2012 | School voor Informatietechnologie
Kennistechnologie, Informatica, Wiskunde, ICT

DOCTORAATSPROEFSCHRIFT

Touch-based Interaction and Collaboration in Walk-up-and-use and Multi-user Environments

Proefschrift voorgelegd tot het behalen van de graad van doctor in wetenschappen, informatica, te verdedigen door:

Davy Vanacken

Promotor: prof. dr. Karin Coninx
Copromotor: prof. dr. Kris Luyten

D/2012/2451/46

Maastricht University

universiteit hasselt
KNOWLEDGE IN ACTION

Acknowledgments

After working one year as a researcher at the Expertise centre for Digital Media (EDM), a research institute of Hasselt University, I took on the challenge of doing my PhD in 2005, as a teaching assistant. I had the pleasure of getting to know and work with many people who not only provided knowledge, guidance and encouragement, but also a lot of friendship. Needless to say, I owe them a great deal of gratitude.

First and foremost, I would like to address special thanks to my advisor, prof. dr. Karin Coninx, for giving me the opportunity to work in the HCI group of EDM. Her valuable feedback and advise had a great influence on my dissertation, but also on how I developed as a researcher and teaching assistant over the years. She instilled in me a confident and professional attitude that will be valued throughout the rest of my career. Next, I want to thank my co-advisor, prof. dr. Kris Luyten. His never-ending drive and creativity was a source of inspiration, and had an important impact on the research presented in this dissertation.

I would also like to thank prof. dr. Eddy Flerackers, our managing director, and prof. dr. Frank Van Reeth, our deputy managing director, for giving me the opportunity to work at EDM, and for being part of my jury. Furthermore, I want to express my thanks to the other members of my jury, prof. dr. Johannes Schöning, prof. dr. Miguel Nacenta and prof. dr. Jan Borchers, for their valuable comments with regard to this dissertation. Thanks also to the chairman of my jury, prof. dr. Marc Gyssens.

Working at EDM would have been a lot less interesting without such great colleagues, and I want to thank all of them, not only for their direct or indirect contributions to my research, but also for all the fun we had together. In particular, I would like to say thanks to my co-authors, Maarten Cardinaels, Raf Ramakers, dr. Joan De Boeck, prof. dr. Alexandre Demeure and dr. Chris Raymaekers. I would also like to single out dr. Mieke Haesen and dr. Anas-

tasiia Beznosyk, for the pleasant collaboration and all the help during the last months of my PhD.

A big thanks to all the other colleagues I collaborated, travelled, lunched and assisted courses with over the years: Jo Vermeulen, Tom De Weyer, dr. Lode Vanacken, Sofie Notelaers, Stijn Agten, Jolien Schroyen, Heleen Van Loon, Karel Robert, Kris Gabriëls, Jan Schneider, Tim Dupont, Tom Haber, dr. Bert De Decker, Fredy Cuenca, dr. Jan Van den Bergh, Peter Vandoren, and many others. Finally, I want to thank Luc Adriaens for all the video recordings, and Ingrid Konings and Roger Claes for taking care of the administrative tasks and logistics.

On a more personal note, I would like to thank all of my friends and family for their enduring encouragement and friendship. I especially want to thank my parents, for giving me the opportunity to finish my studies and for the abundant comfort provided throughout these years. Last but not least, I want to thank my wife, Bieke, for all her patience, encouragement and love. You always made sure that there is more to life than work.

Abstract

Touch-based interfaces are becoming increasingly ubiquitous: they can be found on a variety of hardware platforms, ranging from mobile phones to large public displays, and they allow a wide variety of applications, from single-user casual games to multi-user tools that facilitate brainstorming. Although touch-based interaction is supposed to be intuitive and "natural", it nonetheless imposes specific requirements on the accessibility of a user interface. Due to a lack of common conventions and consistency across applications, gestures are often difficult to discover and learn. Furthermore, since most touch-sensitive hardware supports multi-touch input nowadays, it allows multiple people to interact simultaneously on a shared surface. This not only brings about new opportunities with regard to collaboration, but also new research challenges on how to support this kind of collaboration effectively.

In walk-up-and-use environments, accessibility of the user interface is of great importance, as the limited interaction time and need for immediate use of the system do not allow for much training or exploration. Therefore, we explore the concept of making touch-based interfaces in walk-up-and-use environments self-explanatory, for both single-user and multi-user settings. With TouchGhosts, we propose a help system that demonstrates interaction techniques to the users, for instance through visual "guides" such as animated virtual hands. The graphical nature of our approach allows a clear view on the synchronization of multiple inputs, which are typical for multi-touch interaction. User studies indicate that animated help allows users to quickly discover the available interaction possibilities, with a positive effect on collaboration, as users work together to learn the application.

When multiple users interact simultaneously in a highly collaborative setting, additional challenges emerge, especially if the environment can include both co-located and remote participants. Collaboration may cause conflicts and misconducts, for instance. Therefore, collaborative environments are in

need of floor control policies that resolve and prevent such problems gracefully, without interrupting the dynamic work flow. We apply a Focus+Roles approach to a digital meeting system, iConnect: the user's roles define a user's access rights and privileges during particular activities, while tracking the users' focus provides a means of handling problems associated with the typical lack of mutual awareness.

Continuing our research on collaborative systems, we investigate the requirements of a storyboarding tool to support the various disciplines in a multidisciplinary team. Storyboards are well suited to attain a common understanding during user-centered software design and development, because they allow each team member to contribute to the decision making process. To take full advantage of the different viewpoints and approaches that members of a multidisciplinary team bring to the table, we explore how such teams create storyboards through an observational study. Based on the lessons learned from this study, we formulate a set of requirements to inform the design of a tabletop tool for collaborative storyboarding.

Throughout our exploration of collaborative systems, we repeatedly encountered the need for identification of the different users around a shared surface. Multi-touch hardware can track multiple inputs simultaneously, but the majority of those systems are unable to associate contact points with specific users. Therefore, we present Carpus, a non-intrusive identification technique for mapping touches to their corresponding user by analyzing the back of the users' hands. This feature can improve a multi-user interface in a number of ways, for instance by customizing help on a per-user basis, or by tracking a user's activities to prevent conflicts or unequal contributions.

Another aspect to consider, is the design and evaluation of touch-based and multi-user interaction. We investigate how model-based design can facilitate the development process by modeling environments through the use of high-level diagrams. In this context, we discuss NiMMiT, a graphical notation for expressing and evaluating multimodal user interaction. Because NiMMiT is presently focused on single-user 3D virtual environments, we explore how NiMMiT can be extended beyond these boundaries by reflecting on its current limitations with regard to touch-based and multi-user interaction.

In summary, our main contributions include making touch-based interfaces self-explanatory in single-user and multi-user walk-up-and-use environments, and providing conflict-free multi-user environments that can non-intrusively identify the user behind each action, for instance to facilitate collaborative storyboarding. Furthermore, we present a model-based approach to design and evaluate multimodal interaction techniques.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 A brief history of touch-based interaction

I would like to start my dissertation with a quote from Ben Shneiderman, who somewhat prophetically stated the following in 1991 [Shneiderman 91]: *"I suspect we will be seeing touchscreens being used for more applications than ever before."* Today, we can safely say his prediction came true. When I began my PhD in 2005, as a teaching assistant, the technological landscape of touch-based interaction was still quite different from what it is today. The touch-sensitive hardware we had available in our research lab, was initially limited to a few PDAs and a drawing tablet. At present, we have a wide variety of other devices, such as smartphones, tablet computers, tabletops, workbenches, and whiteboards. Considering the widespread use of commercial hardware such

as satellite navigation devices, smartphones and tablets, we can even say that touch is a prevailing form of input nowadays. As the research presented in this dissertation covers a period of nearly seven years, it offers us an up-close view on this fast evolution in the field of touch-based interaction.

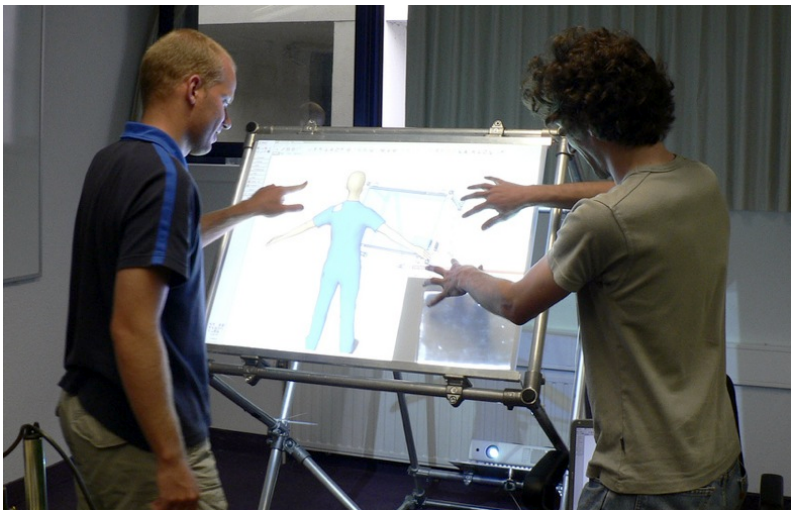
The history of touch-sensitive controls actually pre-dates the age of the personal computer [Buxton 12]. The concept of a touch screen was first introduced in the 1960s, in the context of air traffic control [Johnson 65, Orr 68], and gradually found its way to automated teller machines (ATMs), information kiosks, and personal digital assistants (PDAs). The hardware evolved from being able to detect only a single input point (i.e. single-touch input), to recognizing the presence of two or more contact points (i.e. multi-touch input), and even identifying physical objects that are put on the surface (i.e. tangible input). Likewise, touch-based user interfaces progressed from simply responding to discreet actions such as tapping with a stylus or finger, to more advanced gestural interfaces that support continuous and coordinated actions such as pinch and swipe gestures.

The year 2005 was an important landmark in the evolution of large multi-touch surfaces, as Jeff Han presented a fairly straightforward technique that enables robust multi-touch sensing through frustrated total internal reflection (FTIR), an inexpensive and scalable approach [Han 05]. Not only the build-it-yourself hardware caught people's attention, but also the very attractive multi-touch interaction techniques and applications that were demonstrated. However, it was the release of consumer devices such as the Apple iPhone and iPad that really drove the touch screen revolution we have seen in the last couple of years.

In our research lab, we started building large interactive surfaces around 2005. Our first prototype was a "multi-touch" table that in fact consisted of four single-touch panels, as depicted in Figure 1.1a. We gradually acquired and constructed various setups, such as the FTIR-based surface shown in Figure 1.1b. Our hardware setups were extensively used in numerous student and research projects, situated in domains such as education (e.g. IBBT GBO project *MuTable*), media production (e.g. IBBT ICON project *PISA*), media search (e.g. IWT SBO project *AMASS++*) and intelligent meeting environments (IBBT GBO project *iConnect*). These projects allowed our research lab to gain a lot of hands-on experiences regarding surface computing. Eventually, this knowhow led to the founding of TinkerTouch[1], a small company that specializes in developing highly interactive touch-based applications.

[1] `http://www.tinkertouch.com`

(a) Tabletop consisting of four capacitive single-touch panels.



(b) Setup based on frustrated total internal reflection.

Figure 1.1: Two of our earlier custom-built interactive surfaces.

1.2 Challenges in the field at large

Although touch-based interaction has been around for quite a while, it still provides numerous challenges for designers and developers to this day. Some of the well-known challenges [Shen 06, Ryall 06b, Müller-Tomfelde 10] relate to the orientation of content when people are sitting on all sides of a tabletop, the limited input precision of fingers compared to a computer mouse, the occlusion of the content beneath a user's fingers, accidental input by leaning on the surface, the inefficiency of virtual keyboards for text input, and unreachable or uncomfortable to work in regions in case of very large workspaces. In multi-user interaction, some of the prominent challenges revolve around sharing data, handling conflicts and privacy, integrating personal and shared devices, and the identification of users.

