterns.

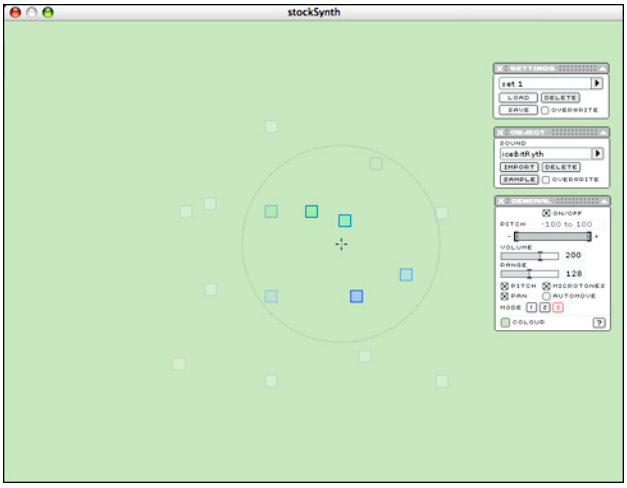

Figure 1: StockSynth. Here the crosshair cursor serves as a microphone that picks up sounds from the boxes that represent sound samples. The mic has adjustable scope (the circle). The boxes are moveable and the mic moves by drawn or automatic trajectories or by dragging it with the mouse.

2. A SHORT NOTE ON INSTRUMENTS

"Even simple physical instruments seem to hold more mystery in their bodies than the most elaborate computer programs" [10]

Both acoustic instruments and music software incorporate and define the limits of what can be expressed with them. There are special qualities found in both, but the struggle of designing, building and mastering an acoustic instrument is different from the endeavor of creating musical software. The acoustic instrument is made of physical material that defines the behaviour of it in the form of both tangible and aural feedback. These material properties are external to our thought and are something that we fight with when we design and learn to play instruments. Such features or characteristics of the material instrument are not to be found in software. Software is per

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
NIME 06, June 4-8, 2006, Paris, France.
Copyright remains with the author(s).

definition programmed (etymology: "pro" = before, "graphein" = written); its functionality is prewritten by a designer or an engineer and the decisions taken in the design process become the defining qualities of the software, determining its expressive scope.

Different languages are based on different paradigms and lead to different types of approaches to solve a given problem. Those who use a particular computer language learn to think in that language and can see problems in terms of how a solution would look in that language.[1] [12]

This is not the place to go into the cognitive processes involved with learning and playing an instrument. But we are faced with an important question: what material (instruments) is the computer musician composing for and where does he or she get the ideas from? In other terms: where does the thinking (or composing) of the computer musician or digital instrument inventor take place? It happens most likely in the form and structure of the programming language in which he or she is working. The environment defines the possibilities and the limitations of what can be thought. But what does it mean to "learn to think in a language"? What are we gaining and what are we sacrificing when we choose an instrument or a programming environment? And what are the reasons for some people preferring one environment for another?

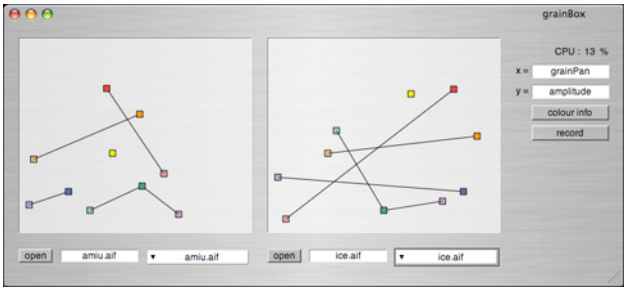

Figure 2: GrainBox. It can be hard to create interfaces for granular synthesis. The GrainBox is a suggestion how to represent the complex parameters as boxes with X and Y dimensions in 2D space and with connections to other parameters such as reverb and random functions.

