RWTH AACHEN UNIVERSITY

Evaluation of Focus Methods *in a Ubiquitous Computing Environment*

Diploma Thesis at the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Eugen Hon Wai Yu

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. -Ing. Karl-Friedrich Kraiss

Registration date: October 05th, 2005
Submission date: May 17th, 2006

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Schriften entnommen sind, sind als solche kenntlich gemacht.

Aachen, den 17. Mai 2006

Contents

List of Figures

List of Tables

Abstract

More than 25 years have passed since the first introduction of desktop computers. Most people are now familiar with the use of a mouse and keyboard to control a single desktop machine.

Researchers predict that our life will be enriched by even more devices in the near future. Unlike desktop machines these devices are not strictly coupled to a single machine. Multitasking will not only be supported by switching applications but also by changing the way how we use and combine devices.

To further understand how users can interact with a room populated with computer enhanced devices, this thesis analyses the differences between such an environment and a traditional desktop environment. In particular, we are interested in how users can actively control the way devices communicate with each other.

Further, we conduct several experiments to compare existing techniques and point out relevant factors and obstacles that should be considered in future design.

Überblick

Seit der ersten Einführung von Desktop Computer sind nun mehr als 25 Jahren vergangen. Der Umgang mit fensterbasierten Betriebssysteme gehört mittlerweile zum alltäglichen Wissen.

Forscher sagen voraus, dass wir in naher Zukunft in einer Welt leben werden, die mit vielen intelligenten Geräten ausgestattet sind.

Im Gegensatz zum Desktop Computer sind diese Geräte nicht fest an einem einzigen Computer gebunden. Multitasking, die gleichzeitige Bearbeitung mehrerer Aufgaben, wird dann nicht nur durch den Wechsel der Anwendungsprogramme unterstützt, sondern auch durch ein flexibleres Zusammenspiel von Arbeitsgeräten. Mit welchen Eingabe- und Ausgabegeräten ein Nutzer seine Aufgaben erledigen wird, wird nicht vom Systemdesigner vorher bestimmt werden.

Um die Interaktionen in einer solchen Umgebung besser zu verstehen, analysieren wir im Rahmen dieser Diplomarbeit die Unterschiede zwischen einer derartigen Umgebung und einem traditionellen Desktop Computer.

Insbesondere interessieren wir uns für die Frage, wie ein Nutzer die Kommunikation zwischen verschiedenen Geräten aktiv verwalten kann.

Um bekannte Methoden miteinander zu vergleichen, wurde ein Prototyp gebaut, der eine solche Umgebung simuliert. Mit diesem haben wir mehrere Experimente durchgeführt, um entscheidende Faktoren und Einschränkungen dieser Methoden zu ermitteln. Ebenfalls wollen wir mit den gewonnene Kenntnissen neue Methoden entwickeln.

Acknowledgements

First of all, I would like to thank Prof. Dr. Jan Borchers for giving me this interesting and challenging thesis topic.

On my way to grasp the vision and the spirit of ubiquitous computing, Tico Ballagas gave me a lot of guidance and helpful hints. Apart from his comments and tips for my experimental design, I would like to thank him for his reviews and comments on my thesis work.

I want to thank Prof. Karl-F. Kraiss for being my second examiner, besides I am still grateful for the excursion to Liège three years ago.

I would also like to thank all past and present members of the Media Computing Group for the joyful time. I enjoyed the spontaneity, creativity, and humour of every member of the chair a lot.

Since my thesis deals with a networked environment, I have to thank the system administrators, Christoph Wilhelm and Stefan Werner, for providing me a stable platform to work with. They were excetpionally helpful during the migration from Panther to Tiger.

Many thanks go to the reviewers, without whom this work would not exists. I want to express all my thanks to Anke Sprödefeld, Michael Bieseke, Robert Kalis, Volker Schönefeld, and especially Torsten Sattler for all their help and support. Well, and, did I mention Torsten Sattler?

Without the works of preceding diploma students, this thesis would not have the form it currently has: I would like to thank Thorsten Karrer for his lovely Textemplate, Nils Beck for his title design, and Jan Buchholz for his stencil packages.

I would also like to thank the numerous students who participated on the experiments; their efforts, comments, and creative suggestions are an important part of this thesis.
Finally, I would like to thank my parents and my brother for their love and support.

Thank you.

Conventions

Throughout this thesis we use the following conventions:

Text conventions

Definitions of technical terms or short excursus are set off in coloured boxes.

EXCURSUS:
Excursus are detailed discussions of a particular point in a book, usually in an appendix, or digressions in a written text.

Definition:
Excursus

Source code and implementation symbols are written in typewriter-style text:

```
myClass
```

The whole thesis is written in British English.

Chapter 1

Introduction

"Machines that fit the human environment, instead of forcing humans to enter theirs, will make using a computer as refreshing as taking a walk in the woods."

—Mark Weiser, The Computer of the 21st Century

1.1 Interactive Workspace

Mark Weiser's vision of tomorrow's computing [Weiser, 1991] consists of multiple computationally enhanced objects that become part of the fabric of our everyday lives.

We are approaching this vision as hardware costs keep on falling. Hardware like large displays, wireless network, surround sound systems, mobile devices, like cell phones, digital cameras, Personal Digital Assistants, finds its way into living rooms and offices. These devices form an environment that is named the Interactive Workspace [Johanson et al., 2002a].

INTERACTIVE WORKSPACE:
Interactive Workspace is a room with multiple computer-enhanced devices. Multiple users are expected to work in such a room collaboratively.

Definition: *Interactive Workspace*

To illustrate the advantages of using an Interactive Workspace for collaborated working, consider the following scenario:

Scenario: editorial staff

Peter, Clark, Barbara, and Mary are members of the editorial staff of 'Business Weekly', a weekly published magazine. Currently they are holding a meeting in an interactive room. This room consists of two plasma displays, each of which is attached to a separate machine. After all members have taken their seats, Mary walks up to a wall-mounted display (Display A). She transfers some data from her laptop to the display, presenting the cover of the upcoming issue using a standard graphical program. 'Our cover story of our next issue will forecast the global economy in the next decade.'

Clark suggests, 'Hey Mary, the design looks great. But how about using a font that creates a more dramatic feel? '

'I am not sure what you mean? '

'Let me show you.'

Instead of walking up to the display to modify the document, Clark requests the system, saying, 'Let me control the graphical editor on Display A.' A tone sounds to confirm the new setting. Clark now uses the mouse and the keyboard in front of him to apply various effects on the font. A few clicks later, he says, 'There you go.'

To compare the different designs Mary opens the same document with the previous design on another display (Display B). Mary agrees with the alternate design, 'That's a good idea, looks more appealing. Let's use the new one.'

Peter is also satisfied, 'I think I can also use this style in my article as well,' opening a document on display B. 'Mary, do you mind if I close the document? '

'Go ahead. I still have a copy on my machine.'

Peter uses his preferred pointing device, his digital pen, to close the window. Then he copies the cover's style and pastes it to his article. Barbara, who is sitting next to Peter, asks, 'Peter, can I borrow your pen for a while? I need to sketch a diagram.'

'Here you are,' passing the pen to Barbara. 'Thanks,' says Barbara, grabbing Peter's device and starts scribbling.

In this scenario we see how Mary shares her work on mul-

tiple displays. We observe how Clark uses a speech command to change the configuration of the room, including the behaviour of its devices. Both Mary and Clark can control a document concurrently to exchange their ideas. Any member can join the interaction as needed. Devices become properties of the room that can be shared among users.

Following interactions are important to make a room appear to be such a coherent system:

- Sharing information on multiple machines.
- Concurrent manipulation of content.
- Sharing of devices among multiple users.

Sharing information on multiple machines

People often change their working environment. They leave their office for a meeting, a conference, or a presentation. In a shared environment workers will need to share information such as their current results or their schedules. For that, data from the private workspace has to be easily accessible. Further, users should be able to transfer data to other workers or from device to device (see Figure 1.1).

Sharing content

Concurrent manipulation of content

Beside conversation, collaborative modification or creation of digital artefact are means for users to exchange ideas in multiple user workspaces (see Figure 1.2). In the physical world workers can write on the same sheet of paper. In the digital world sharing of devices is not that easy or even unsupported. Therefore an Interactive Workspace should support this feature as well.

Concurrent manipulation

Sharing of devices among multiple users

In everyday's life people can pick up tools like pen and paper, use them, pass them to another person, and she can continue using them seamlessly. Computer devices do not share this flexibility. Although the use of wireless networking technologies increased a lot in the past decades, devices such as mice and keyboards are still bound to a single machine. When a room is equipped with multiple computers

Sharing devices

and devices, we would like to combine their use in a way that suits our task (see Figure 1.3). Thus the system should provide users a way to manage communication between devices.

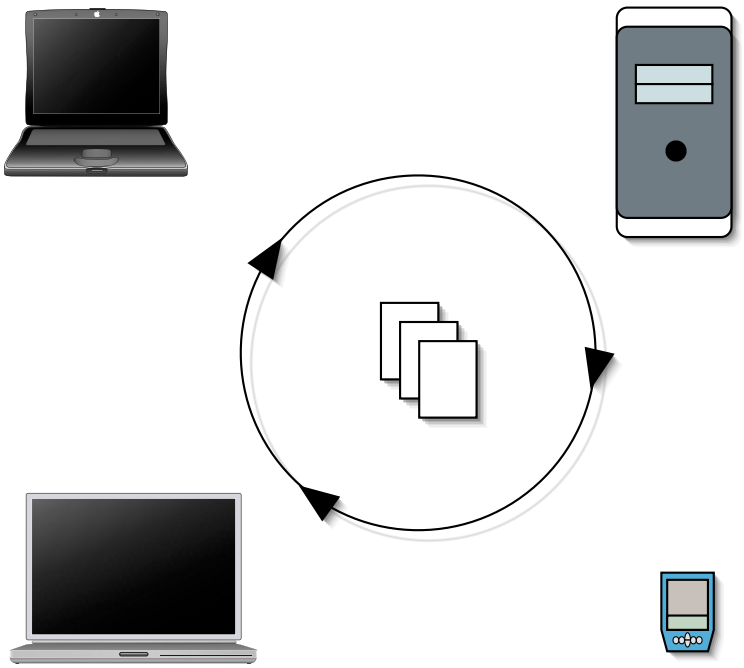


Figure 1.1: Sharing information on multiple machines.

Each of these problems forms a research area of its own. We will focus on the third problem and investigate how users can manage multiple devices. In the following work, we will call a room with such qualities an Interactive Workspace. Figure 1.4 shows an example of an Interactive Workspace.

1.2 Control of multiple devices

Devices mediate the communication between users and the system. An input device transmits the users' action to the system whereas an output device delivers the system's message back to users. If a room contains multiple devices, each user has to choose which devices she wants to use.

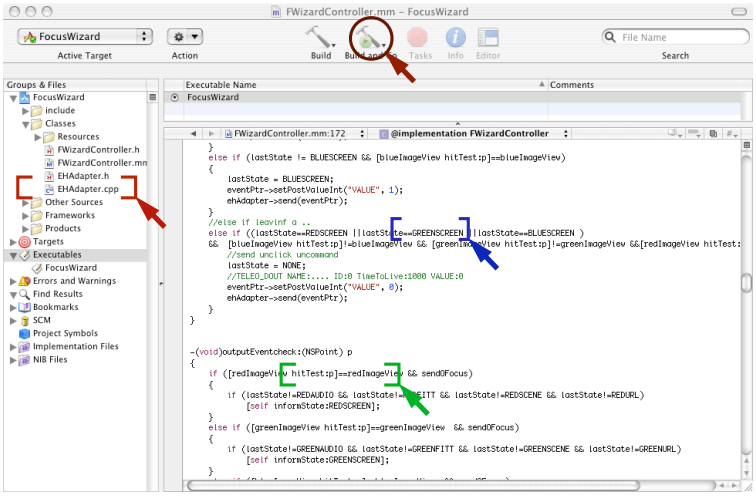


Figure 1.2: Concurrent manipulation of content.

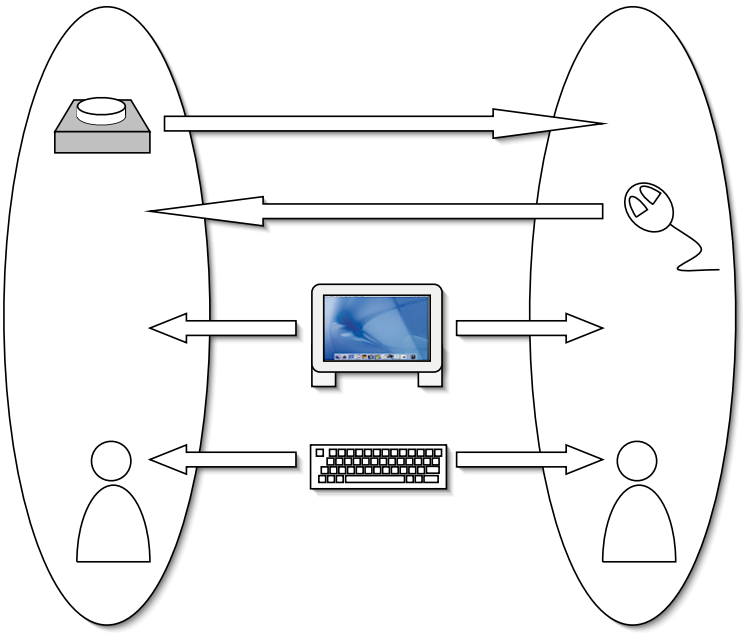


Figure 1.3: Sharing devices among users.

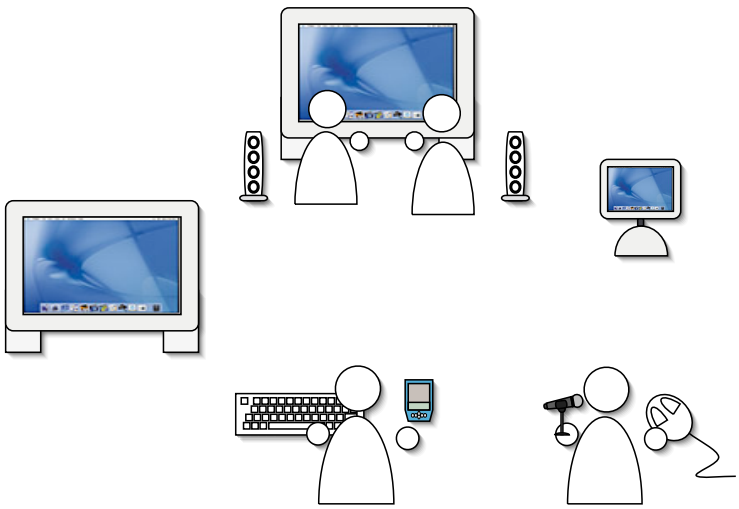


Figure 1.4: An Interactive Workspace with multiple users and multiple devices.

Non-interrupting

However, the primary interest of users is to complete a task. Spending too much time on setting the room's configuration will create an unsatisfying experience. In the worst case a user might be so distracted by the configuration task that she forgets her primary task. This is especially undesired if the Interactive Workspace is used for fast and critical decisions such as in a fusion control room [Wallace et al., 2004].

Therefore we are searching for techniques that are natural, easy to learn, and easy to perform. They should provide a fast way to change the room's settings and to inform the system what a user wants to control. Advances in this field also have benefits for other research area: techniques that work well for Interactive Workspaces can also be applied to an intelligent home.

In a traditional desktop environment each user is assumed to process a single pointing device and keyboard, which can be used to control an application on a display. Events generated by the keyboard are redirected to a specific component inside this application. This component is called the *focus*. To change this *focus*, a user has to select an active window using the pointing device. First, she moves her

cursor icon above the widget of a window, e.g., a textfield, then she clicks on the mouse button. The textfield will be highlighted and all keyboard inputs will be redirected to it. Some events insert characters into this widget, while other key combinations invoke special commands like copy and paste.

The scenario above illustrates a focus policy named *Focus-on-Click*, which is a common interface of many modern operating systems. There are also other window managers supported by UNIX systems that introduce further focus policies such as *Focus-follows-Mouse* and *Sloppy-Focus*, see [Alex Hioreanu's Window Manager][1] . In both cases windows are activated when the cursor enters the frame of a window, no further clicking is required to activate a window. Another common method to change the focus is to use key combinations to switch through all running applications.

Focus-on-Click

We see that even for the traditional desktop, there are different focus techniques.

In general, the shift of focus can be understood as the establishment of a virtual route that redirects input events to the application associated with the active window. The result of the application will be shown on an output device.

FOCUS METHOD:
Focus method, also called focus task, is a set of actions taken with the aim to change a system's focus. After which, the system redirects events different than before.

Definition:
Focus method

Users shift their focus to resume their task using another set of devices. It is important to distinct the *focus task* from their *primary task*. Thus we define the primary task as follows:

PRIMARY TASK:
The actions a user takes to archive her goal that she wants to solve with an Interactive Workspace.

Definition:
Primary Task

[1] http://people.cs.uchicago.edu/ ahiorean/ahwm/

1.3 Scope and Contribution

In this study, we will compare different focus methods in terms of performance and acceptance by users. To date, the concept of focus is still closely related to the practises and understandings of the traditional desktop environment. In this work we will explore a model that extends the current understanding of focus. We will see what makes up the focus in the context of an Interactive Workspace.

To explore various aspects of focus techniques, we apply the RWTH MediaSpace to emulate an Interactive Workspace. We are interested in physical interaction techniques such as pointing, speech command, or selection by physically touching a device. For these techniques, no graphical display is required.

Although many projects explored collaboration on a single display, there are not many works that address the problem of input management. Especially the following questions are of our interest:

- What is the difference between the focus of a traditional desktop and an Interactive Workspace?
- Do we have to introduce new devices dedicated to this function?
- What factors are relevant for the design of focus techniques?
- What factors are important in deciding which technique to use?

In general there are two types of focus methods: implicit focus techniques

IMPLICIT FOCUS TECHNIQUES:
Implicit focus techniques are also known as system-initiated focus methods; they require a continuous monitoring and interpreting of the users' behaviour; it is the system's responsibility to resolve the users' intention and to set the event redirection route(s) accordingly. These techniques belong to the category of adaptive interfaces.

Definition: *Implicit focus techniques*

and explicit focus techniques.

EXPLICIT FOCUS TECHNIQUES:
Explicit focus techniques are also known as user-initiated focus methods. With an explicit focus technique, a user has to consciously take a concrete action to define the focus.

Definition: *Explicit focus techniques*

In this work, we will explore explicit focus techniques. Implicit techniques differ from explicit techniques that they require more effort and cost to prototype and to implement. For instance, to gather implicit information, a tracking system that covers the users' position in the room is required. Furthermore, implicit techniques often require technology from the area of artificial intelligence and predefined models to resolve the users' intention. We will discuss and compare these two classes of techniques in detail in chapter 3.