Over the years, these and other research challenges led to developments in various fields of touch-based and multi-user interaction (the references listed here are merely a few examples, and by no means an exhaustive overview):

- the development and evaluation of novel interaction techniques (e.g. interaction techniques to transfer objects across large distances on tabletops [Reetz 06, Voelker 11], techniques that improve control through separability of spatial manipulations [Nacenta 09], multi-touch graph interaction techniques [Schmidt 10c]),

- enhancements to the accessibility of user interfaces (e.g. using feedforward and feedback [Bau 08] or physical metaphors [Bragdon 10] to help users learn gestures, supporting the transition from implicit to explicit interaction [Vogel 04], designing for engagement [Jacucci 10]),

- improvements to the collaborative aspect of multi-user environments (e.g. cooperative gestures [Morris 06], virtual embodiments to improve awareness and coordination [Pinelle 08], coordination techniques to reduce conflicts arising from indirect input [Pinelle 09], interaction techniques for group support [Nacenta 10]),

- the exploration of various application domains (e.g. geographic information systems [von Zadow 10], composing and playing music [Lynch 11], gestural interaction in cars [Döring 11], touch-based input for elderly users [Wacharamanotham 11], child-computer interaction [Tse 11]),

- studies on behavioral effects (e.g. deployments in public spaces such as city centers [Peltonen 08] or museums [Hornecker 08a], territoriality in

collaborative workspaces [Scott 04], collaborative coupling around tabletops [Tang 06]),

- new advances in hardware setups (e.g. user authentication using mobile phones [Schöning 08], tangible controls [Weiss 09, Weiss 10] and tangible private spaces [Möllers 11], haptic feedback on tabletops [Marquardt 09, Weiss 11]),

- innovations with regard to the development process (e.g. pattern languages for interactive tabletops in collaborative workspaces [Remy 10], toolkits for haptic interactions [Ledo 12], engineering approaches for multi-touch interfaces [Luyten 10, Luyten 11]).

Detailing all the research on these topics is, however, beyond the scope of this work. We come back to these research fields in Section 1.4, to outline the scope of our work.

Although these developments happened on a variety of devices, we mainly focus on the larger interactive surfaces in this dissertation, such as multi-touch tabletops. Some of our research is nonetheless also applicable to other devices, as we will address in the subsequent chapters. There are several dimensions of large interactive surfaces to explore. First of all, there are various ways of interacting with such a surface: one person can use it individually, multiple persons can work in parallel without any real connection between them, or those persons can tightly cooperate with one another. Secondly, the interactive setup can be situated in a variety of settings, ranging from walk-up-and-use surroundings to work environments. Present-day settings that are becoming progressively more common include museums, exhibitions, hotels, restaurants, retail stores, meeting rooms, and so forth.

Each of those examples comes with its own characteristics and requirements. In a single-user environment, for example, users are on their own, with no immediate support or hindrance from others. In a multi-user environment, on the other hand, users can cooperate to accomplish a task, but actions of multiple people may conflict with one another, both intentionally and accidentally. In addition, different settings lead to different requirements. A walk-up-and-use system needs to be very accessible, for instance, since people will simply walk away if it takes too long to figure out the user interface, or if the user experience does not live up to their expectations. Of course, accessibility is also important in a work environment, but the focus will often be on productivity and effective collaboration.

As we (and others) explored the different dimensions throughout several

projects and studies, we encountered various challenges that had to be overcome. In the next section, we reflect on our experiences to identify the characteristics and requirements of different environments, and to pinpoint and clarify our research challenges.

1.3 Hands-on experiences and research challenges

In the following sections, we present some of the experiences with multi-touch setups in walk-up-and-use and collaborative environments that we gathered in the course of doing this PhD. The different topics are based on our practical experiences with a variety of projects, ranging from large-scale projects together with industrial partners to small student projects, as well as a number of user studies we conducted over the years. These hands-on experiences led us to a number of research challenges that are addressed in this dissertation. In this chapter, we primarily focus on the context in which the projects and studies were situated, which we exemplify through a few illustrative scenarios. More specific details on the various projects and studies will be provided in the upcoming chapters.

1.3.1 Walk-up-and-use environments

Walk-up-and-use environments differ from the other examples presented in this chapter in that it is necessary for the interface to be so self-explanatory that first-time users can use it effectively without any prior introduction or training. Everyday examples of walk-up-and-use systems are ATMs, public transportation ticketing systems, and interactive information displays in public places such as museums.

Background and experiences

Since the arrival of commercial surface computing platforms such as the Microsoft PixelSense[2] (formerly called Microsoft Surface), we have seen a lot of new walk-up-and-use applications. Interactive tabletops in hotel lobbies, for instance, enable guests to explore all kinds of information about the city while using an interactive map to plan their day trips. In a sports stadium, VIPs can call up the statistics of the players and club at any time through a touch-based interface. Restaurants and bars allow customers to order and pay their meal or drinks by means of a tabletop application, or play a few casual games

[2] `http://www.pixelsense.com`

while waiting for their order. Museums try to enhance the visitor experience by providing interactive content and by allowing visitors to explore some artifacts digitally, such as precious books that are otherwise inaccessibly displayed in a closed showcase. All these examples fit the profile of a walk-up-and-use application.

Another set of applications that are well suited for walk-up-and-use are casual games. Casual games share some traits with walk-up-and-use systems, as they should also be very accessible, thereby requiring a minimal amount of training. Casual games target a mass audience, usually have very simple gameplay mechanics, and can be played in short bursts, for example while waiting for the train to arrive or during a lunch break. Those games can be played individually, for example on a smartphone, but there is also a market for collaborative casual gaming, for example teenagers visiting a museum during a school excursion, learning about history by competing in teams in a casual game on a tabletop.

(a) A collaborative presentation tool. **(b)** A casual game, *Where's Wally?*

Figure 1.2: Two MuTable applications that were deployed in a secondary school [Schneider 10].

Our experiences with walk-up-and-use and multi-touch systems are primarily based on a wide range of projects and user studies that were carried out in our research lab. One of the main goals of the IBBT GBO MuTable project was, for instance, to develop interaction techniques that allow natural interaction with all kinds of multimedia. In the context of this project, a col-

laborative application to prepare presentations and a few casual games were deployed in a secondary school [Schneider 10], as seen in Figure 1.2. Furthermore, TinkerTouch set up a lot of setups in walk-up-and-use surroundings, such as exhibitions, museums and Pukkelpop, a large Belgian music festival. An additional source of valuable information came from observing visitors while they interacted with some of the interactive surfaces in our lab.

Illustrative scenario

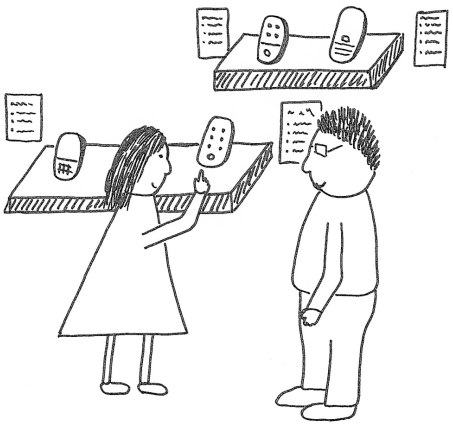
This section describes a typical usage scenario of an interactive application that can be deployed in, for instance, a mobile phone retail environment. Families or friends shopping together for a new mobile phone can collaborate on a multi-touch tabletop located inside the store to find more information about products and compare specifications. Figure 1.3 includes a few of the main scenes from the scenario.

John and Jane are looking for a new mobile phone, as Jane's current phone is outdated. At home, they look at some advertisements and they decide to visit a nearby mobile phone retailer. The display window of the shop shows a rather narrow selection of phones, but a nice looking model catches Jane's eye. John and Jane look for that particular phone inside the shop, and when they find it, Jane tries it out for a few minutes. However, Jane is not entirely convinced, since the phone costs more than she is willing to spend, and she wants more detailed information.

Jane picks up the phone and she walks with John to an interactive display in the shop. Because it is the first time that they have visited this store, the application asks them to quickly register with the system. John enters his name and e-mail address, and the system creates a new user profile. Jane then puts the mobile phone on the interactive display to get more information. The system recognizes the phone by means of a visual tag that is attached to its back and the specifications of that particular model are displayed, together with some promotional photographs.

While Jane is quietly going through the detailed specifications of the phone, the application updates John's user profile in the background, as it keeps track of all products each customer looked at. This enables the application to observe a user's interests in order to offer targeted advertisements, such as product recommendations. As a result, several other phones are now publicized at the top of the display, in the same price range and with more or less similar features as the phone Jane chose from the display window.

While John reads through the specifications of some of the recommended



(a) John and Jane are looking for a new mobile phone inside the shop.

(b) Putting a phone on the interactive display gives detailed information.

(c) While John explores recommended phones, Jane compares two products with a two-handed gesture.

(d) When John later returns to the display, it recognizes him and loads his profile.

Figure 1.3: A few of the main scenes from the scenario regarding walk-up-and-use environments.

phones, Jane sees another interesting mobile phone and compares the two products. To compare products, she performs an easy two-handed gesture. John notices Jane's actions and also wants to take a look at the new phone in which she is interested. Jane then makes a copy of the information panel through the use of another gesture and moves it over to John. John and Jane are now both interested in the phone suggested by the system and go find it in the shop. When returning to the interactive display, they notice that other customers started a new session. However, when they approach the display, the system recognizes them and automatically restores their list of mobile phones. After looking at several phones for a while, Jane decides she wants to sleep on it and they both leave the store.

After a few days, Jane has made up her mind and she returns to the store to buy the phone. John is with her and while Jane is waiting in line at the cash register, he walks up to the interactive display. The system recognizes John, loads up his user profile and shows him all the phones they looked at in the past. While waiting, John browses through some of the recommended phones.

Research challenges

Multi-touch interfaces in walk-up-and-use environments such as public spaces impose specific requirements on the accessibility of the user interface: most users are not familiar with the interface and the user experience needs to be exciting, since users simply walk away if the interface does not meet their expectations. The interface is therefore supposed to support attractive, intuitive and very "natural" interaction, but there are few common conventions regarding the use of gestures. This absence of conventions leads to a lack of consistency across different applications and a low memorability of the gestures. Furthermore, touch-based systems rarely support a hover state and the typical aesthetic design of a gestural interface often provides little to no perceived affordances, making it hard for users to find out which of, and how, the various interface components respond to touches.

Complicating matters further, the limited interaction time and the need for immediate use of the system do not allow for much training or exploration of a walk-up-and-use interface. If a user is not immediately successful in interacting, that person will simply give up and walk away. This leads to a rather complex situation, in which the user interface has to be completely self-explanatory. This might not be too big of a challenge if the application offers little functionality, but as the interactions become increasingly more

complex, so does the challenge of providing a fully self-explanatory interface. Looking at the scenario in particular, John and Jane must be able to interact with the unfamiliar application without any real training. However, Jane needs to somehow learn about the gestures to compare products or to copy an information panel.

Another research challenge is related to the hardware setup itself. Multi-touch surfaces are well suited to co-located collaboration because of their ability to track multiple inputs simultaneously, but the majority of those interactive surfaces are unable to associate a contact point with the particular user performing the touch. In the abovementioned scenario, Jane performs a two-handed gesture to compare two products. For this kind of interaction to work unambiguously, the system needs to be able to distinguish between Jane's and John's input, so it knows that Jane is performing the two-handed gesture, as opposed to both Jane and John each performing a single-handed gesture. In addition, recognizing the users that are currently interacting with the hardware enhances the walk-up-and-use characteristics of the system, seeing that users no longer need to manually log into the application each time it has to load their user profile.

1.3.2 Co-located and remote collaboration

The walk-up-and-use scenario in the previous section involved a fairly small application, with a very limited amount of collaboration, as users mainly browsed through some information. Professional work environments, on the other hand, often necessitate far more extensive systems, with much more advanced forms of collaboration. For that reason, we discuss a comprehensive meeting environment in this section.

Background and experiences

Achieving a common goal typically requires meetings on a regular basis. Defining a project proposal, discussing medical results, brainstorming on an advertising campaign or putting together a storyboard for a new system: it all requires people to meet from time to time. However, a lot of these meetings are organized inefficiently. First of all, people usually all get together in the same physical room, often requiring some participants to travel, which can be both time consuming and costly. Furthermore, participants may bring along one or more personal devices, such as laptops, tablets, PDAs and smartphones. Incorporating those personal devices is challenging in traditional meeting room

setups, thereby frequently limiting the participants' capability of sharing particular information stored on a personal device during the meeting.