When musicians use software in their work, they have to shape their work process according to the interface or structure of the software. As with acoustic instruments software defines the scope of potential expression. The musician is already tangled in a web of structured thinking but the level of freedom or expressiveness depends on the environment in which he or she working.[2] To an extent, the musical thinking takes place at the level of the interface elements of the software itself.

It is misleading then to talk of thinking as of a 'mental activity'. We may say that thinking is essentially the activity of operating with signs. This activity is performed by the hand, when we think by writing; by the mouth and larynx, when we think by speaking; and if we think by imagining signs or pictures, I can give you no agent that thinks. If then you say that in such cases the mind thinks, I would only draw attention to the fact you are using a metaphor, that here the mind is an agent in a different sense from that in which the hand can be said to be the agent in writing.

If again we talk about the locality where thinking takes place we have a right to say that this locality is the paper on which we write or the mouth which speaks. And if we talk of the head or the brain as the locality of thought, this is using the 'locality of thinking' in a different sense. [21]

If here I am attempting to find the "locus" of musical thinking/performing in both acoustic instruments and screen-based digital instruments – a discussion that is much deeper than can be delved into here – it is important to consider the difference in embodiment and incorporated knowledge of the player in those two types of instruments. When learning an acoustic instrument, the motor memory does most of the job and your learning "happens" as interaction with the body of the instrument. Due to the material qualities of it, one can never master an instrument, it always contains something unexplored, some techniques that can be taken further and investigated. With software however, it is more or less visual and procedural memory that is involved, as software doesn't have a material body that the musician learns to operate. The only “body” of software is in the form of its interface elements, and they (as opposed to the indicative nature of physical material) are simple, contingent and often arbitrary design decisions.[3] The "body" of the software has to be created and it does not depend upon any material qualities, but rather the style and history of graphical user interface design.

3. HCI AND SEMIOTICS

Designing is essentially a semiotic act. Designing a digital instrument or programming environment for music is to structure a system of signs into a coherent whole that incorporates some compositional ideology (or an effort to exclude it). The goal is to provide the users with a system in which they can express themselves and communicate their ideas in a way that suits their work methods and sometimes provide new ways of thinking and working. But what kind of a tool is the computer and what kind of communication are we talking about here?

3.1 Interaction Paradigms

We can roughly define three primary **interaction paradigms** in computer software as: *computer-as-tool, computer-as-partner,* and *computer-as-medium.* [6] Different research communities address these paradigms. The HCI field investigates the computer-as-tool paradigm but the attention is mainly on how to design understandable and ergonomic software for the user of the tool. What is lacking is a better understanding of creativity itself and how creative and experimental minds use software (and often have to misuse it to get their ideas across). We have learned from user feedback that there seems to be a general need for better sketching environments that can be modified according to the needs of the user. An interesting fact here is that many cutting-edge art works are created by hacking or modifying software or simply creating one’s own tools. There are schools of artists that respond to the limitations of commercial software with their own software in the form of software art.[4] [11][16]

[1] Try to replace "language" with "instrument" in McCartney's paragraph above – the same applies for musical instruments as well.

[2] From this perspective SuperCollider and Pure Data are arguably more open and free than Logic, Protools or Reason, to name but a few.

[3] Often made by the wrong people: an engineer and not an ergonomist; a graphic designer and not a musician.

[4] The www.runme.org repository is an excellent source for information and examples of what is happening in the field of software art and generative art. It is closely related to the ReadMe festival, which was the first software art festival.

3.2 The Semiotics of a Creative Tool

The most common of semiotic practises is to look at the signifying channel from the sender to the receiver through some medium such as signs, language, text, or film. [5][9] The "work" here is a static construction that doesn't change after it has been published or released.[5] By contrast, computer-based works are interactive and can be changed or modified after their release either by users themselves or by updates. *Interaction becomes a new sign-feature.*[2] Some studies have been done on this new semiotic quality of the computer [1][2][3][7], but very few in the field of music software or other creative software.

