# ixi software: The Interface as Instrument

Thor Magnusson
Creative Systems Lab
Department of Informatics
University of Sussex
Brighton, United Kingdom
T.Magnusson@sussex.ac.uk

## ABSTRACT

This paper describes the audio human computer interface experiments of ixi in the past and outlines the current platform for future research. ixi software [5] was founded by Thor Magnusson and Enrike Hurtado Mendieta in year 2000 and since then we've been working on building prototypes in the form of screen-based graphical user interfaces for musical performance, researching human computer interaction in the field of music and creating environments which other people can use to do similar work and for us to use in our workshops. Our initial starting point was that computer music software and the way their interfaces are built need not necessarily be limited to copying the acoustic musical instruments and studio technology that we already have, but additionally we can create unique languages and work processes for the virtual world. The computer is a vast creative space with specific qualities that can and should be explored.

## Keywords

Graphical user interfaces, abstract graphical interfaces, hyper-control, intelligent instruments, live performance, machine learning, catalyst software, OSC, interfacing code, open source, Pure Data, SuperCollider.

## 1. INTRODUCTION

The interface is an instrument. It is a graphical manifestation of musical ideas and work processes. An interface is at the same time the aesthetic platform defining musical structures and the practical control-base for the underlying sound-engine. In a way it can be seen as a musical ideology. It defines possibilities but also the limitations of what can be composed or played. Here we are mainly thinking of the graphical user interfaces of audio software, but this argument could be extended to audio programming languages as well: the objects or classes ready at hand in a given language define what can be expressed.

ixi has been exploring a basic idea of abstract graphical user interfaces over some years and created various prototypes, each of which contain a certain mode of interactivity. We see it as a proposition for how the musician could work with his or her soundengine. We believe the interface (and here we take the word very literally: that which faces two systems, i.e. the sound-engine and the human performer) is an important factor in a musical performance. It can evoke emotions, encourage direct responses, trigger ideas and open up unknown paths in a live performance. Musicians working with acoustic instruments know very well how an instrument can have a unique personality which makes you play differently depending on its character: just consider the difference between playing a Fender Stratocaster and a Gibson Les Paul and how the type of guitar affects the style of playing.

An essential part of our activities with ixi software, not related to this paper (but I feel it should be mentioned here) is the community aspect of our work. We have a website where people can download our applications for free, a mailing list, and we publish a zine where we promote open source audio programming and introduce things that we find relevant to what we are doing. We also give workshops where we introduce the technologies and ideas we have come up with or use and help people getting started with audio-visual programming.

## 2. BACKGROUND

ixi began as a response to our discontentment and questioning of the way commercial music software businesses build their interfaces uncritically on already established work processes known from the analog studio or from musical traditions such as score writing and reading. The two dimensional computer screen and the mouse are good for many things, but not particularly effective for controlling a mixer with hundreds of knobs. We are not dismissing the ingenious work being done in some of the software houses, but we found there was a big gap which had not been investigated where the computer (with the typical equipment people use such as the screen, mouse, keyboard, sound card, etc.) is taken on its own premises and its intrinsic nature explored.

Most of the software musicians use today can roughly be divided into 3 categories: sequencers (ProTools, Logic, Cubase, etc), editors (Peak, WaveLab, Cool Edit, etc) and patchers (Pure Data, SuperCollider, Max/MSP, Reaktor, etc); the last category being the most interesting for creating instruments and for use in live performances. Using software like Pure Data or Super-Collider, the distinction between composing and instrument building becomes blurred. The musician composes the instrument and decides upon its expressive scope. [7, 9, 12] The instrument does not aim at generality like the sequencing software that tries to provide the ultimate "studio solution", but rather like traditional instruments the programmed instrument is limited and restricted in its musical capabilities. And the musician is happy working with its limitations and its quirkiness.