From 2005 to 2007, our research lab was involved in the IBBT GBO project *iConnect*, shown in Figure 1.4, which provided us with some valuable insights on multi-user environments. iConnect [Cardinaels 06] is a distributed system to support meetings involving collaboration among both co-located and geographically dispersed participants. Although remote participation is unquestionably very useful at times, eliminating the time and costs associated with travelling to one location, we acknowledge that some degree of co-location is often preferred. Setting up a "mixed" meeting with both co-located and remote participants carries configuration costs: a multitude of assorted devices need to be connected to the system, enabling smooth collaborative interaction and an effective way to share data. Allowing users to bring along their own personal devices complicates the configuration, especially when people frequently enter a meeting late or leave early.

Figure 1.4: The iConnect meeting environment, with a variety of devices connected to the system [Cardinaels 06].

The iConnect system focuses on the continuously evolving nature of meetings, providing a framework to facilitate the dynamics and location independence of those meetings. In particular, iConnect aims to minimize configura-

tion steps, while maximizing smooth device integration. A "connected meeting room" is realized by means of a flexible software platform, which is supportive in that it does not constrain users, but allows to embed all kinds of files, such as office documents, images and videos. By sharing these embed files over a network, we ensure participants, either co-located or remote, have a synchronized view while being able to annotate and interact with one another's data.

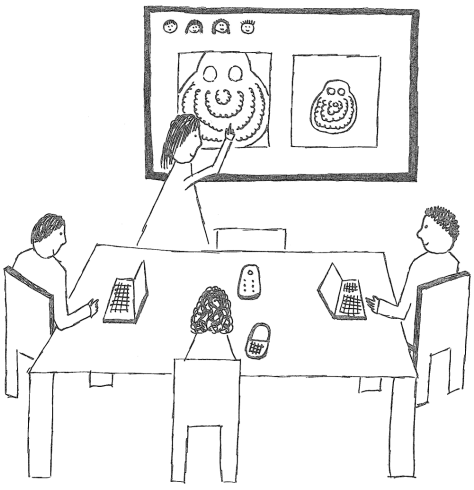
Illustrative scenario

The following scenario illustrates how the iConnect system supports a real-life meeting. Figure 1.5 shows a few of the main scenes from the scenario. A team of neurologists gathers to discuss the results of a particular brain scan, and a secretary is present to take minutes of the meeting. Most of them have brought along their personal laptop and/or a mobile device, such as a PDA or mobile phone. As soon as they enter the meeting room, the system notices the participants' presence and each person is assigned an avatar in order to identify the different users during the collaboration.

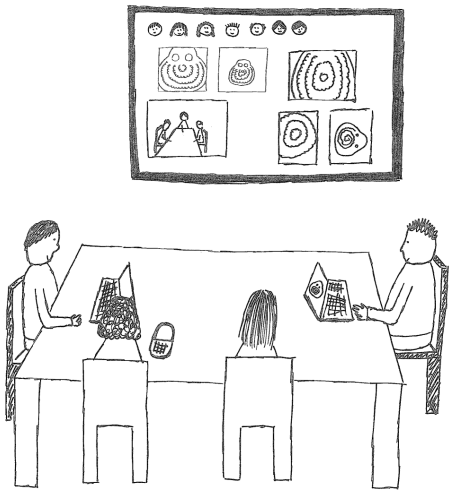
For starters, Jane, the department head and team leader, would like to present an outline of what the meeting is about. She opens her Microsoft PowerPoint file, counting several images of a brain scan, on an interactive whiteboard. iConnect takes care of all presentation and interaction aspects, as the slideshow application is smoothly integrated in the overall user interface of the iConnect system. Jane browses her slides and explains the specifics of this particular case. To accentuate a certain area of the brain scan, she annotates an important part of one of the images shown on the slides.

Another neurologist, Jake, proposes to discuss the future treatment. To clarify the patient's condition, Jake wants to show an older brain scan of this patient, which is stored on his laptop. He uploads the file to the whiteboard, where Jane opens it. Jane sits down for a moment and studies the scan. She indicates the problematic areas by encircling a few anomalies on the brain scan, using her newly bought mobile phone as a remote touchpad. The other participants are also able to use their personal device as a remote control, but when Jake accidentally tries to close Jane's PowerPoint presentation, the system refuses to do so, since Jane is the one who is currently presenting on the whiteboard.

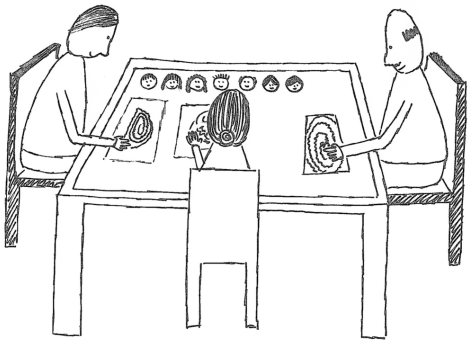
One of the neurologists, Lucie, who is visiting from another hospital, requests a copy of the scan so she can have a closer look. Normally Lucie would have been able to copy the file herself, but as she is registered as a visitor and



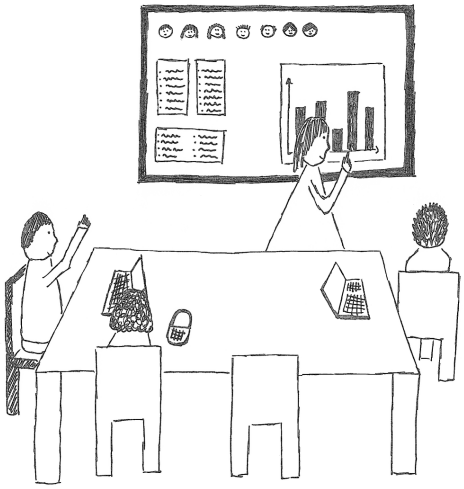
(a) Jane presents an outline of the meeting on an interactive whiteboard and annotates the brain scan.



(b) The whiteboard shows a video stream of a remote team, and additional brain scans provided by Jake and Jacob.



(c) Three expert neurologists discuss a specialized problem on an interactive tabletop.



(d) Meanwhile, Jake and Lucie compare files, while Jane debates a financial report with Jacob.

Figure 1.5: A few of the main scenes from the scenario regarding co-located and remote collaboration.

the scan has to be treated as confidential patient information, she does not have the appropriate access rights. Jane, who has the necessary access rights as department head, therefore has to drag the scan to Lucie's avatar, which results in the file being copied to Lucie's personal device.

A senior team member, Jacob, wants to show a brain scan of a patient suffering from a similar disorder. Unfortunately, the iConnect system does not support the aged file format Jacob is still using. Jacob's laptop, however, is capable of opening such scans. To cope with this situation, the desktop screen of Jacob's laptop is duplicated on a small part of the whiteboard, next to the other brain scans, thereby avoiding the difficulties of changing the current hardware setup, such as connecting the laptop directly to the whiteboard's video source. The system takes care of possible issues with new content being shown on the whiteboard on the fly, by ruling out unfavorable occlusions (i.e. visual overlap between the new content and the previously opened images that are still being discussed at the moment).

Meanwhile, a faraway team joins the meeting remotely. The iConnect system transparently duplicates the workspace to their remote site, allowing them to view and interact with the available scans. The mutual awareness is enhanced by audio and video streams, illustrating the overall situation in the other meeting room. After some time, three expert neurologists decide to discuss a specialized problem. The group splits up: the trio moves to an interactive tabletop, while the others remain at the whiteboard. Seated around the table, the experts cooperate extensively, exchanging data between one another. Again, care has to be taken with, for instance, people closing or occluding another's files.

In the meantime, the others are also having an open discussion. Jake and Lucie opened several new files to compare the approaches of the two hospitals. Because Jane quickly wants to discuss a financial report with Jacob, she opens it on the whiteboard and starts going over the first page. When the three experts reach a conclusion, they send the data to the whiteboard, and they present their findings to the others. Finally, as the meeting comes to an end, the meeting minutes are uploaded to a shared document space, remaining available for future review. However, access to this data has to be treated carefully, since it involves confidential information regarding a patient.

Research challenges

Allowing multiple people to interact in a highly collaborative environment gives rise to several types of conflicts. In the scenario, we mentioned possible

issues with Jake accidentally trying to close Jane's presentation at an inopportune moment, or opening new files that could occlude a file that is currently being discussed. These conflicts should be prevented or resolved, if possible without interrupting the dynamic work flow by enforcing an explicit response of the user. An intuitive approach is to assume that social protocols, such as polite behavior and social standards, are adequately observed and suffice to coordinate the actions of the group of users. Even though social protocols perform well in some cases, they cannot prevent or resolve numerous types of conflicts effectively, especially when remote users are involved. Consequently, the highly collaborative nature of the environment calls for some interaction management.

Additionally, with both co-located and remote users being present, sufficient attention needs to go to mutual awareness and access control to shared data. It is important that users are well aware of the possible effects and ramifications of their actions, even if several smaller subgroups are working separately from time to time, and that access to confidential information is restricted. Just as with preventing or resolving conflicts, gaining information regarding mutual awareness should be non-intrusive, without any interruption to the work flow. It should enable a user to adjust to the observed circumstances, so his or her actions come seamlessly together with the collaborative efforts of the others.

As in the aforementioned walk-up-and-use scenario, the identity of users comes into play in a number of ways. First of all, when Jane is presenting on the whiteboard, others should be restricted in what they can do on this shared surface. Here we not only take Jane's identity into account, but also the role that Jane is currently performing, namely the role of presenter. Secondly, access to shared files is based on the access rights of a user, which again can be linked to both the identity (e.g. Lucy, who is visiting from another hospital) and the roles of a person (e.g. Jane being the department head and team leader, or Jacob being a senior neurologist).

Since the scenario is situated in a professional work environment, the limited time for training or exploration that is prominent in walk-up-and-use systems is less of an issue. However, there is always the possibility of a first-time or infrequent user being asked to join the meeting unexpectedly. That should not pose an actual problem when other, more experienced participants are involved, but that first-time user might also be a lonesome remote participant. In such circumstances, it is again important to be able to quickly discover how the user interface is operated, without having to read an instruction manual, for instance.

1.3.3 Storyboarding in a multidisciplinary team

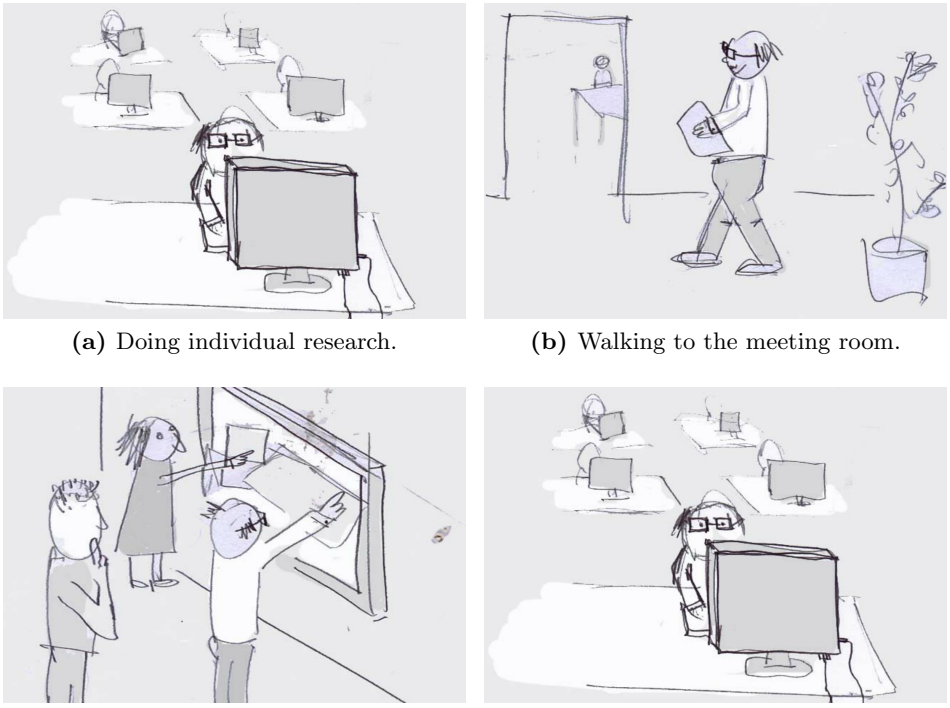
In this section, we have an in-depth look at one particular kind of collaboration, namely collaborative storyboarding among co-located members of a multidisciplinary team. Our main goal is to investigate the specific requirements this particular context imposes on the design of digital tools for shared interactive surfaces.