Explicit focus technique only

A further experiment compares two graphical user interface (GUI) based techniques with a physical one. One of the GUI-based techniques applies a miniaturised iconic map for the user to select. Another technique uses the cursor to identify the selected machine.

To reduce the complexity of our experimental design, we will only consider a single user environment . If we want to scale our environment to multiple users, we have to find a way to keep track of who is sending an event. Many projects avoid this problem by assuming that the users interface with the room though their personal devices. This could be Personal Digital Assistants (PDAs), notebooks, or mobile phones. We do not make this assumption in our work, because we want to allow arbitrary persons to use the system. Since we are dealing with a setting designed

for a single user, we neglect the social effects of multiple users.

Generative model

Contribution In the theoretical part of this work we will introduce a framework about focus in Interactive Workspaces. This framwork is based on the work by [Ballagas et al., 2003]. It is applied to analyse both traditional and post-desktop environment. We will discuss the generative features of this framework which can help future designers to look for interesting interactions. Another model presented is a 3 steps model that organises a selection technique into three phases. It is based on Raskin's elementary task [Raskin, 2000]. It can be used for heuristic evaluation of new selection techniques. To identify further factors that have an impact on the performance of focus techniques, we map [Nacenta et al., 2005]'s evaluation of content relocation to the domain of focus techniques.

During the development of our testbed, several tools have been developed to improve the existing system. A new Patch Panel scripting language is implemented to overcome some shortcomings of the original script. The new grammar is documented in appendix A.

To control applications remotely, a scripting language AppleScript is used in the testbed. For example, it is used to provide audio feedback or to rearrange windows. We increased the performance of its proxy to create a more fluid interaction for testpersons.

Another attempt to reduce delays of windows rearrangement is to access the window manager of each machine locally. A proxy for the NSWorkspace object in the Cocoa objective C language is created to allow the users to remotely change the active window (see appendix B).

With the AppleScript proxy mentioned above the users can only send AppleScript scripts to a machine for execution. We implemented the other direction, so that the users can send events by calling an AppleScript script (see Figure 1.5). Since many software products for Macintosh use AppleScript to extend their functionality, this interface enables

further possibilities to prototype new interactions. For example, in combination with third party applications such as SallingClicker[2] or FlyGesture[3] , the users can send events to the Event Heap with their mobile phones or invoke services in the room with a mouse gesture.

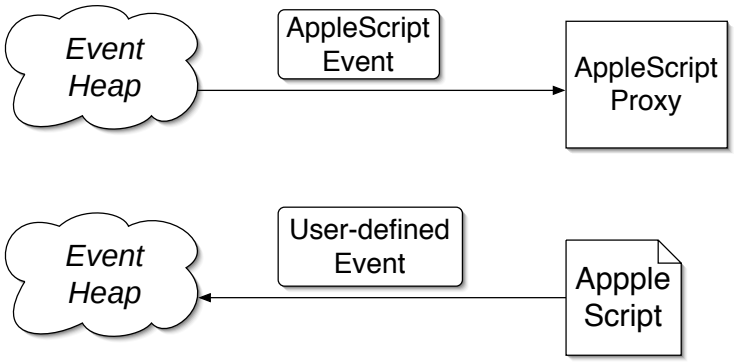


Figure 1.5: The upper case describes how an AppleScript event can execute its content on a machine running the AppleScript proxy; the lower case shows how the execution of an AppleScript script sends an event to the Event Heap.

Finally, we will discuss a series of experiments we conducted. We use them to compare physical focus techniques and GUI-based focus techniques. Based on the results we gathered from the experiments we will propose a new focus device.

1.4 Structure of this thesis

Chapter 2 is a review of literature and related works in the field of event redirection in a post-desktop context. There are three main categories of review: we start with systems that provide a single display shared by multiple users; then we take a look at other existing interactive rooms; we also review literature about technologies that can be used to implement focus techniques.

[2]http://www.salling.com/Clicker/mac/
[3]http://flyingmeat.com/flygesture/

In chapter 3 we analyse the term focus in the context of the traditional desktop; and generalise this concept to Interactive Workspaces. A comparison of implicit and explicit focus techniques shows their key differences and explains why we prefer explicit focus methods in our research. To ease later discussions, we introduce a model to describe focus techniques. Distant interaction is another research field in Human Computer Interaction(HCI), we analyse the relevancy of some of their results in our context.

In chapter 4 we describe the components that make up our experimental platform. We present the hardware and software we have used to prototype some aspects of an Interactive Workspace. We also talk about the problems we encountered during the development of the testbed and how we solved them. Beside the testbed, we also discuss the method we apply in detail. This includes the task our subjects have to solve and the procedure of each trial. Both, the described testbed and the method, apply to the subsequent chapters, in which we compare different sets of focus techniques.

In chapter 5, 6, and 7 we present the results of our experiments and our observations; then we discuss the characteristics of each focus technique. The experiments also uncover problems in our method and testbed. Thus they help us to refine our evaluation method incrementally.

Finally, chapter 8 summarises this thesis work and discusses the effects of the assumptions we made in our experiments. A stepwise removal of these assumptions might induce further research. Furthermore, we introduce a focus device prototype that regards the results we collected from the experiments.

Chapter 2

Related work

"If God has created the world, his primary worry was certainly not to make its understanding easy for us."

—Albert Einstein

2.1 Single Display Groupware

Since the introduction of large displays, much work has been done to improve their usage. Many of these works investigate how to use a large screen space in a more effective way and how collaborated computing enables multiple users to control a single shared display. These systems often identify users by their private device such as personal digital assistant or personal computer. Note that in these systems, inputs from private devices will be directed to the shared one. We can consider these projects as the first attempts to break away from the strictly coupled computer environments. We will call this class of systems *Single Display Groupware*.

Away from strictly coupled devices

2.1.1 Multi-Cursor window manager

Distinguish users

The Multi-Cursor window manager introduced by [Wallace et al., 2004] is a system that enables users to connect to a shared display using their local machine. Upon connection a coloured cursor will be displayed. If the user selects a window with her cursor the border of that window will turn into the same colour (see Figure 2.1).

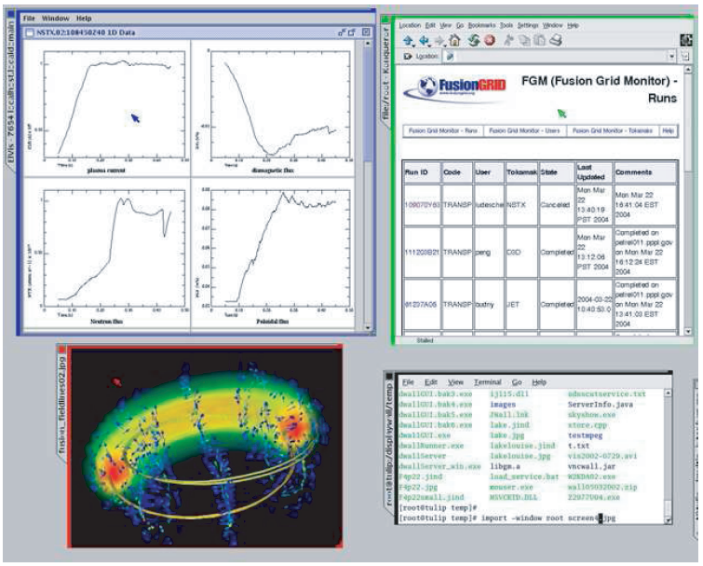


Figure 2.1: The Multi-Cursor window manager uses colour-coding to distinguish users from each other. Taken from [Wallace et al., 2004].

To differentiate events from different devices, an identity number has to be attached to each of them. In their work only three bits are available, therefore at most eight persons can use the shared machine simultaneously, although an average person can discriminate up to 150 different hues.

It is comparable to the PointRight system (see section 2.2.2) because it extends the Windows-Icon-Menu-Pointer (WIMP)[Chignell and Waterworth, 1991] metaphor to a new environment: mouse and cursor are still regarded as primary tools to manage input redirection.

2.1.2 The Pebbles Project

In the Pebbles Project, Brad Myers and his fellow researchers explored various interaction techniques between personal computers and mobile devices such as Personal Digital Assistants (PDAs) [Myers, 2000]. One of their applications, the Remote Commander, allows users to control the same cursor on a shared machine. Further, users can also send keyboard inputs using Graffiti, a standard gesture recognition program for PDAs.

However, the system does not offer explicit control over the cursor's ownership. It relies on the social protocol among the users to decide whose turn it is. Thus, there is no semantic of device ownership or dynamic control reconfiguration in their work.

Among all applications that are created in the Pebbles Project, the PebblesDraw application is most related to our project (see Figure 2.2). Each user can configure the application to accept events from her PDA. On the GUI of PebblesDraw, there is a special button labelled with "Add User". If a user wants to join the interaction with PebblesDraw, she can ask a collocated user, who is already using the application, to press the "Add User" button to grant her access, after which she has to type in her name and the serial port number of her PDA to identify her inputs. This can be seen as a focus technique at the level of input devices.

Input device joins application

In this work they use a different approach to distinguish between different cursors than the Multi-Cursor window manager. Beside drawing cursors in different colour, the shapes of cursors are also varied. Further, the current drawing tool selected is also visible in a icon next to the cursor.

PebblesDraw, Wasinger's ShopAssist [Wasinger and Krüger, 2005], and Slay's Ukey device [Slay and Thomas, 2006] share a common feature: users interact with their environment through their PDAs.

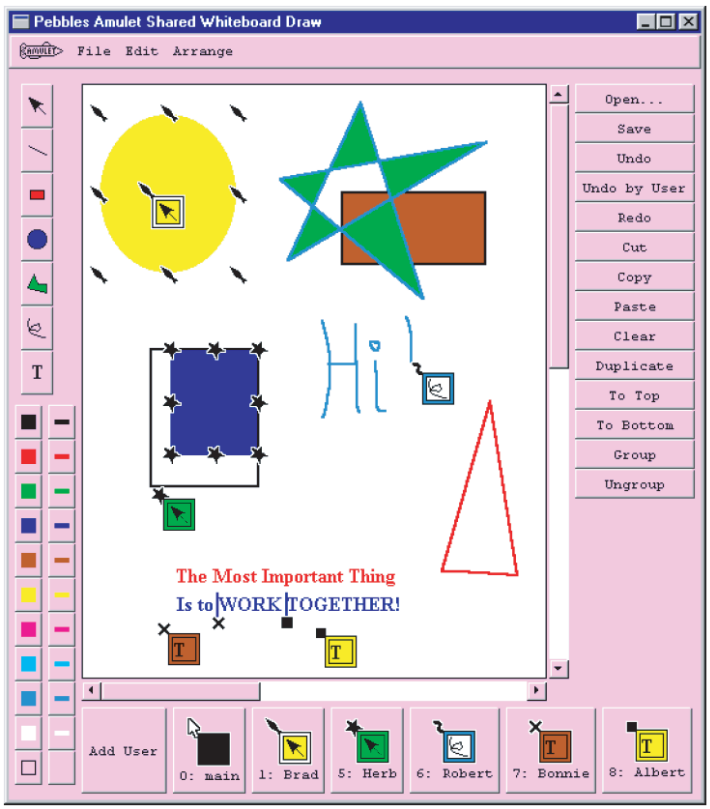


Figure 2.2: PebblesDraw is controlled by multiple PDAs.

2.2 Interactive Workspaces

Multiple devices

We have seen how different projects exploit a single output device to support collaborative interaction. As stated in chapter 1, Interactive Workspaces are equipped with more than just one display. Moreover, we can find multiple types of device in such an environment. The main challenge of the following projects is to improve the management of information across multiple displays.

2.2.1 ARIS

[Biehl and Bailey, 2004] introduced the ARIS (Application Relocation in Interactive Space) system, which is based on

the Gaia architecture [Romàn et al., 2002]. The motivation of their work was to enable users to move running applications among displays. This is more convenient than pure content relocation as the user does not have to reopen an application to modify or view the transfered content.

On each desktop, running applications are visualised by windows. Windows that can be relocated contain a special button in the title bar. Upon clicking on this button, an iconic map appears on the screen. This map shows a top-

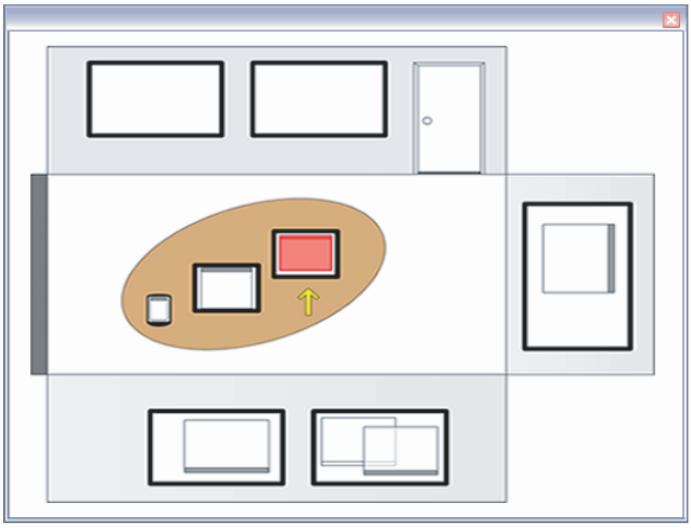


Figure 2.3: An iconic map. Taken from [Biehl and Bailey, 2004].

bottom layout of the room. The displays in the room and applications running on them are represented by icons in this map. The icons of each display are positioned relative to their physical position. Similarly, the icons representing running applications are sized and located relatively to their current display. Users can relocate an application by dragging an application-icon and dropping it on the icon of a target display.

Drag and drop application

It is likely that the user wishes to continue manipulating the content after relocating a window on the new display. Hence, a second feature is introduced which allows users to redirect their mouse and keyboard input to another display. To specify the target display, the user moves the cur-

sor above its icon. After two seconds, the system confirms the new setting by changing the icon of the target display.

2.2.2 PointRight

In the traditional desktop environment a pointing device is the primary input device to change the focus. This technique still works if multiple displays are attached to a single machine. For instance a software called nView by Nvidia support this feature [nView][1] . However, if the displays are connected to independent machines, the borders of the local display become the boundaries of the cursor.

Virtual topology

To solve this problem [Johanson et al., 2002b] introduced the PointRight system in Stanford's iRoom. Each user of the system is required to bring along their personal computers with a pointing device and a keyboard. The cursor on their personal computer can be understood as an indicator of "where the user is". If the user moves the cursor out of the display's border the cursor "reappears" on another display according to a predefined room model (see Figure 2.4). This creates the illusion that all displays are connected to the same machine in a multi-display configuration. These connections between the various display are also referred as "Virtual Paths". Furthermore, the input of the keyboard follows the cursor.

2.2.3 The Intelligent Conference Room

Speech command

MIT's Intelligent Room [Brooks, 1997] is a long-running facility, that is used for formal and informal meetings. It is equipped with numerous computer-controlled devices such as lights, projectors, LED displays and sound equipment. People in the room are detected and checked by cameras that are also applied to record meetings. To detect sound in the room, an array of microphones is mounted on the ceiling. Although a wall-mounted LCD touchscreen is the primary interface to configure the room and to show its

[1] http://www.nvidia.com/object/feature_nview.html

Figure 2.4: When a user moves her cursor out of the display, it "reappears" on another one.

status, users can also use a hand-held microphone to access services supported by the room.

To create a more natural speech interface and explore affective interfaces, a spin-off project works on an avatar as an interface to control the room's configuration. It also applies eye-tracking technology so that the avatar responds to speech commands only if the speaker looks at it.

This project makes extensive use of technologies in computer vision, robotics, speech understanding, and natural language processing to capture the context of a group meeting.

Compared to the projects PointRight and ARIS, the Intelligent Room does not discriminate users; there are only two parties: the room and the group of users in the room. One of the users is chosen as the main user. Therefore the collaboration happens rather on a social level.

2.2.4 Universal Interaction Controller

Slay and Thomas introduced the Universal Interaction Controller (UIC) in [Slay and Thomas, 2006]. It consists of four

components: the Interaction Manager, the Ukey device, the Clipboard Manager, and the Ve-World. Similar to Stanford's iRoom, their communication architecture builds on the Event Heap and Data Heap.

The Interaction Manager is responsible for event redirection. Depending on its state it translates input from heterogeneous modalities and forwards them to the correct recipients.

A user's focus changes as he reconfigures the Interaction Manager. This is done through the Ukey device (see Figure 2.5). A tracking framework mounted on the ceiling informs the system where the Ukey device is pointing to.

However, the primary task of Ukey is to copy content from a display and paste it onto another. For that Ukey and the Clipboard Manger work together to transfer data from a display to the Ukey, then from the Ukey back to another display. Users point with the Ukey device to determine the source and target of data. It is not further mentioned how Ukey might cooperate with further input devices.

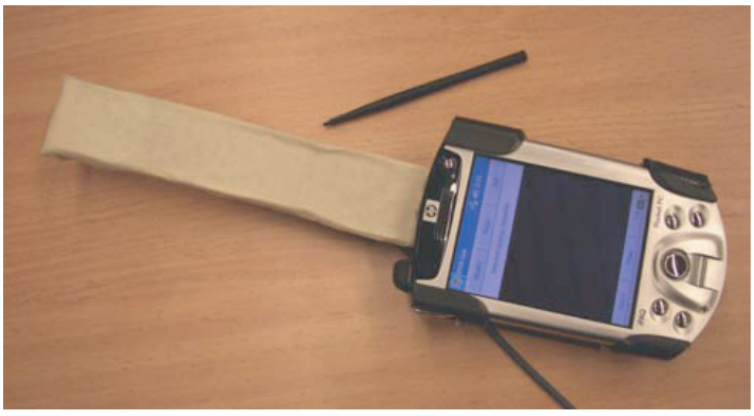


Figure 2.5: The Ukey devices can tell the system where a user is pointing to. Taken from [Slay and Thomas, 2006].

The Ve-world is a virtual reality based application; it uses information from the tracking framework and visualises the devices in the room with all its significant furniture for orientation. Further, a virtual ray shows which display the

Figure 2.6: To interact with a room a user points her device to a display and uses the touch sensitive PDA to copy and paste content. Taken from [Slay and Thomas, 2006].

user is pointing at.

2.3 Post-desktop interaction

Beside the methods proposed in the projects mentioned above, we can find further potential techniques to control devices that use post-desktop technologies.

Also we will discuss some projects that explore the use of speech, gesture, or spatial information. We will consider their applicability in our domain. Further, these techniques can be combined into a multimodal or bimanual interface.

2.3.1 Speech interface

Among post-desktop interfaces speech is an important research area. In [Rosenfeld et al., 2001] Rosenfeld points out two characteristics of speech:

- "Speech is an ambient medium rather than an attentional one."
- "Speech requires modest physical resources."

The first point suggests that a speech interface does not requires focused attention. In an Interactive Workspace the users' primary concern is to solve a task using the room while the management of its devices should not interrupt the primary task. Therefore speech is an appropriate technique to handle the management.

