EG UK Theory and Practice of Computer Graphics (2005)
L. Lever, M. McDerby (Editors)

The Data Surface Interaction Paradigm

Rikard Lindell and Thomas Larsson

Department of Computer Science
Mälardalen University
Sweden

Abstract

This paper presents, in contrasts to the desktop metaphor, a content centric data surface interaction paradigm for graphical user interfaces applied to music creativity improvisation. Issues taken into account were navigation and retrieval of information, collaboration, and creative open-ended tasks. In this system there are no windows, icons, menus, files or applications. Content is presented on an infinitely large two-dimensional surface navigated by incremental search, zoom, and pan. Commands are typed aided by contextual help, visual feedback, and text completion. Components provide services related to different content modalities. Synchronisation of data surface content sustains mutual awareness of actions and mutual modifiability. The prototype music tool was evaluated with 10 users; it supported services expected by users, their creativity in action, and awareness in collaboration. User responses to the prototype tool were: It feels free, it feels good for creativity, and it's easy and fun to use.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information interfaces and presentation]: User Interfaces

1. Introduction

The graphical user interface of today's computers has its origin in the findings and inventions of Alan Kay at Xerox PARC in the 70s. With it came a lot of interaction elements, techniques, and metaphors to explain and convey the behaviour of the computer. With the limited computational power and memory capacity then available they succeeded to make machines "user friendly". They were so successful in the process that their design has remained a constant factor for graphical user interfaces since then. However, the desktop metaphor suffers from several drawbacks related to how content information is retrieved, visualised, and manipulated. Therefore, there is a constant need to explore and develop the area of graphical user interfaces. In this study we have constructed a different interaction paradigm for the graphical user interface of computers, which we call the Data Surface Interaction Paradigm (DSIP). It is designed to work for large deposits and flows of information, user collaboration, open-ended creative tasks, and multi modal interfaces.

The DSIP is content-centric instead of tool-centric, which means that small embedded software plug-in components implement the functionality. The idea has many similarities with the now discontinued OpenDoc project, which brought a component framework for creating compound documents that removed the need of monolithic applications on desktop platforms [CDE95].

With the DSIP users do not have to conduct any explicit file management. Further, the user is not forced to initiate the transfer of data from the machine's primary memory to its secondary memory through an explicit save action. And finally, the system takes care of the information for the user in the same manner as with handheld devices for instance the heap storage framework of the Palm OS or the data soup storage framework of the Newton OS [Smi94]. For this type of bookkeeping of users content neither file systems, nor relational databases, are suitable [GSL00].

For the DSIP, all content remains visually present in its context and the surrounding set of elements for a specific content sets its context. All content is visualised on a flat, infinitely large surface. The content may have varied scaling that permit hierarchical relations between different content information elements. Tversky has shown that people map hierarchical relations to differently scaled objects of a flat image [Tve91]. People view small-scaled objects as sorted under large-scale objects in the hierarchy. On the basis of cognitive maps [DS73] and cognitive collage [Tve93] spatial

semantics helps navigation. People's different spatial abilities, reported by Dalbäck et. al. [NHS96], have been taken into consideration by not relying on geometric 3D.

The content is navigated by graphical trajectory zoom and pan. The graphical zoom we use, in contrast to semantic zoom used in various applications [KP93, BBB95, PLVB00, BMG00], visualises all the details even in de-zoomed state, thus colour and static structure (shape and geometric relations) serve as cues to the content and users navigation experience, which does not have to be broken by e.g. popping information elements. Large-scale annotations can help navigation. Algorithms for level of detail (LoD) efficiently reduce the amount of data that must be filtered from database to image buffer, and can be designed to avoid popping effects or other artifacts. The trajectory zoom [Spe01] allow zoom and pan in one single action, however, usability evaluations have shown that a specific pan action is more satisfactory when the subjects are pleased with the zoom scale and familiar with the context. Hyper structured information visualisation can be solved with inspiration from lexivisual communication found in newspapers, comics, and movies [Kin91]. Zoom allow "linked" content to be visualised per se in context, thus users does not need to break their navigation context to get the information. Although Barreau and Nardi [DB95] reported that users in most cases rely on spatial semantics for information navigation of their file systems, Ravasio et. al. [RSK04] found that Barreau and Nardi's conclusion came from the poor usability of system search tools, which requires users to search manually. With the large flows and deposits of information today and in the future, improved search tools are required. Navigation of the data surface is further aided by incremental search. All content is in the search scope not just meta-data. Feedback for search condition satisfaction reminds users of content locus. The feedback is immediate for each keystroke, and information that is not of interest becomes transparent. The search also selects the content for command invocation and manipulation. We use the affordance of the selected content for command invocation, i.e. the commands available for the user to invoke are provided by the content of the selected component. Pook et. al. showed that this also works in a semantic zoom system [PLVB00].