This is where we find that our work with ixi fits in; as a front end to more complex underlying sound engines. From discussions with the users of the software we find that it is as if the graphical and the abstract forms relate to some more intuitive and emotional parts of the musician. The raison d'être of the interface is not that of nice visuals or slick front end, but rather that of designing user interaction or workflow for a live performance. By hiding the engineered and compound sound-engine and providing playful and simple interfaces with their unique expressiveness and limitations, the musician is able to forget about technology and concentrate on simply playing music.

## 3. IXI TECHNOLOGY/HISTORY

Initially we wrote applications where the user would import his or her own samples into the instrument and work with them in the way that the instrument proposes. These applications could record sound input so the performer could sample other players and work with their sounds as well as their own. Slowly we created a visual language which would be utilised in various ways when designing new interfaces. For example, the location of objects on the vertical axis could control pitch, the horizontal could be panning, size could indicate volume, rotation some other parameter. A movement or blinking of the object could indicate tempo or repetition. But instead of determining this language to rigidly, we have tried to keep it open and flexible. There are no definite rules for what a horizontal location should mean, but rather - in the sense of Wittgenstein's language philosophy [15] - we have got language games where meaning is defined by its context.

The instrument paradigm worked for a while but we outgrew it eventually. This development happened for many reasons, but the main ones were that the users of the software were asking for new features that we had to try to implement, but more importantly the open source software was getting so powerful and interesting that it made sense to work with software like Pure Data and SuperCollider. After participating in NIME 2002 we started experimenting seriously with the Open Sound Control protocol and it became clear to us that it was more interesting to create open controllers that send out OSC rather than the limited instruments we had been working on until then.

OSC provided us with an opening where various programming languages can communicate in a protocol that is very open and not bound to the irritating limitations of MIDI. Contrary to MIDI, OSC does not contain any musical concepts and that is a very good idea when writing a protocol for music as such an effort would always be limiting to the musician. The next ixi application was an OSC controller where the sound engine was built in SuperCollider. For us, this was a kind of paradigm shift. Instead of a fixed instrument that would only do what we had programmed it to do, we had a controller, which provided parameter settings and control for arbitrary synthesis components. This way the musician could use the ixi application as an interface for his or her own Pure Data, SuperCollider or Max/MSP patch and use the incoming OSC messages (the output from ixi) as they like.[1] We found this extremely liberating and it changed the meaning of the software quite a lot. Not only did it make a contribution to the open source community and people working with these programming environments, but it also gave us much more freedom in experimentation with interactive modes of the interface. Instead of one defined use of the software as a certain type of instrument, it now became open for all kinds of applications and use. In short: the responsibility of mapping the GUI elements to sound were no longer only ours, but also the user's who would redefine the interactivity. The exploration of mapping is one of the basic ideas for our work with ixi and there is good existing literature that shows the importance of mapping in virtual instruments. [2, 3, 1]

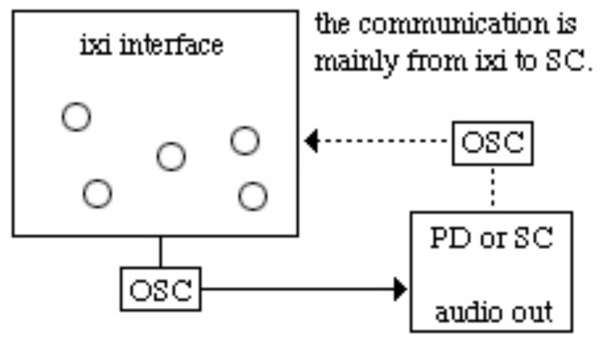

Figure 1. A typical ixi software structure.