Further, the users' locations are not fixed, they might need to access devices that are out of their reach. Using speech commands, users do not need to travel across the room.

To date, technologies that recognise concrete word becomes more and more common in mobile devices, such as in [Leong et al., 2005]. Although their work mostly concentrates on the implementation details and acurracy of the system, it indicates that one can use speech to control projector screens, projectors, and lights in their experimental settings. This further motivates the use of speech commands as focus techniques in this work. We will take a closer look at how speech performs in our context in chapter 7.

2.3.2 Pointing gesture

When someone asks you, 'Where is the post office?', you probably will say, 'Turn left around that corner', pointing with the finger in the direction you are referring to. During a conversation in the real world, pointing is a natural gesture to supplement a message. In the following we will review some projects that use this gesture to interface with a system.

[Vogel and Balakrishnan, 2005] explored different instances of freehand pointing. Their prototype is based on a motion tracking system where the user needs to wear passive reflective markers. Their aim was to create a "point and click" user interface for a large display. Three different pointing

and two clicking gestures were implemented and evaluated in their work. Because they observed high error rates in absolute pointing techniques, they recommend techniques that use relative pointing and a clutch for freehand pointing. Freehand pointing can easily be transfered to be used as a focus technique in an Interactive Workspace.

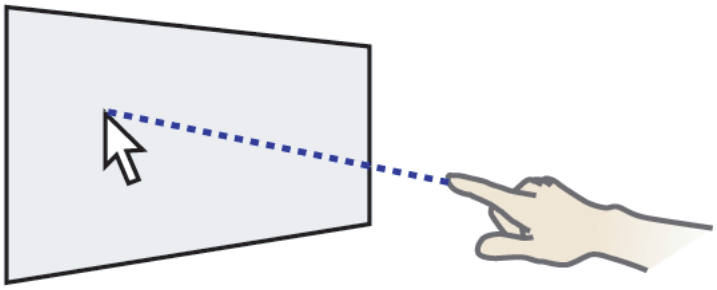


Figure 2.7: Absolute freehand pointing. Taken from [Vogel and Balakrishnan, 2005].

Wilson and Shafer 's XWand project [Wilson and Pham, 2003] uses accelerometers and magnetometers to track the position and the orientation of a wand-shaped device (see Figure 2.8). Furthermore, Wilson and Pham introduced the WorldCursor, a ceiling mounted laser pointer that provides feedback of the current position [Wilson and Pham, 2003]. In their project, the XWand is used as a primary input device to interact with an environment. For example, one can turn on a light by pointing the XWand to it and push a button. It is also possible to interact with a display: when a user moves the projected laser spot onto a display, it turns to a cursor.

2.3.3 Multimodal interfaces

Often speech and gesture are combined to make sense of the users' focus. [Siracusa et al., 2003] showed how to combine video and audio cues to determine who is talking and to whom she is speaking to. [Stiefelhagen et al., 1999] introduced a similar approach which also applies speech and camera images. Although the intention of these technologies is to support automatic capture of meetings, we believe

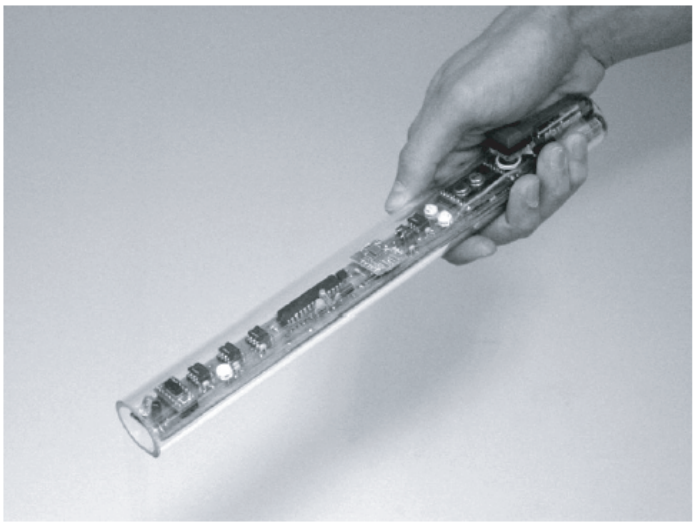


Figure 2.8: XWand - a pointing device with integrated tracking sensors. Taken from [Wilson and Shafer, 2003].

that this technique can also be used for redirecting voice input similar to what has been done by Bolt in [Bolt, 1980].

2.3.4 Eye-Tracking

[Kaur et al., 2003] showed that humans fixate an object before interacting with it through speech commands. Instead of pointing to a device for activation we can use eye-input to focus devices.

[Vertegaal et al., 2005] demonstrated some devices that can track whether they are being attended. A device knows that a user intends to use it, when it is fixated by her eyes. To enable a device to see whether someone is interested in using it we can attach eye-contact sensors to them (see Figure 2.9). Such devices are called *EyePliances*.

When a user looks at the device, an EyePliance opens a communication channel to the user. It is important to mention that eye-inputs are only used to build up a communication channel with a device. The primary task still has to be solved by exchanging data over this communication

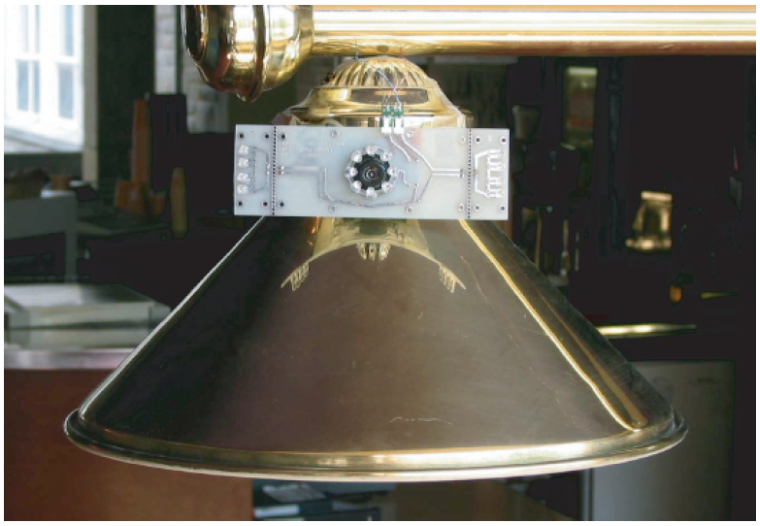


Figure 2.9: A device that can sense whether one looks at it. Taken from [Vertegaal et al., 2005].

channel.

2.3.5 Continuous tracking techniques

As we have seen in some projects with multi-modal and speech interfaces, eye inputs, the posture of the head, and the direction of speech are all information that can describe the context in an interactive room. We can use these inputs to infer the current task of a user. Thereafter the system can change the input management for the user implicitly. No active involvement of the user is required.

[Brumitt et al., 2000] gives an example that uses positional information to determine the event route.

Often an intelligent component is needed to resolve the user's intention. Much research in the field of artificial intelligence deals with this problem as well [Isbell et al., 2004]. However, as stated in the introductory part, we do not concentrate on this type of techniques in this work.

Chapter 3

Theory of focus

"A theory is something nobody believes, except the person who made it. An experiment is something everybody believes, except the person who made it."

—Albert Einstein

The word "focus" has many different meanings across different scientific fields. Physic scientists use the word in a different way than a psychologist, mathematicians, theologians, linguists, communication scientists, or computer scientists. However, all the different meanings share one property: they describe something that converges into a single entity.

Convergence

Another property that is important to the following discussion is taken from the definition of communication science which is the dynamic character of focus. During a dialogue the focus of conversation undergoes a continuous change [Grosz, 1978].

Frequent change

In the field of computer science, according to [Brockhaus, 2002], "focus is a special semaphore, that is assigned to a window being activated". Furthermore, "in a graphical user interface[...], an active window is the window, on which the cursor resides. All inputs and actions refer to the content inside this window." As one can see, this definition

WIMP interpretation

is closely grounded on assumptions made with the WIMP metaphor in mind [Chignell and Waterworth, 1991]. For the approaching age of post-desktop computing this definition needs to be extended.

3.1 Focus in traditional desktop environment

Focus on traditional desktop environment(TDE)

The support of multitasking contributes largely to the popularity of today's desktop environments. Users can switch among active applications by moving the cursor and clicking on their visual representation. Input from the keyboard will be redirected to the active application. This application becomes the focus of the keyboard; a routing channel is established between the keyboard and the application. Within the GUI of each application there are multiple widgets. They also form a different level of focus. One can specify the active widget by clicking on its visual representation (e.g., an URL-addressbar or a textfield). This exemplifies the so-called Focus-on-Click policy (see Figure 3.1)

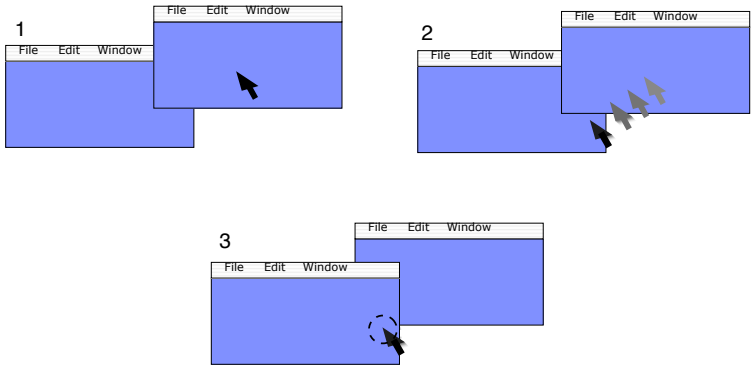


Figure 3.1: Focus-on-Click focus policy.

Focus techniques for TDE

Another focus policy is the Focus-follows-Mouse policy. With this policy every time the cursor enters the boundaries of a window, the window becomes active. No explicit clicking is necessary (see Figure 3.2). The Sloppy-Focus policy is similar to Focus-follows-Mouse, but it differentiates the root window from application windows, see [Hioreanu,

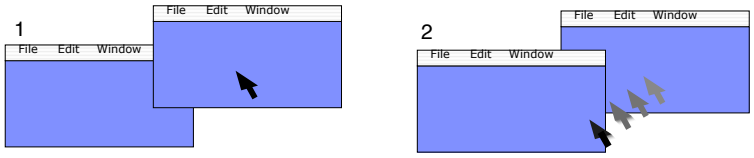


Figure 3.2: Focus-follows-Mouse focus policy.

2002]. This shows that even in the traditional desktop there are different techniques to manage input.

Internally it is the window manager which determines which window should receive an event. It keeps track of the active window and change the focus by reordering registered windows. All windows have an order assigned by the window manager; this order is often referred to as the Z-order [Baudisch and Gutwin, 2004]. Although the implementation of their internal structure can vary from window manager to window manager, all window managers can determine the window that is the current focus of the system.

3.2 Focus in Interactive Workspace

To determine the focus on a single-user desktop the active window is the only information required. All inputs are generated by a single user and sent to this window.

Focus in Interactive Workspace(IW)

In an Interactive Workspace however, we need more information to describe an event route. [Ballagas et al., 2003] points out five key differences when comparing Interactive Workspaces with a traditional desktop:

5 M's paradigm

- Multiple users.
- Multiple machines.
- Multiple input devices.
- Multiple output devices.

- Multiple applications.

This corresponds to [Wasinger et al., 2003]'s analysis which states that an Interactive Workspace consists of: devices, users, and services. All in all, the key idea to keep in mind is that devices should not be strictly coupled to any computer. They should become part of the room as a whole.

Focus Space(FS)

In an Interactive Workspace, each user is associated with a set of tuples that we define as the Focus Space:

$$\{(u, i, m, o, a, w) \mid u \in U, i \in I, m \in M, o \in O, a \in A, w \in W\},$$

where

U is the set of all users

I is the set of all input devices

M is the set of all machines in the room

O is the set of all output devices

A is the set of all applications

W is the set of all widgets

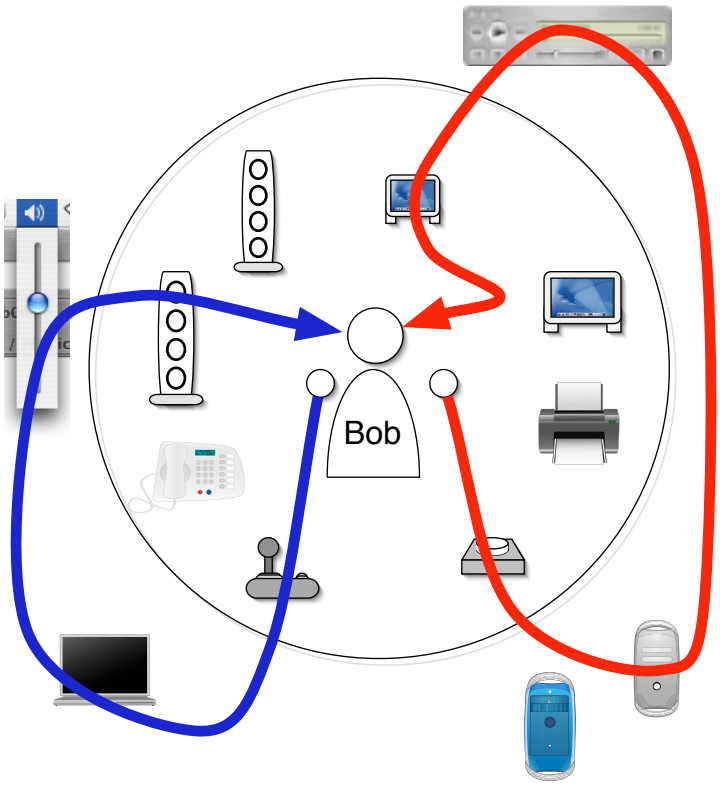
Figure 3.3 shows the information that such a Focus Space delivers.

Event route as tuple

Each Focus Space consists of tuples that have five components. They identify

- which user a tuple belongs to,
- which input device she is using,
- to which machine the event is redirected to,
- from which device the user is expecting a response,
- which application is processing the input event,
- and which widget reacts to the input.

Our model is organised as a set to allow concurrent input. Using a sequential list prohibits multiple users to input at



{(Bob, Knob, Bob's desktop, Display A, DVD player, Timescale),
(Bob, Joystick, Sever for audio, Loudspeaker #2, Loudspeaker's volume)}

Figure 3.3: If Bob is the owner of these two tuples, then he can use his knob to control the time scale of a media running on Bob's desktop. At the same time he can use a joystick to control the volume of loudspeaker #2.

the same time. The system would have to apply a turn taking strategy to manage inputs from multiple users.

With the proposed model multiple users can be in the same room and work independently. They can also join and work together when needed. This gives the system a greater flexibility.

3.3 Shifting focus

In desktop systems, we change the active window to shift our focus; in Interactive Workspaces we modify its focus space.

Allocate, share, release

According to Wasinger's analysis in [Wasinger et al., 2003] the control of a device can be: *allocated*, *shared*, or *released*.

A user *allocates* the control of a device when she wants her input to be directed to it. A user *shares* her control for collaboration (e.g., sharing a display with multiple users). Finally, a user *releases* the control of a device when she terminates its use.

Thus there are three types of modification for a Focus Space:

- When a user allocates the control of a device, a new tuple is inserted into the focus space. Example 1 in Table 3.1 shows how Bob's keyboard input will be redirected to the TextEditor on a machine with the IP-address 137.226.53.64.
- Sharing control is a more complex topic than allocate and release: the system does not only has to reconfigure routing channels, it also has to synchronise information between the shared control. When a device is shared, the Focus Space contains more than one tuple that share the same component. Example 2 in Table 3.2 shows how Jim can use the keyboard #2 to assist Bob.
- When a user releases the control, a tuple is removed from the tuple space. Example 3 in Table 3.3 shows resources are freed from Bob's control.

A more detailed table[1] with storyboards shows different interactions the Focus Space can describe. Using this model we can classify existing interaction for Interactive Workspaces and identify gaps for further research.

[1] http://media.informatik.rwth-aachen.de/~yu/thesis/index.html

Example 1	Focus Space
Before allocation	{}
After allocation	{(Bob, Keyboard, 137.226.53.64, Plasma display, TextEditor, TextField)}

Table 3.1: Focus Space modification: the system *allocates* the devices fro Bob by inserting a tuple into the Focus Space.

Example 2	Focus Space
Before sharing	{(Bob, Keyboard #1, 137.226.53.64, Plasma display, TextEditor, TextField)}
After sharing	{(Bob, Keyboard #1, 137.226.53.64, Plasma display, TextEditor, TextField), (Jim, Keyboard #2, 137.226.53.64, Plasma display, TextEditor, TextField)}

Table 3.2: Focus Space modification: the plasma display is non-exclusively *shared* by Bob and Jim.

3.4 Tuple construction

Assumptions to decrease complexity

When a user requests for the control of an output device, a tuple will be put into the focus space. Her input will be redirected to this device. Before a tuple is inserted into the focus space, users need to create this tuple in the first place.

We have seen that each tuple consists of multiple components. The specification of each component yields an extra action cycle. Thus six additional steps are needed before a tuple and its according event channel is created. This clearly contradicts to the requirement of focus methods to be fluid and transparent and poses a great challenge to any implementation of a fully flexible Interactive Workspace.

To get around this problem the context of the Focus Space can be preserved and only part of the Focus Space needs to be changed on a focus transition.

In many existing systems, the specification process is simplified due to various assumptions a system made. Here are two examples:

Desktop. A desktop system assumes that it is operated by a

Example 3	Focus Space
Before release	{(Bob, Keyboard, 137.226.53.64, Plasma display, TextEditor, TextField)}
After release	{}

Table 3.3: Focus Space modification: when a tuple is removed, the devices allocated are free again. Their control are *released*.

single user with a fixed set of devices. It also shows its output on a single screen. Thus the user only needs to specify the application or the widget she wants to control. Further, there is no need to specify the application, if a user selects a widget, because every widget belongs to a single application.

ARIS. The ARIS system mentioned in chapter 2 assumes that each user brings their laptop along and interact with the room's display through these laptops. Thus each user has a fixed set of input devices which is the keyboard and the mouse of their laptops. Further, each display is connected to a single machine. As a consequence, a user only needs to choose the display, on which she wants to interact with an application.

3 steps model

The specification of these components share a similar pattern. We can divide the interaction into three steps: recognition, indication, and selection (see Figure 3.4).

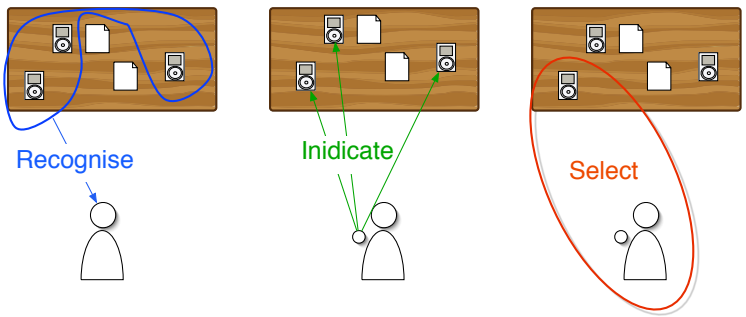


Figure 3.4: 3-steps-model consists of: recognition, indication, selection.