However, by removing the command selection menu and replace it with typed commands, the tools of a data surface system are said to be non-visual. Context help and text completion aid the users to quickly find the suitable command. A single model sustains recognition for learning and recollection for efficient use. Users who have formulated their intentions can directly invoke the correct command without navigation for it. Pre-visualisation of the command results allows users to investigate the outcome of their actions before the actions take place. This mechanism is advantageous to creative tasks and actions [TM02].

In the data surface environment, shared synchronised surface areas favour collaboration. Users can more easily work together on a project on the shared surface [MIEL99]. According to Mynatt a flat surface is also suitable for creative and open-ended collaborative tasks [MIEL99]. Visual feedback make users aware of each other's actions and provides an external referent for negotiations [NBK03]. The interaction styles in a data surface environment favour multi modal user interfaces instead of pixel precision and window manipulation. 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 recognition.

2. Scenario

The following scenario was retrieved as result from interviews with professional music tool users: David and Lisa plan to perform their music live at a dance club. On each of their music creativity devices they have all samples, sounds, and songs in a shared synchronised content data surface environment. Most of the time they sit together, but sometimes they create music at different locations connected through the Internet. During collaboration the synchronised visual and acoustic space of the devices help by providing an external referent to their discussion, when online they use 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. The system's beat alignment eliminates rhythmic glitches. They rehearse a couple of times before conducting their performance.

3. Design Values

The design of the DSIP was inspired by the values in the Human Interface Principles chapter of the Macintosh Human Interface Guidelines [Com92]. It states 11 design values that have brought the success of the desktop metaphor interface. However, although these values - despite Gentner and Nielsens' attack on them [DG96] - in themselves are sound, they were to compromised by the technology available in the late 70s. Today with hardware accelerated graphics, powerful CPUs, and vast storage capacity we can reassess these values into new designs. Here we will focus on user control and modelessness. For the DSIP we have added the values of creativity and collaborativity. Defining the meaning and the implication of these qualitative values in this context helps understanding the DSIP.

3.1. Creativity

A value for design creativity holds within itself both the activity of the user, and the activity the tool affords. Users

creativity, within their activity, may it be writing a report or a piece of music, should not be challenged, but encouraged by the tool. The creativity afforded by the tool connects to users' expectations. For instance, if we imagine another model than the typewriter used for word processors, why not something similar to "fridge poetry".

Schön's theory of how peoples' creative work [Sch83] and Mynatt's et. al. design implications for creative and open-ended task software [TM02] were also taken into consideration. The implication of their work was to allow users to evaluate their actions in advance before they happen; the environment is designed to go with the flow.

3.2. Collaboration

Collaboratively working on a report, or a piece of music, is done today by turn-taking; mailing files back and forth or putting the latest version on a server. Current systems are designed for one on one interaction [DFAB98, LÖ1] and, despite the success of Internet, does seldom support collaboration for other tasks than communication. In observations of and interviews with computer music creators, we have seen that collaboration at the same place and time were done by one primary user controlling the computer and with a secondary user as adviser. This impedes creativity. Collaboratively - as design value - must therefore be supported as a part of the fundamental interaction paradigm for computer systems.

3.3. User Control

In a strict sense user control means that the user, not the computer, initiates all actions [Com92]. This may be true for office applications, however, the process the user interacts with may very well be dynamic or collaborative. User Control - in context of the work for this paper - means the users control the process, they may be over-viewing (zoomed out) or inspecting its details (zoomed in) and in control of what is going on with the process, but not necessarily initiating all actions. User control isolates the process from the tools by which users interact with the process. Tools should not initiate actions. Illustrated by an example, a tool should not auto-correct misspelled word (but can of course indicate possible misspells), whereas, game avatars may trade nova-crystals for ore with you.