In music software, the user is at the same time *the receiver* and interpreter of information from the designers of the software and *the sender* of information in the form of the music being composed using the tool. This dual semiotic stance is important in all tools (whether real or virtual) but becomes vital in contingently designed tools such as music software. Music software is a sign system in its own right, but the important question here is: which are the relevant layers of signification and communication and from where do the originate? This can be analysed into strata of different practices. The hardware designers, the programmers of the compilers, the language API and the software itself, the designers of the interaction and the programmers of the interface. A creative tool has history of important design decisions all shaping its scope and potential. This is a complex structure, but the user is faced with the question: what is the meaning conveyed in the interface? And is this system of signification not essentially of compositional nature? Who took those decisions and by which criteria?

The contingency of design mentioned above in relation to the digital medium is one of the most definable characteristic of it. We don't have this "contingency problem" when designing acoustic instruments as the properties of the material we work with leads us in our design: closing a hole in a flute increases the wavelength in the resonant tube and the tone deepens; pressing the string against the fingerboard of a guitar – shortening the wavelength – produces a note of higher pitch. When designing screen-based computer interfaces we can choose to imitate physical laws as known from the world of acoustic instruments, we can draw from the reservoir of HCI techniques or we can design something entirely new. It is here that interface design, the interaction design, and mapping becomes very important factor in the creation of interesting screen-based instruments for the computer.

4. INTERFACE ELEMENTS IN IXI

Most modern operating systems are graphical or allow for a graphical front end. The WIMP (Window, Icon, Menu, Pointer) interface [4] has become a standard practice and we have become used to the direct manipulation [20] of graphical objects. The traditional method is to translate work practices from the real world into the realm of the computer, and thus we get the folders, the documents, the desktop and the trash. In music applications we get representations of keyboards, buttons knobs and sliders, rack effect units and cables. This is also suitable where the aim is to translate studio work practices into the virtual studio. But when we are creating new instruments using the new signal processing capabilities and artificial intelligence of the computer, there might not exist any physical phenomena that we can use as source for our interface metaphors.[6]

4.1 Interaction Models

Each of the ixi applications is a prototype or a suggestion and it explores a specific mode of interaction. The whole of our software can be grouped into a specific kind of **interaction model**: a language, a semiotics or a design ideology that informs and en-forms the work. An interaction model can be defined as more operational than an interaction paradigm (computer as tool, partner or medium). [6] It can be evaluated according to the descriptive, the evaluative and the generative power of the model. These dimensions of evaluation are all important when creating an interaction model. The descriptive power is the ability to describe a significant range of existing interfaces; the evaluative power helps us to assess multiple design alternatives; and the generative power is the ability of the model to inspire and lead designers to create new designs and solutions.

4.2 Interaction Instruments

It is the generative aspect of ixi's interaction model that is the subject here. Beaudouin-Lafon's definition of *instrumental interaction* [7] is the closest description the author has found that relates to our work with ixi software. The interaction instrument is a tool that interfaces the user with the object of interest. A scrollbar is an example of such instrument as it gives the user the ability to change the state/view of the document. A pen, brush or a selection tool in a graphics package is also a type of such instrument.

There are three design principles that define the methodology of instrumental interaction: *reification* - the process by which concepts are turned into objects; *polymorphism* - the property that enables a single command to be applicable to objects of different types; *reuse* - the storing of previous input or output for another use. When an ixi application combines all three design principles into a successful interface, we have what we could call a semiotic machine. The interface is multifunctional and can be used in a variety of different contexts.

4.3 The Terminology of ixi's semantics

As explained in earlier papers, [14][15] most of the ixi software applications are controllers that send and receive OSC (Open Sound Control) [23] information to sound engines written in other environments such as SuperCollider [12] or Pure Data [17]. We separate the interface from the sound engine in order to be able to reuse the control structures of the abstract interface in other contexts, for example allowing a sequencing interface to control parameters in synthesis if the user configures it so. These controllers are all made from a common ideology or an interaction model that we see as a semiotic system.