3.4.1 Recognition

Recognition

The system has to inform users which device can be used in the current context by distinguishing them from non-interactive ones. It either provides this information actively or replies upon a user's request. This information can be transmitted using different modalities. They can be tactile, visual, or auditive.

In addition, Interactive Workspaces can provide cues that are in the physical world. For example, we can give devices a certain colour or we can attach labels, tags, or LEDs to them.

Visible windows represent the selectable applications on a desktop. There is an active visual feedback. However, if a window is covered, the specification of a component breaks down. Exposé [Apple Computers, 2003] solves this problem by resizing and rearranging selectable windows in a way such that all of them are visible (see Figure 3.5).

3.4.2 Indication

Indication

When a user indicates a certain device, she informs the system about which device she would like to control. The system updates what it thinks the user is referring to.

This operation corresponds to the same elementary operation that Raskin suggested in [Raskin, 2000]. According to Raskin's description, users indicates an object by moving a cursor to point to a single object without any other action. We can also consider indication as an act to inform the system about the *locus of attention*[Raskin, 2000] of a user. In Interactive Workspaces this act of referencing can use gaze, gesture, or spatial information of users to indicate an object.

Further, indication depends on how the targets are structured. If we navigate through a map such as in the ARIS project, we navigate in a two-dimensional space. If we point to an object in the real world, the devices are organised in a three dimensional space.

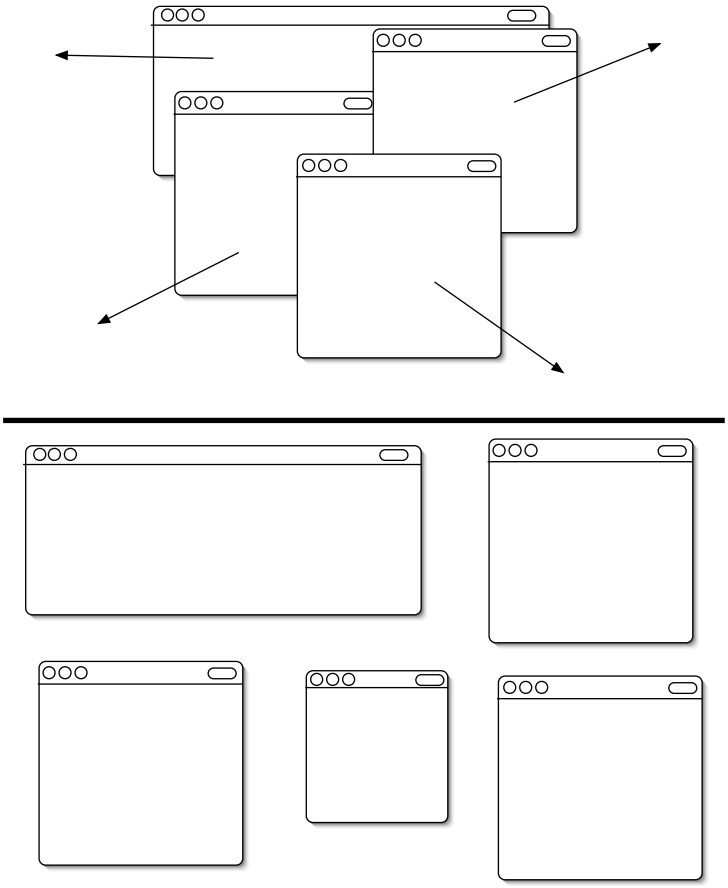


Figure 3.5: Exposé resizes and orders all application windows. The users can *recognise* selectable applications.

3.4.3 Selection

Selection

Users consciously submit a request to change the Focus Space to confirm that the indicated object should become focused. It is comparable to the ”activate” operation in Raskin’s set of elementary actions, although it has another semantic in our context. Such a request can be triggered by pressing a button, releasing a button, or producing a voice command. Upon a successful transition the focus of the system is synchronised with the user’s locus of attention.

3.4.4 Feedback

Each of the three steps mentioned above can provide different feedback. It is important to make the following information clear to users:

Feedback

- What devices are available?
- What device a user is referring to?
- Was the transition of focus successful?

Take for example a desktop environment that applies the Focus-on-Click policy:

- Visible windows show what application can become the activate window.
- The window, on which the cursor is floating above, is the indicated window.
- If a window becomes active, the system brings that window to the front.

If the system does not provide appropriate feedbacks for all of the 3 steps, the interaction will break down. For example, if a user is working from afar and cannot clearly see where the cursor is, she does not know where the cursor is pointing to; changing the active window becomes difficult. Another example that shows a breakdown in the last step: if the window does not pop up to the front when activated, users do not know whether the focus has been changed.

Break down

3.5 Implicit and Explicit focus techniques

As described in chapter 1, there are implicit and explicit focus techniques. Implicit methods differ from explicit techniques in that users do not explicitly take action to change the focus. The configuration is managed by the system. It

keeps track of users' behaviour while they solve their primary task. The system infers the appropriate settings based on the data collected and reconfigure the input management automatically.

Implicit focus technique

In terms of the 3 steps model indication and selection become a single step. For example, using the focus-follows-mouse policy, when a user moves the cursor to indicate her window of interest, it becomes the active window without an additional step to click on the window. Similarly [Brumitt et al., 2000] uses the spatial information, and [Fono and Vertegaal, 2005] uses eye-input to infer what a user is indicating.

Compared to explicit techniques implicit techniques may have a lower cognitive load and are more transparent for users. The focus task is left to the system, so that users can concentrate on their primary tasks.

However, implicit technique suffers from inaccuracy. One source of error is the misinterpretation of focus. In [Kaowthumrong et al., 2002] only 75- 85 percent of system's prediction are correct. Another source of error is the system sensitivity: when a user accidentally moves the mouse away or when a user looks away for a short time, the system immediately reacts to these unintended indication. Users are likely to find these unintended focus changes frustrating or disturbing.

3.6 Distant interaction

Distant interaction

Interactive Workspaces offer many ways to use their equipment. In some scenarios users may change their position frequently and walk around in the room. In other scenarios they work from where they are sitting. In these situations it may be disturbing, if users have to stand up and walk to the devices to set up event routes. Thus the room has to provide interactions that can bridge the spatial gap between users and devices, which are often spatially scattered in the room.

This problem is related to multi-display reaching. In the domain of focus techniques we deal with the question: "How can I send my inputs to *that* device?" In the domain of multi-display reaching we ask: "How can I move a digital object from this device to *that* device?" Their common property is the reference of a device. Nancenta has done a thorough comparison of several techniques for the latter domain in [Nacenta et al., 2005]. Some factors he identified for the multi-display reaching techniques also apply to focus techniques.

Topology of the environment
[Nacenta et al., 2005] refers topology to the way the physical space and its virtual conception is related. During the indication step users navigate through a space of selectable items. This space is formed by the virtual conception of how the selectable objects (in our case, they are devices) are organised. The structure of this space can be seen as an attribute of a focus technique.

This space can be one-dimensional (e.g., a list), two dimensional (e.g., a top-bottom map), or three dimensional (e.g.,referring to items in a virtual reality or the physical world). It can also be discrete or continuous. The navigation though this space can be absolute or relative.

Range
Every interaction has an effective reach range. Suppose we have a setting in which users are working around a shared table, [Nacenta et al., 2005] identified three ranges to transfer content: within hand's reach, beyond hand's reach, and discrete points in the periphery. Adapting this classification to our problem this means that we can touch and manipulate devices that are *within hand's reach*, using our hands or fingers.

But if they are *beyond hand's reach*, users have to walk to those devices or they need to apply other techniques to move the devices close to them.

Users can also refer to a distant device directly. For that, they need a technique to specify the direction they are pointing to.

Accuracy and resolution
Focus techniques can be compared regarding to their error rate. We have seen that explicit focus technique is more accurate then implicit one, this is because implicit techniques can falsely change the focus.

Explicit focus techniques can also produce errors. Some explicit focus techniques use pointing in the indication step. [Myers et al., 2002] reported different error rate depending on the pointing technique used.

Feedback
The importance of feedback has already been mentioned in section 3.4.4. Feedback can be implemented with different modalities in different ways. Since the same technique can have different type of feedback, it opens a further dimension to vary during the design of a focus method.

Focus Device
There are focus techniques that do not require any device to operate (e.g., freehand pointing or eye-input). But if one is needed (e.g., a wand, a mouse, or a microphone), its ergonomic factors effect the overall experience. It might introduce extra time to switch among devices.

Frequency of Change
From the results of our experiments we identified this quality as a further dimension to consider. If the settings in the room is changed frequently, users will prefer a fast and accurate focus technique. If the room just needed to be set at the beginning of a meeting, users are probably willing to use techniques with a higher effort (e.g., the Touch-to-Focus technique in chapter 4) to solve the focus task.

Number of Devices
The number of devices is also relevant for the design of focus methods. If we have a large number of devices, they have to be organised, so that users can identify what they need easily. Thus this attribute is closely related to how to design the topological structure for a focus technique.

3.7 Focus methods

In this section we present five categories of focus techniques that coarsely classify available techniques. We point out their characteristics and different possibilities for implementation.

Pointing techniques
This class of focus techniques uses a virtual directional ray to refer to a device of interest. We can use wand-shaped devices, laser pointers, handheld cameras, or head-mounted cameras to afford a directional ray. If it is required that users' hands are not occupied, we can also use freehand pointing. Pointing using hands is in general not reliable because it is hard for a human's hand to stay steady. It is natural that a human's hands produce sudden movements with a small amplitude, however the effect is amplified because of the distance a ray has to travel. As a result, users often miss the target when they use hands to point to a distant object. This problem is often referred to as the *hand jitter* problem. Beside using hands we can also use our eyes or the direction of our head to specify a direction.

Speech interface
If an Interactive Workspace is equipped with microphones that cover the entire room, users can control the room from any location such as in MIT's Intelligent Room [Brooks, 1997]. If the system have to differentiate multiple users, each of them can use a personal microphone to access services in the room. This class of techniques does not occupy users' visual channel. No additional device is required for these techniques. Thus users' hands can remain on their input devices and continue with their primary task.

ID-based techniques
To overcome the spatial gap between device and user, we can assign "proxies" to each device. Suppose these proxies are within hand's reach, we can select these proxies instead of the real device. For examples, RFID tags or visual codes can be used this way. This technique becomes hard to realise if the amount of devices is large. It will become hard to search for the required proxy. It will also be hard to distinguish one proxy from another.

List-based selection
This class of focus techniques structures available items as a list. In the indication step user navigates through a one dimensional structure. Thus we only need three commands to implement this technique: ”forward”, ”backward”, and ”select”. To implement these commands we can use keys, voice command, or freehand gesture. Although this technique is easy to implement, it breaks down, if the list is too large. A way to diminish this problem is to use different logical ordering. If the system uses visualisation techniques to tell users what devices are available, we can use a different ordering to show available devices are, we can also filter out distractors so that the desired target is easier to identify.

Context-based techniques
We denote *Context-based techniques* as those that use implicit input to redirect events in Interactive Workspace. From the hardware point of view we need a tracking system has to be installed to capture data such as position, gaze, body-posture. Cameras and computer vision techniques are often used. Further, we need software that make sense out of these data. Implicit techniques can be made explicit by introducing an explicit selection step. For example, we can use a voice command, a button or a gesture as triggers for a focus transition.

GUI based techniques
We will call focus techniques that requires display *GUI based focus techniques*. They cannot be operated without a display. Desktop, laptop, or PDA are often used to provide one. For example, the World-in-Miniature technique needs a display to show the miniaturised map of the room.

Chapter 4

Prototyping Interactive Workspace

"Our thoughts create our reality – where we put our focus is the direction we tend to go."

—Peter McWilliams

In chapter 3 we saw that focus involves the selection of multiple entities at different levels. Those are input devices, output devices, machines, applications, and widgets.

The traditional desktop environment provides techniques to activate applications and widgets. The PebblesDraw application demonstrates a technique to connect input devices with an application (see chapter 2).

We do not consider the selection of machines, because machines should be transparent for the users. Users interact with machines through input devices and output devices. Often the reference to a machine is mostly given only through its output devices. Therefore we consider in our experiment output focus techniques.

Our aim is to keep interactions simple so that people will use it. We can use physical interactions to achieve this, because they are more intuitive, natural, and make the technology less intrusive and more transparent.

Further, we make the following assumptions to reduce the complexity of our experimental settings:

- Displays and machines have a 1-to-1 relationship. We will use the terms display and machine synonymously.
- All output devices have the same type, in our settings we will redirect inputs to plasma displays.
- We emphasise on focus techniques at the level of output devices. We can apply focus-on-click to select applications.
- Our system does not support multiple users and implicit information based techniques. To prototype these features we would need tracking technology to be installed or a testbed with multiple wizards.
- Users do not need a personal computing device such as a PDA to interface with the room.

4.1 MediaSpace

To explore the performance and acceptance of different focus techniques, we applied the MediaSpace[Borchers, 2006] to prototype different interactions. The MediaSpace is a room with a length of 7.4 metre and a depth by 6.4 metre. It is located at the RWTH computer science building on the second floor in room 2212.

This room is designed for group meetings, video conferences, project demonstrations, and presentations.

To maintain multiple functions in the room, all furnitures are mobile. Tables, chairs, and even large plasma screens can be rearranged as needed.

The atmosphere in the MediaSpace is light and spacious. It does not have a dark and cinematic atmosphere as in Stanford's iRoom because large windows are installed on one side of the room (see Figure 4.1).

Figure 4.1: The MediaSpace at RWTH Aachen.

For our experiments, we use three touch sensitive plasma displays. When a user is working on one display, the other two are out of her visual field, so that the displays make use of the 3-dimensional space and do not imitate a large wall display. Each of them is connected to a notebook running Apple's OS X 10.4 operating system (see Figure 4.2).

We use a keyboard, a gyro mouse, a laser pointer, and a head-mounted microphone as input devices (see Figure 4.3).

Gyroscopic mouse

A gyro mouse is a relative pointing device. It tracks the movement of the hand using gyroscopic sensors and can be considered as a extension of the traditional mouse into the third dimension. Thus users do not need a surface to operate on but can carry it around. To move the cursor users need to press a clutch. In this experiment we use the GyroRemote produced by Gyration.

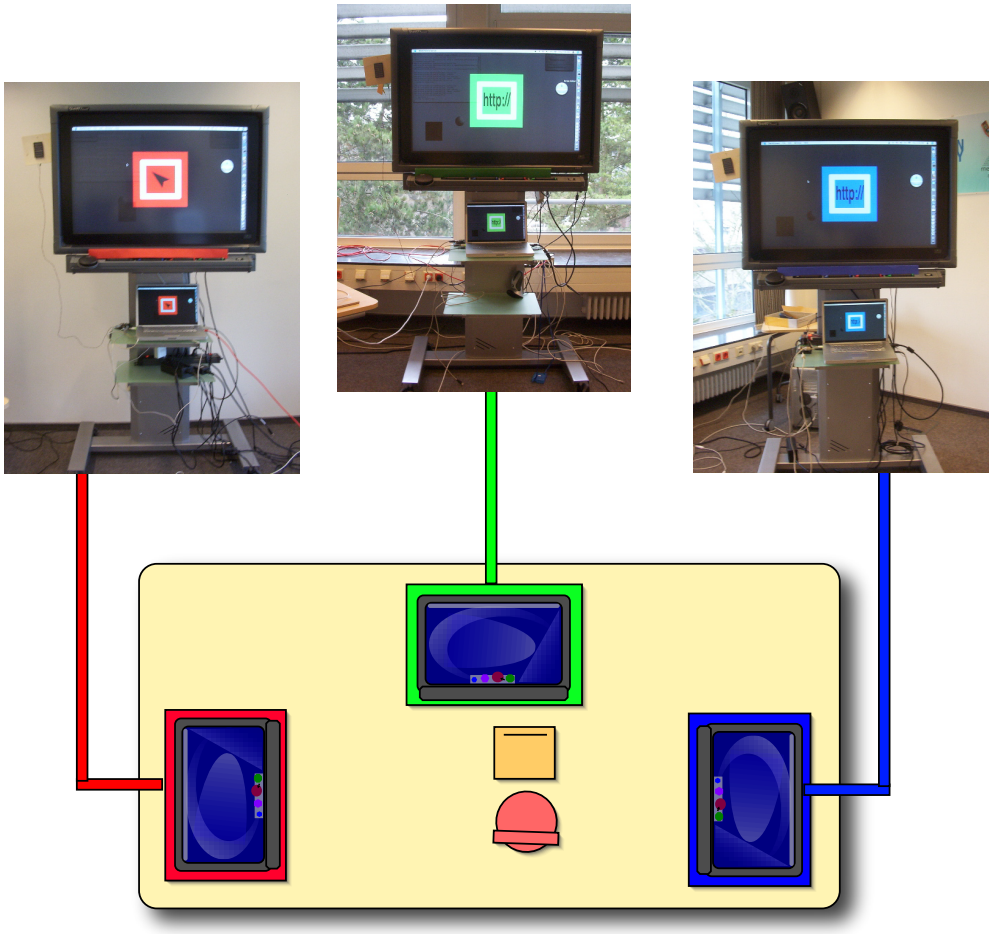


Figure 4.2: The layout of plasma displays in MediaSpace.

4.2 Infrastructure

Tuple-based middleware

The iROS software is a TCP and JAVA based middleware that acts as a blackboard for events. It provides the means for devices and applications to communicate with each other. Each event is structured as a tuple. The machine that manages these events is called the Event Heap server. Every user, machine, and application can generate and post events to the Event Heap. We use an Apple G5 desktop machine with Dual-core processors to run this software.