3.4. Modelessness

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 [Nor92]. The desktop paradigm bring about a number of different modes, such as, for instance, dialogue boxes that prevent users from further actions in an application until they have completed the dialogue. Gentner and Nielsen advocates one should yield to that it is impossible to create a modeless interface [DG96]; they point out that even in the Macintosh Human Interface Guidelines the bigger part of the modelessness section is devoted to guide the correct use of modes. The argument against modelessness; that humans concentrate and focus at one task at hand, does not sustain the design for the user to give away overview and context. To concentrate and to focus transfers to be zoomed in, which means to be within the information and activity, in contrast to de-zoomed i.e. overview.

4. Method

We have used a variety of methods in order to fully grasp the potential positive or negative outcomes. Each method has been adopted to the specific needs in the process and fall in one of these three categories: conceptual studies, empirical studies, and technological studies. The conceptual studies contain related work and the field of cognitive psychology. Empirical studies were in-depth interviews, focus groups, qualitative evaluations in controlled environments as well as through field studies. The in-depth and focus group interviews were based on open form questionnaires with six (6) interviewees. Interviews of users revealed that they find music software tools based on the desktop metaphor cumbersome and constraining for their creativity. The primary qualitative evaluation criteria for these studies were the users' attitudes toward these tools. The technological studies were implementation of a series of prototypes with increasing interactivity and complexity.

5. Prototype Music Tool Design

The prototype tool was designed and implemented to test the DSIP approach. It has a set of available components. One text component allows users to write text in any empty region of the data surface. One sound component plays and displays streamed sound contents. One sound controller component is used for manipulation of sound length, pitch, volume level and pan. And finally one song arrangement matrix component helps in the arrangement of sound controller components to form a song. Note that since the subject users were all Swedish, text displayed in some of the screen shots in the figures is in Swedish.

5.1. Components

An overview of the visual appearance of the content for which music tool components were created is displayed in figure 1. The small clustered rectangles in the top of figure 1 (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).

5.2. Sound Loop Component

The loop data is visualised by the loop component illustrated in Figure 2. At the top, the loop caption is shown (1). The

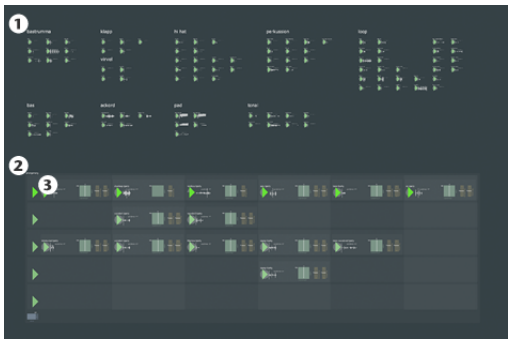

Figure 1: *Overview (zoomed out) of the data surface, (1) shows the loop clusters, (2) the arrangement, (3) the loop component embedded into a loop controller component in the arrangement.*

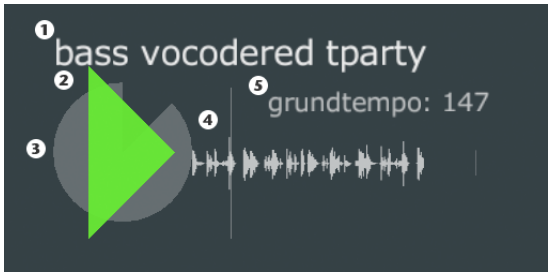

Figure 2: *The loop component (zoomed in), (1) the caption, (2) play/stop button and indicator triangle, (3) pie diagram animation of loop progress, (4) loop waveform and progress indicator (thin vertical line), and (5) the base tempo of the loop.*

loop's play state is displayed by the intensity of the triangle (2) opaque saturated green for playing and transparent green for 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 loop's waveform, 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).

5.3. Sound Loop Controller Component

When a user copy a loop to the arrangement it becomes embedded into a loop controller, see Figure 3. 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 and 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).

