Mälardalen University Licentiate Thesis
No.39

Towards new Interaction

A Content Centric Data Surface Approach

Rikard Lindell

November 2004

Department of Computer Science and Engineering
Mälardalen University
Västerås, Sweden

ISSN 1651-9256
ISBN 91-88834-78-6
Printed by Arkitektkopia, Västerås, Sweden
Distribution: Mälardalen University Press

Abstract

Point-and-click, multiple views in windows stacked over each other, and menu selection were breakthroughs in the 70s and what was achievable with the computational power then available. The desktop metaphor explains to users the structure of file, directories, and programs. The computer industry has developed enormously since then. Computational power, the flow and deposits of information have now increased to a point where new approaches to interaction have to be considered. This thesis presents a content-centric interface paradigm, which I call the data surface paradigm. The data surface paradigm contrasts with the desktop metaphor and elements of the desktop metaphor: windows, icons, menus, document files and application programs. The data surface paradigm is based on a reassessment of the fundamental design values of the desktop metaphor interface. It takes into account information navigation and retrieval, collaboration, and ongoing creative open-ended tasks and processes. As a design case I have studied computer music creativity throughout the thesis. Interviews and observations of novices and expert users of music creativity tools identified their needs and inspired prototype designs. I have used an iterative user-centred design process to build and evaluate a series of three prototypes. Content is visualised on a flat infinitely large two-dimensional surface. Users navigate their content by zoom and pan, and incremental search. There are no windows. The unlimited area avoids the need to stack multiple views on top of each other. There are no icons. Content becomes its own icon when users zoom out, miniaturised in size but with preserved structure and metric relationships. There are no menus. Content affords typed commands and context help makes it easier for users to learn. Visual feedback and text completion of command substrings create a uniform model for command invocation and shortcuts. Users do not experience document files. The information content is visualised directly on the surface. Users have no need to deal with explicit file manipulation. The system manages the persistency of

content. Users do not experience application programs. Small plug-in components provide services related to different information modality. Components are attached to the content, one component for each service and information modality. I have, on the basis of cognitive science, observations, interviews, and usability evaluations of prototypes, found strong indications sustaining my approach. The final prototype was evaluated with 10 subject users. The prototype supported the services expected by the users, their creativity in action, and awareness in collaboration. Their responses to the final prototype were: ”It feels free, it feels good for creativity, and it’s easy and fun.”

Acknowledgements

First I want to thank my beloved wife Eva for all her support and encouragement. Jussi Karlgren at the Swedish Institute of Computer Silence (SICS) for his supervision, commitment, and language filtering. Kristina Höök at SICS for introducing Jussi. Henrik Björnek and RSC Technology. Claudia Weidner and Ableton. Olli Parviainen for helping me with his SoundTouch library. Peter Funk at Mälardalen University (MdH) for fruitful comments. Ivica Crnkovic and Björn Lisper at MdH for giving me the opportunity to work with these ideas. Preben Hansen at SICS for help with focus group methodology. Victor Miller for language help. Andreas Gustavsson formerly at SICS. Markus Bohlin at SICS and Kjell Post for helping me with LaTeX animations. Andreas Berger at Gränssnisttsakademin for his knowledge in database systems. David Dinka at Gränssnittsakademin for cognitive science discussions. Mette Holmgren at MdH. Thomas Larsson at MdH for taking over heavy duties. Stig Emmoth at Gripen Upper Secondary School in Nyköping. Last but not least, I thank all the interviewees and all the people who participated in the evaluations.

Contents

Chapter 1

Introduction

The work presented in this thesis is one possible path towards new interaction and the craft of interaction design. It takes off in the polemic field between the Macintosh Human Interface Guidelines [7] and the Anti-Mac Interface [11]. The starting point for the work presented in this thesis came from the insight that users are required to perform actions for which computer systems are better suited, and that these actions are integrated into the desktop interface paradigm. Some years ago my grandmother, at the age of 86, bought a used Macintosh SE 30. It took a while before she could understand some of its concepts, for instance why she could not start to write as soon as the computer had booted. Why she had to double click on the MacWrite square icon to launch the word processor application to write text. And, why she had to double click on a rectangular document icon (with the applications icon on it) to open the document she was currently working on. Another action that posed problems was save. She had already written the text so why did she have to save it? ; the text was already in the computer. Instead of disregarding these understandable learners reactions toward the interface; they inspired me to question the current state interaction paradigm. My grandmother could not be the only one confused by the idiosyncrasies of the current interface design.

It was not long ago that people thought of the graphical user interface and the mouse as toys, today, the PC with its graphical desktop paradigm interface is essential to modern society. Alan Kay and his team developed the desktop paradigm in the 70s at Xerox Palo Alto Research Center (PARC). Before this the interface was a blinking cursor in fluorescent green on black display.

”We have to discard fondly held notions that have sneakily become myths while we weren’t watching. An old adage is that no biological organism can live in its own waste products - and that seems just as true for computer designs. If things continue as they are going now, we will soon be able to see the original Lisa and Mac as improvements on their successors!” Alan Kay 1990 [22].

In this work I have intended to find an interface paradigm design for personal computers that better support human cognition, by making use of models for mental representation of the environment. This paradigm should not need a comprehensive metaphor to be explained. By consistently applying a set of design values and by removing typical elements of the desktop metaphor, the paradigm should more easily be a concept on its own. The paradigm should in itself be designed for collaboration and multi modal interfaces.

1.1 Research Context

I have derived the complexity of the interface from the underlying infrastructure of the operating system and the use of metaphors to convey the behaviour of files, directories and application programs. The desktop metaphor interface paradigm was not designed for the enormous contemporary flow and deposits of information, for user collaboration, and for multi-modal interfaces. The thesis presents a new interaction paradigm that takes these issues in consideration. The work towards new interaction begins with these questions:

- Can reassessment instead of rejection of the fundamental design values of the desktop metaphor interface paradigm result in an interface paradigm that better supports human cognition, collaborative work, users creativity in action, and multi modal interfaces?
- Can this new interface paradigm be more consistent with reassessed design values of the desktop metaphor than the current desktop metaphor interface design?
- What happens if fundamental elements of the desktop interface, windows, icons, menus, files and application programs, are removed?
- Would users accept and would they be satisfied with an interface that has no windows, no menus, no icons, no files, and no applications?

Here follows a brief description of the interaction paradigm design and the research context of this work. The paradigm was created with the following seven principles taken into consideration. One: It is content-centric instead of tool-centric. In a content-centric environment, embedded software components implement the functionality. Two: Content remains persistent. The user has not, with an explicit save action, to transfer data from the machine's primary memory to its secondary memory. Three: Content remains visually present in its context. All content is visualised on a flat, infinitely large surface. The content may have varied scaling that permits a hierarchic relation between content information elements. Four: Incremental ubiquitous search simplifies information retrieval. Five: Non-visual tools. Content affords commands. Typed commands with context help and text completion provide functionality. In a single model, the paradigm sustains recognition for learning commands and recollection of commands for efficient use. Six: Shared synchronised surface areas favour collaboration. Users can more easily work together on a project on the shared surface. Seven: Avoid pixel precision and window manipulation to favour multi modal user interfaces.

1.2 Methodology

A number of methods have been used during the preparation of this thesis. The work has been divided into two phases, first a conceptual phase and then a prototype iteration phase.

The empirical methods in the conceptual phase were mainly in-depth interviews, focus groups, and observations. For the in-depth interviews and focus groups, I had relatively few interviewees (6), and I used open question forms with question clusters. Open forms produce more data and unanticipated data, but the large volume of data and its diversity makes it difficult to quantify the results. It is important to consider unanticipated data that can provide inspiration in design work. Question clusters give more results from reserved interviewees. Scenarios based on empirical data and the design ideas were used as a basis for the second phase of prototype iteration.

The methods used in the second phase were quantitative user-evaluations and prototype implementation. The user evaluations were mainly based on the Think-aloud method. The Think-aloud method has, however the disadvantage that when users face difficulties. they tend to become silent, thus records were

kept by means of camcorder, log and notes. I also debriefed users by means of interviews. Prototypes were implemented with a general-purpose programming language. The implementations gave strong indications that the ideas work and it was necessary to have a real prototype to engage the users in an interaction.

1.3 Publications

Users Say: We Do Not Like to Talk to Each Other, in proceedings of Graphical Communication Workshop 2003, pages 87-90, publisher Queen Mary University of London.

When Information Navigation Divorces File Systems - Database Surface Prototype Results, in proceedings of The Good, the Bad, and the Irrelevant: The user and future of information and communication technologies. Conference 2003, pages 76-81, publicher COST Action 269 and the Media Lab of University of Art and Design Helsinki UIAH

1.3.1 Research Results and Discussion

By leverage on cognitive science, observations, interviews, and usability evaluations I have found strong indications that my approach supports the services expected by users, their creative work processes and their awareness in collaboration, in a manner they feel to be fresh, fun, and pleasing. The validity and the reliability of the results come from the triangulation of conceptual studies, empirical studies, and technological studies. The conceptual studies contain the related work, the reassessment of design values, and the field of cognitive psychology. In-depth interviews, focus groups, qualitative evaluations in controlled environments, and qualitative evaluations by means of field studies constitute the empirical studies. The technological studies were the implementation of the prototypes with a multimedia tool, and general purpose programming language and open source game development libraries.

From the activities listed above I can conclude that users can do without windows, icons, and menus. They described the interaction as free, good for creativity, easy, and fun.

1.4 Introduction Summary

This thesis is a presentation of the long and winding road from the insight that the current desktop paradigm interface cannot be the end of the line, to user studies and evaluation of a data surface paradigm.

In chapter 2 I present a scenario to illustrate what this work is about.

In chapter 3 I go into the details of the principles and design values of the desktop paradigm interface (Guidelines 1992). I also present Gentner and Nielsen's polemic view (Gentner 1996) of these design values.

With chapter 5 I go into the details of the principals of the desktop paradigm interface, its design values, and that its conceiver's hoped it would bring [7]. I will also show Gentner and Nielsen's polemic view [11] of these design values.

The chapter 6 present my sources of inspiration and related works.

In chapter 7 I will go through some theory and observations of spatial semantics.

In chapter 8 I describe the case music application area. The chapter will also cover my interviews with computer music creativity tool users as well as a case study of user collaboration with an of the shelf tool.

The chapter 9 contains my initial Concept Prototype design and evaluation of it.

In chapter 10 I continue to iterate the prototype evaluation process by describing my study of my Fluent Zoom Prototype.

The chapter 11 covers design, implementation aspects and users evaluation of the Music Tool Prototype, the final prototype for this work.

In chapter 12 design implications and principals for the data surface paradigm will be presented.

The final chapter, chapter 13, contains discussion and conclusions.

Chapter 2

Scenario

David and Lisa plan to perform their music live at a dance club. They have one computer music creativity device each. All their material, samples, sounds, and songs are in a shared synchronised content data surface environment. While creating the music for their performance, they most of the time sit together in their studio, otherwise they work at different locations connected through the Internet. During collaboration David and Lisa discuss their music material, the synchronised visual and acoustic space of the devices help by provide an external referent to their discussion. When they are online they depend on the embedded chat component. They zoom in on sounds that need to be examined, copy sound from songs to empty regions of the surface as repository for their performance. They zoom out from the region to get an overview of what they have collected, and annotate content reminders and cues.

Performing live and creating songs are two different kinds of creative activities. For each of these activities the data surface environment provides an embedded component. The live improvisation tool easily allows David and Lisa to layout the collected sounds. Starting an array of sounds, aligned to the beat, while the previous sounds are stopped, can create advanced arrangements. The beat alignment eliminates rhythmic glitches. They rehearse a couple of times before conduct their performance.

2.1 Summary

This scenario illustrate the examined task domain, the users in this task domain, and how they user the tool. Bear this scenario in mind for the chapters of the application area, the early concept prototypes, and the music tool prototype.

Chapter 3

The Values of the Desktop

As backdrop to this thesis I will present the intellectual foundation for the desktop paradigm interface. How often do we reflect on what constitutes the interface of the computer we use almost every day? The overlapping windows, files folders, menus, trash can, and of course the desktop itself, are taken for granted. Where did it come from, why did it evolve, and what are the fundamental values of this interface paradigm? Before the graphical interface developed at the labs of Xerox PARC (Palo Alto Research Center) by Alan Kay and others in the 70s, the computer interface was a blinking cursor by which highly trained expert users wrote more or less confusing command acronyms. Xerox PARC fellows saw the capacity of the computer but realised that it had to have a comprehendible interface to become a commodity. They invented the desktop paradigm and the graphical interface that became available to the rest of us through the Apple Macintosh in 1984. They also invented along with the interface the ethernet and the Smalltalk programming language. We have seen a tremendous development since then, capacity of CPUs, memory, disk space, and networks have increased by a magnitude, but still the underlying values for the desktop interface have remained the same. The Macintosh Human Interface Guidelines (referred to as the guidelines throughout this section) **??** neatly lists values that constitutes the Mac interface. Other desktop systems are very similar to the Mac and many of the great minds behind the desktop paradigm user interface at Xerox PARC were recruited to Apple including Alan Kay. Hence, the Mac interface stands here as a representative for desktop interface systems in general. The guidelines state 11 values as the founding principal for the desktop interface which are: Metaphors, Direct Manipulation,

See-and-Point, Consistency, WYSIWYG, User Control, Feedback and Dialog, Forgiveness, Perceived Stability, Aesthetic Integrity and Modelessness. Gentner and Nielsen wrote in 1996 the article ”The Anti-Mac Interface” [11], in which they go through all these values and rejected them in the view of what future interfaces should be. I have found that many of these values are above the Mac, and that a different interaction paradigm can be designed and implemented reflecting on these values. I will run through these values and give a brief review of why Gentner and Nielsen rejected them. I will also give you my view of the value in the light of the data surface paradigm presented in this thesis.

3.1 Metaphors

The guidelines states that by using metaphors to convey concepts and features of your application, you can take advantage of peoples knowledge of the world. They further state that metaphors should constitute of concrete, familiar ideas. The guidelines explaine: ”For example, people often use file folders to store paper documents in their offices. Therefore, it makes sense to people to store computer documents in computer-generated folders that look like file folders.”

As Gentner and Nielsen pointed out out the target domain (the computer interface) does not have the same attributes, properties and features as in the source domain (in this case the desktop). Metaphors are, according to Gärdenfors [16], conceptual spaces that map to eachother. In terms of language we talk about the peak of performance mapping the spatial characteristics of a mountain peak to the graph of the performance. This powerful cognitive construct is tempting, but as Gentner and Nielsen said the source and target domains are different most of the time, thus metaphors are only useful in explaining how a system works for novice users, whereas, over time, the metaphors vanish and the interface stand by itself. Documents are no longer stored in file-cabinets: young people have no concept of the source domain. In that case, introducing limiting constraints from the real world source domain into the virtual world target domain restrains the power of the interface, and thus the power of the computer. To avoid this situation Alty and Knott’s set based theory can be applied to evaluate the use of metaphors [21] . In their theory they classify three sets: V - functions of the application; M - users metal models of the metaphor; and I - manipulations granted by the interface. Intersection of these three sets indicates well-designed interaction. Whereas, the theory designers to change interface or metaphor if either these lies outside the intersection.

Instead of searching for a comprehensible explanation of how the computer works, I have looked at what would suit human cognition and perception. My interaction design approach rests on cognition for spatial semantics, automation and visual perception instead of metaphors.

3.2 Direct Manipulation

The guidelines strongly advocate Shneiderman's [42] direct manipulation. The idea is that objects should be visible on the screen while users perform physical actions on these objects. The impact of those operations on the object should also be immediately visible. The most widely known operation in this manner is the drag 'n' drop action, typically dragging files between folders or to the trash can. Again the assumption is that real world behaviour should hold for the virtual world.

Gentner and Nielsen pointed out that direct manipulation tend to work on an atomic level. For instance try to remove all the PDF-files of a folder and its subfolders. You have to recognise each file and drag it to the trash can. Whereas command line interfaces can be much more powerful for this action the single line: rm *.pdf -R irreparably removes all the PDFs: indeed powerful but also unsafe. Taking a step back and look at the grammar of these actions you will see that for direct manipulation you have noun-verb grammar. First you identify what object you will affect, then you complete the action on it. The command line interface has verb-noun grammar. The advantage with first defining the objects for the operation is that the objects themselves constrain the action. For instance you play a movie but not a picture. Noun-verb interaction grammar may leverage on Norman's affordances [34]: the objects afford the actions. Whereas with verb-noun grammar you have to know the possible actions for the objects in advance.

With direct manipulation comes visibility of tools. In complex and vast applications tools allocate a large proportion of screen space leaving less screen real estate to the information content, while the command interface takes no or very little screen real estate. When users have clear goals of what command to use, they have to find its location. To find a command can be an extensive navigation task, since many commands are hidden behind curtains of drop down menus. In this thesis the presented data surface paradigm design does not throw away the precession from the users knowledge in their head. Written commands allows users to rapidly evoke the correct command. Better design of command interface can be enhanced by the advantage of direct manipulation

without loss of power.

3.3 See-and-Point

The guidelines tell us that: ”users perform actions by choosing from alternatives presented on the screen”. The screen and the visual appearance of objects and the visual appearance of actions available for the object constitute the main interaction style for users of desktop interface computer systems. The mouse is the main tool for interaction. The guidelines instruct interface design to utilise noun-then-verb interaction grammar.

Gentner and Nielsen’s objection to the See-and-Point value was that it throws away the power of language. Extreme amount of information is condensed into human ”natural” language. The computer power should be, according to Gentner and Nielsen, unleashed by reinforcing a language interaction style.

But, both the guidelines and Gentner and Nielsen miss the point. First of all the the See-and-Point value tells us specifically what interaction grammar should be used for a desktop interface computer. The symbols for this grammar are graphical ones. Also, the real power of human ”natural” language lies within reasoning about what is not there, in space and in time. Language helps us select things in a powerful and simple way, for instance out of a basket full of black and white marble balls the utterance: ”give me all black marbles” is much more powerful than See-and-Point for selecting black marbles. The ability to make selection is pervasively built into my interaction paradigm, by incremental search, without losing the visual presents of information elements. Command invocation is defined as a two step action. First define the noun, selected by direct manipulation or incremental search, then evoke a command available for the target object set.

3.4 Consistency

One of the key values for the Macintosh and which has contributed to its success is the consistency value. It means interface element behaviour should be consistent within different applications, making users’ knowledge transferable from application to application. As an example font selection should be similar and evoked by the same command in a simple text editing tool, an e-mail client, and a layout application package. If this holds, the users’ interaction can be habitual and automated making the interaction smooth, efficient and

safe. For instance, as with the example of copy and paste commands (but not cut: cut is unsafe, if you cut then copy you irreparably lose data unless you can undo more than once). Consistency is bigger than this, the guidelines ask you as designer to dwell on a few questions: Is the interface consistent within itself? Is it consistent with earlier versions? Consistent with Macintosh interface standards? Consistent in its use of metaphors? And, consistent with peoples' expectations? In short an extensive amount of design decisions to think through to achieve consistency.