Applications that are interested in a certain type of event

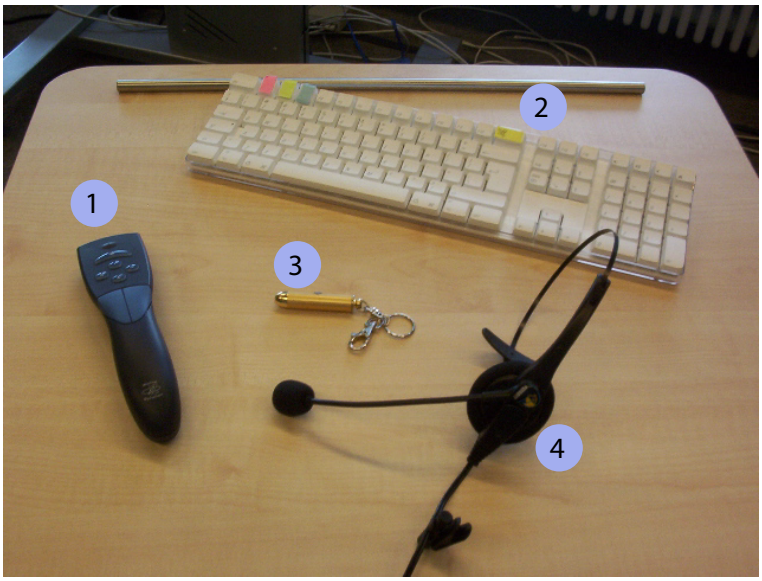


Figure 4.3: Input devices that we used: 1) gyro mouse, 2) wireless keyboard, 3) laser pointer, and 4) head-mounted microphone.

can subscribe and listen for them. Although every machine in the room still runs a standard operating system, they are all connected to each other by the Event Heap (see Figure 4.4). So they do not need an operating system that is exclusively designed for the room.

We created a suite of programs that listen for keyboard- and mouse-events on the Event Heap and use the *java.awt.Robot* object to emulate local inputs. Thus we give users the feeling of remotely controlling a machine with their keyboards and mice.

For our experiments, we also need functionalities at system level, such as reordering of windows and a speech synthesiser. Mac OS X offers a language called AppleScript to communicate directly with these services. Therefore we also run an AppleScript proxy on each machine that is connected to a plasma display.

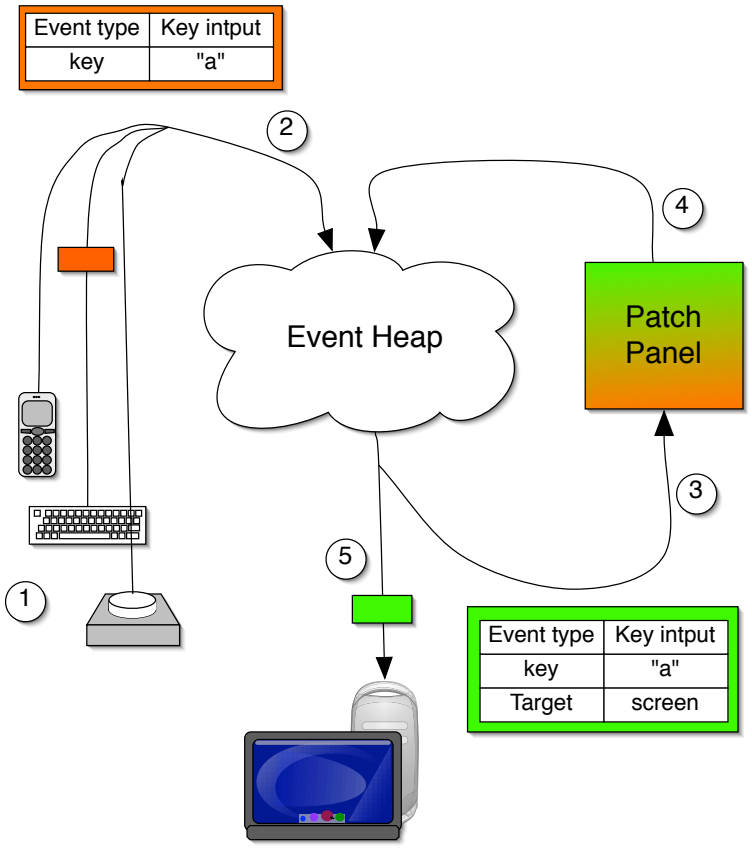


Figure 4.4: 1) Input devices generates input events, 2) Input events sent to the Event Heap, 3) Some events are translated by the Patch Panel, 4) The Patch Panel sends the translated event back to the Event Heap, 5) Event from the Event Heap will be processed by a device.

4.2.1 Patch Panel

As shown in Figure 4.4, a software called Patch Panel is responsible for event redirection. It is an Event Heap client that does not only listen for events, but also emit events.

Event translation

For the Patch Panel, there are incoming and outgoing events. A mapping configuration defines the translation between incoming and outgoing events. When an incoming event that is defined by the Patch Panel's mapping arrives on the Event Heap, an outgoing event is emitted by

the Patch Panel. Thus a mapping configuration defines how the Patch Panel translates events.

Moreover, this mapping can be changed by an incoming event as well. Such an event changes the active event-mapping configuration of the Patch Panel. We can call this type of events *mapping-redefinition events*.

Because of this dynamic behaviour, we can see the Patch Panel as a state machine, where the active event-mapping defines the state, and the mapping-redefinition events define the translations. All possible event-mapping will span the state space.

Focus Model implementation

To emulate focus techniques, we can use the Patch Panel to implement the Focus Space model, where each tuple in the Focus Space corresponds to a subset of the event-mapping. To change this Focus Space, a focus technique generates an event that is sent to the Event Heap. If this event is recognised as a *focus control event* by the Patch Panel it will emit a focus event. As a mapping-redefinition event, this focus event changes the current event-mapping, thus altering one or more tuples encoded in the event-mapping. As described in chapter 3 this yields a change of focus.

To give the Patch Panel the ability to recognise focus control events, we have to assign an initial behaviour to the Patch Panel at the first place. Currently, there are two interfaces to do this: we can configure the initial mapping with a GUI or a scripting language. Since we have to program a complex behaviour, we use the scripting language to configure the initial Focus Space.

4.3 Patch Panel script

During the development of the testbed, we have identified some shortcomings within the original Patch Panel scripting language.

Firstly, it does not fully cooperate with AppleScript events. For example, escape characters are misinterpreted. This

prevents us from using the tabular or the line break character. Therefore Patch Panel cannot handle complex scripts using the AppleScript scripting language.

Secondly, the original parser does not provide any informative error feedback. This results in a high effort in finding errors and makes the debugging of scripts very difficult. While browsing through the source code of the parser, it turns out that the parser's code is hard to understand because parsing and interpretation of scripts is mixed.

New parser needed

While the Patch Panel example scripts compile within 3 to 4 seconds, the script for the experiments needs 30 seconds. This underlines the complexity of the script that will be used. As a consequence, it is necessary to introduce a new scripting language for the Patch Panel and a new compiler for this language.

To achieve a higher clarity and to make the language easier for future researchers to customise, we use an automatic parser generator called SableCC[1] . We define the grammar of the scripting language in a BNF-styled language. Then, SableCC generates a set of classes that form the abstract syntax tree.

To interpret the script, one has to implement a class that traverses though the abstract syntax tree and sets the internal state of the Patch Panel. The definition of the new Patch Panel scripting language is documented in appendix A.

Benefits of the new language:

- Programmers can use comments that span multiple lines.
- Programmers can use timers with different names to create time dependent triggers.
- Definitions of variables and events are not restricted to the beginning of the script.
- Switch statements have a more powerful syntax.

[1] http://sablecc.org/

These points are implemented in an attempt to make large Patch Panel scripts easier to read and to maintain. Allowing multiple line comments makes it easier to provide documentation for scripts and to comment out parts of a script for faster modification.

Each script defines a state machine. With the new language we can define variables and events at arbitrary locations in a script. If an event or a variable is relevant only to a certain state, we can define them in the scope of this state.

A common usage of Patch Panel is to trigger a discrete event when a continuous variable crosses a threshold or enters a certain range. With the modified script language we can define switch statements as Table 4.1. A continuous variable is often translated into a discrete trigger when the value is within a certain range. The style of progamming shown in Table 4.1 allows users to define ranges and behaviours for multiple cases easily.

```
switch float (sliderValue){
       case 0.0:
       case (3,4]:{
            send turnLightOn;
       }
       case (4,5]:{
            send turnLightOff;
       }
}
```

Table 4.1: Example of a switch statement

4.3.1 Future improvements

The ability to hide details is essential to implement a powerful language. The Patch Panel would become more powerful if behaviours inside a trigger can be encapsulated and reused throughout the script. Currently, each Patch Panel script implements a state machine. If we extend the definition to a state chart, we can define more complex be-

State charts?

haviour. With the current Patch Panel we can simulate state charts by running multiple scripts in parallel; however, these scripts cannot share common variables.

As described earlier, Patch Panel is responsible for event redirection by attaching a destination field to incoming events. If the Patch Panel translates an event into another event with the same type this translated event will also trigger further creation of events of the same type. In this way one single event can trigger a large amount of events. Although the Patch Panel contains mechanism to avoid an event to reside on the Event Heap permanently, it still puts a heavy load on the network, making the infrastructure unusable. To solve this problem, we can

- improve the event matching mechanism. The current implementation allows Event Heap clients to listen for events that have certain fields. To solve the problem mentioned above, we can allow an Event Heap client to listen for events that *do not* have certain fields.
- attach a field that marks the level of process, determining whether a low level event is generated by a device or a high level event destined for an application.

4.4 Interaction with the testbed

To capture the performance and acceptance of focus techniques we conducted a series of within-subject experiments. The independent variable in these experiments is the variation of output focus techniques.

4.4.1 Primary Tasks

Primary tasks requirement

Our requirements for the primary tasks are as follows:

- Tasks should cover the use of all input devices.

- Tasks should correspond to interactions in real scenarios.
- Tasks should cover as many modalities as possible.
- Tasks should simulate a natural usage of an Interactive Workspace.

In all experiments each subject has to solve four different types of task on three displays:

- Type in a URL address into the browser.
- Move the cursor to a given position.
- Word repetition.
- Search for a significantly different frame in a video.

Key inputs

In the first task we showed a URL at the title bar of the browser's window. Users' task was to open the website with the URL shown in a browser. The URL was chosen randomly from a set of valid URL addresses. To enter the URL into the address bar, the subject used the gyro mouse to click on the address bar and the keyboard to generate keyboard events.

Cursor movments

In the second task a user saw a golf ball and a hole. Her goal was to move the ball onto into the hole. The ball followed the cursor when it was moved. Each time the initial position of the golf ball and its target position were randomised. The user should use the gyro mouse to perform this task. With this we covered the modality of gesture movements.

Audio task

An audible word was generated by the speech synthesiser for the third task. The user solved this task by repeating the word aloud. Both stimulus and response require auditive mental resources.

Search task

A clip was presented to the user in the fourth task. This clip contained several subsequent frames that were significantly different than the rest. The differences were the not described beforehand. Users were instructed to use a slider to search for the position of these frames. Unlike the tasks

described before, this task requires more interaction cycles to complete and puts a higher strain to the users' attentional resource.

4.4.2 Procedure

After introducing the MediaSpace and all the hardware to the subject, we showed her different coloured icons (see Figure 4.5). We explained their meanings: the colour in-

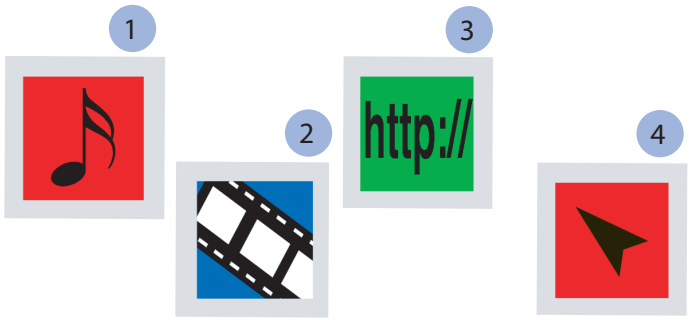


Figure 4.5: Examples of instruction icons: 1)word repetition on red display. 2)scene search on blue display. 3) URL entry on green display. 4) move cursor on red display.

dicates which display they should select and the type of icon indicates the primary task they should solve. Then we demonstrated how each primary task is solved on a single machine.

For each trial the subject applied a different output selection technique. Also each subject had a different order, in which she applied the methods. We permutated this order for each user to eliminate the learning effect.

Before a subject started a trial, we demonstrated the focus method to her and gave her enough time to get familiar with the technique.

Each trial consists of 16 cycles which have the following steps:

1. We instructed the subject by showing her a coloured icon on all displays.
2. The subject applied the assigned output focus technique according to the colour of the icon.
3. Then she chose the application she needed to solve the primary task she was instructed to solve.
4. The user worked on the primary task.

After each trial we gave the subject a questionnaire to fill in.

In each cycle, the subject needed to focus on a different output device. This justifies the number of displays we use. If we had used two displays the subject could have predicted which display would have been the next. She would not have waited for the instruction to appear. Then we would not have has a reliable starting point to measure the time needed to complete the focus task.

4.4.3 The Wizard

Our experiment settings also require a wizard, who has three tasks:

1. He controls the input management of the room, emulates focus technique by observing the subject's behaviour and then changes the room's setting accordingly.
2. When unexpected behaviour occurs, he makes notes about it by adding comments to the log file.
3. He also determines whether the subject has completed a primary task. When the primary task is done, the wizard shows the subject another instruction icon and starts measuring the focus time.

A GUI application assists the wizard in these tasks (see Figure 4.6). The wizard can manage the input by moving the

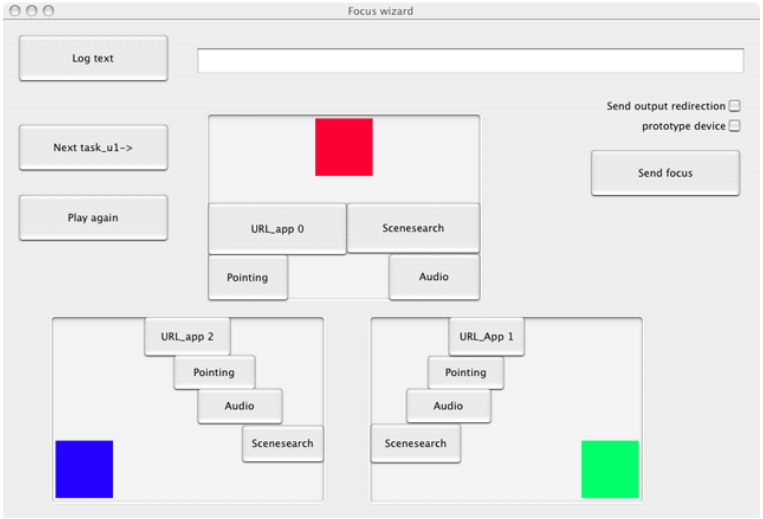


Figure 4.6: Wizard's interface to control event redirection.

mouse. When the escape key is pressed, a new instruction screen appear on all displays; a timer starts to measure the output focus time.

4.4.4 Quantitative metrics

Definition:
Primary task time

PRIMARY TASK TIME:
The time a subject needs to complete the primary task. Its measurement starts when the target application is selected.

Definition:
Output focus time

OUTPUT FOCUS TIME:
An instruction icon informs the user which output device should be used. The output focus time defines the interval between showing this icon and the selection of this output device, see Figure 4.7.

APPLICATION FOCUS TIME:
An instruction icon informs the user which application to use on an output device. The application focus time defines the time between output device selection and the selection of target application, see Figure 4.7.

Definition: *Application focus time*

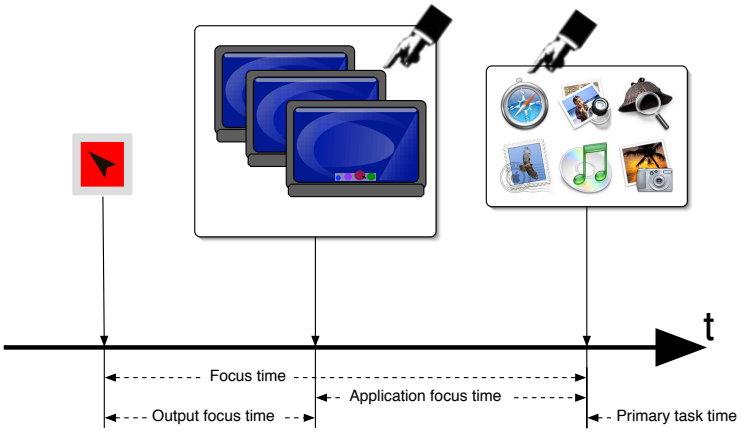


Figure 4.7: Output focus time is the parameter that we will measure.

We are interested in the time that a focus technique takes, so the primary task time is not measured in our settings.

With the procedure described above we start the measurement upon showing the icon. This also includes the time a user needs to interpret the icon, but since the icons are self-explanatory and easy to understand, we claim that the time for interpretation is constant throughout all trials and subjects.

However, the time that we measured can only be used to compare focus techniques. It does not characterise the performance of each technique. It includes time to recognise, indicate, and select the target display.

Chapter 5

Preliminary testing

In this preliminary test we compare following output focus techniques:

- Pointing with a laser pointer
- Pointing with a finger
- Pointing with an input device
- Focus-by-Gaze
- Touch-to-Focus

With this experiment we would also like to uncover errors in our testbed, so that we can refine our method incrementally.

5.1 Focus technique

The three different pointing techniques differ in the way how they specify a virtual ray.

For the first technique, we used a laser pointer to send a ray and a solar cell to sense this ray. As described in section 3.7, it is hard to absolutely point to an area with the laser

pointer because of the hand jitter problem (also see [Myers et al., 2002]). We try to avoid this problem by explicitly instructing the subjects to sweep across the solar cell as if someone would use a pen to check a checkbox.

There are also different ways to specify a virtual ray using the finger: we can consider the ray between the eyes and the tip of the index finger, we can use the index finger itself to cast a ray, or we can extend the arm's direction. For our experiment, we choose the first option.

To keep the comparison fair, we emulate all techniques with the Focus Wizard's GUI, i.e., the input management is triggered by the wizard.

Wizard controlled techniques
Techniques that are not yet fully implemented can be emulated with the wizard's help. Beside controlling the task flow, the Wizard also manages the input redirection for them. The GUI that the Wizard used is described in section 4.4.3.

System's feedback
As stated in section 3.4.4, we can provide feedback in different modalities. In this experiment we use visual feedback to inform the subject which display she is using: when the subject selects a display, the system invokes the Exposé function immediately. All application windows will then be scaled and positioned such that no window is overlapped. On the one hand this serves as a feedback of the selection step, on the other hand this also prepares users for application focus.

5.2 Procedure

We proceed in the manner as described in 4.4.2: a coloured icon was shown on the screen. The subject has to choose a display according to the colour. Then she choose an application to solve the primary task.

After each the use of each technique we gave them ques-

tionnaires to fill in. We also use informal interviews to gather qualitative results.

However, we do not measure the time needed for each focus method, because the wizard's reaction time would have been included in the output focus time, making this variable improper to characterise the technique.

Group plan
12 computer science students (8 females and 4 males) took part in this round of testing. Their age ranged from 23 to 28. All of them had previous experience with graphical user interfaces, but none of them had ever worked in an Interactive Workspace. They reported that they had used at most two computers simultaneously. Their participation was a requirement to get their credits for the course they have taken at our chair.

We also randomised the order of focus methods to minimise learning effect.