In our work with *ixi software*, the fundamental attention has been on the interaction design and not the interface design. The design of interface elements is often highly (but not exclusively) aesthetic and depending on taste, whereas the interaction design deals with the fundamental structure and ergonomic idea of the software. In the example of SpinDrum [14], for example, the wheels contain pedals controlling beats per cycle, the size of the wheel signifies the volume and the

[5] Post-structuralist thought has rightly pointed out how interpretations of the work change in different times and cultures, but the work itself doesn't change - only people's interpretation and reception of it.

[6] As we can derive from the Peircian semiotics, an interface object can be represented in various ways: *iconically* (where the representation is based on resemblance to an object), *indexically* (where the representation is influenced by an object) or *symbolically* (where the representation is based on convention).

colour accounts for which sound is attached to the object. Here the interaction design clearly affects the interface design (size, number of pedals, colour), but the shape of the pedals (whether a square, a circle or a triangle) is simply an aesthetic decision and of little general importance.

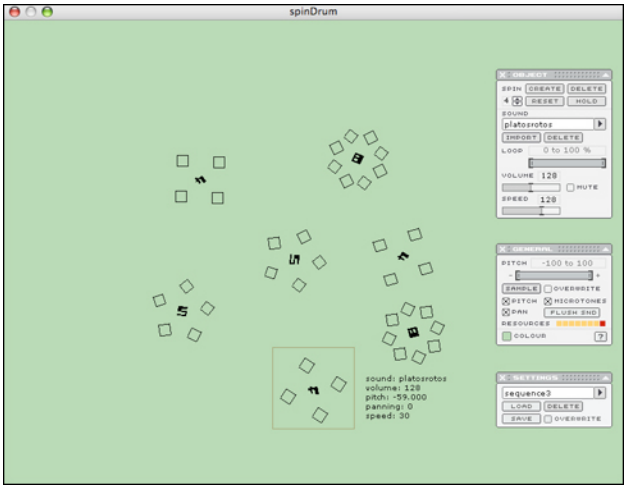

Figure 3: SpinDrum. Each wheel contains from 1 to 10 pedals. The wheels rotate in various speeds, and when a pedal hits top position (12 o'clock) it triggers the sample or sends out OSC info to the soundengine. The X and Y location of the wheels can affect parameters such as pitch and panning.

4.3.1 Actors

The ixi interfaces are pattern generating machines with cogs and bolts of varied significance. To sum up the basic design ideas of ixi software we could say that it was the *reification of musical ideas into abstract graphical objects as control mechanisms that act in time*.[7] We call these abstract objects *actors,*[8] as they are graphical representations of temporal processes that act, enact and react to the user, to each other or the system itself in a complex network of properties, relations and teleology (desired states or end goals). Beaudouin-Lafon calls graphical interface tools "interaction instruments", but we cannot use that metaphor as an ixi application is a musical instrument on it's own but also because of the different nature of the interface units of ixi software. The feature under discussion here is the difference musical applications have from the *ergonomically* "single-threaded" or serial task-processing applications used for painting, text editing, programming, video editing or in architecture. In contrast to these applications, a music application is multi-threaded or parallel, i.e. there are many processes, streams, layers or channels that run concurrently in every composition or performance, all controlled by the user, but, in the case of ixi, usually only one at a time.[9]-[10]

The interface units that we call actors - such as a picker, a spindrum or a virus - are not instruments that the musician uses for some task and then chooses another instrument for the next task. The actors in the ixi software applications are put into use at some point in time and they continue working in a temporal flow (rotating, moving through a trajectory or interacting) until the musician decides to stop or pause their activities.