Gentner and Nielsen attacked the Consistency value by interpreting consistency very literately. They argue that the guidelines only nurse the hope that users would more easily learn and use similar tools if they look and behave the same. They also claimed that the real world do not work this way, and exemplified this by the pointing out that people have no difficulty switching between ball-point pens and fibertip pens. But even though these pens are different from each other their interface is rather consistent. You operate them in the same manner and their behaviour are consistent with the users expectations. Gentner and Nilsen also said that similar functions does not need a consistent appearance. However, take for instance the car that has ignition on the floor between the seats instead of next to the steering wheel, to get the keys out the driver has to put the gear in reverse possibly preventing juveniles from hijacking the car. The same functionality as any other car but with different behaviour and appearance. This car is confusing to use for those who expect a consistent interface with other cars.

I think the whole issue of consistency comes with the design of desktop interaction paradigm, which originates from application programs that provide the functionality and of the computer. A lot of functionality has to be rewritten for each application for instance the behaviour of copy, cut and paste. Since much of this functionality is the same, consistency helps users to learn and use. In view the work presented in this thesis, a content centric model removes unnecessary function redundancy, as opposed to tool centric. Each content modality has behaviour implemented by components ubiquitous in the entire system. This brings consistency to the interaction.

3.5 WYSIWYG (What You See Is What You Get)

According to the guidelines WYSIWYG is about displaying the document on the screens in the same way as it will look on paper. Contesting this value statement is not in the scope of this thesis.

3.6 User Control

The computer should not ”take care” of the users by protecting from detailed judgements and decisions. User control means that the user, not the computer, initiates all actions. This may be true for office applications however the process the user interacts with may very well be dynamic.

Gentner’s and Nielsen’s response to this was that users have to be in control in a desktop interface environment. That the users have to monitor boring tasks or perform them themselves. They also point out that agents and daemons are incompatible with the User Control statement.

In view of what is presented in this thesis, user control means users are in control of a process. Users may be over-viewing (zoomed out) or inspecting details (zoomed in). Users are in control of what is going on with the process, but not necessarily initiating all actions. User control strictly delineates the process from the tools for the users to interact with the process. Tools should not initiate actions. Illustrated by an example, a tool should not auto-correct misspelled word, whereas, game avatars may trade nova-crystals for ore with you. In a content centric environment, content may be a process or an agent. User control of an agent is like the parents control of their children. Although the children play on their own, their parents are monitoring their actions, interrupting them if they do something wrong. In collaboration with others, user control is in asking the other participant or agent to perform the actions desired action and thus be in control of the process.

3.7 Feedback and Dialog

The guidelines feedback and dialog value advise interaction design to always inform users about the application status. Feedback should be as immediate as possible, lengthy actions should inform of progress and when they are ready.

Gentner and Nielsen said that detailed feedback is needed for detailed control, and that a background agent only needs to provide sparse feedback.

This thesis suggests that in a content centric environment a process provides detail feedback if the users zoom in on them. Whereas, if the users focus is within another context, then users do not need to be fed with details. The feedback value plays a central role, but there is no and should be no model for alert dialogue boxes. For direct actions immediate feedback and feedback forecast provides users with information to evaluate the effect of their actions.

3.8 Forgiveness

The forgiveness value means that actions should be reversible. The guidelines also instructs you to use alert dialogue boxes for possibly dangerous irreversible actions, such as emptying the trash can. Forgiveness is supposed to make people feel more comfortable and safe while using the computer.

Gentner and Neilsen's view of forgiveness was that the desktop interface computer does not keep track of the user's actions. Thus information that could have been anticipated by bookkeeping of the user's intentions is thrown away which may lead to repeated annoying alert dialogue boxes.

Alert boxes are unsatisfactory and unsafe. Repeated "Are you really sure..." type of alert boxes makes the users automate the OK-action, thus the one mistake out of 1000 will be irreparable. But to us advanced AI schemes to remedy the problems of the alert box is no solution. Instead do without the alert box. The model presented in this thesis is that each content information element keeps track of the actions that led to its current state. Smooth and user satisfactory forgiveness is applied by the reversibility implemented by action bookkeeping.

3.9 Perceived Stability

According to the guidelines perceived stability makes the computer environment predictable. Users that leave a document file on the desktop expect it be there when they come back. The interface is filled with consistent graphic elements (window frames, menu bar, buttons, etc.) that maintain illusion of stability.

Gentner and Nielsen said to this that you only have to look at the web, if no content were changed by others the world wide web would be boring. Other situation where stability is undesirable is in computer games and learning applications.

In this thesis spatial semantics and visual appearance of persistence content on a data surface promotes the perceived stability value. Nevertheless, content can be changed by other users which leaves traces of other users in a shared space. Perceived stability still makes sense, the users know what place to look for changes, just like the favourite news web site URL does not change.

3.10 Aesthetic Integrity

This value means that the graphical element layout should be consistent with visual information design principals. The guidelines encourage keep a minimum of interaction elements with a consistent elegant look to enhance usability. The screen layout should, according to the guidelines, not be cluttered and overloaded.

Gentner's and Nielsen's objection was that for increasing magnitude of information repositories coherent visual appearance of information elements make the navigation experience unpleasant and confusing.

Again differentiate tools from content. Content is diverse, even text documents have very different visual appearance. This diversity combined with spatial semantics helps navigation, whereas tool usability still rely on the aesthetic integrity value.

3.11 Modelessness

The guidelines argues for the interface design to be modeless. This means that all possible operation should be available at all times and unrestricted by software modes. Users should be free to perform any operation at any time.

Gentner and Nielsen argued that one should yield to that it is impossible to create a modeless interface, they pointed out that even the guidelines devote the bigger part of the modelessness section to guide the correct use of modes. However, many of the most devastating user errors come from the user not recognising the mode of the system: these errors are known as mode-errors [34]. Desktop paradigm brings about a number of different modes, for instance dialogue boxes that prevent users from further actions in an application until they have completed the dialogue, the overlapping windows, applications, open and closed files.

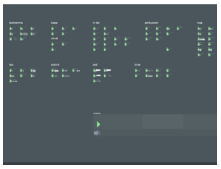
The argument against modelessness; that humans concentrate and focus at one task at hand, does not sustain the design for the user to yield overview and context. In a content centric interface the properties of the content provide context for activities and afford the actions available for the content. Modelessness in this respect means that the users are free to focus and select any content at any time.

3.12 Summary

The Macintosh Human Interface Guidelines contains a number of values and principals for the desktop interaction paradigm interface. The Table 3.11 summarises and compares the design values of the guidelines and the Anti-Mac interface. I have reassessed this values and found that the work presented in this thesis comply many of them. These values were: consistency, user control, feedback and dialog, forgiveness, and modelessness. The values of perceived stability and aesthetic integrity were less important. See-and-point and direct manipulation were more or less merged into noun-then-verb-grammar. The value of metaphor as explicit explanation of a services was rejected. Finally the WYSIWYG value statement was not considered at all.

Table 3.1: Macintosh UI Guidelines, Anti-Mac Interface, and Data Surface

Value	The Guidelines	Anti-Mac
Metaphors	Convey concepts and features. Take advantage of peoples knowledge in the world.	Target domain is different from source. Preserve limitations from source domain.
Direct Manipulation	Physical actions on visible objects on the screen. The effect is immediately visible.	Direct manipulation works on atomic level and can be very inefficient.
See-and-Point	Actions preformed by choosing from alternatives presented on the screen.	Throws away power of language. Language interaction style should be reinforced.
Consistency	Interface behaviour consistency among applications make users' knowledge transferable.	The real world is not consistent. People have no trouble switching between different tools.
User Control	The user, not the computer, initiates all actions.	The users have to monitor or perform boring tasks.
Feedback and Dialog	Always inform, as immediate as possible, the application status.	Detailed feedback is only needed for detailed control.
Forgiveness	Actions should be reversible. Dangerous irreversible actions must be alerted.	Anticipate users intension by book-keeping actions omits repeated alert boxes .
Perceived Stability	Makes the computer predictable. Things remain in place.	Stability is undesired in web, games and learning applications.
Aesthetic Integrity	Minimum of interaction elements, with elegant and consistent look.	Coherent appearance makes navigation confusing.
Modelessness	All possible operation are available at all times.	Modes are natural and people can not cope with everything at once.



Chapter 4

Sources of Inspiration

It is easy to criticise the desktop paradigm, but more difficult to suggest a usable alternative in its place. Removing the parts such as files, folders, applications, scrollbars, icons, windows and menus that typically constitute the every day computing experience was only my first step. The next step was to find something else that provides functionality. This chapter contains description of relative work, theory, and practise that were the sources of inspiration for the work presented in the thesis.

4.1 Persistence

Other models than traditional file-systems for persistent storage was utilised in the Data Soup object storage framework of the Apple Newton OS [46]. The inspiration from the Newton was an eye-opener. Apple's perspective was that developers with traditional file systems had to write extensive code for object serialisation. Developers were relieved from this by an object oriented persistent system along with the embedded NewtonScript programming language. In my view the idea of a content aware persistent storage has a more deep and profound impact on the over all interface paradigm. For instance with object storage embedded in applications omits explicit file management, users can focus on content rather than be the file system janitor.

4.2 Zoom

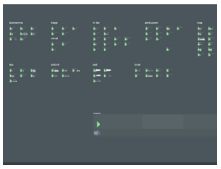
The idea of use semantic zoom for interaction was put forward by Perlin and Fox [23]. They proposed to siggraph 93 the Pad as an alternative to the desktop paradigm. In the pad information is put on a big white board, hierarchy is created by different scales where small scale object show only a caption of themselves, the rest of the document is hidden. The users zoom in on the captions step by step, the closer they get the more details of a document is shown and the caption fades away. Perlin and Fox called this: semantic zoom. For instance in small scale a clock displays only the hour arm, whereas when the clock is zoomed in it displays minutes and seconds too. Detailed sections of a view in the pad can be view by portals, similar in behaviour to overlapping windows.

Bederson and Hollan, whom are the most cited for zoomable user interfaces, developed the Pad ideas into Pad++ [2]. Pad++ is a layer based on Tcl/Tk that allow semantic zoom into portal subspaces of an elastic surface of information. The surface resides in a window. Various lens techniques allow data to be visualised in different ways. For instance a graph can be visualised in a lens when the lens is dragged over a table chard of numbers.

The inspiration form Pad and Pad++ was the idea of use zoom as navigation technique. However, they are not compound content centric. Applications in Pad and Pad++ are bound to specific locations. In case of Pad++ the ordinary widgets are still present, although the widgets can be zoomed. The semantic zoom used in these cases also hides content. Zoomed out content may not be readable, but letting it remain present in its small scale provides visual cues for it. Holmquist's Focus+Context visualisation [19] inspired the use of visually stable but miniaturised views of content.

4.2.1 Focus+Context

Focus+Context visualisation or was first presented as Flip-Zoom for reading text documents on small screen PDA devices, and later as WEST [20] a web browser for small screen devices. The basic idea for context-zoom was to let the current page be displayed in the centre of the screen, while all other pages are visualised as small thumbnails. A click or a tap on a thumbnail page, which zooms it into focus, flips pages. The small screen real estate on a PDA coerces the thumbnails to be resorted for each flip, which caused user confusion in the original design. Large documents are hierarchically organised, the users first flips chapters then pages.



An approach to face the context problem of zoom interfaces were the ideas of Pook et. al. [38]. The semantic zoom hides displays condensed information in overview state. When users zoom in details pop up. However, the information pop cause users to lose context; the inspected information and overview information does not have the same visual appearance and visual cue. Pook et. al. introduced a historic layer and a tree overview layer for people to guide their way through the information space. To save screen real estate these layers have to be brought forward by the user. A combined mouse movement gesture driven menu displays these layers. The menu is also used to control zoom and pan and to invoke commands. The inspiration from this work was the menu; it is one of many possible modalities that are enabled by a content centric zoom interface approach.

4.3 Spatial Semantics

Another approach to find new ways to think about interaction is Dieberger and Franks' City Metaphor [1]. A metaphor that embodies Lynch's Image of a City [28] Dieberger and Frank propose that cities can stand as roll model for navigation of information. The Description of the metaphor is detailed and literal. It leverage on elements of cities; streets, squares, landmarks, quarters, buildings, facades, door, windows, rooms, subways, trams, busses, taxis, air-planes, and many more. It adds a set of magic utilities, where entire cities can resides inside a room of a building in a city. On way to imagine an implementation of Dieberger's work is the Matrix, in the movie with the same name [51] or even or the Matrix in William Gibsons Newromancer [17]. I was inspired by the firm dependency on spatial semantics in the city metaphor.

Dourish et. al. have developed another use of spatial semantics for information navigation and retrieval of documents in an ordinary desktop system [36]. The basic idea is that attribute meta-tags and spatial semantics can help finding and reminding document files. Files are localised in different rooms; an open room displays its content files and the files attributes, a closed room is displayed as a pie shows its size. Attributes may be icons, thus one document file may have many icons. These attributes and properties of the files provide the mechanism for manage, retrieve, search and interact with the document files. Although the Presto significantly improves the file navigation experience its main contribution for me was their proof of spatial semantic and that information navigation is aided by contextual information. The Presto project was acquired from Xerox PARC to Microsoft, and the forthcoming versions

of Windows (Longhorn) and Apple Mac OS X (Tiger) will include many of Presto's features.

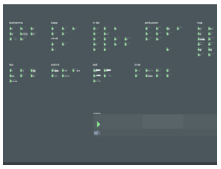
Flatland [30] is an augmented whiteboard interface. The interface is a big white board where everything resides during a collaborative work-phase. Flatland was designed for activities of thinking, pre-production tasks, everyday content, clusters of persistent and short lived content, and semi-public to personal use. The input is handwriting, however, writing is not interpreted into characters and text, the input stream remains the data type ink. Content is organised into segments, these are not allowed to overlap. Old segments are downscaled when more space is required. Flatland has no pan or zoom technique. The most intriguing thing about flatland is that it successfully supports a collaborative creative pre-production phase. It adds ubiquitous computing power to white-boards without tamper the behaviour of the white-board.

Sens-A-Patch is hierarchical context preserving spatial semantic label cluster visualisation technique invented by Löwgren [27]. Labels can either be leafs, nodes, or ping labels. Pings are indicators of relations between clusters. Node labels, displayed in bold-faced font, open clusters. Unopened labels are partially transparent by the rule: the further away from the open cluster the more transparent. But, a label will never be totally transparent; hence they leave a trace of its existence. The closer to the cursor a label of an open cluster is the more opaque. A touch of a node label (the cursor is moved over the node) opens its cluster. Child and parent node labels are active from the current cluster; users can thus float up and down in the hierarchy. The main inspiration from this work was in the early prototypes to display a hierarchical and condensed view of information content.

Islands of Music [35] is a geographic maps metaphor to music collections. Islands represent music genres. Closeness in space maps similarity of genre and piece. Music is analysed by Self Organising Map (SOM) clustering technique [25]. SOM generates a 2-dimensional map of multi-dimensional data. In Pampalk's work the streams of ones and zeros that makes digital music files were analysed and mapped onto a perceivable data surface for users to explore and to learn about unknown genres and pieces. What Pampalk suggests is that spatial semantic for organisation of content information can be automated.

4.4 Related Works

The Morphic interface was introduced with the Self programming-language project at Sun Microsystems [45]. The concept of a morph (from Greek for



shape or form) is that it should be concrete and encapsulate content, behaviour and appearance. The interface is tightly coupled with the underlying Self programming language model, which is prototype based and classless. Both users and developers work by copy-then-modify.

The Morphic users interface framework found in the Squeak Smalltalk system implements a developer friendly and consequent interaction style [29]. In Squeak each morph is design to be an illusion of a tangible object like a little piece of paper that can be picked up and moved around. Morphs can overlap like windows and be arranged into compound morphs. Morphs can contain entire composite world of morphs that users open by double click, analogous to open folders. The Morphic framework is foremost a development-model that challenge the Model-View-Controller framework, but with it comes a flatter concept for how to create window systems.

The Croquette project brings innovation in multi-user immersive 3D interfaces [43]. Croquette is the latest contribution from Alan Kay. It is built on Squeak Smalltalk with legacy from the original Star system. Croquette was constructed with focus on deep collaboration between user teams. It relies on the hypothesis that a shared 3D space better set the context for users collaboration.

Apart from the dynamic late binding Smalltalk language, Croquettes infrastructure is based on an OpenGL 3D render engine, and a peer-to-peer object distribution system called TeaTime. Croquette emphasises on the 3D users experience, the key technologies are dynamic movable portals, floating 3D windows, true 3D creation tools, live teleporting snapshots, and a new concept of 3D portals [44].

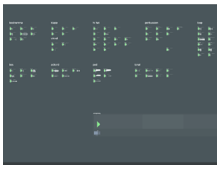
The elements of this project look interesting: the late binding object model, the distributed object model, and the OpenGL render engine. Nevertheless users spatial abilities do not concede the deep emphasis on 3D interaction. For many users find it difficult to perform mental rotations, for instance some people rotate maps in their heads while others must align it with the world. Dahlbäck et. al. [31] found a strong correlation between users spatial abilities and their navigational skill in a virtual space. Users task completion time differed as much as 20 times. Another disadvantage is that many services in Croquette are in 2D and pasted to surfaces called 3D windows. The 2D information is distorted by the 3D projection transformations. According to Spence [47] 3D is good for qualitative view of information but not for quantitative visualisation.

Another related work is the OpenDoc project a discontinued joint venture of IBM, Novell, and Apple [8]. OpenDoc was a platform independent compo-

nent model based on the System Object Model (SOM) for creating compound documents. It removed the need of monolithic applications. OpenDoc tightened the document creation process with the file system. Each document consisted of parts; each part had a component editor or viewer component. Each part editor had a stationary file that users could drag into their document if they needed its service.

4.5 Summary

The projects described and listed in this chapter were those that in design, practise and results inspired me for the work presented in the theses. These papers give insight in the different approaches to solve problems of the desktop interface in different contexts. We have seen that persistent storage can be solved in other ways than by a file-system on operating system level, we have seen a couple of approaches to zoom interfaces; sematic zoom and focus+context zoom. We have also seen various projects that rely on spatial semantics and the context of city metaphor to computer interface, file management, augmented whiteboard, label cluster visualisation, and music genre categorisation. With all this in mind the next chapter will take a little deeper look into the psychology of spatial semantics, what it is, and how it can be used.



Chapter 5

Spatial Semantics - Theory and Observations

This chapter will look into the human cognitive ability for mental representations of the environment. The computer screen is a two-dimensional graphical environment hence design can leverage from this knowledge. I have also observed and interviewed users who used spatial semantics extensively for organise project related files. The data surface interaction paradigm depends on the human ability to map semantics to positions and location of objects. This is called spatial semantics. With a vast range of different types of objects and information a firm support in theory and observation had to be found for the data surface approach.