One of our criteria when choosing a programming environment was that it should be cross platform with good Linux support (which makes PD and SuperCollider good candidates as sound engines), it should have OSC libraries and it should be high level so that we can develop our prototypes quickly with minor effort. We thought of building interface components directly for Pure Data or SuperCollider as part of their GUI's, but it did not make sense in our case as the SuperCollider GUI is platform specific and if we would create something specifically for PD, then Max/MSP and SuperCollider users would not be able to use them. We also wanted the act of programming to be closer to sketching or improvising on an instrument rather than serious software engineering. It is also important for us when we give our workshops that the programming environment is friendly with an easy learning curve. So we chose Python and Java for the graphical programming and Pure Data and SuperCollider for the audio programming. We have built a library in Python that uses OpenGL for the graphics and we built a little API on top of Daniel Holth's OSC library [14] so now we have quite a useful platform for use in workshop situations. In Java we use Chandrasekhar Ramakrishnan's Illposed OSC library [4] and in our workshops we also use Processing [10] as a graphical programming environment for teaching.

## 4. CASE STUDIES

In this part I will try to illustrate the change of direction we had when we started to use OSC. On the ixi website there are currently over 17 applications available to be downloaded, but they have been created with a mix of different technologies. I have chosen to show the basic functionality of 3 applications - SpinDrum, Connector and Picker - which illustrate quite well the history of the ixi technology.

### 4.1 SpinDrum

SpinDrum is one of the earliest ixi prototypes, an application that experiments with sequencing techniques and in fact parodies traditional sequencer software. Instead of having linear boxes representing the steps, typically a 16 step sequencer, there are rotating wheels with 1 to 10 pedals which can rotate at different

[1] We still distribute the software with example patches in either PD or Supercollider, so the software is both for inexperienced and experienced users of those environments.

speeds. Each wheel has a sound sample attached to it and when the pedal reaches the top position (12 o'clock), it triggers the sample. The location of the wheel controls the pitch and the panning, and the size of the wheel represents the volume.

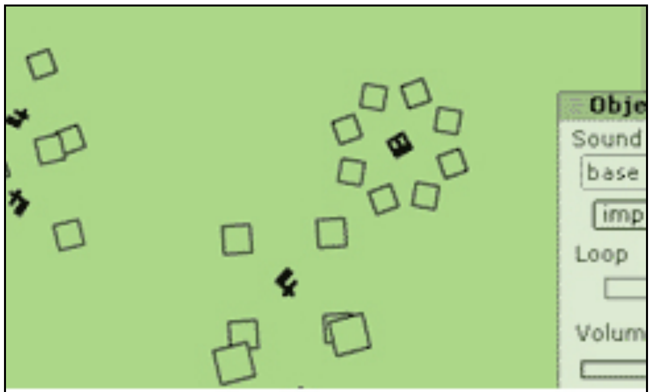

Figure 1. Screenshot of SpinDrum.

The Reichian concept of "phasing" comes to mind here and the software might be labeled a "phase sequencer". [11] When playing, the musician typically moves the wheels, adjusts their speed, samples audio input, adds wheels or deletes them, saves patterns and moves between stored settings often resulting in interesting polyrhythms.

## 4.2 Connector

Connector is an application experimenting with stochastic movement and change with varied degree of determinism. It consists of "connectors", i.e. units in a "plumbing system" with different amount of outlets, and "actors" which move within the system.

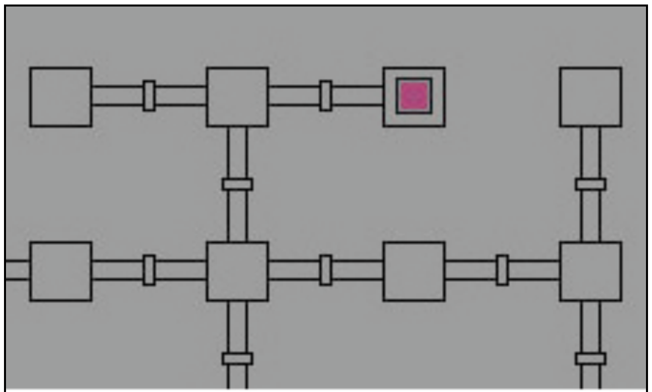

Figure 2. Screenshot of Connector.