Background and experiences

The storyboarding process was developed at the Walt Disney Studio during the early 1930s, as they were in the habit of drawing scenes on sheets of paper to tell the story of an animated cartoon. Storyboards are still used extensively in the film industry, to help visualize the scenes and story beforehand. Nowadays, however, storyboards are also widely used in various other domains. Writers use them for plotting the story of their books, or for planning advertising campaigns and commercials. In industry, storyboards are for instance used to develop process flowcharts, illustrate product usage scenarios, and visualize promising new activities in business presentations. We are, of course, most interested in the use of storyboarding in the field of computer science and human-computer interaction.

In the context of software development, a storyboard can be seen as a narrative that uses rich and iconic illustrations or images displayed in sequence for the purpose of visually representing systems' scenarios of use, as illustrated in Figure 1.6. Storyboards can be hand-drawn or created digitally, and can be used to describe the user's daily doings, as well as the possible system designs and the influence the system would have on the user's activities. The storyboarding approach is well suited for depicting the requirements of systems in different contexts of use and for creating a common understanding of the application domain, much better than an abstract description. The practice of visual thinking and brainstorming allows a group of people to collaborate, concretizing ideas while fostering across-the-board ideas, resulting in a consensus reached by the group as a whole.

The research on storyboarding in multidisciplinary teams was done together with dr. Mieke Haesen, who finished her PhD on combining user-centered design and software engineering in 2011 [Haesen 11a]. Among other things, she introduced the COMuICSer storyboarding approach and tool, and investigated them through several user studies, which resulted in a number of valuable experiences for our research lab.



(a) Doing individual research. **(b)** Walking to the meeting room.

(c) Discussing search results. **(d)** Assembling selected videos.

Figure 1.6: Part of a storyboard created for the development of an application to visually explore video archives [Haesen 11a].

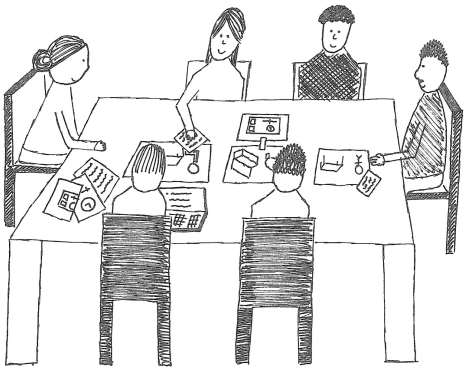
Illustrative scenario

In this scenario, we describe part of a collaborative storyboarding session. Figure 1.7 depicts a few of the main scenes from the scenario. During this session, a multidisciplinary team tries to come up with the functional and non-functional requirements of a future home automation system. The team includes a human-computer interaction expert, two systems engineers and a user interface designer, as well as the client and an application domain specialist.

All team members were briefed in advance, as they received a number of personas that represent typical end users and a scenario that revolves around a home automation system to control the heating and lighting, which can assist a household in saving money on energy consumption. This home automa-

(a) Kate goes over the scenario, displayed on her tablet. Meanwhile, John takes notes on hardware requirements using his laptop.

(b) Different threads of discussion emerge, as Oliver and John discuss the hardware, and Jim asks Kate about the user profiles.



(c) The storyboard scenes are arranged in the middle of the table, and different artifacts are attached.

(d) To preserve the results, Kate takes a few pictures of the final storyboard and the associated artifacts.

Figure 1.7: A few of the main scenes from the scenario regarding storyboarding in a multidisciplinary team.

tion system should allow family members, differing in age and technological aptitude, to control settings using various devices, such as touchscreens, laptops and smartphones. It should also be possible to automatically adapt the settings according to personal profiles and the current activities of the family members. Furthermore, the system should prevent environmentally unfriendly situations from happening, for example people leaving the house without turning off the lights.

As the meeting starts, the client, Jim, introduces himself and his company, and shortly recapitulates the goal of this get-together. The human-computer interaction expert, Kate, then takes the lead, as she is the most experienced in storyboarding. Kate goes over the scenario, displayed on her tablet computer, highlighting the important aspects step by step. Meanwhile, one of the systems engineers, John, who did not prepare the meeting beforehand, takes notes on the possible hardware requirements for each of these steps on his laptop. When Kate finishes, Jim has some additional requests, as he thinks it would be nice if the system could also monitor the water consumption in each room of the house. The application domain specialist, Oliver, immediately remarks that the additional sensors would increase the costs quite a bit. John does not agree entirely, but before he can speak his mind, Kate takes control of the meeting again, because she does not want to deviate on this topic for now. She makes a note of the client's request, however, so they can return on it in a later stage.

They start discussing a very basic scenario: how to control the heating and lighting from the living room when only one person is present. Kate takes a sheet of paper and quickly sketches this situation. Oliver suggests to mount a touch-sensitive display against the wall as a control panel, and while Oliver and John are discussing the hardware possibilities, the user interface designer, Olivia, starts sketching a basic user interface based on a design she made a while ago for a similar system. In the meantime, Jim asks Kate a few questions about the user profiles. After a few minutes, the different threads of discussion merge, and everyone focuses on the profiles and what should happen when one user initiates a personal profile that conflicts with another user's profile that is currently active.

The resulting storyboard scenes are arranged in the middle of the meeting room table. Olivia attaches her user interface design to the scene of the living room. As the meeting progresses, they debate on the use of mobile devices, and more and more scenes are collected on the table. Kate arranges the storyboard scenes in the right sequence, and now and then attaches a note with an important comment to a scene. She also checks her tablet, containing

her preparations, to make sure that they do not lose track of some important aspects regarding the end users, such as the difference in technological aptitude.

The brainstorm continues like this for a while, and after the meeting comes to an end, Kate takes a few pictures of the resulting storyboard scenes and the associated artifacts with her digital camera. She also collects all the sheets of paper. Leo, the second systems engineer, leaves the meeting with mixed feelings, as he was not able to contribute as much as he would have liked. As the system should be able to detect people's presence in particular rooms, Leo prepared a list of hardware possibilities concerning indoor tracking beforehand. However, Leo is a somewhat reserved person and this specific topic never came up in much detail, so his list was never used during the meeting, and he fears his preparations will go to waste.

Research challenges

As people with different backgrounds and goals gather, they will bring different viewpoints and approaches to the table. Involving all team members in the decision making process results in more comprehensive storyboards, so it is important that each individual has the opportunity to contribute equally and that a degree of mutual engagement is established. In the scenario, John does not always agree with Oliver, but he never got the chance to express that during the meeting. Likewise, Leo's preparations were never used. Kate tries to write down the most important comments, but some of the design rationale will not be recorded, especially when smaller groups of people start working in parallel for a while.

Since Section 1.3.2 also covers collaboration, the research challenges presented in that section are of course also valid for multi-user storyboarding. However, while digital systems such as iConnect can support the collaborative brainstorming process to some extent, care should be taken that the use of such a system does not constrain the team's creativity or flexibility. Members of a multidisciplinary team are already accustomed to specific devices, tools and approaches, and forcing them to change may hinder their creativity and engagement. Some people prefer, for instance, hand-drawn storyboards to digital storyboards, since it allows them all the freedom that comes with working with regular paper. Forcing those people to use a specific drawing tool on a tablet or a tabletop will not only limit their creativity, but also their level of commitment, leading to less diversity in viewpoints and thus inferior storyboards.

While some members of the team may prefer to work with real paper, others will like to use (a certain application on) a tablet or laptop. However, when combining the paper and digital world, there is always the risk of useful information being lost or forgotten, particularly if no tight coupling is maintained between paper and digital artifacts. In the scenario, Kate takes a few pictures of the results, but those do not include any of the digital artifacts, making it harder to consult all the results of the brainstorming session afterwards without overlooking or having to track down any additional information.

Storyboarding in a multidisciplinary team imposes some additional requirements on the research challenges presented earlier in this chapter. First of all, the multidisciplinary team will often include external members, who are more likely to be unfamiliar with any digital systems that might be used during the meeting. Being too involved with training and exploration of a system during the storyboarding session will lead to people being less involved in the actual brainstorm, which should be avoided. Secondly, keeping track of (the identity of) a user becomes more difficult if that user is not only using digital devices and applications, but also regular paper, as we can no longer rely on using identity-aware shared surfaces and tracking of the user's personal devices.

1.4 Summary of research challenges, scope delineation and overview of chapters

In this chapter, we presented some of our experiences with walk-up-and-use and multi-user environments, which we illustrated through three scenarios about get-togethers involving collaboration among co-located and/or remote participants. Based on these experiences and scenarios, we put forward several research challenges:

- Walk-up-and-use environments impose specific requirements on the accessibility of touch-based interfaces. There are few conventions regarding the use of gestures, and the limited interaction time and need for immediate use of the system do not allow for much training or exploration. How can we make touch-based interfaces in walk-up-and-use environments self-explanatory, for both single-user and multi-user settings?

- Allowing multiple people to interact simultaneously in a highly collaborative environment gives rise to several types of conflicts and possible misconducts, especially if that environment can include both co-located and remote participants. How can we provide interaction management

and enhance mutual awareness in a multi-user environment without interrupting the dynamic work flow?

- To take advantage of the different viewpoints and approaches that members of a multidisciplinary team bring to the table, it is important that each member has the opportunity to contribute equally to the decision making process and that a degree of mutual engagement is established. How can a storyboarding tool support the various disciplines and maintain equitable contributions, without impacting the team's creativity?

- Multi-touch tabletops are well suited to co-located collaboration because of their ability to track multiple inputs simultaneously, but the majority of those tabletops are unable to associate contact points with particular users, a feature that can improve a multi-user interface in a number of ways. How can we accomplish non-intrusive identification of the different users around any tabletop, independently of the hardware technology?

To situate these research challenges in the overall field of touch-based and multi-user interaction, we reconsider the research fields listed in Section 1.2. The main focus of this dissertation lies in the fields of enhancing the accessibility of touch-based user interfaces and improving the collaborative aspect of multi-user environments. Since we approach these topics from various angles, from both a user and developer perspective, we inevitably touch upon some of the other research fields. However, as we occasionally venture into those fields, it is always in support of our central goals, and it is never our primary intend to create truly novel interaction techniques, application domains, hardware systems or development processes.

In addressing our research challenges, we explore the abovementioned fields as follows throughout the upcoming chapters:

- In the first part of this dissertation, we investigate the concept of a self-explanatory user interface to enhance the accessibility of touch-based interaction. In Chapter 2, we propose various strategies to assist users in discovering and learning the functionality of a touch-based interface. We approach this from the perspective of both the user and the developer.

- Subsequently, we present a few evaluations of different strategies in Chapter 3, and based on our results and observations, we offer developers some useful insights into how to provide help. The evaluations include strategies that specifically target walk-up-and-use and multi-user systems.

- We continue our exploration of multi-user environments in the second part of this dissertation, by examining how we can improve the collaborative aspects in two application domains: meetings in general and storyboarding. In Chapter 4, we focus on interaction management, mutual awareness and access control in the iConnect meeting environment, supporting co-located as well as remote collaboration.

- Next, in Chapter 5, we discuss an in-depth observational study that focuses on one specific kind of collaborative environment, namely storyboarding among co-located members of a multidisciplinary team. We formulate and concretize a number of requirements to inform the design of a tabletop tool for collaborative storyboarding.

- To conclude the second part, we move a little into the field of hardware advances in Chapter 6, as we tackle a challenge that we encountered on multiple occasions throughout the preceding chapters. We present a non-intrusive solution to identify the different users around a table by using an overhead camera that observes the back of the users' hands.