5.1 Cognitive Maps

Cognitive maps a cognitive model to explain how people have organised spatial information in their heads [12]. Cognitive map is a metaphor for the mental representation of a persons environment. It is not a precise map, it consists of *landmarks* and their relations, *routs* for navigation, finally landmarks and routs are unified with metric *survey* of information. A cognitive map is a rather firm construction, people find it very difficult to rotate the map in their heads. The landmarks are crucial. If the environment is changed, for instance by exchange a crossing with a roundabout, people may feel lost.

5.2 Cognitive Collage

A more recent extension to the model of cognitive maps is Tversky's cognitive collage [50]. Tversky observed that the cognitive map model could not describe peoples' spatial mismatches. She suggests the metaphor of collage as model for metal representation of spatial information. She found that people kept several parallel and overlapping maps in their heads. These maps or pieces of a collage do not preserve metric data but leverage on landmark relations and semantic context. For instance, Tversky showed in one of her experiment that student at U.C. San Deigo had a non-metric mental model of the world. The sudents put Reno Nevada East of San Diego California. Nevada is East of California but Reno is in fact West of San Diego, because California does not run North to South. People straighten up their metal models, as a result of this the map of Sweden is most often viewed, for instance in weather forecasts, as aligned to a North South axis, whereas the correct view of Sweden is tilted to the right.

5.3 Observing Users

Barreau and Nardi studied how users organise and find files on their computers [9]. They looked at all the present major personal computer systems, DOS, Windows, OS/2 and Macintosh. Their result indicated that users preferred spatial semantic and locations as the critical *reminding function* for file retrieval. Users preferred to keep things as simple as possible and avoided advanced filing and detailed finding schemes. Users had three different temporal categories of information, short term information: *ephemeral*, *working*, and *archived*. However, relatively little information was archived, thus, in reality the ongoing work information eventually becomes the archive. This result indicated that that users do not like to tidy the desktop, and if they do, the exchange of files' locations may cause interference between old and new locations.

The result of Barreau and Nardi have more recently been examined by Ravasio et. al. [39]. They too found that a broad variety of users rely on spatial semantics as cue for where they put documents and how they build their structures. However, they conclude that the reason why users do not facilitate the system included search tools for Mac OS and Windows is that this tools are to knotty. Users want to search in text content of files, not only in name and metadata that easily can be forgotten. Metadata such as creation data or file type are also to technical and does not help the users.

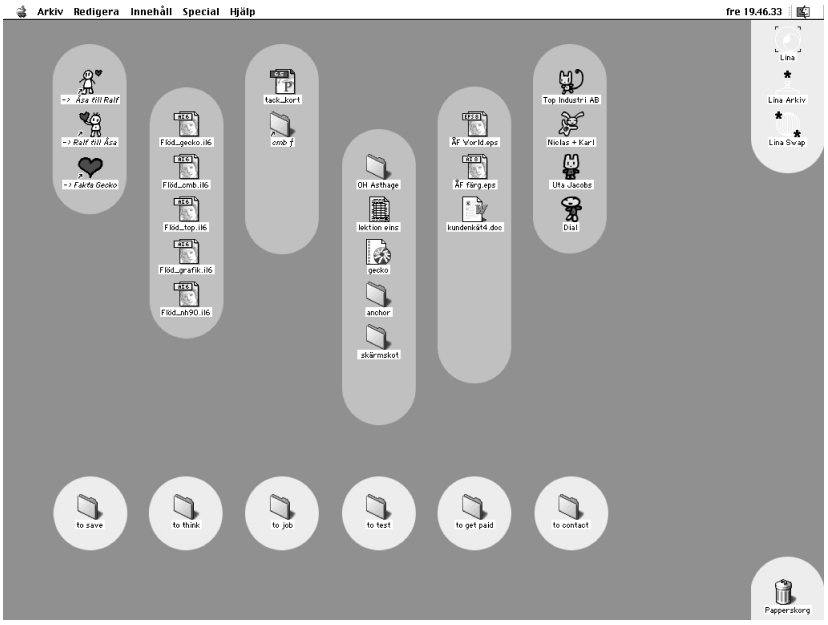


Figure 5.1: Screen-shot of a graphic designers business management system, based entirely on spatial semantics.

5.3.1 My Observation - A Neat Desktop

In user observations I have found support for use of spatial semantics with an ordinary desktop. For instance one user I observed, a graphic designer, created a project management system. A desktop background image created a rule context for manually organised files and folders. See Figure 5.1. Each active project resides in a slot. As each project progressed the files and folders were moved closer to the finish line. A completed project was moved from the desktop into an archive folder. Contacts with customers and other agencies were also included in this system.

5.3.2 My Observation - A Chaotic Desktop

Another fellow at the same office also used spatial semantics, though at the first glance it looked more like chaos see Figure 5.2. The user had no trouble

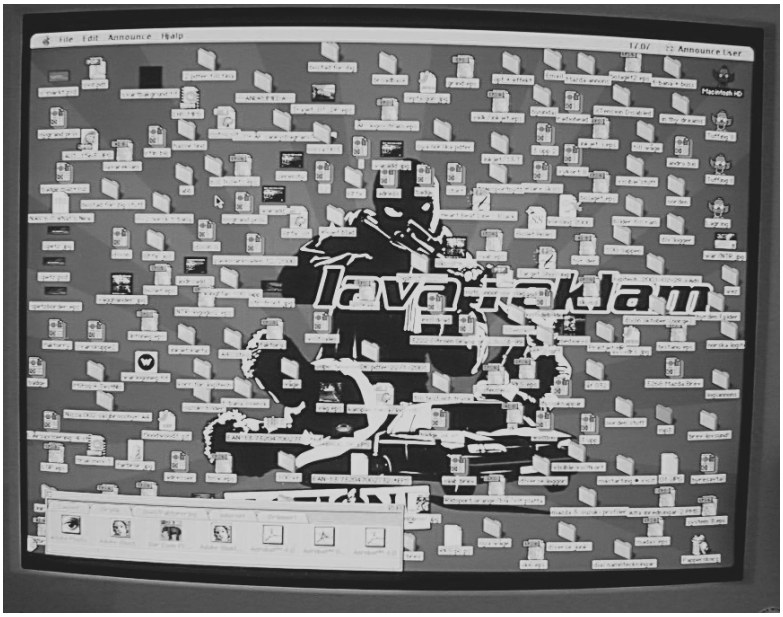
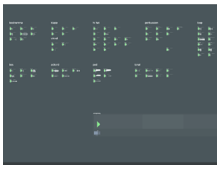


Figure 5.2: A picture of the contents of graphic designer's desktop. All the information is present, different project are located in different regions.



to locate information: everything is on the desktop. Also each customer has a region for which files and folders are located. A completed project is archived into a folder. The archived project folder remains at the file cluster position on the desktop. The user explains: ”I may need to reuse some of the stuff for this [the users pointed at a location of the desktop] customer, it may be a logo or a photograph. If I remove it [the folder] from the desktop I forget where I put it”.

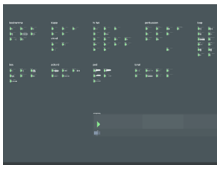
5.4 Design Implications

Spatial semantics is a powerful mechanism that goes well with human cognitive abilities. The data surface flat property along with persistent information was designed to leverage on Stea and Tversky’s work. The theory also rejects three-dimensional representations that are expected to have a malicious impact on navigation satisfaction due to the metal rotations users have to perform. With spatial semantics users know the kind of the information elements from their position. Users can create landmarks by annotations and sub-area background colour. For instance songs that the users have created themselves could be located in a region to the top right of the data surface, with a big label ”my songs”. Songs that the user ripped from CD:s could be located to the right-hand side labelled ”from my CD:s”, and songs shared with some friend could be on the bottom right-hand side perhaps labelled ”shared”. Note that the entire right area is devoted to music, but the different sub-areas have different attributes to them. Tversky [49] also showed that people map hierarchical relations to different scaled objects on a flat surface. Archives can thus be created in-place. The observed user for Figure 5.2 achieved this by replace related files with folder that contain the files. Instead of folders, archived can be created by shrink the content information elements. The visual appearance of the content remains the same, and internal spatial relations between the content elements are preserved. According to Dahlbäck et. al. [31] user navigation performance in a virtual space correlate strongly to their spatial abilities. A flat surface with orthogonal projection is less dependent on users ability to perform mental rotations, than 3D or hypermedia.

5.5 Summary

Spatial semantics is a powerfully human ability to utilise in computer program interface design. The computer screen can display environments of informa-

tion that works well with this ability. However, we have to be careful with rotations, higher Euclidean dimensions, and dynamic information. Spatial semantics utilisation is shown in chapters that describes the prototypes.



Chapter 6

Music Application Area

Users Say: We Do Not Like to Talk to Each Other

Parts of this chapter have previously been reported in the paper ”Users Say: We Do Not Like to Talk to Each Other” in proceedings of IGC2003: Second International Workshop on Interactive Graphical Communication, Queen Mary University of London, (2003).

In this chapter I will examine the music application area. The data surface paradigm was applied to this area for many reasons; music has a temporal modality, music is real-time, music creativity is a creative process, and music creativity is a social activity. This chapter present what users need and expect from a computer music tool, and observation results of users that try to make music together. Cues to a design from these results were found in the scenario about David and Lisa.

The trend in current music technology is toward software emulation of music hardware equipment. The term used by artist and music producers is ”native synthesiser”, an indication of that the software is runs natively on the computer’s processor. With this software approach users do no longer have to document the settings of their equipment by manually write down the parameter values. This process was, during the years of transition from computer-controlled hardware to native synthesisers, only partially automated by the use of MIDI system exclusive compliant equipment. With a software music studio

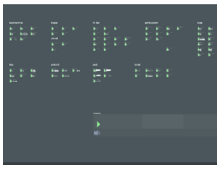
users can comfortably save all the parameters to the computer's disk. The Software also has the advantage to be limited only by the processor for the number of notes played simultaneously. Multiple instances of emulated vintage synthesisers are enabled saving both physical space and money. A vintage synthesiser of the 70s had physical knobs and sliders for every parameter. Knobs have direct audible and tactile feedback. With software this interface is replicated with photo-realistic interactive sliders and knobs on panel backgrounds that indeed look like synthesisers. However, the tactile feedback and precision are gone. Interviews with software music tool users shows that users find this as one of the most disturbing idiosyncrasies about music software tools, apart from no tactile feedback, audio feedback is often poor due to operating system event latency, and the mouse permits only one parameter to be altered at a time.

The desktop paradigm interface of today's popular operating systems is designed for one on one interaction [10, 26]. Hardware units are replaced by software, users' access is limited to only one artifact: the personal computer. Hence, with software there is no auxiliary hardware, the entire interaction with every instrument is done through the desktop interface of personal computers. This impedes collaboration. Create music used to be a social activity, now it is a solitary one.

6.1 Creativity

Music creativity like other types of media creativity is an open-ended task. The theory of reflection-in-action Schön present a model for how professional artists and designers engage creative work and open-ended problems [41]. A brief description of this model can be summarised in 4 steps: 1. Frame the problem, evaluate, and try to understand it. 2. Make a move, perform an action on the framed problem. 3. Reflect on the move, evaluate the consequences of the action. 4. Repeat the process. What Schön showed was that an artists do not have any standardised approach. Instead they rely on intuition based on experience to form initial hypothesis, which is tested by conceptual designs, experiments and actions. By make moves artists learn more about the problem and in the process transforms the problem to a more desirable state. The artists engage in dialogue with the problem, make moves, and evaluate how the problem ”talks back”.

The famous Swedish designer Olle Eksell describes this creative process for design in a list of verbs, to design is to: plan, sort, clean, organise, calculate, and form [13].



The point here is that the frequently used model of goals in Norman's execution-evaluation cycle [34] can not be applied to interaction for open-ended creativity task. On the other hand to write a document in an office context was very early described by D. Engelbart as a creative process [14]. Engelbart's description of document authoring was that people deal with small units of information, concepts, extracted and condensed from documents and other sources of inspiration. These concepts are considered to be raw material that people transform, integrate, compare, lose, forget, and even throw away.

Mynatt et. al. suggested that the single state document model of the desktop paradigm design [48]. In the context of reflection-in-action they propose that content information should be able to easily support the evaluation of actions, before and after the move. According to their observations users of creativity software packages did evaluation of actions by invoke undo and redo commands repeatedly. They said that users must be able to work with multiple partial copies, and that this process should be free from explicit file management. Creative processes are also collaborative by nature [32].

The mutual awareness problem for music creativity was attacked by Bryan-Kinns et. al. They invented and investigated a musical visualisation and interaction technique for group improvisation called the daisyphone metaphor [33]. It has an arm that circle clockwise over a disk. The disk has holes where users put musical event blobs. Pitch is mapped to the distance from the origin. The shape of the event blob indicates instrument, and the colour indicated the user. The daisyphone was tested with tablet computers so that users could annotate by scribble and draw with a pen. The tool and its music representation particularly supported the cyclic property of music, mutual awareness of actions, and mutual modifiability.

The Dasiyphone is based on a web based client/server architecture for collaborative multi-user music creativity by P. Burk [4]. It synchronises music creativity at control level for multiple remote participants. The architecture together with its demo service called WebDrum was the starting point for remote music creativity on the web. Burk saw, despite its vast number of users, the solitude of the web and its potential as a creativity arena, close to the field of social navigation.

6.2 Background Interviews

I have conducted interviews with the purpose of elicit users attitudes towards and experience with current music hardware and software tools to serve as

inspiration for a design. I made four interviews two in depth interviews with professional music producers and artists, and two focus groups with music software novices. I used a question cluster interview technique where each question is has a set of subquestions that can be used as complement if the respondent leaves something out or misunderstand the question. I wanted to know the interviewees background, and if they preferred to work alone or in a group. I wanted to know about their creative process, if their tool support the process, and if the process is a result of the tools they use. More on, I was keen to know what role the Internet had in their creativity and if they feel bound to a specific location.

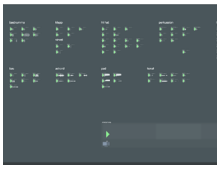
6.2.1 Common Denominator

The common denominator among the interviewees were that a music tool should be fun to use, sound well without have the user to work all preferences, and they want to collaborate in a creative way. They do not feel that the mouse is an appropriate device of control for music software. The worst case is when the metaphor of knob is used to set a value of a parameter; inconsistent behaviour among different applications, lack tactile feedback, poor audio feedback due to operating system event latency, and the mouse permits only one parameter to be altered at a time makes the interviewees uncomfortable with this interaction artefact. One of the experienced user interviewees described the creativity dilemma: ”music is a set of acoustic element along a time line, this makes it reasonable to move rectangles around in Cubase[1] , thou it is not very artistic”. All interviewees agreed that the tools they use affect ways they reflect on music. Music created by playing a guitar forms another concept than music created with computers.

6.2.2 Experienced User View of a Device

The experienced users have given their creative process a great deal of thought, and were much more aware of their actions compared to the novice users. The two interviewees characterised the their creative process, as a process with few rules, that a song can take any form, and that songs often start from zero. They described that they often feel as if the song already exists and that you have to switch of your conscious self, the song talks back to you, and shows you where it wants to go. The artist role is to focus and go with the flow. One of the

[1] Cubase is a sequenser music software package and a trademark of Steinberg Media Technologies AG



artists have found that the popular visualisation of the music along a time-line is indeed very effective as a measure of the physical time property of a song or a phrase, but apart from that he can not see many advantages in the time line model. Part of the process is to look for new tools that can do unexpected things. The other artist worked extensively with the computers abilities for algorithmic note and signal processing. For him the computer is a tool *p*er se.

The experts summarised view of the current music tools' design where against the established metaphors between vintage hardware and current software. Specifically they pointed out that the photo-realistic representation and metaphor of knobs. The tools they use did not support collaboration in any other way than turn taking by send their files back and forth.

The experienced users want to have a tool that is perceived physically, hardware with a sense of touch; an all in one collaborative digital studio should be portable, a gadget, like Game Boy[2]. They would like to have the possibility to improvise across the Internet, however, collaboration are, in their opinion, more fruitful in project groups. Another problem with current technology is that its hard to persistently store sound samples, software synthesizer sound preferences, and songs. The interviewees experience difficulties with todays interfaces, its hard to annotate sound and inflexible to navigate.

6.3 Observing Users

I designed a study to see how current state of the art music software supports or can support creativity, collaboration and improvisation. A possible way of introduce collaboration back into computer-supported composition would be to synchronise a set of computers via MIDI. From a studio point of view this would lead back to the problem of have all the files related to one song spread out on multiple machines. However from an improvisational point of view it might be interesting - which is the basis for this experiment. The study was set to answer a number of questions.

- Would the subject users enjoy the type of improvisation and collaboration the test was designed for?
- Would the users take roles with different responsibility?
- Is a shared acoustic space enough for vital music improvisation and collaboration?

[2]Game Boy is a registered trademark of Nintendo Co. Ltd

- Should there be a shared visible space as well?
- Can users navigate sound files by name conventions usually found on sound collection CD-ROMs?
- Should there be other means of content navigation?
- Can music improvisation be vital across the Internet?

6.4 Collaborative Music Improvisation Test

I created a collaborative environment was for the test. Current state of the art music software enabled the creation of a common acoustic space. Three PowerBook computers were connected and synchronised through MIDI. The MIDI protocol allow only one unit to be master, hence only one computer could be used to set the tempo. The other two computers were slaved to the master unit. The software used for the test was Ableton Live[3], a live performance package (see Figure 6.1) The interface of the software was customised to display only the most necessary panels, those were: the sound collections in the left panel; and the big sound matrix and mixer panel (see Figure 6.2). Ableton Live has a sequencer that performs pre-recorded phrases of a track knows as loops. To add a loop to the song the user drag and drops the file from the sound library into a sequencer cell. A mouse click on the play button of a cell start plays its loop. A column of cells in this matrix represents a track into the mixer. The start is always aligned in time with the rest of the song, hence the user does not have push the button at the exact moment.

The test was divided in three sessions, all in 10 minutes each. Free private improvise tutorial, Internet collaboration scenario, and live collaboration jam-session scenario.

6.4.1 First: Free Private Improvise Tutorial

The goal of this session was to let the users become acquainted with the tool. Before the users started to play, they received an introduction and a demonstration of the equipment and instructions how to use it. The users were only able to listen to their own song into their earphones. The main output to the speakers was muted and the synchronisation was switched off. Hence, the users focused only on what they were doing without the interference from the others.