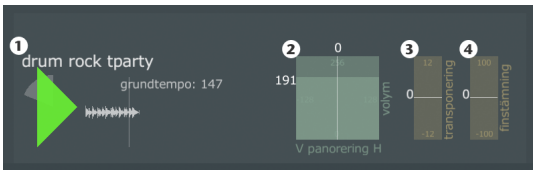

Figure 3: *The sound loop controller component, embeds the sound loop component (1) and adds volume/left-right-pan (2), transpose (3), and fine-tune (4).*

5.4. Arrangement Component

In the arrangement component, see Figure 1 (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 starting the next loop. Furthermore each row has a launch button that starts all the loops in the same row at once, while stopping all the rest of the playing loops. Start and stop of loops align to bars, so the user does not have to worry about timing mouse clicks or commands. The arrangement also has a tempo controller, which affects all loops in real-time for auto tempo correction.

5.5. Navigation

Navigation of the content is done through a zoom and pan technique called trajectory zoom [Spe01]. Trajectory zoom allows both pan and zoom in one action. Think of it as following a trajectory from the viewer through the surface at the cursor point (in reality the scale is simply centred to the cursors position). We use graphic zoom where no details are hidden, thus, both popping of information and user context breaks are avoided. De-zoomed content show miniaturised details that provide a visual cue of it. Zoom interaction is done with the scroll wheel of the mouse. An example of how the zoom action is visualised is shown in Figure 4. A separate pan action was implemented with the right mouse button by drag-and-drop of the entire surface.

5.6. Search

Users can also navigate by search with the find command; the argument substring is used as a filter to select all the content information elements that contain the argument. This is done incrementally with immediate feedback, thus each appended character tightens the condition, which grey out more and more information elements. In Figure 5 this search procedure is illustrated. There is also a leap command allowing users to zoom in and inspect a detailed view of each consecutive information element of the selection.

5.7. Commands

In the design of how command should be evoked, we considered both the desktop paradigm and command line interfaces

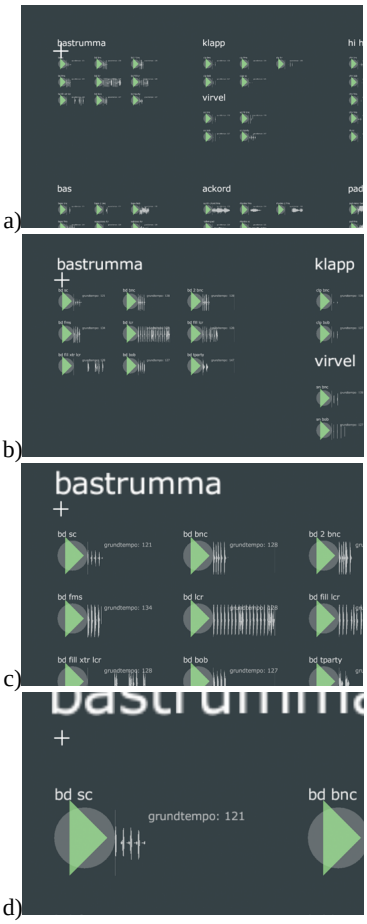

Figure 4: *(a) Before the zoom action. The user aim with the cursor at the position marked with a white cross. (b) The user has started to turn the scroll wheel. (c) Almost there. (d) The users has reach the desired level of zoom.*

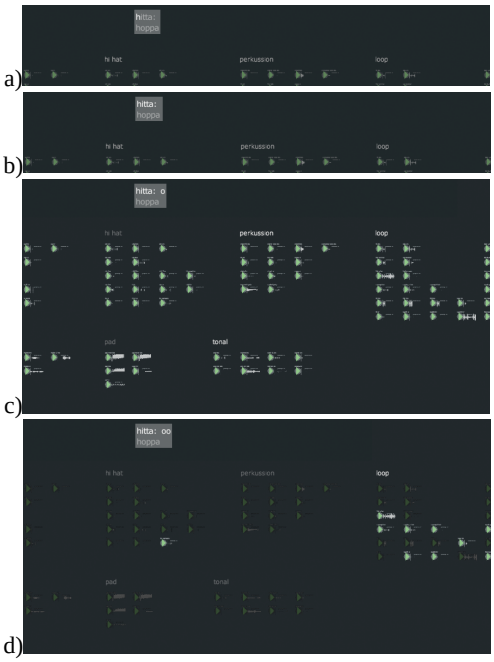

Figure 5: *(a) The user has typed the 'h'-key and thus initiated search for the find command named 'hitta' Swedish for find. (b) The user typed the colon-key that leaps to argument input. (c) The user typed 'o', which selected every information element containing the substring 'o' and filtered away the rest which became transparent. (d) The user typed yet another 'o', which selected everything containing the substring 'oo'. Notice how few element this short substring selected.*