- Finally, in the third part of this dissertation, we consider the development process of touch-based and multi-user interfaces, as we explore a graphical notation to support the development and evaluation of interaction techniques in Chapter 7.

To end with, we summarize our main conclusions, we reflect on our contributions, and we discuss various opportunities for future work in Chapter 8.

Part I

Self-explanatory interfaces for touch-based interaction

Chapter 2

TouchGhosts: visual guides for multi-touch interaction

Contents

2.1 Introduction

In this chapter, we explore the concept of a self-explanatory touch-based interface, and we introduce *TouchGhosts* [Vanacken 08b, Vanacken 09a]. A

TouchGhost interface facilitates the discovery and learning of touch-based interaction techniques "on the spot", as visual guides are merged with the actual user interface to inform users about the available interaction techniques, if possible within the current context of use.

One of the biggest advantages of a graphical user interface (GUI), compared to for example command-line or speech interfaces, is that commands do not have to be memorized. Instead, the possibilities of the user interface can be discovered by exploring the toolbars, menus and other widgets. Designers of a GUI should make the important aspects of a system perceivable to the user, to allow the user to easily translate goals into actions. For that reason, most widgets in desktop environments provide affordances to the user, "suggesting" how the interface component may be interacted with [Norman 88]. In addition, as a result of the standardization of GUIs, a user can usually depend on prior knowledge of other applications when confronted with a new interface. In other words, the core functionality of an interface should be apparent just by looking at it, and by exploring the different graphical components.

As gestural interfaces typically rely almost exclusively on direct manipulation, there are little to no user interface components providing perceived affordances or any means of discoverability [Norman 10a, Nielsen 10, Nielsen 11]. Often interface components are mostly hidden within the "smooth" aesthetic design of the user interface, with no indication which of, and how, the various components respond to touches. Although the smooth design might look attractive, without any visual distractions, established GUI design guidelines dictate that controls such as buttons should look raised, giving the impression of being something that can be pressed, and that scrollbars and other interactive components should be visually distinctive. Traditionally, some sort of lighting or dimensionality effects are used to indicate raised or lowered interface components, but those effects are often absent in the design of gestural interfaces.

To make things worse, touch-sensitive displays rarely support a hover state. On a standard display we use a device such as a mouse or trackpad to indirectly interact with the user interface. The hover state that can be accomplished with such a device is commonly used as one of the important means to make users aware of the user interface components that can be clicked and manipulated, for example by adding a dimensionality effect to a button when a mouse cursor is on top of it. A touch-sensitive display, on the other hand, just uses fingers or a stylus to interact with it. As a result, most touch-based systems do not support a hover state, and thereby lose an important way of providing assistance to the user.

A touch-based interface is supposed to support intuitive and "natural" interaction through gestures, but users experience difficulties finding out how to interact with such an interface due to a lack of familiarity. There are few common conventions beyond a small set of widespread operations (e.g. select, move, resize), resulting in a lack of consistency across different applications and a low memorability of the gestures [Norman 10a, Nielsen 10, Nielsen 11]. Moreover, most gestures are hardly natural [Cao 08, Norman 10b] and some commands elicit little gestural agreement [Wobbrock 09]. Touch affordances can be a first step toward a good design, by communicating touch-based interaction in an implicit way [Schöning 09], but as stated before, affordances are mostly absent in current touch-based interfaces. Due to this characteristic design of touch-based user interfaces, first-time or infrequent users have difficulties figuring out or remembering what can be done with the interface and how it can be done.

Looking at multi-touch interaction in particular, it allows much more than just clicking and dragging, since interface components allow multiple concurrent points of control. Earlier deployments of multi-touch surfaces in public spaces [Cuypers 08] and other observations of tabletops [Ryall 06b, Hornecker 08a, Marshall 11] revealed that users are hesitant to touch the surface at first and often imitate the traditional mouse interactions by trying to operate the interface with merely one index finger, not fully exploiting the multi-touch features. Only over time they gradually discover that multiple fingers or specific hand postures can be used.

While single-finger gestures can be simple enough to discover through trial-and-error, it is less likely that a user will discover interactions involving two or more fingers "accidentally", not to mention cooperative [Morris 06] or whole hand [Wu 03] gestures. People rarely use multiple fingers together to interact with an interface, except in very specific contexts that are well understood, like zooming with two fingers. Therefore, gestures such as scrolling inside a list view control with two fingers (as opposed to scrolling the entire window with one finger) have a very limited discoverability rate.

Especially walk-up-and-use setups in public spaces, such as the ones we described in Section 1.3.1, undergo these problems: the limited time of use does not allow for much training or exploration. However, similar issues can be found in the other scenarios from Section 1.3. Take, for example, the iConnect system we described in Section 1.3.2. Since the scenario is situated in a work environment, we can assume that there is time for at least some training and exploration. However, a first-time or infrequent user might end up in a meeting unexpectedly, which poses a problem if no experienced participants are present

to help out. The same holds true for storyboarding in a multidisciplinary team, as described in Section 1.3.3. External people involved in the meeting, such as external clients or designers for example, will probably be unfamiliar with the system, and appointing a trained facilitator to provide technical support and training [Galaczy 99] is not always feasible.

Figure 2.1: Example of a self-explanatory TouchGhost interface. Visual guides are merged with the actual user interface to inform the user about the available interaction techniques within the current context of use.

To assist users in exploring and learning the functionality of a user interface, most applications include some sort of help system, in the form of an instruction manual, embedded tooltips, an online help system, interactive tutorials, a virtual assistant, and so on. As multi-touch interfaces introduce new and unfamiliar notions, such as multiple concurrent input streams and synchronized interactions, new types of help systems are being explored. In this chapter, we introduce the concept of a self-explanatory *TouchGhost* interface to facilitate the discovery and learning of touch-based interaction techniques. Figure 2.1 portrays an example of such a TouchGhost interface.

After considering the related work in the next section, we discuss different TouchGhost strategies to invoke and visualize help in Section 2.3. We introduce two distinct architectures in Section 2.4, including a Microsoft .NET architecture that allows integration of a TouchGhost interface in new or existing .NET applications. Based on this architecture, we implemented a number

of examples, which are shown in Section 2.5. We finalize this chapter with some conclusions and next, in Chapter 3, the results of several evaluations of different TouchGhost strategies are presented.

2.2 Related work

In this section, we explore the existing work relating to various kinds of help, including the use of videos, contextual help, menus, animations, and feedforward and feedback. We also examine how to encourage active involvement and how to draw people's attention.

A classification of help. Dworman and Rosenbaum [Dworman 04] address a very basic question: why do users fail to use the help systems available to them? To improve the initial interaction between users and help systems, they discuss among other things help-seeking behavior and help system access. They identified the most common types of help in current applications:

- text-based help, e.g. printed or digital manuals, tooltips integrated in the application, online help systems and tutorials;
- interactive help, e.g. systems that try to make a diagnosis through questions and answers, virtual assistants;
- "show me what to do" help, e.g. demonstration videos, semi-transparent overlays or animations;
- human help, e.g. help desks, colleagues and forums.

Most applications include some sort of help system that falls into one of these categories. Plenty of studies unfortunately show that the content, categorization and terminology of that help do not always meet the users' needs [Duffy 93, Krull 01, Martin 05, Novick 06], as users may find tutorial-style documentation too basic, but find a reference manual too technical and overwhelming. Based on numerous studies, Grayling [Grayling 02] states that users are often reluctant to consult help, preferring trial and error tactics. They only open help when they are really stuck and they read text hastily and inaccurately. Some users are not even aware help exists. However, it is possible to improve help by making it an integral part of the user interface, for instance by implementing it as embedded help panes or tooltips. Furthermore, new evaluation protocols can help to identify existing learnability issues in an application [Grossman 09].

Videos as "show me what to do" help. Since we focus on multi-touch setups in public spaces, we are mainly interested in the third category of help presented by Dworman and Rosenbaum, the "show me what to do" help. Videos are one approach of providing this kind of help, and learning through videos has already been studied extensively. Zhang et al. [Zhang 06] explain the positive effects of interactive videos on both the learning outcome and the learner satisfaction in e-learning, while Schwan and Riempp [Schwan 04] conclude that the use of interactive features in videos results in a better processing and understanding of the visual information. The Ambient Help system [Matejka 11] provides multiple videos and textual help on a secondary display to support opportunistic learning, and Grossman and Fitzmaurice [Grossman 10] show contextual videos in ToolClips as assistance for explaining functionality.

Vogel and Balakrishnan [Vogel 04] present a public interactive display that supports the transition from implicit to explicit interaction with both public and personal information. The system makes use of interaction techniques such as simple hand gestures and touch screen input for explicit interaction. On the subject of help, it provides a self-revealing mechanism that shows videos of available gestures. The examples we present in Section 2.3 include demonstration videos as one of the visualization strategies, and we will discuss the advantages and disadvantages of such an approach in Chapter 3, based on our evaluations.

Contextual help and guidelines. Videos are often shown in a separate window, or a standalone video player. Contextual help, on the other hand, is effective for learning how to use an interface by showing instructions on the actual interface components that the user is interacting with, rather than in a separate view. This approach allows hands-on practice and exploration at the same time [Yeh 11]. Based on a case study on user-aided design of online help, Knabe proposes some fundamental and very applicable design goals for Apple Guide [Knabe 95], including:

- help should appear in the same application layer as the application being used to complete the task;
- instructions should be presented in small chunks to minimize the user's reliance on memory;
- rather than showing an illustration of an object on the screen, the help should point out the actual object.

In addition, Grayling [Grayling 02] postulates that help should meet five key characteristics:

- context-specific, only containing facts relevant to the specific context;
- useful, containing all the information relevant to the context;
- obvious to invoke, the mechanism being clear to the user;
- non-intrusive, not distracting users before being invoked;
- easily available, not requiring digging through help topics or an index.

With TouchGhosts, we have similar goals in mind: the help should be graphically presented within the actual context of use whenever possible, and should be obvious to invoke, non-intrusive and easily available. In contrast, some systems show the help in an area separated from the application's workspace. GestureBar [Bragdon 09] is, for example, a separate toolbar that provides animated images, detail tips and a practice area where the user can test a gesture without actually performing the associated action.

Providing help through enhanced menus. A number of approaches enhance menus to improve learning of, for example, hotkeys [Grossman 07] and stroke gestures as shortcuts to menu selection [Appert 09]. The Multi-Touch Menu [Bailly 08] is invoked by touching a multi-touch surface with the palm of the hand. The user can select a sub-menu from a pie menu with the thumb of that hand, and the items from that sub-menu are then displayed near the other fingers. Experienced users can skip going through the menu and quickly invoke a command by just touching the surface with a thumb and finger in the right formation. The Multi-Touch Menu helps users to learn gestures, but is limited to static multi-finger postures.

Kurtenbach et al. [Kurtenbach 94] also propose interaction mechanisms that make gestures self-explanatory for novices, but still allow experts to use efficient command marks. The examples include marking menus and a crib-sheet animator that uses iconic gestures and animated demonstrations. The demonstration is shown in context, with text annotations to explain its features. Although Kurtenbach's system is limited to pen-based input, the crib-sheet animator fits very nicely in the TouchGhost philosophy and is therefore easily replicated by implementing an appropriate invocation and visualization strategy, as we explain in the next section.

Animated help in traditional interfaces. The concept of using animated help in a direct manipulation interface is not new, and has been studied as far back as the eighties [Cullingford 82] and early nineties [Baecker 90, Sukaviriya 90]. Tuck and Olsen [Tuck 90], for example, implemented a "guided task" help prototype, providing instructions that guide the user while performing a task. They state that the system is not only entertaining and fun to use, but also avoids the problems associated with an impatient expert demonstrating the intricacies of some application to novice users.

Harrison [Harrison 95] shows that a graphical representation of help instructions can increase performance and decrease the number of mistakes. Palmiter and Elkerton [Palmiter 93] confirm these results, but indicate that textual help may allow users to remember instructions more efficiently than demonstrations. Tversky et al. [Tversky 02], on the other hand, state in the context of facilitating learning that animations are not necessarily beneficial compared to static graphics, because animations may be too complex or too fast to be accurately perceived, and they often mismatch people's conceptions of motion. Although animated help has already been studied extensively, our work differs in that it uses animated help for multi-touch interaction, so we need to reassess its usability compared to other types of help under these new circumstances.