4.3.2 Context

All actors perform their task in a context. They are graphically represented in a two- or three-dimensional space on the screen and their location might typically influence their properties. The actors move, rotate or blink in this space and are therefore both spatially and temporally active units. The space can have qualities such as temperature, gravity, brightness, etc. which are all qualities that could affect the actor's behaviour or it can contain other actors of different type that influence the behaviour of the message sending actors. Feedback from users of ixi software has shown us that people find the metaphor of an actor presented in time and space useful to represent musical actions and ideas. What the feedback also shows is that people intuitively understand the metaphor of having actors on a stage that perform some tasks that they – the directors of the piece – are controlling.

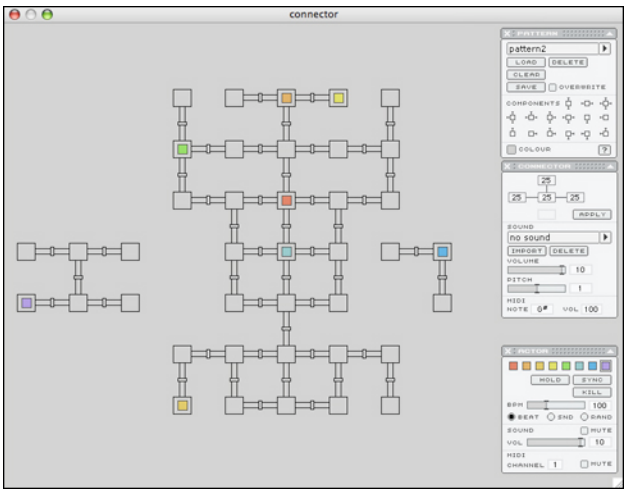

Figure 4: Connector. This software uses generative algorithms to decide where *actors* travel within a network of *connectors*. There are probability charts that decide the next move of an actor and when it enters a connector it triggers a MIDI note and/or a sound sample that is a property of the connector.

4.3.3 Network

When talking about the context and the environment of these actors, we must note the fact that the interface elements are not the only actors in the context of an ixi instrument: the user is one actor, the control hardware (a mouse, keyboard, sensor or controller), the soundcard, the speakers and other communication such as virtual audio cables, MIDI or OSC messages. The whole context of musical action and reaction is the space of the actor, a space in which the heterogeneous network of musical performance takes place. The meaning of the actor is its functionality within the control context and the mapping context. The actor has as many dimensions as it has numbers of control parameters and connections for receiving or sending messages.

---

[7] Musical idea here meaning any pattern generating structure.

[8] We thought about calling the active interface elements *agents* but it was too confusing as the term has very strong connotations in computer science, especially within the field of artificial intelligence.

[9] Another fact that divides those types of software is that the painting software, the video software or the 3D package are not packages that are used in live performance.

[10] This is of course what people are working with in the research field often known as NIME (New Interfaces for Musical Expression www.nime.org) where building physical interfaces to control sound on the computer allows for multi-parameter mapping to one sound-engine.

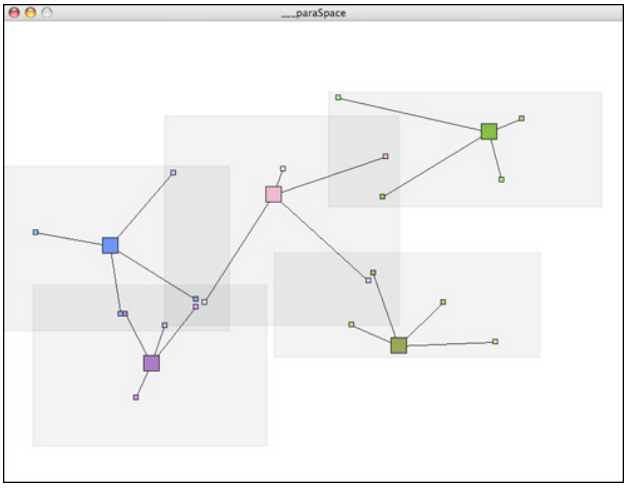

Figure 5: ParaSpace. This application interfaces with audio effects written in SuperCollider (but can talk to any software that supports OSC). Each audio effect has variable number of parameters and they are represented as small boxes in the control interface of ParaSpace. The point here is that the parameters interact on the interface level with automation, artificial life and artificial intelligence.