[3] Ableton Live is a trademark of Ableton GmbH

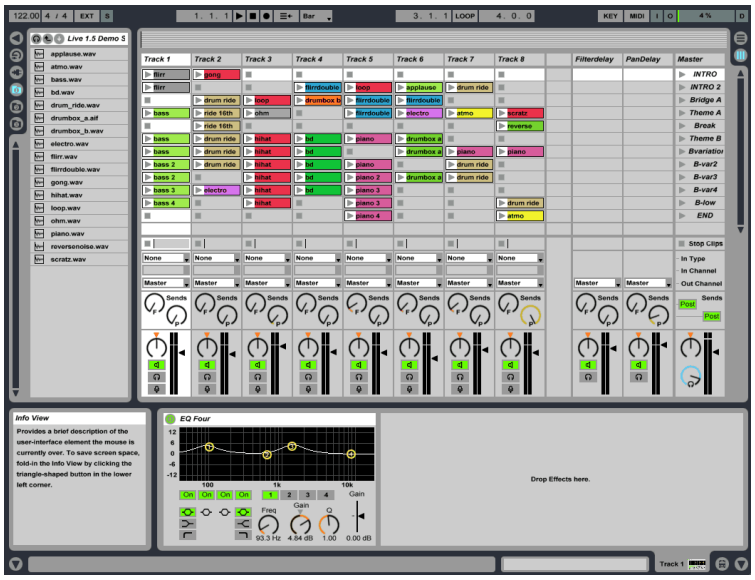


Figure 6.1: Screen dump of the demo song, all panels are visible. The top left panel holds the sound loop library. The main panel holds the matrix sequencer and the mixer. Cells for each loop are in the upper region of the main panel, and the mixer is in the bottom region. The left bottom panel holds an information and feedback panel. The bottom right panel holds effects for echo, reverberation etc.

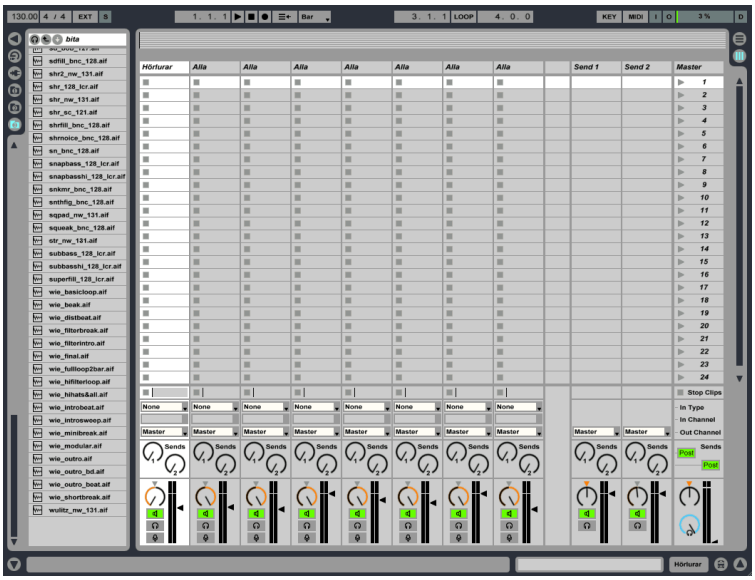


Figure 6.2: The view confronted by the users, only the sound collection in the left panel and the main panel with matrix sequencer and mixer are visible.

6.4.2 Second: Internet Collaborative Scenario

For this session the goal was to simulate how to collaborate with a tool like Ableton Live via the Internet. To achieve this the users were asked only to communicate with each other through the music. They had to refrain from any other form of communication. The computers were now synchronised and all the music came out of the speakers. The subject users' task was to collaboratively improvise music.

6.4.3 Third: Live Collaborative Jam-session Scenario

The purpose of the session was to see if the users would negotiate roles and responsibilities. The users were allowed to communicate in any form; verbally, exchange glances, and gestures. The acoustic counterpart to this session would be the jam-session known from Jazz. The subject users task was to collaboratively improvise music.

6.5 Gathering data

The protocol method for the test was to document observations by take notes with pen and paper. The notebook was prepared with the metrics that the test was design for. The note-form had plenty of space and margin order to get those extra observations that were orthogonal to the metrics.

6.5.1 Metrics

- Usability of the interfaces in general: find audio files, usability of matrix sequencer and the mixer.
- Degree of satisfaction: do the subject users enjoy this form of collaboration or are they frustrated.
- Methods: how users evaluate loops before they are put into the song, and how they build songs in collaboration with the other participants.
- Feedback: do the subject users seam to get enough feedback to be able to collaborate and participate?
- Role: what roles do the subject users take?

- The tools effect on collaboration: how well do the subject users collaborate in the test, what kind of emotions do they express toward each other?

Focus group interview followed up the observations. The users' views, thoughts and comments of the test had to be evaluated. The questions regarding the test were open ended and clustered around: collaboration, role, satisfaction, and usability problems.

6.6 The Subject Users

Two groups with three persons each were invited as subjects for the test. The first group consisted of users with little music software experience; the second group was music software experts. They had to fill out a form if they played any instrument, of their previous experience with music software, and of what genre they preferred. In the second group the third participant chose not to participate, that left me with only two music software expert users. Those who responded to the call eventually turned out to be good musicians; hence the scale novice to expert only refers to the subject user's skill in music software tools.

6.6.1 Music Software Novices

There were two men and one woman in this group. They had profound experience in playing different instruments; hence they were good musicians. However, they had little experience with music software tools. The three users had flute, violin, piano, guitar, drums, and bass in their repertoire. Their knowledge in music enabled them to recognise and use different musical components. For instance, they were familiar with a hi-hat rhythm loop and its applications. Also they were all very skilled computer users, hence the computer itself and its interface presented no obstacle. One of the users had used a pedagogical music tools for kids. The other two users had some, but old, experience of Cubase. The software we used for the test: Ableton Live, has a different conceptual model from Cubase and other conventional sequencer software; Ableton Live, has a different view of time; and you never hit stop to consolidate what to do next, decisions are done live. This and the fact that the users experience with Cubase were about ten years old, made it irrelevant for the test.

The novice users had a broad variety of taste in the range from motown, rap and jazz to renaissance and folk music. As one of the users put it: "Everything

but Eurodisco!". This group would be referred to as the novice group throughout this paper. Where novice, as mentioned above, should be understood as novice in music software tools.

6.6.2 Session One: Free Private Improvise Tutorial

During the first session the users in deep concentration explored and learnt the software. Although they were experienced computer users they surprisingly relied on double-click. For instance they double-clicked the play button of loop cells and the sound source navigation folder button when these buttons only required a single click. As the minutes passes they came up with methods to combine and evaluate sounds. One of the users even started to explore the software, engaged more advanced features such as effects. But no one even thought of change tempo.

6.6.3 Session Two: Internet Collaborative Scenario

In this session the users seemed to be eager to communicate. One of the users tried to use body language and facial expressions to show his frustration with sounds that one of the others had put there. One of the users grabbed the open place whereas the other tried to be polite; they hesitated before they put anything in public. After a while the intensity of the music increased, but they were also expressed thematic patterns of sounds. Also they started to use the mixer in order to adapt the sound volume to the rest of the song. As observer it was entertaining! They also seamed to be frustrated to not be able to express critique, and not to be in control of all sounds. Eventually they became tired from all concentration.

6.6.4 Session Three: Live Collaborative Jam-session Scenario

When the users were given the opportunity to communicate they expressed vigorously that they were tiered of certain sounds, two against one. In the beginning this session they gave each other cues of what they did, for instance: "I'm working with the bass and the drums." They also expressed what they liked in direct utterances, as an example: "that bass is very good, keep it! They also negotiate on what should be included. As listener this negotiation results in better quality of output. When they communicate, they use very short utterances and direct instructions like: Remove your drums!, Listen to this!, or I

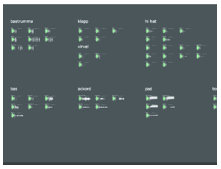
have something here you don't like to hear? ha, ha. The role and responsibilities that they had given one-another made this communication possible.

As the minuets go by the music becomes more rich and full of complex rhythmic and harmonic structures. The users shouted out in appreciation: ”Yes! Wow! They also started to find explicitly expressed themes: ”lets do something with this. followed by musical events that one of the other participants commented: ”Sounds like pac-man”. An interesting term they use to express what they did was: collect. As an example one user called out: ”Lets collect things that are mysterious!” The term collect is very descriptive; hence this suggests that sound collection is a usable label for the set of sound files used this context.

6.7 Debriefing Focus Group Interview

How did you experience the collaboration? The users thought it was difficult to find a role, almost total anarchy. The users needed a form or convention for how to improvise music with this kind of tool; they claimed it was dangerous not to have an idea to start with. One of the users felt like he took to much space, which was confirmed by the observations. The users felt they had no language for discussion of technical details. It is worth to remind that although these users were well versed in music as well as computers, knowledge in how to user music software tools could not intuitively be transform from these related domains, due to the subculture, genre or jargon of music software tools concepts. Surprisingly, the users thought it sounded better and was more fun in session two when they did not communicate. This contradicts observations, from the subjective point of view when the music in session three was richer and more entertaining than in session two, and because two users were so eager express their wish to get rid of some sounds that had annoyed them in session two. Was it entertaining? The users commented: ”Yes, it was fun!” They like the idea to be able to sit at home and create music with others by collaboration via the Internet. They found the Internet scenario most satisfactory, whereas, they did not like the third jam session scenario as much.

Did you experience problems? Bad and coincidental cataloguing of sounds made it time consuming to search for sounds and collect sounds. They would have liked to have a broader collection of sounds. Anonymous sound labels were obstacles to collaborate fluently. They thought that sounds collect should be done from groups of spatial semantic organised sounds where the location of a group in relation to other groups reflects the properties and applications



for the sounds. They also wanted visual feedback of the sounds that were put in public. They wanted to be able to see what everyone else is doing in a shared workspace.

6.8 Music Software Experts

One of the users played the piano. He derived his computer science education and profession from his interest in music. He started to use computer music tools with Cubase for Atari ST. Today he still uses the modern versions of Cubase for recordings. Rock, pop, ambient and classic music was his favourite genres. The other subject user played harmonica, accordion, and guitar. He had experience from many different music software tools; his favourite tool was Emagic Logicł [4]. He also wrote his own MIDI-file generation software in the 80s. The reason for him to start with electronic music was the lack of skilled musicians to collaborate with. He could not single out any particular style as his favourite one. He mostly liked to improvise. These two subject users were both good musicians, experienced computer users in general, and experts in music software tools. This group will be referred to as the expert group.

6.8.1 Session One: Free Private Improvise Tutorial

During the first session the expert users almost immediately found a method to examine sounds. They put sound in a column into the cue track, examined them one by one. They users acted like they already knew how to use the software. They looked for what sounds there were, but did not use the time to explore the capabilities of the software. One of the users explores the sounds one by one, and removed them after examination. They also started to build songs slowly and carefully, but erased everything after a minute; this behaviour was repeated a few times. They are patiently looking for sound, suggests better navigation. Eventually one user calls out: ”Now we know this!”

6.8.2 Session Two: Internet collaborative scenario

The users hesitated for about 30 seconds before they let any sound come out of the speakers. Users were tried out the sounds cautiously before they put anything into the public mix. The music was extremely minimalistic. The user in control of tempo changed it from 130 beats per minute (BPM) to 116. They

started to use the mixer to affect the sound and as a tool for expression, create a rhythm by play with the mixer's sliders. This was also expressed in the focus group evaluation; they felt lack of tools for control but used what was available. Where the novice users had created a rich and thick sound, these users created a much more restricted and contemplated sound. They evaluated every sound and its possible use, and created a song by fade from one sound to the other in a sequential order.

6.8.3 Session Three: Live Collaborative Jam-session Scenario

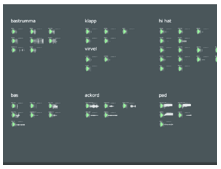
In the third session the users started to negotiate, first the tempo, than what the overall idea should be. "Let's run a backward song". Although the users did not know eachother before the test, backward song was a usable term, they seemed to agree what it meant. They discuss key, and problem of how to find sounds, the problem of how to find a bass, and need for functionality of sounds for instance of those with more overtones. They made expression of need for better control like: "I need two mice to cross-fade". The users thought it was difficult to figure out what sounds came from whose machine. They still did not explore Ableton Liveł for more controls.

6.9 Debriefing Focus Group Interview

How did you experience the collaboration? They found it simpler to collaborate only with sound in the second session. They thought acoustic feedback was enough. They thought it was difficult to put words to what they wanted to achieve. A common language is needed, and a global metaphor.

Did anyone of you take a role? The users felt themselves to be controlled only by the tempo. The tempo did set the frame to what they could do. The overall sound was most important to them; what sound had to do with this and that? They felt that the loops provide little room for expression.

Was it entertaining? They felt it was entertaining, but only one step beyond how one performs as DJ. However they found the software to be more of a toy, and again the loops are very constraining. They thought that you needed a plan. Something they only could form in the third session.



Did you experience problems? They felt that cross-fade from one track to another with the mouse is impossible. They would like to vary beat or melody with a group of sound in each loop to select from, like the 15-game.'

6.10 Discussion

There is much more to these two groups of subject users than the level of experience. For instance it turns out that the novice group have more of lateral thinking; they combined loops from different genres that were not meant to fit together. The expert group, on the other hand, have more of vertical thinking; they built songs by carefully evaluate each loop. The novice users thought only in retrospect that they should have had an idea, but the expert users knew how things should be done; they had the idea of a ”backward song”. However, the users disregard the level of experience, did not like to talk to each other while they improvise music. They expressed that they felt more comfortable and satisfied to use only the music as means of communication. This came as a surprise, especially since the music result was richer, more interesting and of higher quality during the third speech communicative session. This also contradicts their observed behaviour during the second session, in which the novice users are annoyed by elements in the music beyond their control they dislike. If there is a difference between observed behaviour to what users claims, the former would be more valid. My conclusion must be that it is more comfortable not to complain about annoying sounds, however for collaboration in the long run it is an unacceptable situation and it is more satisfactory to communicate. According to Cohen [5] spoken references to objects that are perceptually available to the listener are nearly always indirect. However, although sounds are perceivable they are difficult to describe verbally. The subject users in this experiment complained about their lack of a language, why they had to be as direct as possible, thus they took the risk to offend each other. Also, creative work as an open-ended task has an unstructured form of conversation [10], thus it is not possible to push theory of a generic structure on this experiment such as conversation for action [52].

6.11 Design Implications

For the design of a collaborate tool this aspect has to be really though through. The users should have some degree of control of the elements of a song that was put there by another participant. The subject users suggested a common

visual workspace with feedback of what everyone else is doing should simplify collaboration and communication. Another problem the users points out is the navigation of sound files. The users suggest spatially semantically organised groups according to characteristics of genre, tempo, timbre, and key.

6.12 Summary

This Chapter presented empirical result and theory from the investigations of the music application area. The design implications concluded from the result of these investigations were considered in the designing of the data surface paradigm and the music creativity improvisation collaborative prototype tool.

Chapter 7

Concept Prototype

This chapter will describe the concept prototype used early in the project for usability assessment by user evaluations. The prototype was a mock up interactive animation build with Macromedia Flash [1]. The first sections of this chapter summarise the essences of the previous chapters and sets a context for the design rationale of the concept prototype. The previous scenario sets the context for the design of this concept prototype.

The contents of this chapter have previously been reported in the proceeding of: The Good, the Bad, and the Irrelevant, Helsinki Finland, in the Media Centre Lume of the University of Art and Design Helsinki (September 3-5 2003) Cost269 User Aspects of ICTs

7.1 Identify Troubles

The initial step toward the design of the first concept prototype was to identify the source of the troubles with the desktop paradigm interface. Historically all personal computers have had file systems. The design of this most vital part of personal computers has consequently affected the design of user interfaces. The file system can be traced in interfaces from today's graphical user interfaces to yesterday's command line user hostile DOS (Disk Operating System) interfaces. Systems like DOS were even explicitly devoted to file management,

[1] Macromedia Flash is a registered trademark of Macromedia Inc

and the word ”Disk” in its name points at peoples concept of what a computer is, or rather was. The current desktop paradigm adds only a layer to the early systems, the whole desktop metaphor is devoted to explains how users should treat files; as documents, folders, and applications. This is a result of the desktop paradigm intellectual foundation in the metaphor value.

What if, as an intellectual exercise, one was to exchange the spine of computer system from a file system to a database? Imagine the unstructured stream of ones and zeros you normally put in a location on a hard drive instead be put in a content aware database. Break apart the monolithic applications into components and put these in the database too. All the different content files turned into compound objects that are put into the database and so forth. Such an exercise introduces a number of questions that include how to handle: information access; information visualisation; user collaboration; multi modal interface usability; program architecture; scalability over different platforms; to name a few.

7.2 The Trouble of Metaphors

There are many situations where interaction designers used peoples’ knowledge in world outside the computer to explain the behaviour of an application program. In the case of the desktop metaphor, peoples’ knowledge about desktops, documents and folders were utilised to explain the behaviour of file-trees [11]. On the desktop you may put all the files related to current activities. If you do not archive your work when the finished with it, you eventually end up with a mess. The trouble with metaphors, as described in the chapter The Values of the Desktop, is that they introduce limitations from the source domain. In the real world you have to tidy your desktop due to the lack of space. But when you do, you will later on have trouble recalling where you put things. This situation is ascribed to the metaphor value. The desktop obeys real world constraints such as limited size, but the computer world is virtual world and obeys the rules we make. For instance, what happens if we remove the size constraint for the desktop?

7.3 Tools vs. Contents

The desktop metaphor manage to explain many of the essential attributes of computer systems, but did not really change them. One of them is the functionality or service that extends the computer’s system abilities, in other words

the application programs. Programs are still today distributed as files. They are big and monolithic. However one application can not cover all the users' desired services. Thus, users have to deploy a number of application programs for a specific task, most likely these applications share a basic foundation of tools. Due to the structure of the computer system these tools are implemented separately by different vendors, they have slightly different behaviour, appearance, and attributes. The consequence of this is that the consistency value does not hold, and users have to abide inconsistency. It is the users task to appoint a file type to the application program. Even though Macintosh have used file type and file creator identities since 1984, they work only under the assumption that one file goes with only one application program. Some formats have received more acceptance than others, still due to¿ the design of desktop systems, applications, *t*ools, are in focus and not the document *c*ontent. From a users view with tools in focus, the question is: *W*hat can I do this tool? instead of *W*hat tools work with this content?.

7.4 Prototype Design Rationale

With the early prototype I focused on organisation, navigation, and visualisation of persistent state content. The two state model for files, opened and closed, was replaced by a unified model to better satisfy the modelessness value. I have chosen to create an illusion of a data surface. Think of is as a flat infinite map of information, like a magic paper that can be stretched to contain everything you need. The layout provides spatial semantics, spatial context and hierarchical relations. Content remain persistent in place attaching attributes of the perceived stability value to the prototype. The identity of the graphic modality information is the information content itself. Icons are unnecessary, since out zoomed visual appearance of content serve as icon. Acoustic information such as sounds also has a graphical modality representation; this may be a wave graph. Synthetic generated sounds may even have a linguistic representation as source code of the sound. Different modalities of contents afford the visual appearance, the design of the appearance goes back to the aesthetic integrity value. The forgiveness value induce that every manipulation of an information element can be traced back by unlimited steps of undo/redo. All manipulations are implicitly persistently saved. The data surface is navigated by zooming and scrolling; by an overview index map, analogous to a site-map called Index Map; and by incremental search.