The user, of course, sets up the desired system. Each connector has a probability graph, meaning that when an actor is about to leave it and move into another connector, it chooses the destination according to the user-defined probabilities, set up in the connector panel. The connectors have a MIDI value and they can also contain a sound sample, which is triggered when the actor enters the connector. There are 8 actors and they move with controllable speed and randomness. In this application we have extended ixi to become more than an instrument by the use of MIDI. It can now control external devices or virtual synths on the same computer.

## 4.3 Picker

Our first experiment with OSC is called Picker. As the name partly indicates, it picks up colour values from the interface and sends them as OSC to a sound-engine, here written in SuperCollider. The application has a video channel and a webcam channel which can be mixed in various ways, and it also has layers where one can display bitmap images and control their transparency.

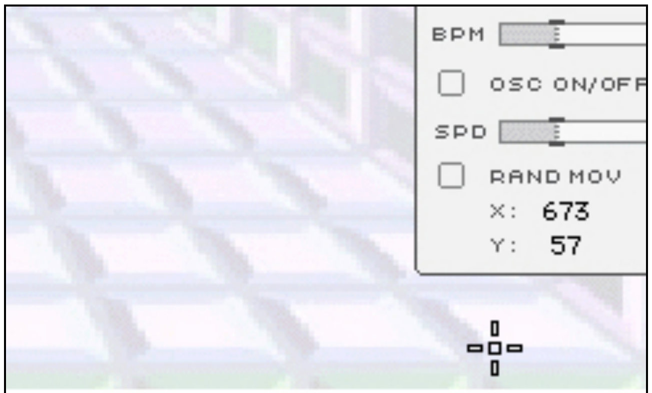

Figure 3. Screenshot of Picker.

All this is to create an exciting colour field which, by the use of video, can have a moving background. There are 4 pickers that can be placed wherever on the screen, but they can also be taught some movement or just move randomly around the screen. The pickers send out the RGB values and the XY location through OSC, generating a flow of numbers that can be useful to work with. The repetitive video or patterns in the bitmap can result in interesting sound structures and the software has been used both to control the macrostructure (tonal information) and the microstructure (low level synthesis) of a piece. In a performance situation, the webcam functionality can be practical, as the performer can point it at him/herself and control the music by movement of the body, or just point the webcam at objects in the surroundings.

## 5. FUTURE WORK

Our experience is that representing sound, from the synthesis to the compositional level, as graphical objects in a two or three dimensional space can help the musician to relate to the material he or she is working with. The human brain is limited in its capacity and representing complexity in an intuitive and ergonomic way is one of the goals of HCI. Furthermore, if the interface is well designed, it can serve as a useful instrument with unique characteristics, which can be played intuitively and expressively. [6] We believe there is huge amount of work ahead of us in exploring the language of such human computer interaction design of screen-based instruments. We have chosen to concentrate on this small but exciting field and currently we are not concentrating on physical controllers or sensors with our works as ixi, although it remains a personal interest for us. Future research will involve studies in how musicians understand the spatiality of the screen using the ixi interfaces and how visual memory relates to motor memory and musical decisions in a live performance.

The word "interface" explains very well what our research is about. It is used in HCI as signifying that boundary across which two systems communicate (the human and the program), but it is also used in software engineering as describing the "visible" methods of a class. A typical UML [13] diagram will show you what variables and methods are available in a given class, but the class itself might have much more complex and lower level inner workings that need not to be seen. The visible methods are the interface of the class and the rest is happening under the hood. This is the way we see the use of ixi apps: as interface between the human and the more complex synthesis engine, where the performer instructs the instrument in a simple manner, but controlling some much more integrated processes that might be taking place underneath.