To clarify this idea of actors being all the elements that affect the interaction in an instrument, let us have a look at the software *Connector*. Here actors move in a system of connectors (a plumbing-like system) and trigger sound samples or MIDI notes that are properties of the connectors. The connectors are actors themselves as they are the receivers of an action and contain the information that yields the sound. It is through the interaction of all the actors and their properties that interaction takes place – interaction between elements *within* the instrument and also with the musician using the instrument – and this interaction is simply the automation that controls the various parts of the music set into motion. In *StockSynth* (Figure 1) the microphone is one such actor (with its properties of trajectory and scope) that interacts with the sound objects that contain the information about the sound and its properties.

4.3.4 Semiotic elements and mapping

The actors and the contexts in which they function are all elements in a semiotic language. This language has dialects or rather idiolects (each application is unique) where the meaning of an element can change as in Wittgenstein's concept of the usage as the word's meaning [15][22] or as in the Saussurian conception of the lack of natural connection between a signifier and the signified.[11] [19] We provide a semiotics or suggest language games where the behaviour of an actor maps onto some parameters in a sound engine. For example, vertical location of an actor could signify the pitch of a tone or playback rate of a sample. Size could mean amplitude, rotation triggering, and direction could mean a tendency for some action. But, it could also signify something entirely different as the controllers are open and it is up to the musician to map the actor's behaviour onto a parameter in the sound engine.

5. CONCLUSION

This paper has tried to show how the materials we work with when we design instruments (digital or acoustic) are the foundation for what can be expressed with the instrument. Whereas the expressive possibilities of an acoustic instrument are highly dependent upon the physical material it is built out of (wood, iron, strings, etc.), the situation is very different when we create digital instruments, especially screen-based. We have shown some examples of the semiotic system we are working towards in our work with ixi software and suggested a terminology of actors, context and network to better understand and modularise the interaction and interface design of virtual instruments. We have also illustrated how an interface can have its own meaning system independent of its relationship to the sound-engine, where the interactive patterns of an instrument can be mapped in many different ways onto the parameters of the sound-engine.

6. FUTURE WORK

Future plans involve exploring the dimensional spaces of the screen-based actors as the interface for musical interaction. The computer is becoming quite good at imitating the properties of acoustic instruments but it excels as an interesting instrument on its own where interaction is designed from the premise of the qualities of the computer and not by imitation of real world objects.

Our work involves experimenting in creating semiotic systems that can be taken further and extended into new dialects and systems of meaning. This system is not exclusive to one type of applications, but can rather be seen as a semiotic toolbox from which elements can be taken and reused in new contexts. Computer music software is a highly interesting area in the field of HCI as it is used in live performances and should contain depth that can be explored and practiced, thus allowing for musical virtuosity. In semiotic interfaces such as the ixi software there is always the filament of concurrent mappings or parallel streams of musical events happening at any one time. The temporal aspect of computer music software makes it also quite unique in relation to other types of software. Facing these incredible demands and challenges of music software we feel that we are just starting our journey into the possibilities of new meaning systems, metaphors, pattern generators and control of synthesis techniques through the creation of semiotic machines in the form of interfaces.

7. ACKNOWLEDGEMENTS

The *ixi software* project (www.ixi-software.net) is a collaboration between Enrike Hurtado Mendieta, Thor Magnusson and various musicians and artists that are involved with our work. I would like to thank Chris Thornton of University of Sussex, Marcelo Wanderley and the people at the IDMIL Lab at McGill University, Montreal and Julian Rohrhuber at the University of Cologne for good discussions and important feedback for this paper.