(CLI). In the desktop environment, commands are selected from menus by point and click. There are key chord shortcuts for commands that are expected to be used frequently. Whereas CLI 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 presentation; the users do not have to remember the command only recognise it. 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. Menus take screen real estate, which is why there are menu bars and hierarchical menus. For CLI users have to remember exactly how to type the commands; online help, "man", guide only the right use of commands. For efficiency commands are often named with acronyms, for instance "list" is "ls", "remove" is "rm", and "change directory" is "cd". This impedes learning. CLI force users to ask for feedback.

5.8. Commands in the Data Surface Environment

We designed a text-based command evocation method. The idea behind typed command was that the users should not have to look at the tool, only the content information. The number of available commands is limited by the selected content. We can think of it as a noun, whereas commands can be regarded as what-can-we-do-with-this-noun-verbs, following the idea of a noun-then-verb interaction grammar [Com92]. For each key press, a feedback help list displays the commands that contain the written substring. The first command item of the feedback list is due to be evoked when the user press the command completion key. In Figure 6 (top) the user typed: "k", which resulted in a list of two commands. In Figure 6 (bottom) the user typed: "ko", which selected the command "kopiera" (Swedish for copy), and pre-visualised immediate feedback enable users to investigate the results of their intended action in advance. This has been shown as particularly usable for creativity tools

[TM02]. A help key is also available displaying all the commands for the selected content context. A unique enough substring selects the desired command rapidly, for instance with “kopiera” (copy) the user may type “o” and then the command completion key to evoke it.

Figure 6: *First and second key press while the cursor hover over a sound component. (a) Feedback of the command that in this context match the typed key is displayed. (b) The first command of the list is due to be invoked, indicated by the stronger contrast. Note that a pre-visualisation of the copy command is also shown.*

5.9. Collaboration

Real-time synchronisation of data surface contents on different devices supports collaboration, mutual awareness of action, and mutual modifiability. All actions and commands are echoed to the other collaborating participant’s machine. The appearance of the visual and acoustic contents 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 typing messages onto the data surface. The author of the message is indicated by colour.

6. The technology of the prototype

In the implementation of the prototype, the Simple Direct Media Layer (SDL) is used as platform for events, timers, threads, network, and sound. SDL makes it easy to set up an OpenGL context. It is also designed for game development, which gives some further advantages, for instance, it makes it easy to develop custom user interfaces and it runs natively with little overhead.

The basic functionality of SDL is very simple; more advanced features are built as add on libraries. In case with audio output, SDL simply sets up a stream buffer for the audio card and generates a callback whenever the buffer needs to be filled. We use 16 bit stereo samples, and the latency for our prototype is 10 ms, which is responsive enough for the task. In our implementation of the mixer and sequencer, the SoundTouch library, running in a background thread, is used for time stretching and pitch shifting of the sound samples. The SDL_Net library is used to send UDP-packages containing commands for synchronisation of collaborating participants’ machines.

The data surface information content is linked to a database containing components arranged in hierarchical scene graphs providing the components relative position and scale. We note that scene graph structures have already become the de facto data structure in 3D graphics simulation and animation packages, due to their flexibility and performance features. For example, classic scene graph packages like Open Inventor [SC92] and Performer [RH94] have found widespread use within many 3D graphics applications. However, together with commodity graphics hardware, the scene graph programming paradigm is suitable for many more applications. It seems natural to use the power of today’s graphics hardware in our implementation. Scene graphs and standard OpenGL allow dynamic features without compromising performance.

In the rendering traversal of the scene graphs, we use the components bounding quads to perform hierarchical view-frustum culling. As the traversal progresses, the components’ transformation matrices are push and popped on the OpenGL model view matrix stack. An automatic selection of an appropriate LoD is also done per component; currently a choice between rendering the components geometry or using cached texture mapped versions is done. Note that this way of rendering provides interactive performance for pan and zoom actions, even on graphically rich data surfaces, which is a key feature for the DSIP. Furthermore, during the rendering traversal, the components are visited in back to front order, which enables utilisation of the alpha-channel, for instance the transparency of unselected contents.