5.3 Results and Discussion

Pointing with Laser Pointer
Although we explicitly instructed the subjects to sweep across the solar cell, most of them used the direct pointing technique instead. The laser pointer's afforance for direct pointing seems to be very strong. Now that the laser pointer is used for direct pointing, some users found its use intuitive. However, many users reported that it was unclear why they should point to the solar cell. They prefer pointing to the display directly instead. Those who disliked this technique stated that they found the use of the laser pointer as a focus device disturbing. They stated that the functionality of the laser pointer is limited because it was only used for selecting displays. For them, it did not "pay off" to switch between the laser pointer and another input device.

Point to screen

Although this technique has a laser spot as visual feedback, it was not fully reliable. There were a few users who

lost track of the laser spot after they made a sudden hand movement. The same problem occurred, when some users pointed the device to a non-reflective surface such as the window panes in the room.

Cursor

Some users found it confusing to have a laser spot and a cursor on the display at the same time. This problem can be solved by WorldCusor's approach: they synchronise the laser spot with the virtual cursor on the display [Wilson and Pham, 2003].

We did not explicitly instruct the subjects to use a particular hand for a device. Most of them held the gyro mouse with the right hand. They put down the mouse and picked up the laser pointer with their right hand when they needed the laser pointer. Only one subject used both of her hands. With her left hand she held the laser pointer; with her right hand she used the gyro mouse. However, we observed that she had great difficulties in combining the usage of the two devices. She often moved the false device. The fact that the laser pointer is an absolute pointing device whereas the gyro mouse is a relative pointing device even confused her more. An explanation of this problem can be found in [Wickens, 2002]. According to attentional theory, if users use the same modality for the same output channel (in this case both hands produce manual responses), both tasks will compete for the same cognitive resources which leads to an interference of both actions.

Wicken's cube

Focus-by-Gaze

With this technique users can select a display by looking at it. Since the technique is controlled by a wizard, we cannot replicate the Midas touch effect. The Midas touch problem refers to the fact that eyes are always active, thus the challenge of many eye-input based interaction is to implement a suitable clutch to control.

Midas touch

In general, users stated that they like this method. However, some found the time between looking at the device and showing the feedback disturbing. We conclude that this technique is not well represented by a wizard-based version. In reality this technique would perform much faster. What we can take from this test is that users found this technique natural and intuitive, but needed an explicit

Not well-represented by wizard

action to trigger the focus change.

Pointing with Finger

With this technique users indicate a direction by pointing with a finger. Users would like to explicitly request for focus transition similar to the Focus by Gaze technique. We can use a shoot gesture or a voice command to implement that. All users used their dominant hand to point to a display. In general they found this technique rather awkward.

Awkward, also need selection

Pointing with input device

The quality of this technique depends strongly on the characteristics of the input device. Whereas the gyro mouse can afford a direction, the subjects found pointing with a keyboard non-intuitive. We believe that the posture of hand, when using the device, is important for the quality of pointing. Other physical attributes such as weight and shape also determine whether a device is applicable for pointing. Remote controls, mobile phones, and pens are such devices.

Which input device?

Pointing with the gyro mouse had the highest grading among all other techniques. However, when subjects compared this technique with the technique that uses a laser pointer, many of them found that it would improve the technique if there was a feedback which showed where they were pointing at.

Where am I? Feedback?

Touch-to-Focus

At the beginning of the tests we expected that this technique would have the worst grading. However, the average grade of this technique was higher than "pointing with finger" and "focus by gaze". Some subjects mentioned that they found walking across the room neither troublesome nor exhausting. The reason might be the application focus technique. In other techniques, when users selected a display, they use the gyro mouse to select an application. With Touch-to-Focus they could touch the window on the plasma display directly. The gain of using this technique seems to be greater than the cost of travelling across the room.

Touchscreen support

General comments

Some users stated that they had difficulties in working with the plasma display, because the visual elements were too

small: sometimes they lost the cursor and had to wiggle the mouse to find out its position. The title bar and address bar of a browser were also hard for them to read.

5.4 Method refinement

Visual feedback is not appropriate in our settings as displays may be out of users' visual field. Some users did not notice that a display had been falsely focused. Thus we use the speech recogniser to provide audio feedback in the next round of tests.

Trial took too long

The primary task was too complex for the subjects. The introduction phase took about 30 minutes in average, each trial took further 15 to 20 minutes to be accomplished. Subjects expected the experiments to last 30 to 60 minutes. As a consequence, we only applied the URL-entry tasks and the golf game in the following experiments.

Questionnaire and database

Evaluation using paper-based questionnaires costs a lot of time and effort to postprocess. Therefore we created a digital questionnaire that store the data in a database. The questions are defined in a XML-file. The experimenter can customise the questions and their types by modifying this definition file. Also this program supports open questions and questions that need ratings. The labels and scale size of a rating is defined in the same file. Also we can order the questions hierarchically.

Zooming

Because some visual elements are hard to see from a distance, we offer the subject two function keys to access Mac OS X built-in magnifier for impaired people. The display becomes a lens and the subject can move the lens using the mouse.

From Wizard's point of view

In general, when a subject points with an input device, it is hard for the wizard to see whether the subject has hit the target or not. In future experiments, we should consider attaching a camera to the pointing device.

Chapter 6

Comparison of Focus Techniques

In the following experiment, we are interested in the performance of four physical focus techniques.

- Dedicated-Keys (C1)
- Speech commands (C2)
- Pointing with a laser pointer (C3)
- Touch-to-Focus (C4)

6.1 Focus Techniques

We choose the laser pointer to represent pointing techniques, because it was the only one that can be operated without a wizard.

Touch-to-Focus is considered as a point of comparison between distant focus techniques and focus techniques that require users to walk to the device.

6.1.1 Dedicated-Keys

We use three physical buttons to represent the displays. The mapping between displays and keys is shown in table 6.1.

F1 → Red display
F2 → Green display
F3 → Blue display

Table 6.1: Mapping between keys and displays

If the Patch Panel senses that one of the keys is pressed, it emits a focus event to reconfigure itself. To make the mapping clear to the user, we coloured the physical keys as well. We can consider each button as a physical representation of a display, therefore this technique can be seen as an instance of ID-based techniques.

6.1.2 Speech commands

The Mac OS X operating system is shipped with a speech recogniser that does not need training. The iStuff package includes a proxy that mediates between theEvent Heap and the speech recogniser. When the subject says the command: ”red”, ”green”, or ”blue”, the proxy posts an event which the Patch Panel translates into a focus event. This focus event in turn changes the state of the Patch Panel. On each machine that is connected to the displays we run a speech recognition program. With these settings, we hope the subject can control input management from anywhere in the room.

However, some pretests showed that the sound quality we captured this way was not good enough for the system to recognise. This is one of our observation: when the system failed to recognise a speech command, the user walked closer to the microphone, attempting to improve the audio quality. If it failed again, she walked one step closer and so on. In the end, the whole interaction became the same as

the Touch-to-Focus technique. So as long as speech recognition does not work flawlessly, considering the location of the capture device seems to be important.

To work around this problem we tell the user to wear a head mounted microphone for the tests (see Figure 4.3).

6.1.3 Pointing gesture

We use a laser pointer to afford a directional ray. To sense the laser spot, one can apply an array of photo transistors or a solar cell. Our implementation uses the latter solution, therefore we attach a solar cell sensor to the upper-left corner of each plasma display (see Figure 6.1). If the laser

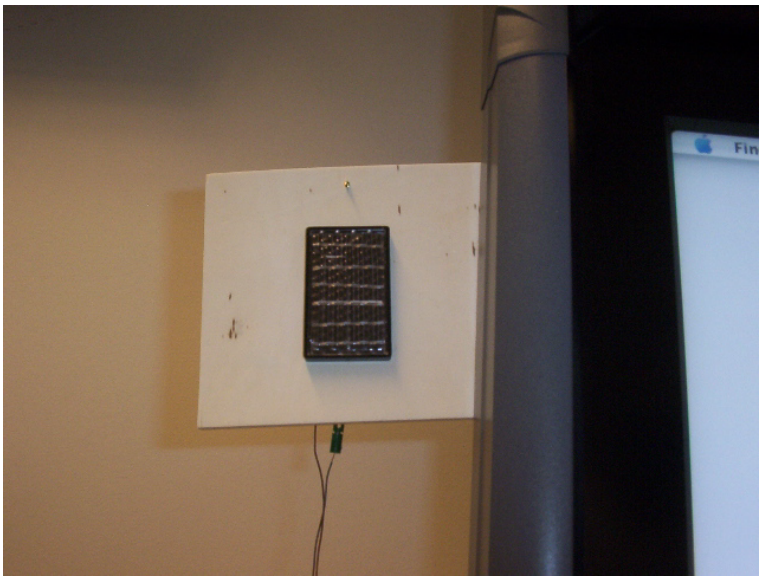


Figure 6.1: Solar cell attached to the upper-left corner of a plasma display.

spot hits the solar cell, the cell generates a small amount of voltage; the amplitude of this signal is then magnified by an operation amplifier. Figure 6.2 shows the circuit we applied to amplify the signal.

The amount of voltage is monitored by an I/O interface

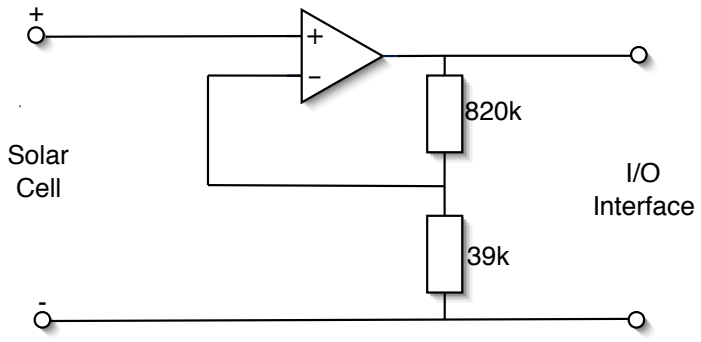


Figure 6.2: We magnify the voltage of a solar cell with an operation amplifier.

called TELEO module[1] .

If the voltage crosses a certain threshold, the TELEO-proxy emits an event that the Patch Panel translates into a focus event. Users do not need to hold the laser pointer steady, it suffices to sweep across the solar cell. Thus the problem of hand jitters should not corrupt this technique.

6.1.4 Touch-to-Focus

For our experiments we use plasma displays that are touch sensitive. When a user touches the instruction screen, it posts a focus event to the Event Heap.

The key characteristic of this technique is that users need to walk across the room. If no focus techniques are provided by the system, users must walk to a device and configure the settings manually. Thus the interaction without the support of focus techniques would take at least as long as touching the device.

Therefore we use Touch-to-Focus as an anchor to compare focus technique. Comparison with this technique reveals the benefits of a focus technique.

[1] http://www.makingthings.com

6.2 Procedure

6.2.1 Modification

With the experience we gathered in the prior tests, we decided to use audio feedback rather than visual feedback. When users select a display, speech synthesiser will voice the colour of display that was selected.

As described in the last chapter, we used a digital questionnaire to gather data. Furthermore, we offer users two hot keys to zoom in and zoom out of the screen.

Experimental design
We applied the same procedure as described in section 4.4.2. However, due to the tremendous time consumption, we adapted the modification suggested in section 5.4 and worked with two types of primary tasks instead of four. We have chosen to use the URL-entry task and the task that requires the subject to move the cursor in a Fitt's styled test. We chose these two because text inputs and moving cursor are common operations that are supported by every desktop machine.

Unlike the experiments in chapter 5, we also gathered quantitative data. The dependent variable in this experiments is the output focus time, see Figure 4.7.

Group plan
We conducted the following experiment with twelve computer science students (five females and seven males). Their age ranged from 22 to 29. We permutated the order of the techniques C1, C2, C3 to avoid learning effects. The subjects were distributed as in Table 6.2.

Hypothesis
We expect the Touch-to-Focus technique to be the slowest and most rejected technique. Further, we hypothesise that speech commands are the fastest among all techniques, because users do not have to switch devices to solve their primary tasks. We also expect that the pointing technique is slower than Dedicated-Keys as users have to switch their

Order of techniques	Number of subjects
C1, C2, C3, C4	2
C1, C3, C2, C4	2
C2, C1, C3, C4	2
C2, C3, C1, C4	2
C3, C1, C2, C4	2
C3, C2, C1, C4	2
Total subjects	12

Table 6.2: Group plan in chapter 6

device.

6.3 Results

Unlike our last experiment, we measured the output focus time for each trail. We gathered 182 data points.

To prepare the data for an ANOVA test we applied a logarithmic transformation on them. A one-way ANOVA test showed that interaction technique has a significant effect on output focus time ($F = 26.6$, $p < 0.001$). We applied Tukey's post-hoc pairwise comparisons on our data. It shows that speech interface is significantly slower than focus techniques using dedicated keys or laser pointer (both $p < 0.01$). Further, the Dedicated-Keys are significantly faster than pointing with the laser pointer or Touch-to-Focus ($p < 0.01$). With a lower certainty, the focus technique using the laser pointer is faster than Touch-to-Focus ($p < 0.05$).

6.4 Discussion

Speech commands
In this setting the speech interface had the worse performance. However, this was caused by the inaccuracy of the technique. Although this speech recognition system does

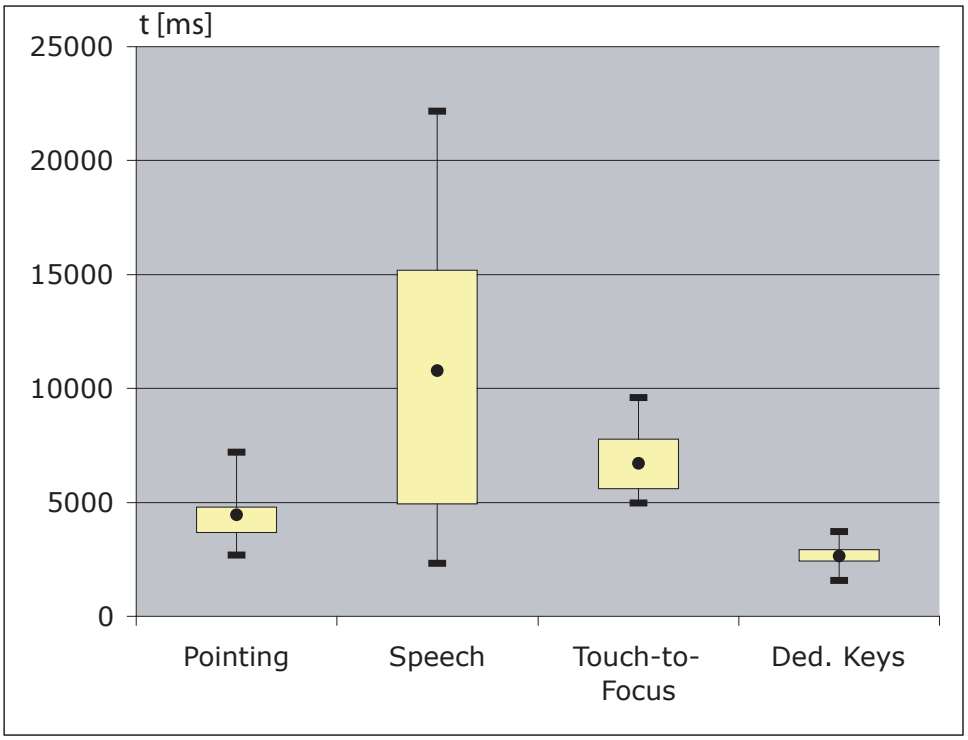


Figure 6.3: Output focus time of pointing with a laser pointer, speech commands, Touch-to-Focus, and Dedicated-Keys.

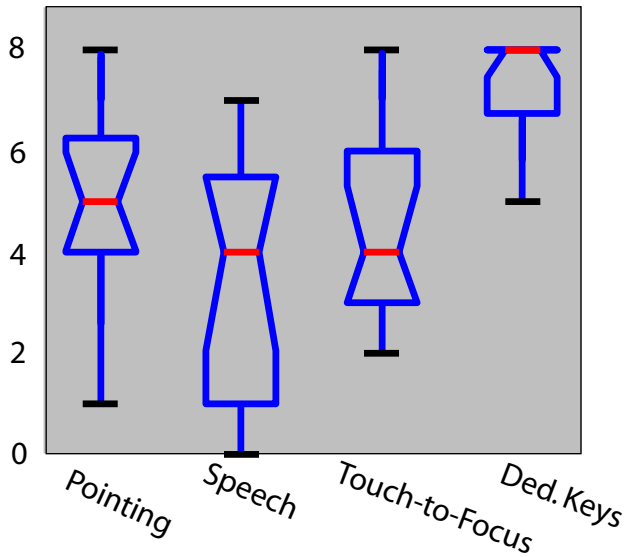


Figure 6.4: Grading of pointing with a laser pointer, speech commands, Touch-to-Focus, and Dedicated-Keys.

not need training, it were the users that needed to learn how to talk to the system.

Comparing to manual interaction, speech is more dependent on an individual's ability. Commands by users with a strong accent were hard to recognise. However, there were also users whose commands were recognised without any problem. This explains the large deviation of our data. For the latter group of users, their obstacle lay in the choice of words. While the system can easily recognise the commands "red" and "green", it has difficulties in recognising "blue".

An interesting effect that we observed was the users' behaviour when their command are not recognised. They normally changed the prosody, changing a command into a question.

Public space and training of speech interface

One might argue that speech interface can perform better if we use a system that needs training. However, an Interactive Workspace that we envision is a public environment. Training is usually not appropriate in such an environment. By doing the experiment with a modern commercial speech system without training, we give a realistic view into how a system might perform today. Also there are wizard of oz speech prototyping systems that provides mechanism to deliberately inject speech recognition error (see [Klemmer et al., 2000]). This points out that errors in speech interface should always be considered in the interface design.

Dedicated-Keys

The winner

According to the test we discussed above, Dedicated-Keys seems to be the fastest technique. This suggests a system control such as in a TV studio's control room, where the director presses keys to switch among cameras.