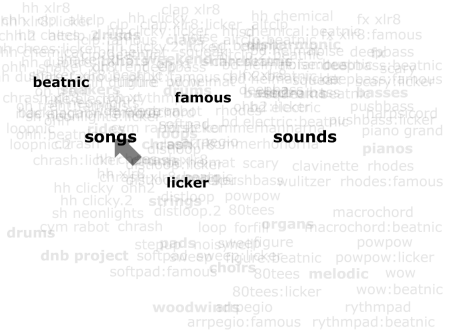


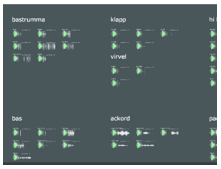
Figure 7.1: The user touches the ”song” node with the cursor indicated by the arrow.

7.4.1 Index Map

The index map (Figure 7.1 7.2 7.3) is Löwgren’s Sens-A-Patch applied to this environment [27]. The Sens-A-Patch, or SAP for short, is an interactive visualisation technique by which users collects overlapping clusters of labels in 2D-space. Information in a SAP contains spatial semantics and context. Each cluster has a caption; each caption can have a parent caption, thus, a label is either a leaf representing content of the data surface or a caption representing a collection of content in the data surface. The index map provides a shortcut to information of the data surface, and enables users to gather information outside current focus. The SAP displays, in a small screen area, both the hierarchical relations between content different scale as well as the spatial relation within the same scale. Hence, the context of information is better preserved by the SAP than with the hierarchical tree view used by Pook et. al. [38].

The index map is navigated by moving the cursor over a node that opens up all the labels clustered around it. Parent nodes and sub-nodes can be reached in each cluster. In Figure 7.1 the user has touched the ”song” node, there are only another set of nodes available for this ”node”. In Figure 7.2 the users has touched the ”beatnic” node that opens up all the labels cluster around it. In the next step Figure 7.3 the users is touching different labels of the the ”beatnic” cluster. The labels that are close to the cursor are more opaque, whereas the other labels are transparent. A touch of a node label plays a sound ”earcon”, representing the node cluster contents.

The index map is visually present as a thumbnail icon located in top left



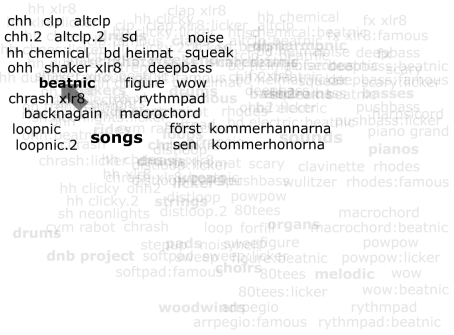


Figure 7.2: The user touches the ”beatnic” node.

Figure 7.3: The user browse the ”beatnic” cluster.

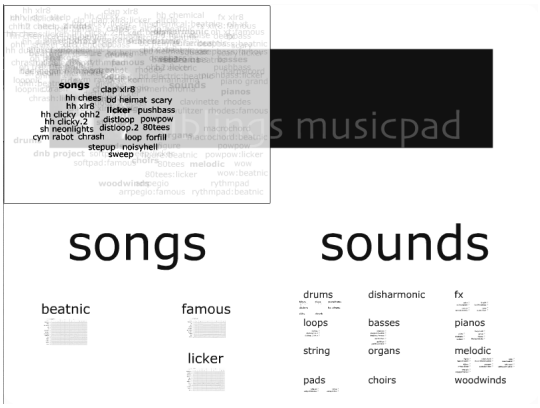


Figure 7.4: The overview of the most out zoomed state of the Concept Prototype. The index map is enabled and browsed by the user.

corner of the screen. It comes into view when the user clicks or taps on it. A click or a tap on a touched label moves the view of the data surface to the labels corresponding scale and location. In Figure 7.4 the user has touched the ”licker” node label. In Figure 7.5 the node was clicked shrinking the index map to a thumbnail and moving the view to the scale and location of the song ”licker”.

7.4.2 Zoom

When users want to see a detailed view of their information, they zoom in on the element of interest by taping or clicking on it, a design inspired by the Pad by Perlin and Fox [23]. When a zoom action is invoked, the previous view is displayed as thumbnails listed with increasing depth to the left of the display (see Figure **??**). This design gives the users historic navigation context. When zooming out, the user taps the thumbnail representing the desired magnitude of zooming from the list. Zooming is done in discrete steps to the context of the information element the user taps at. Previous zoom states are shown as thumbnails. (Only the upper left part of our prototype data surface is visible in this figure). Using thumbnail views embedded in the current view is often used as visualisation technique by news papers for locating countries within contents or cities within countries, in this particular case the inspiration came from the Swedish newspaper ”Dagens Nyheter” and from Holmquist’s FLip-

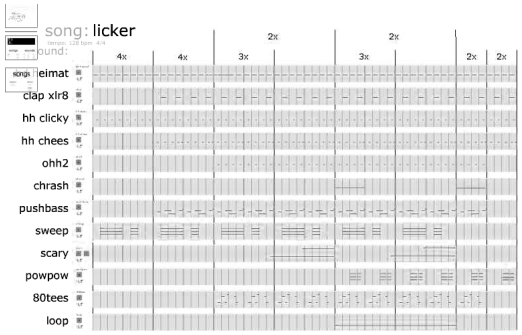


Figure 7.5: In this figure the user have browsed the index map and clicked the node label ”licker”. The surface view is transferred to the nodal label’s corresponding position and zoom scale.

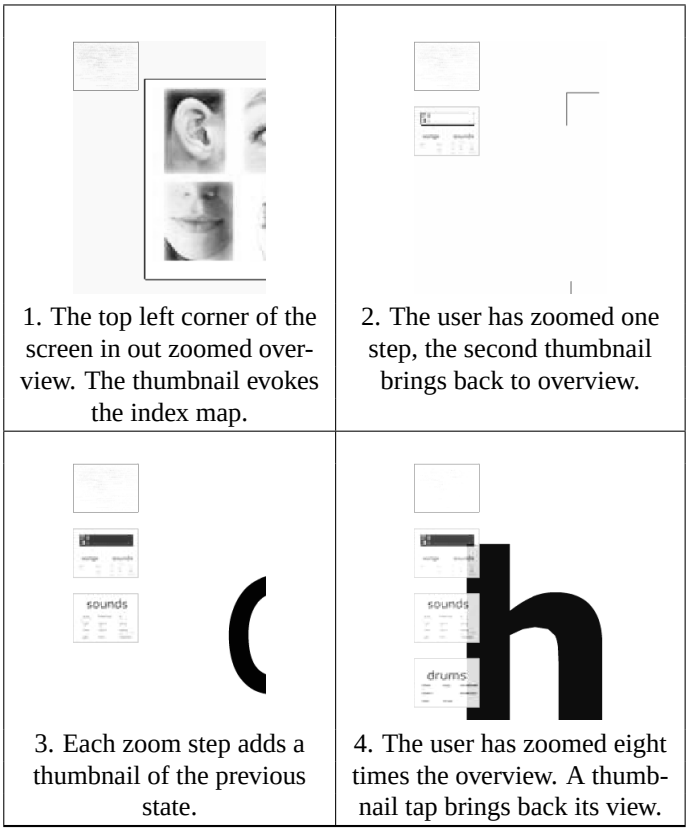
Zoom thumbnails [19]. To scroll the data surface users grab the surface and move it.

When this grabbing and move gesture is done rapidly, the view snaps to a grid displaying a minor part of the previous view as context for it. Scrolling the data surface is only preferred when users know that the information needed is in the close neighbourhood of the current view. Zoom out-and-in action is preferred when the user has to inspect an overview to find the needed information. Information at larger distance in data surface can be collected with the index map.

The effect of zoom is shown in Figure 7.5 The data surface display the contents of the songs score. Here, time runs from left to right. To the left there is a list with all the sound labels for each track. Between the label and the track’s score is a shrunken view of the sound parameters and source code. The tracks hold the score note values. All note values are displayed at once; users do not have to open an extra window to inspect the notes values for a track.

7.4.3 Incremental Search

Incremental search is invoked by writing a letter on the intended search pane. A selection hit is immediately indicated, both in the local scope and the global index map scope, for the content element that contain the letter. The user may



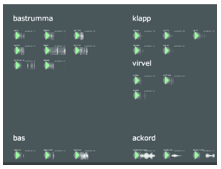
1. The top left corner of the screen in out zoomed overview. The thumbnail evokes the index map.

2. The user has zoomed one step, the second thumbnail brings back to overview.

3. Each zoom step adds a thumbnail of the previous state.

4. The user has zoomed eight times the overview. A thumbnail tap brings back its view.

Figure 7.6: Zoom history thumbnails in the top left of the screen.



tighten the condition by writing a second letter, now all elements containing the substring become selected.

The only logical conjunctive is: AND, represented by space. The natural language OR is not the same as the logical OR, this ambiguity may lead to confusion [18].

7.5 Evaluation

The Macromedia Flash concept prototype was shown to five users who were asked to complete five navigation tasks in a cooperative evaluation. WebPAL[2] tablet hardware ran the Flash prototype. The hardware was to slow to run the CPU intensive prototype fluently, however, in order to better grasp the concept of the data surface I wanted the users to switch conceptual context from the PC with display, keyboard and mouse. The subjects were made aware of this flaw.

7.5.1 Evaluation tasks

- Find the song licker. A simple navigation task to find a song. The purpose for the task was that the subject users should get acquainted to zoom navigation, and to the prototype's data surface layout.
- Edit the sound clp xlr8. The layout of the data surface was divided into songs and sounds, this and the state in which the subject users were predicted to leave the prototype in from the previous task, coerced the subject users to find the zoom out method; the zoom out thumbnails in the top left region of the display.
- Find the sound distloop. To complete this task the subject users did not have to zoom out to the overview state of the data surface.
- Read the source code of distloop. The purpose for this task was to see if the subject users recognised shrunken text elements on the data surface such as the source code text of a sound
- Go to the song famous, inspect a representation of the sound ohh, go back to the song famous. The task was designed to see if the subject users utilised the index map.

[2] is a registered trademark of RSC Technology AB

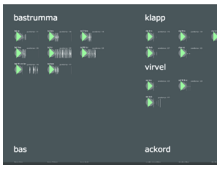
7.5.2 Subject Satisfaction

The subjects were all men, skilled computer users and computer musicians, age ranging from 24 to 36. Before the evaluation started the users were explained in a few sentence what the data surface was, however the zoom out buttons was not explained, and the exact workings of the index map wasnt explained either. All but one of the subjects completed the tasks without any trouble. One of the subjects completed the tasks in optimum number of action. The subject that had problems to complete the task saw the images on the screen as steps in a process and not as zoom scales on a surface. He didnt perceive the zoom out thumbnails as objects of interaction. He also complained about the lack of an update or save button. He thought the design was too good looking! He further stated: Kids will love it! The most successful user claimed it difficult to change thinking from the desktop metaphor to infoscape, however the precision and efficiency of his actions contradict his own claim. He was also fond of the index map: ”I really like this cluster view of the surface, it feels nice!”.

The overall attitude towards the interface was that they liked it, they liked the way you navigate, and they liked the structure. Those who discovered the index map like the fast navigation from one part of the data surface to the other. The feature that the subject users appreciated the most was that all note data was visual in every track. However they thought that this design should be extended to include all sound parameters, allowing control of attributes such as volume, timbre, dynamics, reverberation, etc. Another suggestion was that, in live performance situations, rules and algorithms could generate musical elements.

7.6 Conclusion

The evaluation of the concept prototype showed that the navigation components: the hierarchical context preserving SensA-Patch, and the navigation history providing zoom status thumbnails were sufficient tools for navigation of the data surface. I have found indications that users enjoy content browsing and the experience of a data surface view. However, zoom in discrete steps may confuse users and prevents them from experiencing the focused view as zoom.



7.7 Summary

This chapter investigated the user experience of a content data surface and zoom navigation. The results were elicited by user evaluations of a concept prototype. The prototype was constructed with a multimedia tool and had limited functionality. Still, encouraging results was found in favour of my approach, but not entirely satisfying. As a result of the unsatisfactory zoom experience, the subsequent chapter will look more deeply into fluent zoom and pan experience.

Chapter 8

Fluent Zoom Prototype

This chapter will describe the fluent zoom and pan concept prototype. The result from the previous chapter showed that the zoom and pan style of the concept prototype has to be redesign. The prototype was a C/C++ implementation mainly based on the Open Graphics Library (OpenGL). The prototype had to be a ”real” implementation; interactive multimedia tool packages lack the ability for realising fluent zoom. The fluent zoom prototype as evaluated with qualitative users studies.

The contents of this chapter have previously been reported in the proceeding of: The Good, the Bad, and the Irrelevant, Helsinki Finland, in the Media Centre Lume of the University of Art and Design Helsinki (September 3-5 2003) Cost269 User Aspects of ICTs

A new prototype was constructed based on a revised design from the evaluation result of the concept prototype. Strong indications were found that zoom interaction was a successful technique for navigating the data surface in the context of a music creativity application. However, some of these results were not satisfactory. For instance why did one of subject users not perceive a flat surface, but instead a sequence of images? Would fluent zoom with smooth transitions from overview to focused view clarify the visualisation of the data surface and be more satisfactory than the discrete zoom steps of the concept prototype?

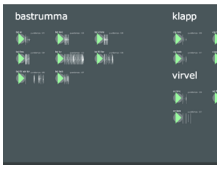
It is worth mentioning, again, that the concept prototype was strongly constrained by the restrictions of Macromedia Flash capabilities, a trade of I made to rapidly design an acceptable visual appearance. The fluent zoom prototype

was actually two prototypes. I wanted to test fluent zoom with two different approaches to pan. The first of these utilised a grab-and-move metaphor to pan the data surface. Zoom was view centred. The other prototype used a trajectory zoom method that allow both zoom and pan in one action [47]. An elegant method but would it be satisfactory for the subject users? Since the scope of what I needed to establish did not include the complex functionality of music creation the data surface was restricted to contain only a couple of articles from the BBC web news site.

8.1 Grab-and-Move

The storyboard in Figure 8.1. illustrates how grab-and-move pan works. The user wants to read the text document to the right. The first step is to move the surface so that the text to be read is in the centre, the user presses down the left mouse button, which is the ”grab”-action, then moves the surface to the left by moving the mouse in the same direction, illustrated in the transition from left to middle in Figure 8.1. The user rolls the scroll-wheel away from himself/herself to zoom in on the surface. Zoom is always, both in and out, centred to the screen and unrelated to the cursor’s position. In Figure 8.1 transition from middle to displays the zoom, the middle image view the state before and the right image the state after zoom.

Figure 8.1: Transition from left to middle shows when the user moves the surface. Transition from middle to right shows when the user zoom in on the text in the centre of the screen, accomplished by an upward roll on the scroll-wheel.



8.2 Trajectory Zoom

Trajectory zoom was used for the other prototype, trajectory zoom enabled both pan and zoom in one action. The Figures 8.2 8.3 shows an example for how this is done. Again the user wants to read the right document. To move the content on the data surface down to the left, the user puts the cursor somewhere in the bottom left area. Then they roll the scroll wheel towards themselves to zoom out.

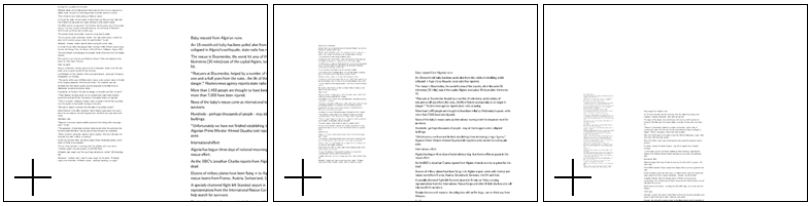


Figure 8.2: Pan done by zoom out centred to the trajectory marked with a cross. The left image show the cursor position before zoom out. The middle image is in the transition from in zoomed to out zoomed view. The rightmost image display the out zoomed state and how the surface was centred to the cursor.

Figure 8.3: The images from left to right show zoom action from overview to respectively detailed view of the content.

Figure 8.2 displays how the data surface shrinks and centres around the orthogonal trajectory from the view plane to the cursor's position on the data surface (marked in Figure 8.2 (right) with a cross). The user now has an overview of the document, but in order to read the document they need to zoom in on it. In Figure 8.3(left) the cursor was moved, and thus the trajectory, aiming to where the user wants to read. In Figure 8.3 the zoom magnification is displayed, where Figure 8.3 (right) displays the readable view.

8.3 Prototype Evaluation

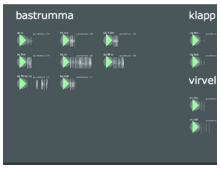
Evaluation was done by observing four subject users. One of these subject users was a female graphic designer, the others were computer science students: two female and one male. They were given a written introduction to the test in which the different zoom techniques were explained. The test was set up in two sessions, the first tested the grab-and-move prototype, and the second tested the trajectory zoom prototype. In each session the tasks was to write down the answers to four questions. The answers could be found the articles on the data surface. Users were observed while performing the test. A video camera and manual notes were used for protocol. I wanted to know the subject users opinion of these interfaces, hence, debriefing interviews completed the evaluation.

8.4 Subject Satisfaction

All the subject users agree to that this interface felt very different from what they were used to, except for one subject user who had implemented a zoom function as an exercise in a computer graphic course. All subject users did navigate the data surface without any difficulties, except for one subject user who had to put a great deal of effort in handling trajectory zoom.

One of the subject users had been using a graphic package that made her perform the zoom actions, using the grab-and-move prototype, with impressive precision. First she zoomed out to get an overview of the database surface. Then she moved the content to its proper position. Eventually she zoomed in on the document in one swift swoop, all other subject users had to do these actions in many repeated steps; first move a little and then zoom a little.

All the subject users easily mapped zoom direction to the scroll-wheel's roll directions of the mouse. They preferred the grab-and-move method for pan: they experienced it: ”like moving around a big paper”. One user claimed that he got a better overview when using the trajectory zoom. Two subject users made sure that the text had a broad margin to the right, this margin was used as a bar, which was grabbed each time a users wanted to pan, they did not want to ”touch” the text. Only one subject user gained interest in the trajectory zoom technique: she would have liked to continue to use it, and train for it. Her comment was that she intuitively felt trajectory zoom to be more powerful than view centred zoom.

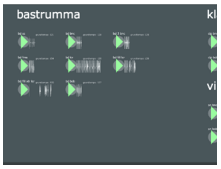


8.5 Discussion

The evaluation showed that fluent zoom was more satisfactory than the previously tested discrete zoom. Separated actions for zoom and pan was preferred by the users, thus a pan technique such as the grab-and-move metaphor should be used in the local scope, however trajectory zoom could not be rejected as navigation method for the data surface in the global scope. I have found indications that experienced users of graphic software packages perform zoom with high precision and accuracy. The second prototype should be extended with an overview panel to indicate the user's current visual locus. The result from the debriefing interviews shows that users should be able to engage multiple view of the data surface.

8.6 Summary

This chapter covered a more profound investigation of the zoom and pan technique. Users do not appreciate a combined zoom and pan in one action. Fluent zoom remedies the risk for confusion of discrete zoom. Results from the investigations reported in this and the previous chapter will be used in the next chapter in which we see the results of usability evaluation of a real music tool prototype.