8. REFERENCES

[1] Andersen, Peter, B. & May, Michael. "Instrument Semiotics" in *Information, organisation and technology. Studies in organisational semiotics*. (eds. Liu, Kecheng; Clarke, Rodney J.; Andersen, Peter B.; Stamper, Ronald K.). Kluwer: Boston/Dordrecht/London. 2001: 271-298.

[2] Andersen, Peter, B. "What semiotics can and cannot do for HCI" in *Knowledge Based Systems*. Elsevier: 2001.

[3] Andersen, Peter, B. "Computer semiotics" in *Scandinavian Journal of Information systems*. Vol. 4: 3-30. Copenhagen, 1992.

[11] For Saussure, the meaning of signs is relational rather than referential, i.e. the meaning lies in their systematic relation to other signs in the semiotic system and not by direct reference to material things, for example.

[4] Apple Computer Inc. *Human Interface Guidelines: The Apple Desktop Interface*. Boston: Addison-Wesley, 1987.

[5] Barthes, Roland. *Mythologies*. London: Paladin, 1972.

[6] Beaudouin-Lafon, Michel. "Designing Interaction, not Interfaces" in *AVI Proceedings*, ACM: Gallipoli, 2004

[7] Beaudouin-Lafon, Michel. "Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces" in *Proceedings of ACM Human Factors in Computing systems (CHI 2000)*, The Hague: ACM Press, 2000.

[8] Beaudouin-Lafon, Michel. "Reification, Polymorphism and Reuse: Three Principles for Designing Visual Interfaces" in in *AVI Proceedings*, ACM: Palermo, 2000

[9] Eco, Umberto. *A Theory of Semiotics*. London: Macmillan, 1976.

[10] Edens, Aden. *Sound Ideas: Music, Machines, and Experience*. Minneapolis: University of Minnesoda Press, 2005.

[11] Gohlke, Gerrit. (ed.) *Software Art - A Reportage about Source Code*. Berlin: The Media Arts Lab, 2003.

[12] McCartney, James. "A Few Quick Notes on Opportunities and Pitfalls of the Application of Computers in Art and Music" in *Ars Electronica*, 2003.

[13] McCartney, James. "Rethinking the Computer Music Language: SuperCollider" in *Computer Music Journal*, 26:4, pp. 61–68, Winter, 2002.

[14] Magnusson, Thor. "ixi software: The Interface as Instrument" in *Proceedings of NIME 2005*. Vancouver: University of British Columbia, 2005.

[15] Magnusson, Thor. "ixi software: Open Controllers for Open Source Software" in *Proceedings of ICMC 2005*. Barcelona: University of Pompeu Fabra, 2005.

[16] Magnusson, Thor. “Processor Art: Currents in the Process Orientated Works of Generative and Software Art”, Copenhagen: 2002. www.ixi-software.net/thor

[17] Nadin, Mihai. “Interface Design: A Semiotic Paradigm” in *Semiotica* 69-3/4. Amsterdam: Mouton De Groyer, 1988.

[18] Puckette, Miller. "Pure Data." in *Proceedings of International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 269-272, 1996.

[19] Saussure, Ferdinand de. *Course in General Linguistics*. London: Duckworth, 1983. p. 121.

[20] Shneiderman, Ben. "Direct manipulation: a step beyond programming languages," *IEEE Computer* 16(8) (August 1983), 57-69

[21] Wittgenstein, Ludwig. *The Blue And Brown Books*, Oxford: Blackwell, 1993. p. 6.

[22] Wittgenstein, Ludwig. *Philosophical Investigations*, Oxford: Blackwell, 1994.

[23] Wright, M; Freed, A; Lee, A; Madden, A. and Momeni, A. "Open Sound Control: State of the Art 2003" in *Proceedings of NIME 2003*. Montreal, Canada. 2003.