The selection or picking mechanism is also implemented as a search traversal in the scene graphs, the main difference being the usage of the OpenGL selection buffer, for which we set up a limited frustum, centred at the image space picking coordinates, and we also invert the normal rendering order.

Finally, for the text component, we use true type fonts that allows sophisticated text to be rendered. Font rendering, however, is an involved and complicated task. Fortunately, the FreeType and FreeType OpenGL (FTGL) libraries support texture mapped font rendering as one of its rendering styles, which turned out to give us an acceptable balance between visual appearance and rendering speed.

7. Evaluation

The described design ideas behind the DSIP were tested on ten (10) subject users: six (6) solo user experiments and two collaboration pair experiments. The screen area was set to 1024 x 768 pixels on 14" displays. 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. The subject users were handed a written introduction that described the prototype and a few simple tasks to carry out to get them acquainted to the interface, navigation, command invocation, the loops, and the arrangement. Then, during the solo-test, the subject users where instructed to think aloud, whereas collaboration test users were allowed to engage in verbal communication with each other.

The criteria for the evaluation were: usability of the interfaces in general; usability of the command invocation; degree of satisfaction (enjoyment vs. frustration); what roles were taken by the users; and tools effect on collaboration. The evaluation was divided in two phases, first the users completed a set of task by themselves, and secondly they created a song together. The users who collaborated sat in the same room; they could see each other but not each others' screens. During the collaboration phase, one set of loudspeakers played the synchronised sound from one computer.

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

8. Evaluation results

The usability of the navigation tools was satisfactory to all the users. Note that in previous investigations [Lin03] of a desktop paradigm music tool, the users where bored and unsatisfied with the navigation style. The users for these tests, however, really enjoyed themselves, one user even shouted out: "this is fun!". During the second phase the users started to look for sounds. After having found a set of sounds they felt comfortable with they started to create their song. The users negotiate what sounds to select and collect. 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 due to implementation bugs. Two users, one expert and the other a novice, hailed the command invocation style. One expert user disapproved it. On the other hand, the novice user, without any pre-knowledge about command-line interfaces, thought it was really "cool" and revolutionary to write commands.

A few stray comments from the other subject user were: "you are free - good for creativity", "I loved to have everything on top", "good to not to have to load loops", "super nice to escape menus", "easy and fun", "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. 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 and used the commands to launch and stop the loops to create a song. Eight (8) out of ten (10), created songs with the arrangement component as expected.

9. Conclusions

The Data Surface Interaction Paradigm showed very good results for a loop-based improvisation and live music creativity tool. Users were very pleased with the navigation and the approach support their creativity for the task. The command model was not fully embraced, on the other hand visual and audio content synchronisation simplified collaboration. With visual synchronisation an external referent to sound modalities enables users to engage in negotiations for their creative work. Clearly, the users enjoyed the music tool implemented based on the Data Surface Interaction Paradigm. There were strong indications that the content centric Data Surface Interaction Paradigm approach can serve as an efficient, beneficial, and pleasant interaction paradigm for graphical user interfaces of computers. However, to fully understand the potential benefits and pitfalls other application areas and larger data sets have to be investigated before the general case can be proved.

References

[BBB95] BENJAMIN B. BEDERSON J. D. H.: Advances in the Pad++ Zoomable Graphics Widget. In *Third Annual Tcl/Tk Workshop sponsored Toronto, Ontario, Canada* (July 1995). 2

[BMG00] BEDERSON B. B., MEYER J., GOOD L.: Jazz: an extensible zoomable user interface graphics toolkit in java. In *UIST* (2000), pp. 171–180. 2

[CDE95] CURBOW D., DYKSTRA-ERICKSON E.: The OpenDoc User Experience. In *Develop, the Apple Technical Journal Issue 22* (1995), pp. 83–93. 1

[Com92] COMPUTER A.: *Macintosh Human Interface Guidelines*. Addison-Wesley, 1992. 2, 3, 5