Chapter 9

Music Tool Prototype

Previous investigations of the Fluent Zoom Prototype showed that users were satisfied with continues zoom, in other words, the zoom scale was to be set to any value instead of discrete steps. Previous investigations also showed that users wanted to aim toward the target of inspection for zoom. Although this may works as pan technique, the users preferred a separate action for pan. The previous chapter Music Application Area suggest better support in collaboration for open-ended creativity tasks, and that users find listed file labels for exploration and navigation of rich media content boring. This chapter will cover the design and evaluation of a prototype for collaborative music improvisation and live performance creativity. The previous scenario sets a context for the design and evaluation of the Music Tool Prototype described in this chapter.

9.1 Prototype Design

The Music Tool Prototype is a music creativity collaborative live performance tool. Users collect and arrange pre-recorded loops into songs. The functionality of the Music Tool Prototype was design to mimic the capabilities of Ableton Live to enable comparison between there evaluation results of the Music Tool Prototype with the result described in the Music Application Area chapter.

The design of the interface was based on Tversky's theory of Cognitive Collage [50] and Barreau and Nardi's "Finding and reminding:file organization from the desktop" [9]. I have cautiously taken peoples different spatial abilities, reported by Dalbäck et. al. [31], into consideration. Contradictory to Barreau

and Nardi's result I suggest that users need a search tool for content navigation. Schön's theory of how peoples' creative work [41] and Mynatt's et. al. design implications for creative and open-ended task software [48] were taken into consideration.

9.2 Prototype Implementation

The system can change tempo and transpose loops in real-time. OpenGL enables smooth graphics and fluent zoom. Ethernet and UDP packages allow synchronization of visual and audio content for collaboration on multiple machines. The implementation was done with the C++ programming language based on open source libraries from the game development community. The mainly used libraries were: Simple Direct Media Layer (SDL)[1] that provides a platform independent framework for events, sound, threads, and OpenGL context; SDL_Net which is an add on library to SDL for TCP and UDP network data communication; SoundTouch [2] supplied real-time tempo adjustments and transposing of PCM-sampled loops; FreeType [3] provides TrueType [4]; FTGL [5] was used to glue FreeType with OpenGL; and eventually all graphics was rendered with the Open Graphics Library (OpenGL) [6].

9.3 Navigation

Navigation of the content is done through zoom and pan technique called trajectory zoom, which I reported about in the previous chapter [47]. It allows both pan and zoom in one action. As an example see Figure 9.1, in which the cross marks the position of the cursor and where the user aimed. In Figure 9.2, zoom is half way through, and in Figure 9.3 the user zoomed to desired detailed level. Also see the animation in the margin of the text by flip through the pages of this these. Zoom interaction is done with the scroll wheel of the mouse. Look at the mouse dangling in its wire: roll the wheel upwards zooms in, roll it downwards zooms out.

[1] http://www.libsdl.org
[2] http://sky.prohosting.com/oparviai/soundtouch/
[3] http://www.freetype.org
[4] TrueType is a trademark of Apple Computer inc
[5] http://homepages.paradise.net.nz/henryj/code/
[6] OpenGL is a trademark of Silicongraphics inc

9.3.1 Zoom Algorithm

Scroll wheel events are recognized by SDL as button up events of button number 4 and button number 5. A continues stream of these events is created by a roll of the scroll wheel. For each event the zoom algorithm is divided in two steps, first set the where the users aimed with the cursor, and then set the zoom scale and current view. Zoom in, button number 4, increases the zoom scale by the zoom factor 1.0 + t. Zoom out, button number 5, is done by simply change to negative sign of t.

```
setZoomCursor(event);
setZoom(1.0 + t);
```

The zoom cursor is in screen space coordinates. The event coordinates are compared to the centre of the view port bounds origin.

```
zoomCursor = event.where - (viewpb.where + viewpb.extent * 0.5);
```

The second step of the algorithm is to set the zoom scale and the new current view position. The zoom scale is simply to increment or decrement it by the zoom factor. The new current view is calculated from the old current view and the zoom cursor multiplied by the different between the old inverted zoom scale and the new inverted zoom scale.

```
oldZoomScale = zoomScale;
zoomScale = zoomScale * zoomFactor;
invZoomScaleDiff = 1.0 / oldZoomScale - 1.0 / zoomScale;
currentView = currentView + zoomCursor * invZoomScaleDiff;
```

Result from the previous chapter also showed that users wanted a separate pan action. This was implemented with the right mouse button. A right button press grabs the data surface, the user can move it around while holding the button pressed, a button release locks the view of the data surface.

Additional navigation context is provided by an overview pane located in the bottom right corner of the screen. It displays the entire data surface with an outlined rectangle that indicated the current view (see Figure 9.4.

9.4 Commands

In the design of how command should be evoked, I looked at both at desktop paradigm and at command line interfaces. In the desktop environment com-

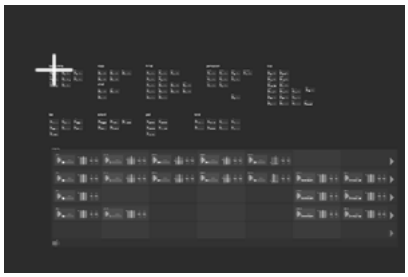


Figure 9.1: The user aims zoom action marked with a cross.

Figure 9.2: The zoom action is halfway through.

Figure 9.3: The user have reached the desired level of detail for inspection.

Figure 9.4: Overview pane with zoom navigation context rectangle positioned in the bottom right corner of the screen.

mands are selected from menus, in a direct manipulation manner, by point and click. There are also key chord shortcuts for commands that expect to be frequently user. Whereas, in a commands line interface environment commands are typed with the keyboard. There are two distinct differences: visual vs. non-visual presence and recognition vs. recollection. Command selection from menus is more easy to use by its visual presents, the users do not have to remember the command only recognise it, this comes from the see-and-point value. The drawback in this design is that menu navigation is slow for users who already know the command they want to invoke. Shortcuts patch this problem for most frequently used commands. Also, menus needs screen real estate, why there are menu bars and hierarchical menus. For command line interfaces users have to remember exactly how to type the commands; online help, ”man”, guide only the right use of command. For efficiency commands are often labels as acronyms, for instance ”list” is ”ls”, ”remove” is ”rm”, and ”change directory” is ”cd”. This impedes learning. Command line interfaces also do not implement the feedback and dialog value, and they force users to ask for feedback. For instance to see the effect of a remove command users have to list all the files of the current working directory.

9.4.1 Commands in the Data Surface Environment

I designed command evocation method for the data surface paradigm to be text based. But since its basically is a graphical user interface it is also a direct manipulations interface, where manipulation is enabled for those visual elements that indicates state of contents. The Music Tool Prototype’s indicators for play state, tempo, volume, left/right pan, transpose, and fine tune are also clickable direct manipulation controllers. Although this compliance to the direct manipulation value provide users good control over the content state elements, graphical representations suffer from precision and are cumbersome to use when users have formulated their intentions in advance. For instance

consider try to use a slider to set the value 77 or -1 in the range of 0 - 100, or worse 0 - 1000. Zoom helps to increase the fidelity, but it is still not sufficient. Whereas, typed commands on the other hand has the advantage to be precise, efficient, and less bound to a graphical representation. I considered the design values for the data surface paradigm command invocation method to be: noun-verb grammar, feedback, pre-visualised feedback, context, context help, pervasive short cuts, non-visual tools, and ambidextrousness.

- Noun - verb grammar, the users points to a selection or a single component first then evoke the command. This goes back to the see-and-point value, but here the grammar is used for typed commands instead of point and click. Still the content affords the action.

- Feedback, for each key press the prototype provided feedback in form of a help list that displays the commands that contain the written substing. In Figure 9.5 (top) the user typed: ”k”, which resulted in a list of two commands. In Figure 9.5 (middle) the user typed: ”ko”, which selected the command ”kopiera” (Swedish for copy), there is also a pre-visualisation feedback for the command. In Figure Figure 9.5 (bottom) the user typed: ”kom” which the system cannot find and thus displays an error message.

- Pre-visualised feedback, enable users to investigate results of their intended action in advance. This is particularly usable for creativity tools [48]

- Contexts, a command affects only the users defined selection set of information elements or the element under the cursor. In other words, while writing, the feedback help list displays the command for the selected context.

- Context Help, a help button, set to be the tab-key see mark (1) in Figure 9.6, displays all the command for the selected context.

- Pervasive Shortcuts, when the user presses a key, the first command item in the list is due to be evoked when the command completion key is pressed, see mark (2) in Figure 9.6. Type a unique enough substring to select the desired command rapidly, for instance with ”kopiera” (copy) the user may type ”o” and then the command completion key to evoke it. Command lists are sorted in descending frequency, but are not adaptive, which help users to habitually select their favourite command shortcut

regardless of context. The command lists are supplied by the selected contents service components. The stability of the command item sort order goes back to the consistency value.

- Non-visual tools, the idea behind typed command was that the users should not have to look at the tool, only the content information. When using the same menu commands over and over again with the desktop paradigm interface means that the users tediously have to look at the menu. The data surface paradigm design does not take away screen real estate from the information content.
- Ambidextrousness, each command seldom needs more than two or three letters to be selected and evoked. The exact substring for command selection can be, to some extent, chosen by the users. A left-handed user may be able to choose letter in the right half the keyboard whereas a right-handed user can choose letter to the left, thus minimise homing of the sword hand from mouse to keyboard and allow the users to work with their both hands.

9.5 Music Tool Description

The prototype has a set of available components. A text component allows users to write text in any empty region of the data surface. A sound component plays and displays streamed sound contents. A sound controller component is used for manipulation of sound length, pitch, volume level and pan. A song arrangement matrix component helps the arrangement of sound controller components into a song. The matrix has rules for playback that simplifies live arrangement of different phrases of a song. For instance users can, aligned to bars, launch a row of sound while stop all the rest of the sounds. Only one cell in each column can be played at once.

9.5.1 Overview

An overview of the visual appearance of the content for which music tool components were created is displayed in Figure 9.7. The small clustered rectangles in the top of Figure 9.7 (1) show all the sound loops in the experiments. The bottom half of the figure shows the song arrangement component (2). Loop components that are copied to the arrangement becomes embedded in a loop controller component (3).

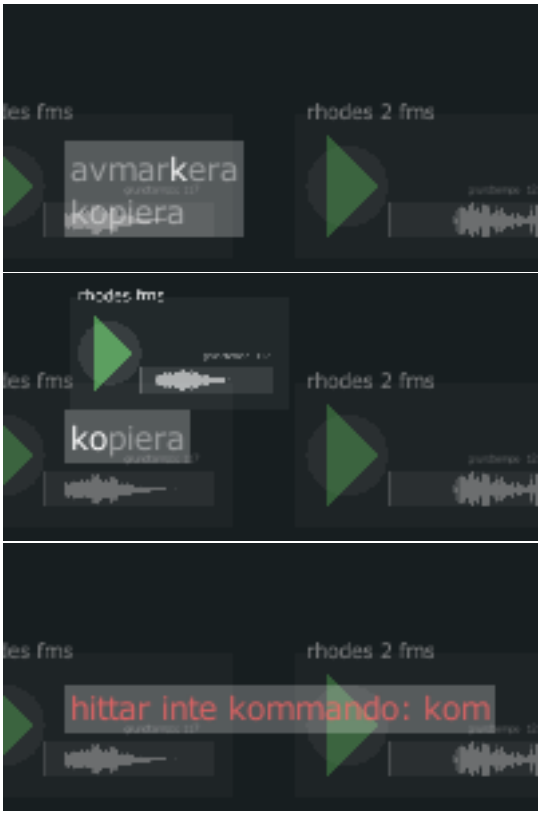


Figure 9.5: First, second and third key press while the cursor hover over a sound component . Feedback of the command that in this context match the typed key is displayed. The first command of the list is due to be invoked, indicated by the stronger contrast. The second image show the pre-visualisation of the copy command. The last image show an error message: ”could not find command: kom”.

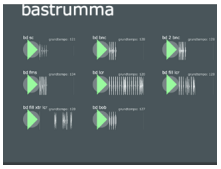


Figure 9.6: Tab-key was redefines as help key. Enter and space bar keys were command completion keys.

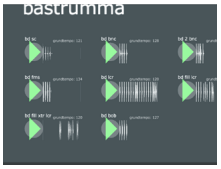
Figure 9.7: Overview (zoomed out) of the data surface, (1) shows the loop clusters, (2) the arrangement, (3) the loop component embedded into a loop controller component in the arrangement.

9.5.2 Sound Loop Component

The loop data is visualised by the loop component. Figure 9.8 shows the loop caption (1). The loops play state is displayed by the intensity of the triangle (2) opaque saturated green: playing, transparent green: not playing. The triangle also behaves as a direct manipulation push button. The circle (3) animates a pie diagram for how much was left to play in a loop sequence; a non-playing loop displays a full circle. The loop component also displays the loops waveform graph, and a progress indicator is visualised as a thin vertical line (4). Finally, there is text information about the base tempo for the loop (5).

9.5.3 Sound Loop Controller Component

When the users copy a loop to the arrangement its embedded into a loop controller, see Figure 9.9. The loop is displayed as usual (1), but there are three new controllers. A controller for volume and pan (2), the vertical axis holds the volume the horizontal holds the left/right pan. There is a controller for transposing the loop -12 to 12 semi tones (3), and a fine tune controller (4).



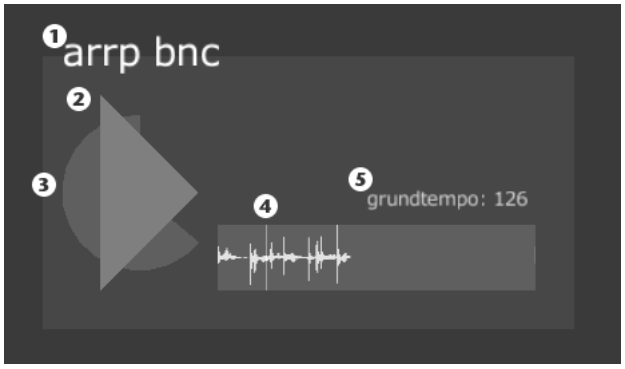


Figure 9.8: The loop component (zoomed in), (1) the caption, (2) play/stop button and indicator triangle, (3) pie diagram animation of loop progress, (4) loop waveform and progress indicator (thin vertical line), and (5) the base tempo of the loop.

Figure 9.9: Loop controller component (1) the loop, (2) volume and pan, (3) transpose, and (4) fine tune.

9.5.4 Arrangement Component

In the arrangement component, see Figure 9.7 (2), loops are organised in rows and columns. Loops in the same column are mutually exclusive. These rules allow users to move forward in the song only by start the next loop. Furthermore each row has a launch button that starts all the loops in the same row at ones, while stop all the rest of the playing loops. Start and stop of loops aligns to bars, so the user does not have to worry about timing mouse clicks or commands. The arrangement also has a tempo controller, it affects all loops in real-time for auto tempo correction.

9.5.5 Available Commands

Here is a list of all the available commands for the environment and the components. The commands with argument end with a colon. A colon key hit leaps to argument input when commands are typed. For example with the ”find:” command, the key sequence f, i, and : leaps to argument input. The commands in the prototype were in Swedish, however here they are listed in English. The commands of the environment are.

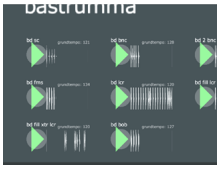
- Undo. Undo deletion of a component of a cursor appointed region.
- Unselect. Clear the component selection set.
- Find: *argument*. The first argument character keystroke adds all components that contain the typed character to the selection set. Each consecutive keystroke after the initial keystroke removes components of selection set that do not match the argument substring.
- Leap. With this command users can leap to position and scale of each component of the selection set.
- Write. Initialises a text component for annotation, equivalent to hit enter on an empty region.

The basic commands common to all components are.

- Undo. Undo a command by pop and execute an undo action from the components undo stack.
- Move. Move the selection set components to the position and scale of the cursor. The spatial relationship among the components is preserved.
- Copy. Move copies of the selection set components to the position and scale of the cursor. The spatial relationship among the components is preserved.
- Delete. Delete the selection set components from the data surface. Add the components to the undo stack of the environment. The actual deletion is postponed to when the region is filled with new content.

The sound component commands are.

- Play. Start to play the sound components of the selection set. Only start the components that are not playing.



- Stop. Stop to play the playing sound components of the selection set.

The sound controller component commands are.

- Volume: *argument*. Set the volume of the sound to the argument value in the range of 0 to 256.
- Pan: *argument*. Set the left right pan of the sound the sound to the argument value in the range of -64 to 64, left respectively right.
- Transpose: *argument*. Set the key of the sound from -12 to +12 semitones.
- Finetune: *argument*. Set the fine-tune percent of the sound from -99% to +99% of one semitone.

All sliders have the increment and decrement shortcut commands of ”+” respectively ”-”. However, the sound controller component has the volume and pan sliders interleaved, so the increment and decrement shortcut commands for these components are ”v+”, ”v-”, ”p+”, and ”p-”.

The arrangement component commands are.

- New Track. Add a new column into the arrangement matrix. itemTempo: *argument*. Set the tempo to any value in the range 30 beats per minute (bpm) to 180 bpm. When this value is changed all the playing sounds are time stretched or time compressed in real-time to align with the beat.

The row component of the arrangement component commands is.

- Play. Launch the entire row of sound and while all other sound are stopped.
- Stop. Stop all the playing sounds of the row, but does not stop sound of any other row.

9.6 Collaboration

Real-time synchronisation of data surface contents supports collaboration, mutual awareness of action, and mutual modifiability. Every action and command is echoed to the other collaborating participants machine. The visual and acoustic contents appearance is exactly the same on all machines, thus all participating users are aware of all actions. To further help and support the

collaboration, users can annotate the contents by type messages onto the data surface. The writer of the message is coded in colour. Mutual awareness of action and mutual modifiability is inconsistent with the original interpretation of the user control value, since other agents (here other users) than the user itself can change the content, thus all actions are not initiated by one user.

9.7 Evalutation

The data surface paradigm design ideas was tested on ten subject users: six solo user test and two collaboration pair test. All users had fairly good skills in music. Among the users were four studio producers, with expertise in computer aided music production. The users were observed for their emotional expressions towards navigation, command invocation, and collaboration.

Logging was implemented to compare the amount of command invocation and navigation. Previous investigations and observations reported in the chapter ”Music Application Area” showed that users spent a lot of loops navigation time. The hypothesis was that the less the users spent time on navigation compared to creation of music the better.

A tape recorder in the solo tests and video camcorders for the collaboration tests, combined with written notes were used for protocol. The context for the producers was set to their studio environment, see Figure 9.10. The subject users were handed a few simple tasks to get them acquainted to the interface, its navigation, how commands worked, the loops, and the arrangement. Eventually they were asked to create a song. The subject users during the solo-test where asked to think aloud, whereas collaboration test users were allowed to engage in verbal communication with eachother. After the test debriefing interviews was used to collect users comments and to clarify the observations.