Feedforward and feedback in gestural interfaces. Visual hints for discovering, learning, and assisting with the completion of gestural interactions have already been applied in tangible augmented reality systems [White 07]. The hints are graphical representations of potential actions and their consequences in the augmented physical world. Similarly, Octopocus [Bau 08] combines on-screen feedforward and feedback to help users learn, execute and remember single-touch gestures. The help system dynamically visualizes the current state of recognition and the optimal path of the possible gesture strokes a user can perform to complete a gesture. The results of two experiments show that OctoPocus is significantly faster and improves learning of arbitrary gestures, compared to conventional help. One of the main limitations is the level of visual complexity that users face if they enter the help mode without having drawn any portion of a gesture.

Arpege [Bau 10] is a comparable technique that helps users learn and execute multi-finger chord gestures on multi-touch surfaces by providing finger by finger feedforward and feedback. Chord gestures are static multi-finger postures that require the user to simultaneously place two or more fingertips on a touch-sensitive surface (e.g. form an equilateral triangle by placing three

fingers on the surface). An experiment suggests that Arpege enables efficient learning of chord gestures, and resulted in almost three times fewer errors. Techniques such as Octopocus and Arpege make it possible to include arbitrary single-touch or multi-finger chord gestures in real-world applications, but additional research is required to enable similar methods for multi-touch gestures in general.

Gesture previews [Cleveringa 09] take a first step in that direction, as the previews should not only inform the user of possible finishes after having initiated a gesture, but also point out when and how cooperative gestures can be initiated. Initial feedback on this preliminary work suggests that gesture previews can make working with gestures easier for novice users, and that they can encourage cooperation on tabletops. However, discovering all the available gestures with this approach can be bothersome, as different previews have to be shown over time if the set of gestures is too large to display at once. The work is only presented in its early stages, and further research is needed to examine how multi-touch and multi-user gestures can be visualized in a comprehensible manner.

ShadowGuides [Freeman 09] build on this idea, providing assistance by combining visualizations of the user's current hand posture and the available postures and completion paths necessary to finish the gesture. Freeman et al. compared learning gestures with ShadowGuides to learning with video and found that participants learning with ShadowGuides remembered more gestures and expressed higher preference for the system. The ShadowGuides approach resembles our "virtual hands" technique, explained in Section 2.3.1, but focuses mainly on the visualization aspect. We have a broader look at help systems and also propose various invocation strategies.

The Gesture Play system [Bragdon 10] uses the positive reinforcement of physical metaphors to teach gestures, including spring widgets, button widgets, and physical props to afford and constrain input. Evaluations indicate that Gesture Play is more motivating for learning gestures because of its fun nature. However, both ShadowGuides and Gesture Play do not explicitly investigate or address the difficulties encountered in typical multi-user applications, which are very common on multi-touch systems.

Encouraging active involvement and drawing people's attention. Experiences with large displays in public spaces have unearthed an additional challenge: people need to be encouraged to become actively involved, as engagement is often shallow [Hornecker 08a, Marshall 11] and feelings of social embarrassment act as a barrier [Brignull 03]. One way to improve en-

gagement is allowing for gradual discovery as a challenge within the application [Jacucci 10]. In the Worlds of Information application, help topics are presented as spheres that are in constant motion. The main help travels slowly, emitting slogans that encourage people to try the interface, while small help spheres travel at a faster speed. People become engaged in catching these small spheres, and once caught, they explain the gestural interface, for instance by playing a short animation.

The public interactive system by Vogel and Balakrishnan [Vogel 04] that we mentioned earlier includes an implicit interaction phase that is initiated as a user walks past the screen, and is used to demonstrate to the user that the display is interactive. It takes into account the user's body location, body orientation, and head orientation to fine-tune the communication with the user. Likewise, visual feedback such as mirrored silhouettes and images of passers-by can also be an effective way of communicating interactivity [Müller 12]. However, as it takes time to notice the interactivity, passers-by often already passed the public display and have to walk back to interact. Covert interaction through mobile devices [Kaviani 09], for example by sending votes or comments using text messages, is another possibility to encourage interaction. The covert interaction eliminates the social embarrassment and can be learned through large animated instructions, shown on the public display.

Another point of attention is making sure that users notice (all the related components of) the provided help when it is being shown. Users of traditional tutorials and help systems often have difficulty finding the interface components described or pictured in the procedural instructions. Stencils [Kelleher 05] is an interaction technique for presenting tutorials that uses translucent colored stencils containing holes. The stencils direct the user's attention to the correct interface component and prevent the user from interacting with other components. Multi-layer designs [Kang 03] structure an interface so that a simpler interface is available for users to get started and more complex features are accessed as they move through the more advanced layers. The associated Integrated Initial Guidance overlays help on top of the functional interface, in order to locate and demonstrate the main widgets using preset animations. We use comparable techniques to steer the user's attention toward the right spot, as will be discussed in Section 2.5.

2.3 Self-explanatory TouchGhost interfaces

The foundation of our work is the concept of a self-explanatory touch-based interface, a *TouchGhost interface* [Vanacken 08b, Vanacken 09a]. It is not our

intent to put forward one ideal help system for touch-based interfaces, since one solution rarely fits all. Therefore, we offer the freedom to customize a TouchGhost interface through various *TouchGhost strategies*, either at design time or at runtime. The visualization strategy determines what type of visualization of the help is presented to the user, the invocation strategy determines how the help system can be invoked.

2.3.1 Visualizations

Possible visualizations of a TouchGhost guide include playing short demonstration videos, representing the user's fingers as small annotated dots, imitating real actions through animated virtual hands, depicting the actions in a small storyboard, and so forth. Each visualization strategy has particular advantages and disadvantages, which are discussed extensively in the next chapter, where we present the results of a number of user studies.

Figure 2.2: Example of a "virtual hands" visualization, demonstrating how to resize by directly manipulating the actual picture. The red circles on top of the fingertips are animated to indicate which finger needs to be pressed.

Animated virtual hands can demonstrate interaction techniques by directly manipulating the application's interface. This method has the distinctive advantage of presenting the help within the context of use, on the interface components the user is currently working with, as depicted in Figure 2.2. Demonstration videos, on the other hand, only show a generic example of how to perform a particular interaction, but are easy to implement. The help system can simply show a guide and be done with it, or it can ask the user to repeat the interaction that was just demonstrated. The user will first be offered the explanation, immediately followed by an opportunity to try the interaction technique, either in the actual application or in a sand-boxed environment.

Deciding on the most suitable visualization strategy depends on a number of factors, such as the properties of the hardware, the type of application, the target group, whether it is a multi-user or a single-user system, and so on. Playful animated hands, for instance, might not suit users in a professional setting, while they can be an engaging solution in public spaces. As said before, one solution rarely fits all: the TouchGhost architecture permits different strategies that can be selected beforehand or dynamically at runtime, an approach we elaborate on in Section 2.4.

2.3.2 Invocations

An *explicit* invocation strategy, as found in most help systems, requires the user to explicitly request help. A straightforward example of an explicit invocation is simply pressing a help button or holding a finger on an interface component longer than a predefined threshold. Next, the system can display all the available interactions on that particular interface component in a pie or list menu, as seen in Figure 2.3. Once the user selects the appropriate menu item, the corresponding visual guide is shown. In this way, the user can easily discover and explore the interaction techniques supported by a particular component.

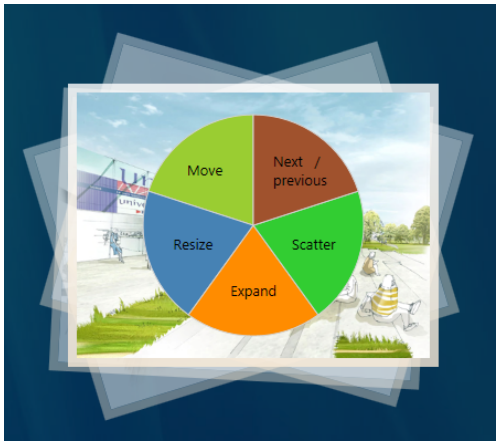


Figure 2.3: Example of a pie menu as an explicit invocation strategy, listing all available interactions on this particular interface component (in this case, a pile of pictures).

Alternatively, if supported by the hardware setup, a user may hover with a finger above an interface component to invoke help, somewhat similar to

hovering with a mouse cursor to prompt a tooltip. The TouchGhost interface can then show one particular TouchGhost guide, or all the guides one after another. The order in which guides are presented, can depend on general usage statistics (e.g. show the guide related to the most common action first). In that case, an "observer" service needs to be running alongside the application, recording all events on the user interface in order to accumulate the necessary statistics.

With explicit invocation strategies, we need to keep in mind that users must be aware of how to access the help system. Users will notice a conveniently located help button, but they might not be able to figure out the "hold your finger on an interface component longer than a predefined threshold" or hover method on their own. Therefore, it may be useful to show a TouchGhost guide on how to use the help system as soon as a new user starts to interact with the application.

Implicit invocation strategies aim to assist the user based on the way he or she is interacting with the application. The user might be operating an interface incorrectly or very ineffectively, without knowing the existence of a more efficient technique. The system might be able to propose improved ways of interacting using the information collected by the observer, such as general usage statistics (e.g. which actions occur often, rarely or never) and interaction patterns (e.g. which actions often, rarely or never occur consecutively). For instance, if the user is only generating single-touch events in a multi-touch interface, the help system can offer information on multi-touch interaction techniques by launching the appropriate TouchGhost guides.

The system can also assist the user after a considerable delay in the user's activities or after repeated erroneous actions. Based on the user's preceding input, it can visualize the possible actions the user can perform to complete a gesture, similar to ShadowGuides [Freeman 09]. In case of erroneous actions, it can be hard to predict what the user is actually trying to achieve. In situations like that, the help system can only recommend the most likely guides to the user. Care should be taken when implementing implicit invocation strategies, since too many uncalled-for or misguided interruptions will likely disturb the user, increasing the cognitive load and significantly reducing the user's overall performance.

In public spaces, a dedicated "engagement" strategy can continuously invoke the available TouchGhosts one after another when the system has been inactive for some time. Passers-by will become more aware of the system and its intended use, while passive bystanders will be attracted to explore the new possibilities of the touch-based interface.

2.4 TouchGhost architectures

The software architecture of a TouchGhost interface determines how the help system is integrated into the underlying application. In this section, we elaborate on two different architectures that we used to develop TouchGhost interfaces, and the requirements they impose on the development of applications. The concept of a TouchGhost interface is, however, not bound to a specific architecture, and can be applied in many other ways, as we make evident in the next chapter.

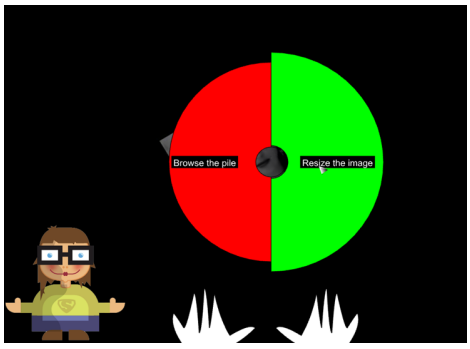
2.4.1 COMETs toolkit

The first TouchGhost interface, shown in Figure 2.4, was implemented with the use of the COMETs (COntext Mouldable widgET) toolkit [Demeure 08]. The underlying application is a very simple picture browser that arranges photographs in several piles, and allows users to look through and resize these photographs. The user can call for help by pressing a large avatar. Next, a pie menu appears on top of the interface components, and the user can select a specific action. The TouchGhost interface then shows the user how to perform that action by means of animated virtual hands.