Pointing with a laser pointer

Fitt's law analysis

We see that pointing with a laser is not as fast as using keys. The reason for this is probably the small target size that a solar cell offers. The width of the solar cells we used is 4.5 cm with a height of 7 cm. We have also measured the distance of the solar cells pairwise: 285 cm, 347 cm, 591 cm. Now we can apply the Fitt's law model to our data to estimate the time it might take, if we allow users to select a

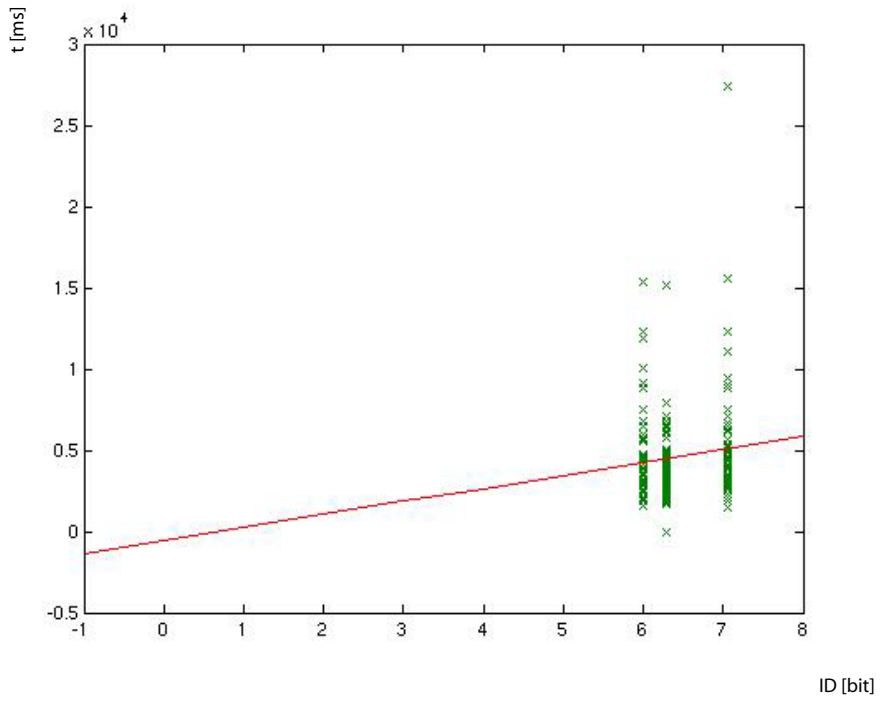


Figure 6.5: Data Regression

screen by pointing to the screen itself. We applied regression on our data (also see Figure 6.5 and Table 6.3): While

$$time = a + b * log_2(\frac{distance}{width} + 1)$$
$$a = -562.5965$$
$$b = 800.1923$$

Table 6.3: We applied the Fitt's law model to the pointing technique.

the solar cells we used have a width of 4.5 cm, the plasma displays had the size 100 cm x 63 cm. Supposed the distance of the targets are in 285 cm far away, according to the Fitt's law, if we use a solar cell as the target area we need

4244 milliseconds to accomplish the pointing task. If we use the complete display as the target area instead, the model predicts that it will take 994 milliseconds. This would event beat the Dedicated-Keys technique. However, we still need real experiment data to back this claim.

6.5 Method refinement

Many subjects complained that the testbed was slow and did not react to their input immediately. Due to this delay the instruction icons did not appear on the displays at the same time. This delay caused extra variance in the focus time measurement. The reason for this delay was the unbalanced load of the AppleScript proxy program. As stated in section 4.2, we heavily rely on the AppleScript language to generate voice feedbacks, activate windows, get the name of the focused window on a machine, and zoom in and zoom out.

Speed up primary tasks

To solve this problem, we redirect some of these tasks to other components. We created a program called NSWorkSpaceProxy which brings windows to the front and responds to requests for the active window's name. Users can communicate with the NSWorkSpaceProxy through events that are documented in appendix B.

Another measure we took to speed up the AppleScript proxy is to replace it by a version written in objective C language. The former version is written in Java. To process an AppleScript event it needs to access persistent memory and invoke commands from the terminal. This was the main reason for its suboptimal performance. Whereas the objective C version receives an AppleScript event and runs the script in the working memory.

We should notice that these measures do not have an impact on the output focus selection, but on the application focus time. With these modification we can measure application focus time, which was not possible in our previous settings. Nevertheless, it gave the subjects a more natural feeling and reduced the time they needed for each trial.

Chapter 7

Comparison with GUI-based technique

In chapter 6, we compared four different focus techniques with each other. Our result was that the Dedicated-Keys performed best in terms of output focus time. In this chapter, we want to compare this technique with successful research and commercial alternatives. One is the Vitrual-Path, which corresponds to the technique applied in the PointRight project (see chapter 2). The other is the World-in-Miniature technique, which uses an iconic map such as the ARIS project does (see chapter 2).

The techniques discussed in this chapter are:

- Dedicated-Keys (C_i)
- Virtual-Path (C_{ii})
- World-in-Miniature (C_{iii})

7.1 Focus techniques

Dedicated-Keys is the same technique we used for the experiment in chapter 6 and does not need to further description.

7.1.1 Virtual-Path

With the Virtual-Path technique, the focus follows the mouse of a user. Users can move their cursor across displays, giving them the feeling as if they are using a tiled large display.

The connections between displays are called *virtual paths*; they define the virtual topology of the displays. PointRight is the currently best-known system that implements this design. We will use a system that is a part of the iRoom operating system for our experiment in chapter 7.

The virtual layout of our settings is defined as follows: the right border of the red display is connected to the left border of the green display and the right border of the green display is connected to the left border of the blue display.

7.1.2 World-in-Miniature

World-in-miniature shows a miniaturised representation of the room for input management. We choose an iconic map similar to the ARIS project (see Figure 7.1). The GUI we provided for the experiments is based on the Wizard's GUI described in section 4.4.3. A key difference between them is that users need to click on an icon explicitly to change the focus. An important part for the interaction is move the map to the front. In the ARIS project users have to press on a special button in the title bar of a window. We have chosen to use the right click of the gyro mouse to show the map, which allows a faster interaction speed.

7.2 Procedure

We used the same experiment design as in chapter 6 but with the improvement as mentioned in section 6.5. An self-explanatory instruction is shown on the screen. The subject has to select the output device according to the colour of the icon.

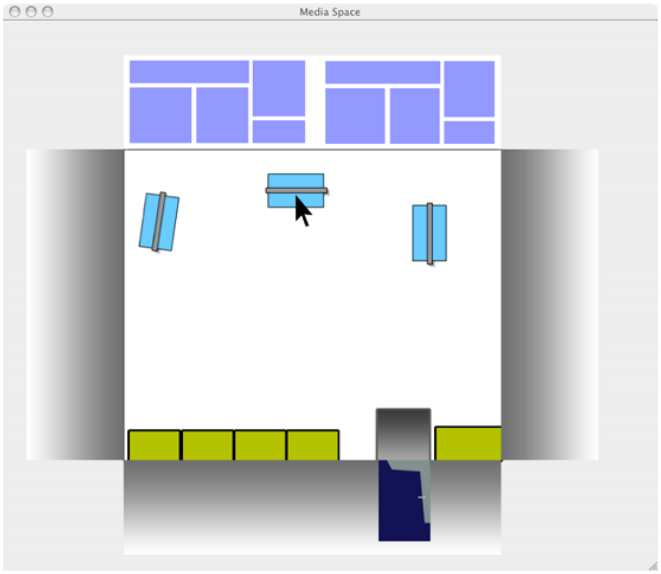


Figure 7.1: An iconic map of output devices in MediaSpace. When user clicks on an icon that represents an output device her inputs will be redirected to that output device.

Group plan

Three techniques were compared. Each technique is used for 16 trials. For this within-subject experiment we invited 12 computer science students to perform 48 trials. Their age varied from 22 to 27. As in the preceding experiments, they needed to attend an experiment to get their credits. We permutated the order of focus techniques to avoid learning effects. The subjects were distributed as in Table 7.1.

Order of techniques	Number of subjects
C_i, C_{ii}, C_{iii}	2
C_i, C_{iii}, C_{ii}	2
C_{ii}, C_i, C_{iii}	2
C_{ii}, C_{iii}, C_i	2
C_{iii}, C_i, C_{ii}	2
C_{iii}, C_{ii}, C_i	2
Total subjects	12

Table 7.1: Group plan in chapter 7

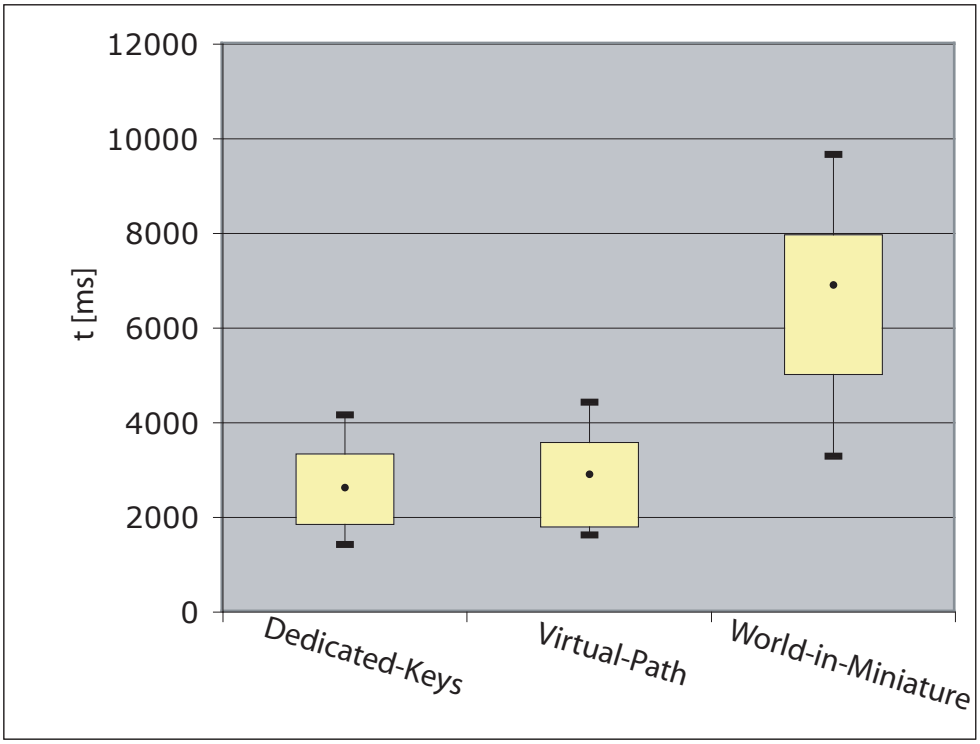


Figure 7.2: Output focus time of Dedicated-Keys, Virtual-Path, World-in-Miniature.

Hypothesis
We predicted that the Dedicated-Keys technique, being a physical focus technique, still has the best performance than the other two techniques.

7.3 Results

An ANOVA test on the raw data showed that the interaction technique has a significant effect on output focus time ($F = 62, p < 0.001$). We can see that the World-in-Miniature technique is significantly slower than the other two techniques (see Figure 7.2).

Surprisingly the performance of Virtual-Path is close to the Dedicated-Keys' performance. Figure 7.3 shows the grading of the techniques. Running a Kruskal Wallace test on

them showed no significant effect.

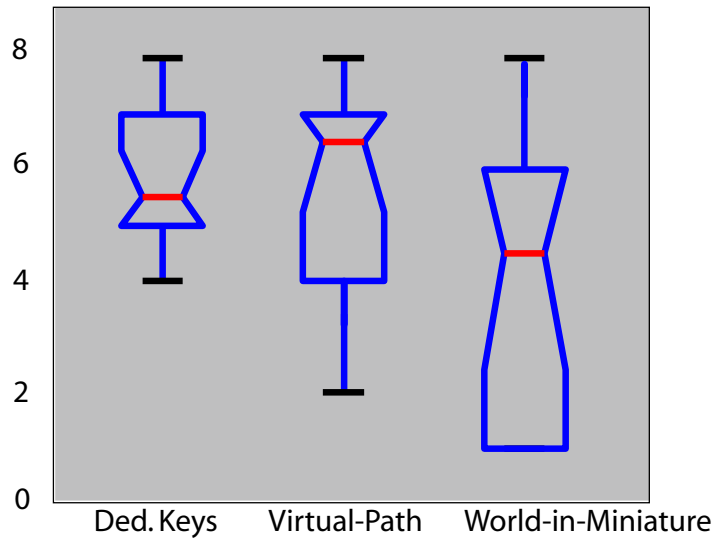


Figure 7.3: Users' grading of Dedicated-Keys, Virtual-Path, World-in-Miniature.

7.4 Discussion

World-in-Miniature
From our observation, the World-in-Miniature takes longer than the other two techniques because of the following reasons:

- Bringing up the map introduces an additional interaction cycle.
- Controlling the cursor from afar is difficult, because most users are not familiar with the gyro mouse.

A further phenomenon we observed in our experiment showed that the World-in-Miniature is a verb-noun interaction: a user was working on the display A and needed to focus the display B; she looked around and searched for the display B in the physical world; then she clicked on the

Verb-noun interaction

right button to show the map. However, she was still looking at display B and expected a response on that display, whereas the focus of her device still was at the display A.

In ARIS, this effect can not be observed, because their main target was to move applications from one display to another. The icon of an application serves as the noun in their context.

Despite the problem mentioned above the iconic map also has some advantages: an iconic map provides feedback that is clear and easy to understand. One subject found the iconic map beneficial to novices that are not familiar with the environment.

Virtual-Path
Although the Virtual-Path techniques is not the statistical significantly fastest technique, its performance is comparable to the Dedicated-Keys. However, the quality of this technique depends on the underlying topology. In our settings, all displays are aligned linearly in the physical world.

When not aligned?

The topology is defined in the same way.

This technique breaks down if displays are not arranged in a linear order. Users will then probably have difficulties in predicting the cursor's behaviour.

Dedicated-Keys
Although this technique is the fastest, the result is not transferable to all settings. This technique will probably break down, if the number of selectable device increases. It will become harder to find an appropriate mapping between devices and keys.

Number of devices

Chapter 8

Summary and Future work

"New media, like any chaotic system, are highly sensitive to initial conditions. Today's heuristical answers of the moment become tomorrow's permanent institutions of both law and expectation."

—John Perry Barlow

8.1 Summary

In this work, we investigated how input control can be managed in Interactive Workspaces. First, we analysed the concept of focus on traditional desktops, where the active window is the destination of input events.

Then we pointed out the key differences between a traditional desktop environment and an Interactive Workspace. Unlike traditional desktops, we have multiple users, devices, machines, and applications in an Interactive Workspace. These entities should be loosely coupled, but easy to combine.

To describe the dynamic aspect of an Interactive Workspace

we used the Focus Space to illustrate how users can dynamically redirect input and what information is needed for this task. A Focus Space consists of multiple tuples, each of which describes an active event route. Such a tuple denotes the following information:

- the ownership of the route,
- the input device that generates events,
- the application and the widget that process the event,
- the machine, on which this application is running,
- the output device that provides feedback.

An Interactive Workspace should offer focus techniques for each of these levels. There is a pattern that these focus techniques share, so we use the 3 steps model to capture this pattern. The three steps are: recognition, indication, and selection. We hope that future designers can use it for heuristic evaluation and to generate new focus techniques.

In the practical part of this thesis, we incrementally developed a testbed and defined a method to compare focus techniques with each other. In an environment with three plasma displays, users applied different focus techniques to select the display. We used the output focus time to measure the performance of the following techniques:

- pointing with a laser pointer,
- speech commands,
- Dedicated-Keys,
- Touch-to-Focus,
- Virtual-Path,
- World-in-Miniature.

In the first round of our experiments we compared speech commands, Dedicated-Keys, pointing with a laser pointer,

and Touch-to-Focus. According to our data Dedicated-Keys was the fastest among the four techniques in our settings.

Ded. keys fastest

With the Dedicated-Keys as the winner of the first round, we compared this technique with Virtual-Path and World-in-Miniature. Both Virtual-Path and Dedicated-Keys were significantly faster than the World-in-Miniature technique.

Ded. keys and Virtual-Path

Finally, during the development process we made several contribution to enrich the iStuff software package:

- A new and documented Patch Panel scripting Language.
- A more efficient proxy for AppleScript events.
- An AppleScript interface for the Event Heap.
- NSWorkspaceProxy to control activation of windows on local machines.

8.2 Future work

There are two main criticisms to the experiments conducted in this study:

- The several assumptions in chapter 4 we made might be too strong.
- The validity of our method depends on the primary tasks and the settings.

8.2.1 Simplified conditions

The assumptions we made to simplify the problem might be too strong. We can put more strain on the techniques by relaxing these assumptions stepwise. Thus we can find out their limits.

Multiple types of devices

For example, we can include multiple types of devices in our settings. It will also have a benefit for group meetings if we can control loudspeakers, window blinds, or lights. The challenge will be connecting these devices to the Event Heap.

Multiple users

In our experiments, each trial is completed by a single subject. We can extend the testbed for multiple users and study their behavioural patterns such as in [Biehl and Bailey, 2006].

When multiple users are working together, they might need to share devices. What will happen when a user hands over a device to another user who is working in a different context? How can she reconfigure the device?

Interaction using PDA

In this work, we searched for focus techniques that does not require additional personal devices to operate. If we drop this constraint, the most promising technique will be [Myers et al., 2002]'s Semantic Snarfing technique. With this technique users point a PDA to a device for coarse grained selection; a small area of the targeted display will be copied to the PDA. Users then can apply the PDA's touchscreen to interact with this area on a fine grained level.

8.2.2 Validity of methods

The user's primary tasks in our experiments was to use the keyboard to enter an URL and to move the cursor using a gyro mouse. We decided to use these tasks because they should emulate a natural working environment.

We also considered using primary tasks based on a scenario in which one or more users have to access different devices to solve a higher level task. However, we did not come up with a meaningful task that is both natural and requires multiple displays. Later in the process we found in [Biehl and Bailey, 2006] a promising approach. In their work, subjects were instructed to create a document collaboratively. Unfortunately, there was not enough time and resources to implement this testbed.

Choice of measure

We used the focus time to measure the performance of a technique. There are also other measures that we find useful to characterise a technique. For example, we can use the amount of head movements to quantify the effort needed for a technique.

Head movements

Another attribute is the learnability of a technique. If we can quantify this attriribute we can verify whether the World-in-Miniature technique is better for people with few experience with computers.

Learnability

8.2.3 Focus device prototype

As stated in section 5.3, users find it more intuitive to point at the screen directly rather than pointing at a sensor area. We attached an iSight webcam to a gyro mouse to prototype this technique(see Figure 8.1 and Figure 8.2).

Figure 8.1: Focus device prototype consists of a gyro mouse, a webcam, a laser pointer and a magneto mechanical switch.

Figure 8.2: Focus device prototype with a coil driven switch.

Furthermore, users complained that pointing with the gyro mouse did not provide enough feedback. So we also attach a laser pointer to our prototype.

Redundant multimodal feedback

Some users mentioned that they sometimes lost track of the laser spot. Thus a redundant feedback may keep the indication step intact. For this we attached a magneto mechanical switch on the device. On the one hand, the laser spot shows where the user is pointing to, on the other hand, the switch produces a "click"-sound whenever the user sweeps the device across the border of a display. The switch also produces a vibration that can be sensed by the user. The device delivers the same feedback when the user sweeps across the border again. Thus she always knows when she enters and when she leaves a selectable area.

For the selection step, a user just needs to click the button of the gyro mouse. We can also implement a redundant selection technique using speech command.

Currently this device is controlled by the wizard, but with the ARToolkit[1] we believe that we can implement a fully

[1] http://www.hitl.washington.edu/artoolkit/

functional device.