9.7.1 Metrics

- Usability of the interfaces in general: navigation of sound, usability of the arrangement component.
- Usability of the command invocation.
- Degree of satisfaction: do the subject users enjoy data surface interaction or do they show signs of frustration?
- Methods: would users find a scheme to complete their tasks?

- Role: would user rely on roles for who is in charge of what?
- The tools effect on collaboration: would users like to talk to each other or not?

The evaluation was divided in two phases. In the first phase the users completed a set of task by themselves. In the second phase they created a song together. The users experience of the prototype is very different from the ordinary desktop interface, so for the users to be able create music together they had to learn the interface first. A written introduction that contained a description of the prototype the tasks for the users to complete taught them how it worked. The users how collaborated sat in the same room 3 meters apart, they could see each other by turn their heads 15 degrees to their left. They were unable to see each others screen. They had each one camcorder behind their backs that captured their actions. During the collaboration phase one set of loudspeakers played the synchronised sound from one computer.

9.7.2 The Tasks

Here is list of the tasks handed to the users for evaluation.

Zoom in on ”ackord” (Swedish for ”chord”). Instructions of how to pan and zoom guided the users. The task was coupled with images of the present state and the goal state to help users to recognise success, and instructions how to zoom in and out and how to pan. The idea of this task was simply to acquaint the users to the navigation and the content of the data surface.

Play the sound ”rhodes sc”. The users were instructed to press the play button of the sound.

Zoom out.

Find the sound ”kw-tone bnc”. This task was coupled with a goal image. The idea of this task was to repeat the first task with fewer instructions.

Copy the sound ”kw-tone bnc” to an empty region. A description of how to invoke command guided users step by step through this task. The users were instructed to first point at the target content, then press the help key (optional, see mark (1) in Figure 9.6) , type a few characters of the command, navigate

to the position and scale they want the copy to be located, and finally hit the command completion key (see mark (2) in Figure 9.6). To press the help key was optional to explain that if the already knew the command they did not need to press it. Here I wanted to users to learn about the command invocation mechanism, the copy scheme that is different from the copy-paste scheme of the desktop interface, and the pre-visualisation of commands.

Move the copy of ”kw-tone bnc” onto the arrangement component and copy a sound of your own choice onto the arrangement. This task also contained a goal image of the arrangement component. The rules of the arrangement were described: that only one sound in each row of one column can be played at once; that an entire row of sound can be launched at once while all other sound are muted just by invoke the play command of the row or hit the rows play button.

Write a comment of your experience on an empty region of the surface. The users were instructed to point at an empty region, press the enter key, and type the comment. This task taught the users how to write annotations and comments that during collaboration work as chat messages.

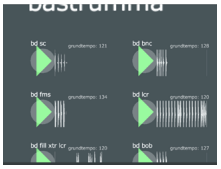
During the second phase the task was simply to create a song together.

9.7.3 Debriefing Interviews

Each evaluation was followed up by debriefing interviews. The solo users were interviewed one by one whereas the collaboration test users were interviewed in focus group pairs. They were asked what the thought of five different aspects, what they liked and disliked. The aspects were: navigation, layout and design, command invocation, note writing, and collaboration.

9.8 Evaluation results

Observations showed that no one of the subject users had any problems with the navigation. None of the users had any help from over view thumbnail in the bottom right corner; in fact it was totally overlooked. The usability of the navigation tools was satisfactory to all the users, but can be further simplified by remove the over view thumbnail. Analysis of log data also showed that navigation early in the test was interleaved with play actions. This was the part when



the users conducted loop navigation. When the users had settled with a set of loops there was very little navigation. In previous investigations of a desktop paradigm music tool, covered in the chapter ”Music Application Area”, where the users opened a loop, discharged it, opened another, and discharged it, repeatedly until they were bored. The users for these tests really enjoyed themselves, one user shouted out: ”this is fun!” On the other hand, command invocation was not usable for all users. One of them had great difficulties and struggled with how it worked.

All the users successfully completed the task of the first phase. However one user failed to learn the command invocation scheme. The learning-style and pace differed among the users. Some of them follow the instructions carefully, while others tried to complete them as fast as possible. During the second phase the users started to look for sounds, the log files show extensive navigation. After have found a set of sound they felt comfortable with they started to build their song. The users negotiate what sounds to select and collect. They continuously talked to each other. One user became the ”doer” who added sounds to the arrangement. All the users, except for one solo user, managed to complete a simple song.

The users reported that they really enjoyed the navigation. One of them initially had to work a little extra to learn the out zoom cursor centred design. They liked the layout and design of the sound components; the expert users were really fond of the combined volume/pan controller. All the users reported difficulties with the arrangement component. The users disagreed the most on the usability of command invocation. Two users, one expert and the other a novice, hailed the command invocation style. One expert user disapproved it. He suggested that there should be a menu bar; he did not at all like to remember and write commands. He wanted to have right-click for context menus. On the other hand, the novice user, to young to have used DOS, thought it was really cool and revolutionary to write commands. They liked the shared surface that enabled both users to be active simultaneously. Note writing was considered easy but lacked capabilities. The users enjoyed the collaboration style; they wanted to see it soon in future software music tools.

9.8.1 Subject Satisfaction

A few stray comments from the other subject user were: ”you are free - good for creativity”, ”incredibly good to use both hands”, ”I loved to have everything on top”, ”good to not to have to load loops”, ”super nice to escape menus”, ”nice with the wheel for zoom”, ”easy and fun”, ”this has enormous potential”,

Figure 9.10: Typical studio context for the expert user evaluations.

”fun to see what the other one was doing”.

The collaboration users communicated with each other both by talking and by chat annotations. The annotations worked as referent in their verbal communication. They were very pleased with this form of collaboration opposed to sit together by one computer.

One user summarised the interaction: ”there are no menus only help list and those were only exceptionally needed, you manipulate information but the tools are invisible.” In debriefing several of the users thought that the approach was very different from what they were used to but also very easy to learn. They felt like they mastered the prototype only after a few minutes.

Unfortunately, the arrangement component revealed to have a number of bugs that made the users less satisfied with it. One user totally neglected the arrangement and copied loop components freely onto an empty space of the surface. He used the commands to launch and stop the loops to create a song, thanks to that the loops were playing aligned to bars and tempo outside the arrangement. The rest of the users, in other word eight out of ten, created songs with the agreement component as expected, despite its hitches and glitches.

9.9 Conclusions

The data surface paradigm works for a loop-based improvisation and live music creativity tool. Users were very pleased with the navigation. The approach support creativity and open-ended task. The command model was not fully embraced. The thumbnail overview pane did not provide contextual cues. Visual and audio content synchronisation simplifies collaboration. With visual synchronisation a visual external referent to sound modalities enable users to engage in negotiations for their creative work. The users enjoyed the music tool implemented based on the data surface paradigm. Other application areas and larger data sets have to be investigated before the general case can be proved. However, I have found strong indications that the content centric data surface approach can serve as interaction paradigm for computers.

9.10 Summary

This chapter summarised the investigations of the prototypes in the previous chapters as well as the observations from the Application Area chapter. By conceptual, technical, and empirical investigations I have found support for the

approach represented in this thesis. The next chapter is a design guideline for those who plan to implement a data surface.

Chapter 10

Design Principles of the Data Surface Paradigm

The chapters so fourth in this thesis have explained and reported about the background, theoretical as well as personal, the conceptual, the empirical, and the technological investigations of the development and design of the data surface interaction paradigm. Even though you by now have a good feeling of what constitutes the data surface interaction paradigm, you do not have principals for constructing and deploying this theory in practise. This chapter presents a set of practical principles for software interface design based on the data surface interaction paradigm.

10.1 Principles

This section repeats the principles listed in the context for the reader section of the introduction chapter. The data surface paradigm was created taking the following seven principles into consideration. One: It is content centric instead of tool centric. There are no application programs. Two: Content remains persistent. The system takes care of the information for the user.There are no files. Three: Content remains visually present in its context. The surrounding set of elements for a specific content sets it context. Content can be annotated. Spatial semantics helps navigation. Four: Incremental ubiquitous search simplifies information retrieval. Feedback for search condition satisfaction remind users of content locus. Five: Non-visual tools. Users who have formulated their

intentions can directly invoke the correct command instead of navigating and looking at menus. Six: Shared synchronised surface areas enable collaboration. Visual feedback make users aware of each others actions and provides an external referent for negotiations. Seven: Enable multi modal user interfaces. By removing the WIMP-components (There are no overlapping **W**indows, no **I**con bars, and no **M**enus) that make the desktop interface so well designed for the mouse **P**ointing device, the paradigm allows other interaction techniques such as eye gazing, gestures, handwriting and speech.

10.2 Persistent Storage

Keep all the content information and components in persistent storage. If this is solved by deploying a database server or by careful bookkeeping of files is of less importance. The main issue is that the users should never be responsible for transferring information from primary memory to secondary memory of the computer. Think of is as content storage modelessness. Use data representation that simplifies search and multiple undo.

10.3 Use Spatial Semantics

Let all the content reside on an infinitely large two-dimensional data surface. Let the users to be able to map the semantics of the content to its position. Navigation and information retrieval is easier if similar information elements are clustered closer together, whereas dissimulated clusters need to be separated with larger distances of space. Enable the users to add contextual cues by annotate the data surface, either by adding text, pictures, or by adjusting background colour. Perceived stability is achieved by letting the content remain in its place.

10.4 Make Things Flat

Try to visualise as much information as possible in one view. Do not hide information. High fidelity details that enable quantifiable judgements can be visualised with a lower zoom scale, whereas qualitative visualisations should have high zoom scale. Try to avoid extensive use of 3D, remember Charles-Joseph Minard's illustration of Napoleon's Russian campaign from 1869 10.1.

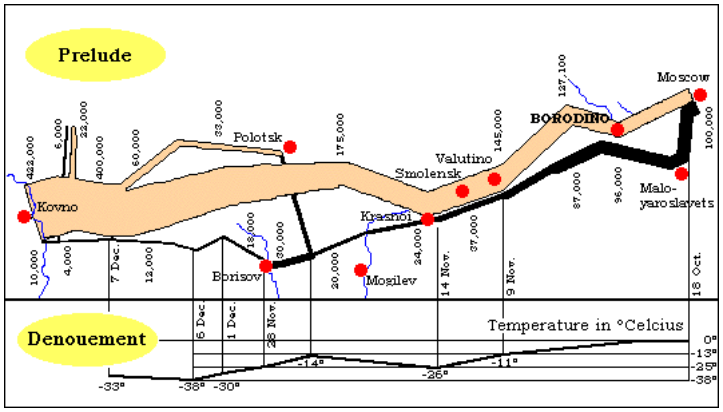


Figure 10.1: On two dimensions Minard mapped, places, dates, temperature, casualties,

10.5 Content Centric Components

Supply services as components. Each component may work on only one information modality, whereas an information element may have multiple service components. All components should be accessible at all time, in other words do not coarse the user to explicitly start or stop a service component. This is regarded as modelessness of service.

10.6 Zoom

Let the user be able to zoom in and out on the content. Enable the users to aim at content by trajectory zoom. Map the action to the scroll wheel of the mouse. In case there is no scroll wheel use mouse drag up and down to zoom in and out respectively while holding the middle button pressed. In case of one or two-buttoned mouse use a meta-key to enable the middle button. Avoid ”poping” of information by visualising fine grain details in out zoomed state. In other words, do not use semantic zoom. Make sure the zoom scale transition is smooth, fluent, and stateless. Fine grain details and fluent zoom provides context that otherwise would be lost with risk of confusing the user.

10.7 Panning

Panning is done with the right mouse button. The user should be able to drag the data surface while keeping the right mouse button pressed.

10.8 Selection

Any left mouse button click on a passive information element should add the element to the selection set. Multiple elements can be selected by keeping the left mouse button pressed and drag it creating a selection rectangle. Any information element with a region within the selection rectangle should be added to the selection set. Selection by click and selection rectangle toggles the information element selected state. The selection set can be deleted by invoking the unselect-command. Mark selection by negative feedback, hence, let all the content except the selected information elements become more transparent and displayed with less contrast. Selection should also be able to be done with incremental find.

10.9 Incremental Find

Use the find command to find and select information elements. The users should receive selection feedback for every keystroke. Let commands affect only the information elements in the visual scope. You may use a jump command to jump to the zoom and panning of a found information element. Subsequent jumps from element to element. Incremental find bring language selection style to the interaction.

10.10 Commands

The users first select the information elements of interest then they type the command. If the selection set is empty, select the information element the cursor points at for the moment. The typing is actually an incremental search among the command set for the selected information elements components. Use substring so that users may learn and form their own acronym schemes, for instance ”op” could be the ubiquitous acronym for the copy command. Provide feedback after every keystroke of what commands were found or if none was found. Use a help-key for displaying all available command in the selection

set. If the command needs an argument, use the colon-key to indicate a jump to argument insertion. Provide feedback for the current value of the argument. Use a command completion key, enter or space, to evoke the command. Sort the components command in descending order of frequency of use, thus the order of the commands are stable and less keystrokes are needed to search for the most frequently used commands. But, the order must be stable, adaptive order fail the users ability to automate tasks. Display the result of the command before the commands is invoked so that users can evaluate the result without using undo.

10.11 Undo

Error recovery should always be done with multiple steps of undo. The undo action should be coupled with its content. Only the components in the selection set are affected when the users invokes the undo command. Do not use modal alert boxes to prevent users from doing mistakes. Task automation renders these boxes useless anyway.

10.12 Copy

The copy information elements command should be of the from this-copy-there. The selection set information elements should be copied to the position of the cursor in the current view when the user hits the command completion key. Think of the cursors as the argument of copy command. Allow zoom and pan to be interleaved thus altering the information elements location and scale. Do not use a clipboard that hides information.

10.13 Direct Manipulation

At atomic level direct manipulation is a powerful tool. Make sure that all your direct manipulation components clearly afford their actions, also make sure that these elements display their current value both quantitatively and qualitatively. For instance a slider may look like a staple diagram but should also have numeric representation. Direct manipulation should always be combined with a command to set values and to invoke actions, since the user may have a clear intension or be working in out zoomed overview state.

10.14 Collaboration

Let user be able to share and synchronise partial space of the surface. Synchronisation should be enabled in real-time so that two or more participants can simultaneously create content and see what everybody else are doing.

10.15 Multi Modality

There are actually two mechanisms that are need for this interaction paradigm. One is a location indicator, and the other one is a symbol selection mechanism. Handwritten commands and gestures as well as sign and gesture languages enables both. Symbols by speech recognition can be combined with for instance eye gazing. Thanks to zoom, low fidelity gaze recognisers are reasonably satisfying.

10.16 Summary

This chapter described a small set of principals for the data surface interaction paradigm. An implementation can be done by following these principals. It is time to (in Table 10.16) fill the fourth column of the table found in chapter 3. The column contains distilled description of how I reflect upon the design value found in the first column for the data surface interaction paradigm. As you can see the big difference between Macintosh Human Interface Guidelines and Data Surface Interaction Paradigm is the view of metaphors. Where the guidelines encourage extensive use of metaphors, the data surface discourage metaphors and states that knowledge in human behaviour should determine how services should be designed. For the modelessness value and the consistency value the data surface is a much better citizen than the desktop interface.

Table 10.1: Comparing paradigms

Value	The Guidelines	Anti-Mac	Towards new Interaction
Metaphors	Convey concepts and features. Take advantage of peoples knowledge in the world.	Target domain is different from source. Preserve limitations from source domain.	A user illusion construct without metaphorical explanation consistent with human cognition.
Direct Manipulation	Physical actions on visible objects on the screen. The effect is immediately visible.	Direct manipulation works on atomic level and can be very inefficient	Fine grained zoomed control of content. But, commands are used in out zoomed scope.
See-and-Point	Actions preformed by choosing from alternatives presented on the screen.	Throws away power of language. Language interaction style should be reinforced.	Language selection of objects by incremental search. Noun-then-verb grammar for commands.
Consistency	Interface behaviour consistency among applications make users' knowledge transferable.	The real world is not consistent. People have no trouble switching between different tools.	Same set of service behaviour for the each data type in the entire environment .
User Control	The user, not the computer, initiates all actions.	The users have to monitor or perform boring tasks.	Collaborating users initiates actions and agents. Inspect agents in detail by zoom.
Feedback and Dialog	Always inform, as immediate as possible, the application status.	Detailed feedback is only needed for detailed control.	Detailed content status feedback, feedback forecast and immediate feedback are fundamental.
Forgiveness	Actions should be reversible. Dangerous irreversible actions must be alerted.	Anticipate users intension by book-keeping actions omits repeated alert boxes.	Content keep track of its changes. All actions can be undone.
Perceived Stability	Makes the computer predictable. Things remain in place.	Stability is undesired in web, games and learning applications.	Content location is stable, it can be changed by others.
Aesthetic Integrity	Minimum of interaction elements, with elegant and consistent look.	Coherent appearance makes navigation confusing.	Visual appearance of content is diverse. Tools are non-visual.
Modelessness	All possible operation are available at all times.	Modes are natural and people can not cope with everything at once.	Contents persistence, recoverability, and content component model makes mode-. lessness pervasive.

Chapter 11

Conclusions

This chapter contains a final discussion and conclusions regarding the data surface paradigm. By leverage on cognitive science, observations, interviews, and usability evaluations I have been able to find strong indications that the approach presented in this thesis supports the services expected by users' their creativity in action, and their awareness in collaboration, in a manner that they find fresh, fun, and pleasing.

11.1 Discussion

The work presented in this thesis is one possible path toward a new interaction that takes off in the polemic field between the Macintosh Human Interface Guidelines [7] and the Anti-Mac Interface [11]. The validity and the reliability of the results depend upon a triangulation of conceptual studies, empirical studies, and technological studies.

11.1.1 Conceptual Studies

The conceptual studies contain the related work, the reassessment of design values for the desktop metaphor, and the field of cognitive psychology. Zoom interfaces have already been investigated. The work of Perlin et. al. and Bederson et. al. [23, 2] is that most closely related to the approach presented in this thesis. They proposed semantic zoom as a possible navigation and interaction technique in large information spaces. Spatial semantics and Tversky's cognitive collage [50] support the two-dimensional static information space

(in which information elements remain where they are placed), and show that this space is in accordance with peoples mental models of spatial information. For example, annotations of out-zoomed content function as cognitive collage landmarks that trigger users mental navigation routes.

11.1.2 Empirical Studies

In-depth interviews, focus groups, qualitative evaluations in controlled environments,and qualitative evaluations through field studies constitute the empirical studies. In-depth interviews and focus group interviews were based on open form questionnaires with relatively few interviewees (6). Interviews of users revealed that they find the desktop metaphor music software tools cumbersome and constrain their creativity. Instead of focusing on their task, they must deal with explicit file management. Interviews were also used to debrief those performing usability evaluations. The qualitative evaluations performed were of a current desktop music creativity tool and of my prototypes. The primary metrics for these studies were the users attitudes toward these tools. Navigation among files in the desktop system was found boring and idiosyncratic, even though a naming convention was used to help the navigation. On the other hand, navigation of content with a data surface tool was much more rewarding and appreciated by the users.