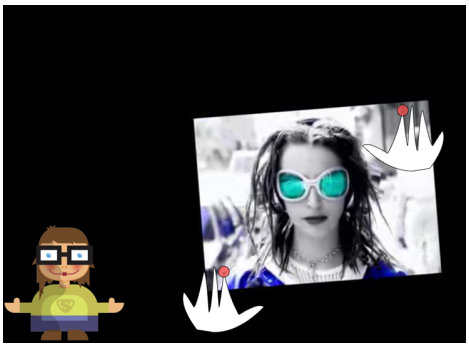
The COMETs toolkit is a special-purpose user interface toolkit that allows querying the underlying semantics of a user interface. This toolkit is based on high-level interactors (e.g. a "choice" widget instead of a radio button widget), and a COMETs widget can be decorated with meta-data, expressing the widget's role in the interactive system. As a result, we can query a concrete user interface, which is defined as a graph of COMETs, for the meaning of user interface components (e.g. which task they support) and for relationships between different parts of a user interface (e.g. whether a "draggable" object is "droppable" into a particular zone). Although the use of the COMETs toolkit is not really necessary, it provides a valuable level of abstraction to link help to an application, which is why we decided on this toolkit to implement a first TouchGhost interface.

To provide the required help, TouchGhost meta-objects can associate visual guides with every action available in the user interface. If an interface supports, for instance, actions such as playing, pausing and rewinding videos, then the visual guides demonstrating these actions are assembled in a video meta-object. By using the COMETs toolkit, such a meta-object can not only be associated with a single user interface component (e.g. a picture, Figure 2.5a), but also with a set of user interface components (e.g. two objects

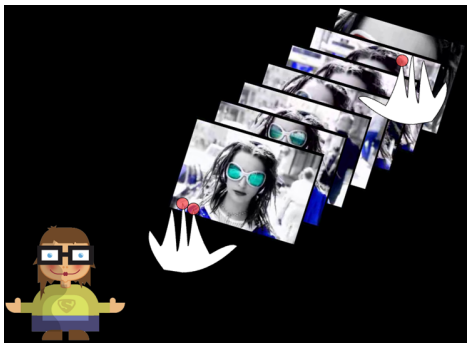
(a) A large avatar is always visible in the left corner of the screen.



(b) After pressing it, a menu appears on top of the interface components.



(c) After selecting the right item from the menu, the virtual hands show the user how to resize the picture.



(d) After selecting the left item from the menu, the virtual hands show how to browse through a pile of pictures.

Figure 2.4: Example of a "virtual hands" visualization and explicit invocation using a pie menu in the COMETs toolkit [Demeure 08]. The underlying application is a simple picture browser that arranges photographs in several piles.

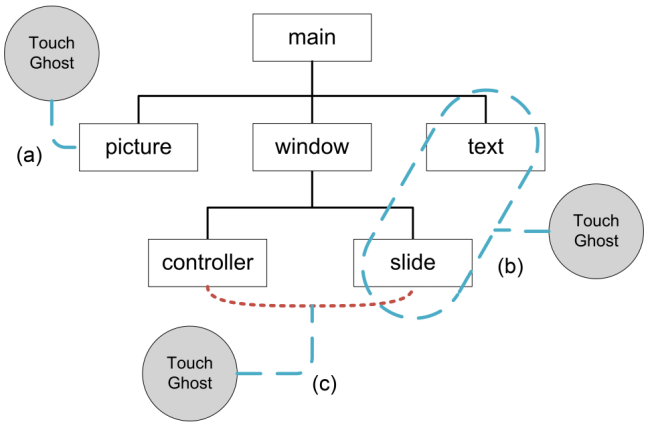


Figure 2.5: TouchGhost objects attached to different parts of a COMETs interface: (a) to a single component, (b) a set of components, (c) a relationship.

that can be merged, Figure 2.5b) or a relationship between several user interface components (e.g. a controller that pilots a slideshow, Figure 2.5c). As a TouchGhost interface usually consists of several meta-objects, a TouchGhost manager traverses the graph of COMETs, making the necessary visual guides available through the user interface.

The COMETs toolkit is suited for different modalities, including multi-touch, but its rich semantics come at a price: the toolkit has a steep learning curve and does not integrate with established toolkits such as the Microsoft .NET framework[1].

2.4.2 Microsoft .NET framework

For our second TouchGhost application, we opted for the Microsoft .NET framework, because of its widespread use, rich feature set and support for multi-touch interaction. The proposed architecture allows easy integration of a TouchGhost interface into new or existing .NET applications and does not impose one single solution, but offers the freedom to select or add appropriate invocation and visualization strategies at will.

One of our goals was to create a software architecture that allows a loose coupling between the help system and the underlying application, in order to minimize the modifications that need to be made to that application. Integrating a TouchGhost interface involves the following three steps:

[1] `http://www.microsoft.com/NET`

1. Create and configure a TouchGhost manager.

2. Associate a TouchGhost meta-object with each user interface component that should support help, and specify relationships if needed.

3. Register all these interface components to the TouchGhost manager.

The TouchGhost meta-object is similar to the meta-object in the COMETs architecture. This meta-object "enhances" a user interface component with the necessary information that typical invokers and visualizers require, such as the list of interaction techniques supported by the component and which guides are associated with those techniques. Compared to the COMETs architecture, a meta-object can only be associated with a single user interface component, not with a set of interface components or a relationship between several interface components. To overcome this limitation, relationships between components can be specified in the TouchGhost meta-object. What kind of additional information the meta-object may need, depends on the type of visual guides that are to be supported, as we will discuss shortly in Section 2.4.3.

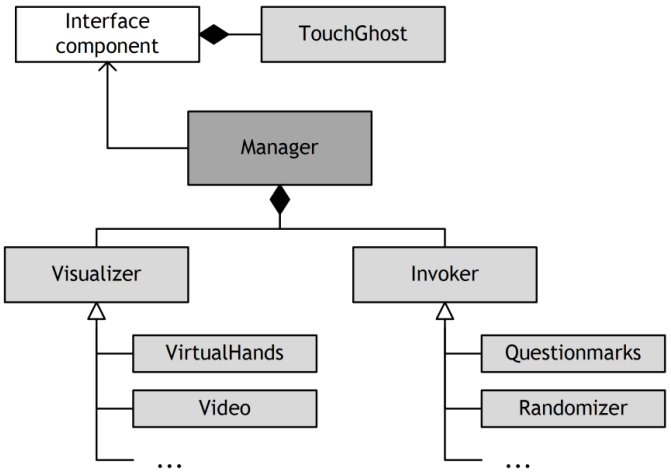


Figure 2.6: The Microsoft .NET architecture. Enhanced interface components are registered to the TouchGhost manager, which sets up the invocation and visualization strategies.

The TouchGhost manager is responsible for setting up the invocation and visualization strategies. To that end, it manages an abstract invoker and visualizer, as summarized in Figure 2.6. Concrete invocation and visualization

strategies have to be added to the manager, either by choosing from existing strategies or by implementing custom alternatives. The selection of strategies is usually made beforehand, during design, but it is also possible to select strategies at runtime, from the collection of strategies defined in the programming code. Finally, the enhanced interface components are registered to the TouchGhost manager, which makes them available to the invokers and visualizers. As a result, a strategy has access to both the interface component itself, as well as the additional meta-data.

2.4.3 Required meta-data in TouchGhost objects

Depending on the type of visualization, the TouchGhost meta-object may need to collect additional information on the user interface component. Guides that are based on videos, text or images are visualized in a separate media player. In that case, the meta-object merely maps each interaction technique supported by the interface component to the corresponding guide. If applicable, the position, size and orientation of the related user interface component can be used to position the media player conveniently in its surrounding area.

The more complicated guides, such as the virtual hands, require additional information to be included in the TouchGhost object. Since the virtual hands reproduce the actual interaction techniques, the visualizer requires information on how those techniques should be performed. To that end, a TouchGhost object can include a collection of "Gestures". Each gesture is composed of at least one "GesturePart", which holds event data such as the type of event (e.g. press, move, release), when and where the event has to occur, which finger should trigger it, and so on. When a guide is invoked, the virtual hands perform the interaction technique based on this meta-data.

2.4.4 Manipulating the actual user interface

To provide the help within the current context of use, the virtual hands manipulate the actual interface that is in front of the user. To maintain a loose coupling between the help system and the application, the virtual hands emulate real input from the user's hands. Whenever a virtual hand moves or touches the surface, it will generate exactly the same event as a real user would (e.g. press, move, release). The underlying application will simply react to events, whether their source is real or virtual. As a result, the virtual hands offer an added advantage during development and debugging, as they can be used to simulate multi-touch gestures on a regular desktop computer.

If actual interface components are being manipulated by the help system, it is sometimes necessary to restore them to their original state. The virtual hands might, for instance, show the user how to delete an object by actually performing the action. Of course, we do not want this action to be permanent, so we need to be able to restore the previous state of the application. A straightforward solution to this problem is to take advantage of the "undo" of the application, if such functionality is at hand.

Another option is to make a temporary clone of the component in question. This implies, however, that the component needs to have a method to clone it. If such functionality is not available, it either needs to be implemented, or the application will not be able to show the help for this particular action on the actual user interface component. To avoid this issue, the application can switch to a different visualization strategy, showing a short demonstration video for example.

2.5 Illustrative TouchGhost implementations

Using the aforementioned Microsoft .NET architecture, we developed a number of invocation and visualization strategies. As a test case, we implemented a simple .NET multimedia application, in which users can browse through and manipulate (piles of) pictures in various ways, and watch video fragments by means of a lightweight video player. Individual pictures can be added to an existing pile of pictures, which means that this particular action involves a relationship between two different user interface components. This relationship needs to be specified in the TouchGhost meta-object associated with the picture component, as stated in Section 2.4.2.

The developed strategies are primarily intended for multi-touch systems in public spaces, since these environments impose some challenging requirements on the accessibility of the user interface. In public spaces, users typically interact with the system over a short time-span, and thus have limited time to learn and explore the possibilities of the user interface. Additionally, we can hardly make any assumptions about the target users, since they are usually very diverse. Therefore, these invocation and visualization strategies need to be as accessible as possible.

In Chapter 3, we present more TouchGhost implementations. Those implementations build upon the results of an initial evaluation of the strategies discussed in this section and have a very specific focus, for instance on supporting multiple users.

2.5.1 Invocations

Our first strategy to invoke help is the explicit "question mark" invoker, where a large help button is always visible at the top of the application, as depicted in Figure 2.7a. When the user touches this button, the help is activated. Figure 2.7b illustrates how the help button changes into a cancel button, while smaller help buttons appear on the interface components that contain one or more TouchGhost guides. All these buttons are slightly animated, to draw people's attention, and will disappear once the user touches the cancel button or after some time of inactivity. Pressing one of the smaller buttons calls a context-sensitive pie menu, portrayed in Figure 2.7c, listing the available guides of that particular interface component. Once the user selects a menu item, the associated guide is shown.

As a secondary invocation strategy, we developed an implicit strategy that continuously invokes random guides when the user interface has been inactive for some time. By touching the surface, the invoker will be deactivated. This strategy is mainly intended to make people more aware of the system and its intended use. To ensure that people are not afraid of interrupting the continuous stream of guides, we put a text message on the screen, encouraging them to touch the surface.

2.5.2 Visualizations

Our first strategy to visualize help, shown in Figure 2.8, demonstrates an interaction technique by showing a short video of someone performing the technique step-by-step. When the invocation strategy activates a TouchGhost guide, a media player emerges in one of the corners of the screen and starts playing the corresponding video. Once the video comes to an end, a replay button appears for just a few seconds, after which the media player disappears. The small cancel button is always present, so the user can close the video at any moment. Since the demonstration videos are typically very short, the usual functionality such as play, pause or seek is not available. This strategy allows users to interact with the interface while the media player is active.

Figure 2.9 illustrates the second visualization strategy, which effectively simulates real hands by displaying two virtual hands on top of the application. These hands demonstrate the interaction technique on the actual interface component the user is interested in. In other words, this guide shows you exactly what to do in the current context of use. The user can interrupt the guide at any time by touching the large cancel button at the top, as can be seen in Figure 2.7. After the demonstration, a replay button will appear next

(a) A large, animated help button is always visible at the top of the application.