We would like to conduct further experiments to compare this prototype with the Virtual-Path technique. The main difference between them is the topology they use. The Virtual-Path technique uses a predefined topology in the indication step whereas our prototype uses the physical world.

Compared to [Myers et al., 2002]'s Semantic Snarfing both techniques use a pointing based coarse grained selection technique combined with a cursor based technique to do fine grained selections.

8.2.4 Coexisting focus techniques

On a desktop system, we use primarily the mouse to change our focus, but we can also use the keyboard to change the active window. The choice of focus techniques depends on the users' preferences and their current tasks.

This principle also applies in an Interactive Workspace. The users' current task is most relevant to the choice of focus technique. Moreover, users will be holding an input device to solve their primary task. Most likely, they will choose a focus method that does not interrupt the use of this device.

Regard input task

Thus we believe that an Interactive Workspace should offer multiple techniques to manage input.

Results from our experiments showed that both Dedicated-Keys and Virtual-Path techniques performed well. We should therefore combine these two techniques in our settings, so that users do not have to switch devices to change the focus. Minimising the homing time should be our primary goal when we design an Interactive Workspace or when we introduce new devices into it.

Minimise homing time

Appendix A

Patch Panel script

All patch panel scripts that use the following grammar must start with the line

//#v2.0

to specify the version of the scripting language. Comments are implemented leaning on common language such as C or Java: a single-line comment starts with **//** and ends with a linebreak; a multiple-line comment block starts with **/*** and ends with ***/**. Grammar:

script	⟸	head-line { body-line }
head-line	⟸	**group** groupname semicolon
body-line	⟸	event-definition | state-definition | variable-definition
event-definition	⟸	**event** event-name **type** event-type-nameevent-block
event-block	⟸	brace-open {field-definition} brace-close
field-definition	⟸	field-lhs **=** field-rhs semicolon
field-lhs	⟸	field-type field-name
field-rhs	⟸	expression [field-template-definition]
field-template-definition	⟸	**,** (**FORMAL** | **VIRTUAL**)
expression	⟸	..tbd
conditional-expression	⟸	constant-expression..tbd
constant-expression	⟸	string defining .... tbd
variable-definition	⟸	(variable-lhs **=** expression | **timer** timer-name) semicolon
variable-lhs	⟸	variable-type variable-name
state-definition	⟸	[**initial**] **state** state-name state-block
state-block	⟸	brace-open {inner-block-definition} brace-close
inner-block-definition	⟸	event-definition | variable-definition | action-trigger-definition
action-trigger-definition	⟸	**on** (**enter** | event-name) action-trigger-block
action-trigger-block	⟸	brace-open {statement} brace-close
statement	⟸	send-statement | goto-statement | cancel-statement | set-statement | switch-statement | if-else-statement | variable-assignment | compound-statement

Table A.1: Grammar of Patch Panel script

send-statement	$\Longleftarrow$	**send** event-name semicolon
goto-statement	$\Longleftarrow$	**goto** state-name semicolon
cancel-statement	$\Longleftarrow$	**cancel** timer-name semicolon
set-statement	$\Longleftarrow$	**setTimer** timer-name (integer | variable-name) semicolon
switch-statement	$\Longleftarrow$	**switch** variable-type parentheses-open expression parentheses-close switch-block
switch-block	$\Longleftarrow$	brace-open {case-group} cases-with-no-behaviour brace-close
switch-label	$\Longleftarrow$	(**case** constant-expression | **case** variable-name | **case** range | **default**) colon
cases-with-no-behaviour	$\Longleftarrow$	{switch-label}
case-group	$\Longleftarrow$	switch-label {switch-label} brace-open {statement} brace-close
if-else-statement	$\Longleftarrow$	**if** condition then-block [else-part]
condition	$\Longleftarrow$	parentheses-open conditional-expression parentheses-close
then-block	$\Longleftarrow$	compound-statement
else-part	$\Longleftarrow$	**else** (if-else-statement | else-block)
else-block	$\Longleftarrow$	compound-statement
variable-assignment	$\Longleftarrow$	variable-name **=** expression semicolon
compound-statement	$\Longleftarrow$	brace-open {statement} brace-close

Table A.2: Statement in Patch Panel script

semicolon	⟸	**;**
semicolon	⟸	**:**
brace-open	⟸	{
brace-close	⟸	}
parentheses-open	⟸	**(**
parentheses-close	⟸	**)**
or-token	⟸	||
groupname	⟸	arbitrary string
event-name	⟸	arbitrary string
event-type-name	⟸	arbitrary string |
		"arbitrary string containing
		reserved keywords or space**"**
field-name	⟸	arbitrary string
field-type	⟸	string | int | float | double | long
variable-name	⟸	arbitrary string |
		in.arbitrary string |
		out.arbitrary string |
		global.arbitrary string
variable-type	⟸	field-type
state-name	⟸	arbitrary string
range	⟸	[numerical-literal , numerical-literal] |
		[numerical-literal , numerical-literal} |
		{ numerical-literal , numerical-literal] |
		{ numerical-literal , numerical-literal}

Table A.3: Terminal symbols in Patch Panel script

expression	$\Longleftarrow$	ternary-expression
conditional-expression	$\Longleftarrow$	conditional-or-expression
constant-expression	$\Longleftarrow$	literal
boolean-literal	$\Longleftarrow$	true | false
numerical-literal	$\Longleftarrow$	**[-]** integer-literal | **[-]** floating-point-literal
literal	$\Longleftarrow$	numerical-literal | boolean-literal | string-literal
primary	$\Longleftarrow$	literal | parentheses-open expression parentheses-close | identifier
unary-expression	$\Longleftarrow$	**!** unary-expression | primary | **cos** unary-expression | **sin** unary-expression | **tan** unary-expression | **acos** unary-expression | **asin** unary-expression | **atan** unary-expression | **sqrt** unary-expression | **exp** unary-expression | **log** unary-expression | **floor** unary-expression | **ceil** unary-expression | **abs** unary-expression
comparative-expression	$\Longleftarrow$	unary-expression | **min(** comparative-expression **,** unary-expression **)** | **max(** comparative-expression **,** unary-expression **)**
multiplicative-expression	$\Longleftarrow$	comparative-expression | multiplicative-expression $*$ unary-expression | multiplicative-expression **/** unary-expression | multiplicative-expression **%** unary-expression | multiplicative-expression $\wedge$ unary-expression
additive-expression	$\Longleftarrow$	multiplicative-expression | additive-expression **+** multiplicative-expression | additive-expression **-** multiplicative-expression

Table A.4: Experssion in Patch Panel script (i/ii)

relational-expression	$\Longleftarrow$	additive-expression |
		relational-expression < additive-expression |
		relational-expression > additive-expression |
		relational-expression <= additive-expression |
		relational-expression >= additive-expression
equality-expression	$\Longleftarrow$	relational-expression |
		equality-expression == relational-expression |
		equality-expression **!**= relational-expression
conditional-and-expression	$\Longleftarrow$	equality-expression |
		conditional-and-expression **&&** equality-expression
conditional-or-expression	$\Longleftarrow$	conditional-and-expression |
		conditional-or-expression
		or-token
		conditional-and-expression
ternary-expression	$\Longleftarrow$	conditional-or-expression |
		ternary-expression **?**
		ternary-case-true-expression **:** ternary-case-false-expression
ternary-case-true-expression	$\Longleftarrow$	conditional-or-expression
ternary-case-false-expression	$\Longleftarrow$	conditional-or-expression

Table A.5: Experssion in Patch Panel script (ii/ii)

Appendix B

NSWorkspace proxy

Event Field	Comments	Required
EventType	**GWM_EVENT**	always
gwmEventTarget	Name of the machine this event is targeted to. If not set, this event holds for all machines.	optional
gwmEventType	**app_forward** or **app_backward**: toggles the active window of the machine specified in the field [gwmEventTarget].	always

Table B.1: Toggle active window event

Event Field	Comments	Required
EventType	**GWM_EVENT**	always
gwmEventTarget	Name of the machine this event is targeted to. If not set, this event holds for all machines.	optional
gwmEventType	**app_activate**: activates the application specified by [appName] on the machine specified by the field [gwmEventTarget].	always
appName	Name of the application to activate. If the application is not running, it will be started. The according window will get the focus.	yes

Table B.2: Activate window event

Event Field	Comments	Required
EventType	**GWM_EVENT**	always
gwmEventTarget	Name of the machine this event is targeted to. If not set, this event holds for all machines.	optional
gwmEventType	**app_getAll**: returns an event containing all running application in the environment.	always

Table B.3: Activate window event

Event Field	Comments	Required
EventType	**GWM_EVENT**	always
gwmEventTarget	Name of the machine this event is targeted to. If not set, this event holds for all machines.	optional
gwmEventType	**app_getActive**: returns an event that contains the name of the active window.	always

Table B.4: Get activate window event

Appendix C

Questionnaire

Date of test	
Time of test	
Technique	

1. Selection of display

1.1 Switching among the screens is

easy to learn difficult to learn
1 2 3 4 5 6 7 8 9

1.2

intuitive non-intuitive
1 2 3 4 5 6 7 8 9

1.3

very easy very difficult
1 2 3 4 5 6 7 8 9

1.4

comfortable very uncomfortable
1 2 3 4 5 6 7 8 9

1.5 Switching among the screens makes me tired.

strongly agree strongly disagree
1 2 3 4 5 6 7 8 9

1.6 Time needed to switch to a screen is

unacceptable acceptable
1 2 3 4 5 6 7 8 9

1.7 You can distinguish your selection easily.

strongly agree strongly adequate
1 2 3 4 5 6 7 8 9

2. Application selection

2.1 Switching among the applications is

easy to learn difficult to learn
1 2 3 4 5 6 7 8 9

2.2

intuitive non-intuitive
1 2 3 4 5 6 7 8 9

2.3

very easy very difficult
1 2 3 4 5 6 7 8 9

2.4

comfortable very uncomfortable
1 2 3 4 5 6 7 8 9

2.5 Switching among the applications makes me tired.

strongly agree strongly disagree
1 2 3 4 5 6 7 8 9

2.6 Time needed to switch to an application is

unacceptable acceptable
1 2 3 4 5 6 7 8 9

2.7 You can distinguish your active application from others easily.

strongly agree strongly disagree
1 2 3 4 5 6 7 8 9

3. Overall interaction

3.1 You can switch between the tasks smoothly.

strongly agree strongly disagree
1 2 3 4 5 6 7 8 9

3.2 Switching to another screen cost you a lot of time.

strongly agree strongly disagree
1 2 3 4 5 6 7 8 9

3.3 It is clear what you are controlling.

Always Never
1 2 3 4 5 6 7 8 9

1. Techniques grading

1.1 Using a laser pointer to select display

Terrible Wonderful
1 2 3 4 5 6 7 8 9

1.2 Pointing with finger

Terrible Wonderful
1 2 3 4 5 6 7 8 9

1.3 Touch to select

Terrible Wonderful
1 2 3 4 5 6 7 8 9

1.4 Selection by gaze

Terrible Wonderful
1 2 3 4 5 6 7 8 9

1.5 Pointing with the gyro mouse

Terrible Wonderful
1 2 3 4 5 6 7 8 9 N/A

1.6 Pointing with the keyboard

Terrible Wonderful
1 2 3 4 5 6 7 8 9 N/A

2. Experience

2.1 The maximum number of computers with which you have worked with simultaneously:

2.2 The number of mobile devices you use at home:

2.3 Your age:

2.4 How familiar are you with Mac OS X? (Shortkeys, Exposé)

Novice Expert
1 2 3 4 5 6 7 8 9

3. Your comments:

Bibliography

Inc. Apple Computers. Mac OS X Exposé, 2003. URL `www.apple.com/macosx/features/expose`.

Rafael Ballagas, Meredith Ringel, Maureen Stone, and Jan Borchers. iStuff: a Physical User Interface Toolkit for Ubiquitous Computing Environments. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 537–544. ACM Press, 2003.

Patrick Baudisch and Carl Gutwin. Multiblending: displaying overlapping windows simultaneously without the drawbacks of alpha blending. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 367–374. ACM Press, 2004.

Jacob T. Biehl and Brain P. Bailey. Improving Interfaces for Managing Applications in Multiple-Device Environments. In *CHI '06: Proceedings of the SIGCHI conference on Human factors in computing systems*, 2006.

Jacob T. Biehl and Brian P. Bailey. ARIS: an interface for application relocation in an interactive space. In *GI '04: Proceedings of the 2004 conference on Graphics interface*, pages 107–116. Canadian Human-Computer Communications Society, 2004.

Richard A. Bolt. "put-that-there": Voice and gesture at the graphics interface. In *SIGGRAPH '80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 262–270. ACM Press, 1980.

Jan Borchers. The Aachen MediaSpace: Multiple displays in collaborative interactive environments. In *Workskop: Information Visualization and Interaction Techniques for Collaboration across Multiple Displays*, 2006.

Brockhaus. *Der Brockhaus-Computer und Informationstechnologie*. F.A. Brockhaus, 2002.

Rodent A. Brooks. The intelligent room project. In *Second International Cognitive Technology Conference (CT'97)*, 1997.

Barry Brumitt, Brian Meyers, John Krumm, Amanda Kern, and Steven A. Shafer. EasyLiving: Technologies for Intelligent Environments. In *HUC '00: Proceedings of the 2nd international symposium on Handheld and Ubiquitous Computing*, pages 12–29. Springer-Verlag, 2000.

Mark H. Chignell and John A. Waterworth. Wimps and nerds: an extended view of the user interface. *SIGCHI Bull.*, 23(2):15–21, 1991.

David Fono and Roel Vertegaal. Eyewindows: Evaluation of Eye-Controlled Zooming Windows for Focus Selection. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 151–160. ACM Press, 2005.

Barbara J. Grosz. Focusing in Dialog. In *Proceedings of the 1978 workshop on Theoretical issues in natural language processing*, pages 96–103. Association for Computational Linguistics, 1978.

Alex Hioreanu. AHWM - Alex Hioreanu's Window Manager, 2002. URL `people.cs.uchicago.edu/~ahiorean/ahwm/`.

Charles L. Isbell, Olufisayo Omojokun Jr., and Jeffrey S. Pierce. From devices to tasks: automatic task prediction for personalized appliance control. *Personal Ubiquitous Comput.*, 8(3-4):146–153, 2004.

Brad Johanson, Armando Fox, and Terry Winograd. The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. *IEEE Pervasive Computing*, 1 (2):67–74, 2002a.

Brad Johanson, Greg Hutchins, Terry Winograd, and Maureen Stone. PointRight: Experience with Flexible Input Redirection in Interactive Workspaces. In *UIST '02: Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 227–234. ACM Press, 2002b.

Khomkrit Kaowthumrong, John Lebsack, and Richard Han. Automated Selection of the Active Device in Interactive Multi-Device Smart Spaces. In *In Workshop at UbiComp'02: Supporting Spontaneous Interaction in Ubiquitous Computing Settings*, 2002.

Manpreet Kaur, Marilyn Tremaine, Ning Huang, Joseph Wilder, Zoran Gacovski, Frans Flippo, and Chandra Sekhar Mantravadi. Where is "it"? event synchronization in gaze-speech input systems. In *ICMI '03: Proceedings of the 5th international conference on Multimodal interfaces*, pages 151–158. ACM Press, 2003.

Scott R. Klemmer, Anoop K. Sinha, Jack Chen, James A. Landay, Nadeem Aboobaker, and Annie Wang. Suede: a wizard of oz prototyping tool for speech user interfaces. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 1–10. ACM Press, 2000.

Lee Hoi Leong, Shinsuke Kobayashi, Noboru Koshizuka, and Ken Sakamura. CASIS: a context-aware speech interface system. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 231–238. ACM Press, 2005.

Brad A. Myers. The Pebbles Project: Using PCs and Handheld Computers Together. In *CHI '00: CHI '00 extended abstracts on Human factors in computing systems*, pages 14–15. ACM Press, 2000.

Brad A. Myers, Rishi Bhatnagar, Jeffrey Nichols, Choon Hong Peck, Dave Kong, Robert Miller, and A. Chris Long. Interacting at a distance: measuring the performance of laser pointers and other devices. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 33–40. ACM Press, 2002.

Miguel A. Nacenta, Dzmitry Aliakseyeu, Sriram Subramanian, and Carl Gutwin. A Comparison of Techniques for Multi-Display Reaching. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 371–380. ACM Press, 2005.

Jef Raskin. *The Humane Interface: New Directions for Designing Interactive Systems*. Addison-Wesley Professional, 2000.

Manuel Romàn, Christopher Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, and Klara Nahrstedt. Gaia: a middleware platform for active spaces. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(4): 65–67, 2002.

Ronald Rosenfeld, Dan Olsen, and Alex Rudnicky. Universal Speech Interfaces. *interactions*, 8(6):34–44, 2001.

Michael Siracusa, Louis-Philippe Morency, Kevin Wilson, John Fisher, and Trevor Darrell. A Multi-Modal Approach for Determining Speaker Location and fFocus. In *ICMI '03: Proceedings of the 5th international conference on Multimodal interfaces*, pages 77–80. ACM Press, 2003.

Hannah Slay and Bruce Thomas. Interaction and visualisation across multiple displays in ubiquitous computing environments. In *Afrigaph '06: Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 75–84, 2006.

Rainer Stiefelhagen, Jie Yang, and Alex Waibel. Modeling Focus of Attention for Meeting Indexing. In *MULTIMEDIA '99: Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 3–10. ACM Press, 1999.

Roel Vertegaal, Aadil Mamuji, Changuk Sohn, and Daniel Cheng. Media eyepliances: using eye tracking for remote control focus selection of appliances. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1861–1864. ACM Press, 2005.

Daniel Vogel and Ravin Balakrishnan. Distant Freehand Pointing and Clicking on Very Large, High Resolution Displays. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 33–42. ACM Press, 2005.

Grant Wallace, Peng Bi, Kai Li, and Otto Anshus. A multicursor x window manager supporting control room collaboration. Technical report, Princeton University, July 2004.

Rainer Wasinger and Antonio Krüger. Modality Preference - Learning from Users. In *Workshop on User Experience*

Design for Pervasive Computing (Experience) at Pervasive. Springer-Verlag, 2005.

Rainer Wasinger, Christian Kray, and Christoph Endres. Controlling multiple devices. In *Physical Interaction (PI03) Workshop on Real World User Interfaces at MobileHCI*, pages 60–63, 2003.

Mark Weiser. The Computer of the 21st Century. *Scientific American*, Sept 1991.

Christopher D. Wickens. Multiple resources and performance prediction. *THEOR. ISSUESIN ERGON. SCI.,* 3 (2), 2002.

Andrew Wilson and Hubert Pham. Pointing in Intelligent Environments with the WorldCursor. In *INTERACT*, 2003.

Andrew Wilson and Steven Shafer. XWand: UI for intelligent spaces. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 545–552. ACM Press, 2003.

Index

Typeset May 17, 2006