[DB95] DEBORAH BARREAU B. A. N.: Finding and reminding:Êfile organization from the desktop. In *ACM SIGCHI Bulletin archive* (July 1995), vol. 27, pp. 39–43. Issue 3, ISSN:0736-6906. 2

[DFAB98] DIX A., FINLAY J., ABOWD G., BEALE R.: *CSCW and Social Issues*, 2 ed. Prentice Hall, 1998. 3

[DG96] DON GENTNER J. N.: The Anti-Mac Interface. In *Communications of the ACM* (Aug. 1996). ACM Press. 2, 3

[DS73] DOWNS R. M., STEA D.: Cognitive Maps and Spatial Behavior. In *Image and Environments* (1973). ISBN 0-202-10058-8 Aldine Publishing Company. 1

[GSL00] GRIMM R., SWIFT M. M., LEVY H. M.: *Revisiting Structured Storage: A Transactional Record Store*. Tech. Rep. UW-CSE-00-04-01, University of Washington, Department of Computer Science and Engineering, April 2000. 1

[Kin91] KINDBORG M.: Visual techniques for orientation in hypermedia structures, 1991. Lic. thesis Stockholm University Department of Computer & System Sciences (DSV) ISSN 1101-8526 1991. 2

[KP93] KEN PERLIN D. F.: Pad an Alternative to the Computer Interface. In *Proceedings of SigGraph 93, ACM Press 93* (1993). 2

[LÖ1] LÖWGREN J.: "Från MDI till interaktionsdesign", 2001. Event: STIMDI'01: Svenska tvärvetenskapliga intresseföreningen för människa dator-interaktion. 3

[Lin03] LINDELL R.: Users Say: We Do Not Like to Talk to Each Other. In *Graphical Communication Workshop* (2003), pp. 87–90. 7

[MIEL99] MYNATT E. D., IGARASHI T., EDWARDS W. K., LAMARCA A.: Flatland: New Dimensions in Office Whiteboardsm. In *Proceedings of CHI99* (1999), pp. 346–353. 2

[NBK03] NICK BRYAN-KINNS PATRICK G. T. HEALEY M. T.: Graphical Representations for Group Music Improvisation. In *Second International Workshop on Interactive Graphical Communication* (2003). Queen Mary University of London. 2

[NHS96] N. D., HÖÖK K., , SJÖLINDER M.: 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). 2

[Nor92] NORMAN D. A.: *The Design of Every Day Things*. Basic Books, New York, 1992. 3

[PLVB00] POOK S., LECOLINET E., VAYSSEIX G., BARILLOT E.: Context and Interaction in Zoomable User Interfaces. In *Published in the AVI 2000 Conference Proceedings (ACM Press)* (May 2000), pp. 227–231. Palermo, Italy. 2

[RH94] ROHLF J., HELMAN J.: Iris performer: a high performance multiprocessing toolkit for real-time 3d graphics. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (1994), ACM Press, pp. 381–394. 6

[RSK04] RAVASIO P., SCHÄR S. G., KRUEGER H.: 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). 2

[SC92] STRAUSS P. S., CAREY R.: An object-oriented 3d graphics toolkit. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (1992), ACM Press, pp. 341–349. 6

[Sch83] SCHÖN D. A.: *The Reflective Practitioner: How Professional Think in Action*. Basic Books, NY, 1983. 3

[Smi94] SMITH W. R.: The Newton Application Architecture. In *Proceedings of the 1994 IEEE Computer Conference, San Francisco* (1994). 1

[Spe01] SPENCE R.: Presentation (Chapter 6). In *Information Visualization* (2001). 2, 4

[TM02] TERRY M., MYNATT E. D.: Recognizing creative needs in user interface design. In *Proceedings of the fourth conference on Creativity & cognition* (2002), ACM Press, pp. 38–44. 2, 3, 6

[Tve91] TVERSKY B.: Distrotions in Memory for Visual Displays. In *Pictorial Communication in Virtual and Real Environments* (1991), S. R Ellis M. K. Kaiser A. D., (Ed.), pp. 61–75. London: Tayler and Francis. 1

[Tve93] TVERSKY B.: Cognitive Maps, Cognitive Collage and Spatial Mental Models. In *Proceeding of the European Conference COSIT* (1993). Springer-Verlag. 1