11.1.3 Technological Studies

The technological studies were the implementation of the prototypes - first with a multimedia tool, and subsequently using a general purpose programming language and open-source game development APIs (Application Programmer Interface). The implementation demonstrated that the data surface could be achieved. The third prototype utilised OpenGL in a straightforward way, no algorithms for cull and cache were used other than those embedded in OpenGL, nevertheless the prototype rendered fluently for visualisation, 100 MB of information in real-time. The game development community have devoted much effort to visualising large worlds in real-time. Algorithms for level of detail (LoD) efficiently reduce the amount of data that must be filtered from database to image buffer without a popping effect or other artefacts. Semantic zoom can be avoided by relying on LoD algorithms.

11.1.4 World of Metaphors

The world of computing has evolved rapidly since the conception of the desktop metaphor interface and the Xerox Star. Speed of operation and the capacity of memories and hard drives have increased enormously, multimedia, the web, and the Internet, have been developed, everything has become faster and bigger. Throughout all this evolution the interface have remained almost unchanged, and the foundation Desktop Metaphor still reigns as the interaction paradigm.

There have been many studies of how the desktop metaphor affects users. Barreau and Nardi's "Finding and reminding" [9] is an often cited report on users file catalogue navigation strategies. Their conclusion was that users did not need search tools, instead they relied on spatial semantics as a cue to their content. Ravasio et. al. [39] extended Barreau and Nardi's conclusion and said that the poor usability of system search tools requires users to search manually. Fällman points out the anomaly of the desktop paradigm with its migration into small devices, mobile phones and personal digital assistants [15]. The PRESTO system by Dourish et. al. [36] attacked the navigation scheme of the desktop interface by making use of metadata and spatial cues, largely based on Barreau and Nardis results. The ideas incorporated in PRESTO have been appreciated in industry and forthcoming versions of both Mac OS X (tiger) and Windows (Longhorn) will incorporate these. Rekimoto invented a metaphor called Time Machine Computing (TMC) [40]. In TMC, navigation is aided by temporal attributes of files and free text search of annotations. A technique called time-casting allow users to browse episodes and procedures. Perlin and Fox suggested a new interaction paradigm based on semantic zoom and portals in their work with the pad [23]. Holmquist suggests that his Flip-Zoom also can be used as a paradigm for file browsing instead of the desktop metaphor [19]. Kay at. al. propose immersive 3D interfaces as a replacement for current interaction style in their work with the Croquet project [43].

In my proposal I have tried to simplify the interface as much as possible. I have tried to remove all the bells and whistles of current interaction techniques. My philosophy is that there should be no decorations; every visual element should either bring a message or perform a function. This is a rather modernistic approach, whereas current systems seem to make up for usability by adaptation to Art Nouveau.

11.1.5 Hierarchy of an Image

Tversky also showed that people map hierarchical relations to different scaled objects of a flat image [49]. People view small-scaled objects as sorted under large-scale objects in the hierarchy. In fact, zoom interfaces are very good for viewing hierarchically organised information [38]. In my studies, I found that users appreciated the hierarchical organisation of sounds in the Concept Prototype.

The semantic zoom interface [23, 2, 38] is similar to the spatial layout view of the desktop. The condensed labels of information content in semantic zoom systems are analogous to folder labels. The similarity is clearly visible in Mac OS, when users double-click on a folder it opens up with a zoom-animation. Ravasio et. al. [39] found that many users of desktop interfaces continue to rely on spatial semantics in sub-folders of the desktop. Their results support my approach. Semantic zoom popping of information, however, destroys context [38], which is why I have suggested non-semantic or graphical zoom. Still, the possibility to annotate and label content at large-scale preserves the hierarchical character of my approach.

11.1.6 Other Views of Information

There are more views of information than hierarchical, for instance hypermedia and view by temporal attributes. Rekimotos Time Machine Computing (TCM) metaphor [40] is a good example of the latter. In TCM, users navigate their files by creation date and modify date meta-tags. Temporal navigation could be included in my model by adding a searchable when clause to undo items. I suggest that users could search for items that have been changed within an interval or around a specified date. Another approach would be to rollback the information space to a certain date.

What I have not dealt with in this thesis is hypermedia, which makes it possible to quickly follow associations and to look up related material. Hypermedia has become very popular with the widespread acceptance of the World Wide Web. The most common format for hypermedia on the Web is hypertext, credited to Ted Nelson with the Xanadu project. A typical link from one page to another is a blue coloured underlined word, but a whole variety of different designs exist on the Web. A simple approach to achieve this for the data surface paradigm would be to introduce links from different parts of the surface to other parts.But links can disorient and confuse users, which, as a consequence, has opened up the field of web-usability. With inspiration from

lexivisual communication found in newspapers, comics, and movies, Kindborg present a possible solution in ”Visual techniques for orientation in hypermedia structures” [24]. The main theme is: information should be presented in its context. For instance, instead of being a label only, a link could be a miniaturised and zoomable view of the related content combined with a map showing the location of the ”original” information and some mechanism for getting there. However, hyper-structures may go deep so context will probably at some point be broken.

11.1.7 Multi-modality

The mouse is an effective device for manipulation of windows. But there are other input modalities, designed to be ”natural” for humans but not necessarily for window manipulation. These interfaces are well designed to bring usability of computers to disabled users [37]. Known technologies for this are, to name a few, recognition of speech, handwriting, gestures, and eye gazing. With my approach users need to do three things, find content, make selections, and command invocation. Finding content is performed linguistically thus without a pointing device or by zooming and panning without pixel accuracy. Selection is performed linguistically and with a pointing device - again without pixel accuracy. Command invocation is based on linguistic modalities. Bernsens modality taxonomy [3] and Technologies such as Stanford Research Institutes (SRI) Open Agent Architecture [6] can be used to incorporate multi-modality into my work.

11.1.8 Complexity of the Task

The application area investigated has throughout this work been improvisational computer music composition. The scenario in chapter 2 sets the context for the task domain, and it is not obvious how to generalise it.

Are the prototype and the task domain complex enough? At first glance, the task of collecting, selecting, and arranging a set of pre-recorded loops into a song may look like a block-world task. But the process of selecting musical elements is more complex than selecting a set of blocks. Users must consider tempo, key, timbre, minor or major, and genre. The prototype also allows users to alter volume, pan, tempo and key. The arrangement permits the use of up to 32 simultaneous tracks. Add to this the social process of collaborative work. Musical composition can be compared to writing a text. Both tasks are open ended, creative, and complex.

The complexity of a tool can be measured by counting the number of menu items and buttons it contains. The number of commands and direct manipulations available for the music tool prototype is 31, for Ableton Live [1], excluding parameters of filter components, preference panel and explicit file management),77, and for the collaborative text editor SubEthaEdit [2] (excluding preference panel and explicit file management), 69. These numbers indicate that the complexity of the prototype is more than one third of that of the commercial Ableton Live software package. The complexity of the Ableton Live music creativity package is roughly the same as the complexity of SubEthaEdit. By following this argumentation, the task domain of the prototype is transferable to any open-ended creative task, and the data surface interface paradigm is valid for these task domains as well. The results of my interviews with music experts match the results of Schön [41], which support the transition from different task domains.

Open-ended task domains are free and users are less restrained. The more complex the environment, the more liberated the users. Open-endedness is consistent with the user control value. However, close-end tasks can be more efficient. For instance it might not be a good idea for organisations to permit its personnel write emails in unstructured free text. A better choice for organisations that give priority to efficiency would be email systems based on Winograd's Speech Act Theory [52], in which each message is a transition in a conversation graph. The same could be said about safety-critical tasks, for which users must follow a checklist strictly to maintain safety, in these cases users control is less important. In this thesis, I have not studied close-ended task domains.

11.2 Conclusions

The content-centric data surface interaction paradigm presented in this thesis is one possible design of the user interface for future personal computing in creative and open-ended task domains. Navigation by fluent zoom of the data surface was satisfactory to the subject users. Content visualised onto the data surface with services supplied by pervasive components liberate users from explicit file management and application management.

The questions posed under the heading Research Context can now be answered.

[1] Version 1.5

[2] Trademark of The Coding Monkeys, http://www.CodingMonkeys.de/subethaedit/

Can reassessment instead of rejection of the fundamental design values of the desktop metaphor interface paradigm render an interface paradigm that better supports human cognition, collaborative work, users creativity in action, and multi modal interfaces? Yes. Although multi-modal interfaces, particularly multi-modal input were not studied, these interfaces were taken into consideration throughout the design process. User collaboration and creativity in action was supported, demonstrated in evaluations, and included in the design process. The fluent context-preserving graphical zoom, the two-dimensional spatial semantics, as well as mode-error prevention provide human cognition with more support.

Can this new interface paradigm be more consistent with reassessed design values of the desktop metaphor than the current desktop metaphor interface design? Yes, many of the issues relating to the Macintosh Human Interface Guidelines deal with inconsistencies with respect to its fundamental design values. In my approach the reassessed guidelines design values, without inconsistencies, formed the intellectual spine of my design.

What would happen if fundamental elements of the desktop interface, windows, icons, menus, files and application programs, were removed? new interaction paradigm would be introduced Elements of an interface are tools for communication with contents. If one removes a set of tools these must be replaced. We must abandon the old to build something new in its place. My approach is one way to do this.

Would users accept and will they be satisfied with an interface paradigm that has no windows, no menus, no icons, no files, and no applications? Yes, the users in my studies accepted without exception the layout and structure of sounds, they appreciated not having to load files, they enjoyed the zoom navigation, and they enjoyed collaboration. The command invocation mechanism were hailed by a few and rejected by one out of 10 subject users.

The subject-users response to the final prototype without windows, icons, and menus were: ”It feels free, it’s good for creativity, and it’s easy and fun”.

Bibliography

[1] Andrew U. Frank Andreas Dieberger. A City Metaphor to Support Navigation in Complex Information Spaces. In *Journal of Visual Languages and Computing*, volume 9, pages 597–622, 1998. LNCS.

[2] James D. Hollan Benjamin B. Bederson. Advances in the Pad++ Zoomable Graphics Widget. In *Third Annual Tcl/Tk Workshop sponsored Toronto, Ontario, Canada*, July 1995.

[3] Niels Ole Bernsen. Multimodality in Language and Speech Systems - From Therory to Design Support Tool. In B Granström, editor, *Multimodality in Language and Speech Systems*. Dordrecht: Kluwer Academic Publishers, 2001.

[4] Philip L. Burk. Jammin' on the Web - a new Client/Server Architecture for Multi-User Musical Performance. In *Presented at ICMC 2000*, 2000.

[5] Philip R. Cohen. The Pragmatics of Referring and the Modality of Communication. In *Computational Linguistics 10:2*, pages 97–147, 1984.

[6] Philip R. Cohen, Adam Cheyer, Michelle Wang, and Soon Cheol Baeg. An Open Agent Architecture. In M.N. Huhns and M.P Singh, editors, *Readings in Agents*, pages 197–204. Morgan Kaufmann Publishers, San Francisco, 1998.

[7] Apple Computer. *Macintosh Human Interface Guidelines*. Addison-Wesley, 1992.

[8] David Curbow and Elizabeth Dykstra-Erickson. The OpenDoc User Experience. In *Develop, the Apple Technical Journal Issue 22*, pages 83–93, 1995.

[9] Bonnie A. Nardi Deborah Barreau. Finding and reminding:file organization from the desktop. In *ACM SIGCHI Bulletin archive*, volume 27, pages 39–43, July 1995. Issue 3, ISSN:0736-6906.

[10] Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale. *CSCW and Social Issues*. Prentice Hall, 2 edition, 1998.

[11] Jakob Nielsen Don Gentner. The Anti-Mac Interface. In *Communications of the ACM*, August 1996. ACM Press.

[12] Roger M. Downs and David Stea. Cognitive Maps and Spatial Behavior. In *Image and Environments*, 1973. ISBN 0-202-10058-8 Aldine Publishing Company.

[13] Olle Eksell. Framååt ... Bakååt. In *Det grafiska uttrycket*. Stochholms Typografiska Gille, December 2000.

[14] Douglas C. Engelbart. Special Considerations of the Individual as a User, Generator, and Retriever of Information. In *American Documentation*, pages 121–125, April 1961.

[15] Daniel Fällmann. Desktop computing as paradigm. In *In Romance with the Materials of Mobile Interaction - a Phenomenological Approach to the Design of Mobile Information Technology*, pages 165–199. Department of Informatics Umeøa University, 2003.

[16] Peter Gärdenfors. *Conceptual Spaces*. MIT Press, 2000.

[17] William Gibson. *Neuromancer*. Ace Books, 1984.

[18] Usability Glossary. Boolean Search. http:www.usabilityfirstċomglossaryterm_920.txl, 2002.

[19] Lars Erik Holmquist. Focus+Context Visualization with Flip Zooming and the Zoom Browser. In *ACM Computer-Human Interaction (CHI) '97*, 1997. ACM Press.

[20] Lars Erik Holmquist, Staffan Bjoerk, Johan Redstroem, Ivan Bretan, Rolf Danielsson, Jussi Karlgren, and Kristofer Franzn. WEST: A Web Browser for Small Terminals . In *Proceedings of the 12th annual ACM symposium on User interface software and technology*, November 1999.

[21] Alty J.L. and Knott R.P. Metaphor and human computer interaction: A model based approach. In C.L. Nehaniv, editor, *Computing for Metaphors, Analogy and Agents*, pages 307 – 321. Springer-Verlag, 1999.

[22] Alan Kay. User Interface: A Personal View. In Brenda Laurel, editor, *The Art of Human Computer Interface Design*. Addison-Wesley, 1990.

[23] David Fox Ken Perlin. Pad an Alternative to the Computer Interface. In *Proceedings of SigGraph 93, ACM Press 93*, 1993.

[24] Mikael Kindborg. Visual techniques for orientation in hypermedia structures. Lic. thesis Stockholm University Department of Computer & System Sciences (DSV) ISSN 1101-8526 1991.

[25] Teuvo Kohonen. *Self Organizing Maps*, volume 30. Springer, 2001.

[26] Jonas Löwgren. ”Från MDI till interaktionsdesign”, 2001. Event: STIMDI’01: Svenska tvärvetenskapliga intresseföreningen för människa dator-interaktion.

[27] Jonas Löwgren. Sens-A-Patch: Interactive Visualization of Label Spaces. In *Proc. Fifth Int. Conf. Information Visualization (IV2001*, pages 7–12, 2001. CA: IEEE Computer Society.

[28] K. Lynch. *The Image of the City*. MIT Press, 1960.

[29] John Maloney. An Introduction to Morphic: The Squeak User Interface Framework. In Mark Guzdial and Kim Rose, editors, *Squeak: Open Personal Computing and Multimedia*, 2002.

[30] Elizabeth D. Mynatt, Takeo Igarashi, W. Keith Edwards, and Anthony LaMarca. Flatland: New Dimensions in Office Whiteboardsm. In *Proceedings of CHI99*, pages 346–353, 1999.

[31] Dahlbäck N., K Höök, , and M Sjölinder. Spatial Cognition in the Mind and in the World - the case of hypermedia navigation. In *The Eighteenth Annual Meeting of the Cognitive Science Society*, 1996.

[32] K. Nakakoji, Y. Yamamoto, and M. Ohira. A framework that supports collective creativity in design using visual images. In *In Proceedings of the Third Conference on Creativity and Cognition*, pages 166–173. ACM Press, 1999.

[33] Mike Thirlwell Nick Bryan-Kinns, Patrick G. T. Healey. Graphical Representations for Group Music Improvisation. In *Second International Workshop on Interactive Graphical Communication*, 2003. Queen Mary University of London.

[34] Donald A. Norman. *The Design of Every Day Things*. Basic Books, New York, 1992.

[35] Elias Pampalk. *Islands of Music Analysis, Organization, and Visualization of Music Archives*. PhD thesis, 2001.

[36] Anthony LaMarca Paul Dourish, W. Keith Edwards and Michael Salisbury. Using Properties for Uniform Interaction in the Presto Document System. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 55–64, 1999.

[37] John Perry, Elizabeth Macken, Neil Scott, and Jan McKinley. Disability, inability, and cyberspace. In Batya Friedman, editor, *Human Values and the Design of Computer Technology*. Centre for the Study of Language and Information,U.S., 1997.

[38] Stuart Pook, Eric Lecolinet, Guy Vaysseix, and Emmanuel Barillot. Context and Interaction in Zoomable User Interfaces. In *Published in the AVI 2000 Conference Proceedings (ACM Press)*, pages 227–231, May 2000. Palermo, Italy.

[39] Pamela Ravasio, Sissel Guttormsen Schär, and Helmut Krueger. In Pursuit of Desktop Evolution: User Problems and Practices with Modern Desktop Systems. In *To Appear in: ACM Transactions on Computer-Human Interaction (TOCHI)*, 2004.

[40] Jun Rekimoto. Time-machine computing: a time-centric approach for the information environment. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 45–54. ACM Press, 1999.

[41] D. A Schön. The Reflective Practitioner: How Professional Think in Action. 1983. Basic Books, NY.

[42] Ben Shneiderman. *Designing the user interface (2nd ed.): strategies for effective human-computer interaction*. Addison-Wesley Longman Publishing Co., Inc., 1992.

[43] David A. Smith, Alan Kay, Andreas Raab, and David P. Reed. Croquet - A Collaboration System Architecture. In *First Conference on Creating, Connecting and Collaborating through Computing*, pages 2–9, 2003.

[44] David A. Smith, Andreas Raab, David P. Reed, and Alan Kay. Croquet: A Menagerie of New User Interfaces. In *Second International Conference on Creating, Connecting and Collaborating through Computing*, pages 4–11, 2004.

[45] Randall B. Smith, John Maloney, and David Ungar. The Self-4.0 User Interface: Manifesting a System-wide Vision of Concreteness, Uniformity, and Flexibility. In *OOPSLA '95 Conference Proceedings, Austin, Texas*, pages 47–60, 1995.

[46] Walter R. Smith. The Newton Application Architecture. In *Proceedings of the 1994 IEEE Computer Conference, San Francisco*, 1994.

[47] Robert Spence. Presentation (Chapter 6). In *Information Visualization*, 2001.

[48] Michael Terry and Elizabeth D. Mynatt. Recognizing creative needs in user interface design. In *Proceedings of the fourth conference on Creativity & cognition*, pages 38–44. ACM Press, 2002.

[49] Barbera Tversky. Distrotions in Memory for Visual Displays. In A. Drunwald S. R Ellis, M. K. Kaiser, editor, *Pictorial Communication in Virtual and Real Environments*, pages 61–75, 1991. London: Tayler and Francis.

[50] Barbera Tversky. Cognitive Maps, Cognitive Collage and Spatial Mental Models. In *Proceeding of the European Conference COSIT*, 1993. Springer-Verlag.

[51] Andy Wachowski and Larry Wachowski. The Matrix, 1999.

[52] Terry Winograd. A language/action perspective on the design of cooperative work. In *Computer-Supported Cooperative Work: A Book of Readings, Kaufmann, San Mateo*, 1988.