The high level control over lower level structures is something that is used in all fields of engineering and technology where the human wants simple and understandable control base. In a NIME keynote, Joel Chadabe refers to the fly-by-wire concept used in aviation :

'fly-by-wire' describes a system in which a pilot tells a computer what the airplane should do and the computer flies the plane. The advantages of such systems include the computer's ability to expand simple but powerful instructions into coordinated controls for multitudes of variables, to redefine controls in different contexts, and to maintain goal-orientation while introducing enough unpredictability to keep the instrument interesting. [1]

This relates to the concept of *hypercontrol* which we have used for some time, where one action (say mousedown on a graphical object, move the object, mouseup) can result in many underlying parameters being affected. This is one of the unique qualities of virtual instruments: they can be controlled by simple input device like the mouse and a simple mouse action can result in a branching web of lower level control messages. An acoustic instrument has typically a one-to-one relationship between action and the sounding result, but virtual instruments can be non-deterministic, non-linear and algorithmic in various ways.

This model lends itself to other exercises and experiments with techniques that people have come up with in the fields of machine learning, artificial life, artificial intelligence, genetic algorithms, etc. which allow us to create instruments that have intelligence, learn about the musician, have a character on their own, and utilise the idea of branched mapping or hypercontrol.

## 6. CONCLUSION

Over the years of working with ixi software, we have received good and useful feedback from the users and the download rate of the applications is ever increasing. This fact and direct communication with the users of the software, have strengthened our belief in the initial conception that abstract graphical interfaces as screen based instruments is a useful idea which can result in some good tools for musicians. With the proliferation of open source programming languages of various kinds which contain the algorithms of cutting edge research and development, it became obvious that we should not be building our own sound engines, but rather make use of what is already out there and in use by a strong and ever growing community. What we like to add to the flora of controller and sensor technology are the ixi interfaces, which can be used to control complex sound engines by OSC communication, a simple but playful addition to an already extraordinary creative research field.

## 7. ACKNOWLEDGEMENTS

ixi software is a collaboration between Thor Magnusson, Enrike Hurtado Mendieta, David Bausola and many musicians and beta testers who give us valuable feedback and ideas for further development. Many thanks to Chris Thornton for inspiring talks and I would also like to thank The Media Centre in Huddersfield, UK and Buchsenhausen in Innsbruck, Austria for a nice residency programmes which we have had the pleasure of experiencing.

## 8. REFERENCES

[1] Chadabe, Joel. "The Limitations of Mapping as a Structural Descriptive in Electronic Instruments" in *NIME 2002 Proceedings*. 2002.

[2] Hunt, Andy; Wanderley, Marcelo M & Paradis, Matthew. "The Importance of Parameter Mapping in Electronic Instrument Design" in *Nime 2002 Proceedings.* 2002.

[3] Hunt, Andy; Kirk, Ross & Wanderley, Marcelo. "Towards a Model for Instrumental Mapping in Expert Musical Interaction" *Proceedings of ICMC 2002.*

[4] http://www.mat.ucsb.edu/~c.ramakr/illposed/javaosc.html

[5] Ixi software: http://www.ixi-software.net

[6] Jordá, Sergi. "Digital Instruments and Players: Part I - Efficiency and Apprenticeship" in *Proceedings of NIME 2004.*

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

[8] New Interfaces for Musical Expression. http://hct.ece.ubc.ca/nime/2002/

[9] Puckette, Miller. "Using PD as Score Language" in *Proceedings ICMC 2002.* Pp. 184-187.

[10] Processing: http://www.processing.org

[11] Reich, Steve. *Writings on Music 1965-2000*. Oxford University Press, New York, 2002.

[12] Schnell, Norbert & Battier, Marc. "Introducing Composed Instruments, Technical and Musicological Implications" in *Proceedings NIME 2002.*

[13] Unified Modelling Language. See http://www.uml.org/

[14] WireTap. See http://wiretap.stetson.edu

[15] Wittgenstein, Ludwig. *Philosophical Investigations*. Blackwell Publishers, Oxford, 1953.