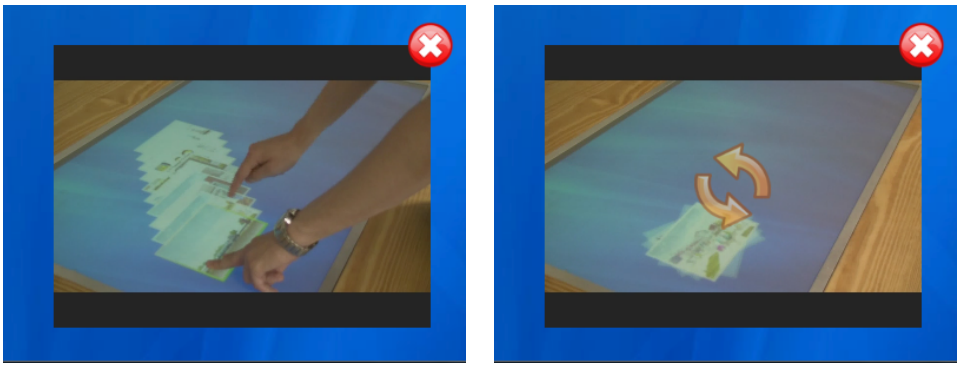
(b) After pressing it, smaller buttons appear on top of the interface components.

(c) Pressing a component's button calls a pie menu that lists the available guides.

Figure 2.7: An example of an explicit invocation strategy, using the "question mark" invoker and context-sensitive pie menus.

to the cancel button for a brief moment, so users can watch it again without having to go through the menu. During the virtual hands demonstration, a translucent colored layer covers the entire interface, except for the related components, similar to the stencils technique [Kelleher 05]. This layer directs the user's attention to the right interface component and prevents the user from interacting with other components, which could otherwise conflict with the actions of the virtual hands.

The virtual hands are currently represented by 2D bitmaps, with animated dots indicating a press or release. The 2D representation of the hands and



(a) Someone showing how to browse through a pile of pictures step-by-step.

(b) When the video comes to an end, a replay button appears for a few seconds.

Figure 2.8: Example of a demonstration video, with someone performing an interaction technique step-by-step. Since demonstration videos are typically very short, the usual media player functionality such as play, pause or seek is not available.

fingers can be limiting at times, for instance when trying to depict a flicking or pinching motion. Although it would require a lot more effort, it is possible to replace the current bitmaps with 3D models that are much more expressive. These models will match real hands more closely and can actually imitate the press or release motion of a finger. We expect this to have a positive effect on the users' first reaction, since they will need less time to get acquainted with the concept of the virtual hands.

2.5.3 Multi-user strategies

Multi-touch interfaces lend themselves perfectly to collaborative applications. However, having multiple users interact with the application simultaneously, introduces some new challenges regarding help. The invocation and visualization strategies we described in Section 2.5.1 and Section 2.5.2 are usable in a multi-user environment, but they might not be the optimal solution. Furthermore, observations of MultiSpace [Everitt 06], CityWall [Peltonen 08] and MuTable [Schneider 10] suggest that there may be ways to support or enhance supportive collaboration and social learning.

Collaborative work often involves periods of tightly coupled group activities, alternated with more loosely coupled individual work [Bergqvist 99, Tang 06]. A group of users frequently starts to collaborate on one topic

Figure 2.9: Example of the "virtual hands" visualization strategies, showing how to browse through a pile of pictures. The animated red circles on top of the fingertips indicate which fingers need to be pressed.

and eventually splits into several subgroups, working separately in multiple threads. Such threads close, split off and merge repeatedly, and a collaborative system should support fluid transitions between activities, and between personal and group work [Scott 03]. Considering these group dynamics, it is important that one user's actions do not unnecessarily interrupt others. For that reason, animated hands are for instance more difficult to implement in a multi-user setting, since the hands take control over the interface and several instances may be demonstrating interaction techniques simultaneously to different users. This inevitably leads to cluttering and confusion. In addition, the virtual hands "lock" the parts of the interface that are involved in the demonstration of the technique, meaning that others might have to wait for the animation to end as well. This locking is necessary to avoid user actions that invalidate the current demonstration, such as closing or deleting an involved object.

Depending on the properties of the hardware setup, there can also be a problem of orientation, for example when users are sitting or standing around

an interactive table. When users invoke the help system of a particular interface component, the help widgets (e.g. the pie menu, media player, virtual hands, etc.) are given the same orientation as that component. We assume that users habitually orient an object toward themselves when interacting with it, or that the system automatically rotates it toward the users. However, some interface components cannot be rotated, and some users like to share the help with other users. As a more advanced solution, the invocation strategy can try to detect the orientation of user's fingers to create an orientation-aware help system [Wang 09]. One might also simply consider the use of a visualization strategy such as demonstration videos, since users can easily rotate the media player in the direction they see fit.

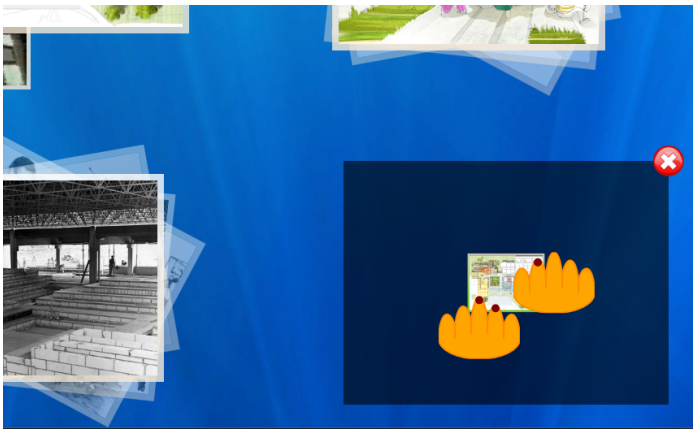


Figure 2.10: Virtual hands visualized in a separate overlay window to accommodate multiple users. The window is positioned in the bottom right corner of the screen, and has a cancel button. After the demonstration, a replay button will appear.

Since we also wanted to have the animated virtual hands available in collaborative applications, we created a new visualization strategy, shown in Figure 2.10. It basically demonstrates the interaction techniques in a separate overlay window, containing a scaled down copy of the related interface components. The overlay window is positioned in one of the corners of the screen and has cancel and replay buttons, similar to the media player in Section 2.5.2. In this way, we avoid cluttering when multiple instances are active, the animated hands no longer interrupt other users, and the window can be rotated to meet the user's wants. However, we also lose the advantage of showing the actions on the interface components of the actual user interface that the user needs to manipulate. In Section 3.3 of Chapter 3, we present an evaluation of a

very similar multi-user strategy, and we look into aspects such as supportive collaboration and social learning.

2.6 Conclusion

A TouchGhost interface facilitates the discovery and learning of multi-touch interaction techniques. Visual guides, such as animated virtual hands, are merged with the actual application to demonstrate the supported interaction techniques, if possible within the context of use. In this chapter, we explored the concept of a self-explanatory TouchGhost interface and discussed a number of strategies to invoke and visualize help. We elaborated on two TouchGhost architectures, one based on the COMETs toolkit and one on the Microsoft .NET framework. The COMETs toolkit is suitable because it is based on high-level interactors, and thus provides a valuable level of abstraction to link help to an application. The Microsoft .NET framework, on the other hand, is very widespread, and our .NET architecture allows easy integration of a TouchGhost interface in new or existing .NET applications due to a loosely-coupled architecture.

In a multi-user setting, problems may arise when several users invoke the help system at the same time. Animated hands are, for instance, more difficult to implement in a multi-user setting, since we need to avoid confusion and unnecessary interruptions. Four or six instances of the animated hands may be demonstrating interaction techniques simultaneously, which will inevitably lead to cluttering. Depending on the hardware setup, there is also the problem of orientation. Other visualization strategies can be more practical, such as showing each user the help in a separate and, if necessary, rotatable widget.

For now, we focus on multi-touch interfaces on large interactive surfaces, since there is an important need for these types of solutions, especially when targeting walk-up-and-use environments such as public spaces. However, we believe our approach can fit other devices (e.g. tablets, smartphones) and ways of interacting (e.g. free-hand gestures, tangible interaction) as well, since a TouchGhost interface helps users to discover interaction techniques that are not directly perceivable by the graphical representation of the user interface.

In the next chapter, we present the results of our user studies on different invocation and visualization strategies that we presented throughout this chapter, in both single-user and multi-user environments.

Chapter 3

Evaluation of different TouchGhost strategies

Contents

3.1 Introduction

In Chapter 2, we presented the concept of our self-explanatory TouchGhost interface, and we discussed various TouchGhost strategies to invoke and visualize help, each with different advantages and disadvantages. In this chapter,

we evaluate a number of those approaches by way of user studies. The first evaluation, found in Section 3.2, targets a few general invocation and visualization strategies, while the second evaluation, presented in Section 3.3, targets strategies that are focused on a collaborative environment. Both sections follow the same pattern, as we present the participants and devices involved in the study, the tasks that those participants had to complete, our experimental design and the procedure we followed, the outcomes of the study, and various observations and points of discussion. We end this chapter with some general conclusions that summarize the outcome of our user studies.

3.2 Evaluation of single-user strategies

We conducted an initial evaluation of one invocation ("question mark" invoker) and two visualization (video and virtual hands) strategies described in the previous chapter. We added a third visualization, namely textual help, to establish a baseline for comparison. With the textual help, we took a minimalistic approach, providing action- and task-oriented instructions [Carroll 98]. Figure 3.1 illustrates the visualizations used in the experiment.

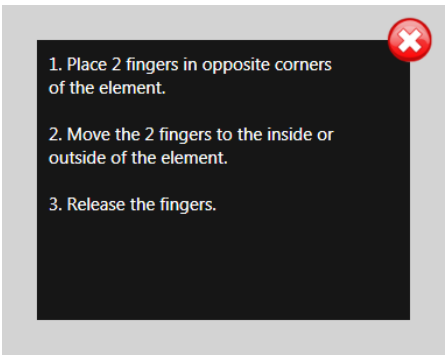
In this first evaluation, we are mainly interested in seeing how a single user interacts with the system and how we can improve the current approach. The primary goal of this evaluation is not to prove that one visualization strategy is better than another, but to make an assessment of the strengths and weaknesses of the various methods.

3.2.1 Participants and apparatus

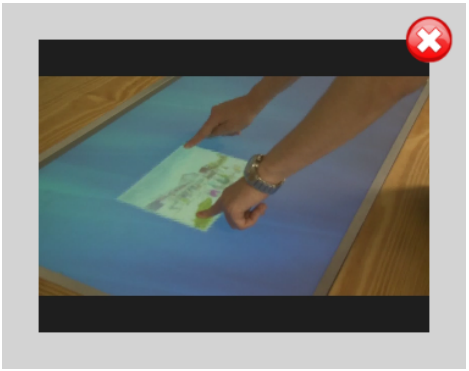
We recruited nine volunteers, six female and three male, ranging in age from twenty-two to thirty-one. All participants were recruited among university staff and students, and either have a computer science background, or have experience working on computer science projects in one way or another. We asked about both their experience with multi-touch surfaces, as well as their familiarity with gesture-based interfaces. One participant was an expert with multi-touch surfaces, while another person indicated to have a lot of experience with gesture-based interfaces, due to extensive use of an iMac Touchpad. However, the average rating across the participants was "very little" to "some", both with multi-touch surfaces, as well as gesture-based interfaces.

We conducted the experiment on a custom-built tabletop, based on Frustrated Total Internal Reflection (FTIR) [Han 05], as shown in Figure 3.2. The interactive tabletop offers a 50-inch (127 centimeters) touch-sensitive surface,

(a) Virtual hands.



(b) Textual description.



(c) Demonstration video.

Figure 3.1: The three single-user visualization strategies we evaluated.

with the rear projection providing a high-definition resolution of 1920 by 1080 pixels. The setup uses a Point Grey Firefly camera to detect input (640 by 480 pixels at 60 Hz) and can handle an arbitrary amount of concurrent touch points.

The recognition software of the tabletop, FTIRCap [Cuypers 08], handles the camera and projection calibration, analyzes the captured frames and sends the detected touch points to client applications. The image capture, analysis and transmission occur at a frequency of 60 Hz, for fluent low-latency interaction. To communicate with a client application, FTIRCap uses UDP together with either a deprecated custom protocol or the TUIO protocol [Kaltenbrunner 05]. The TUIO data can be transformed into native

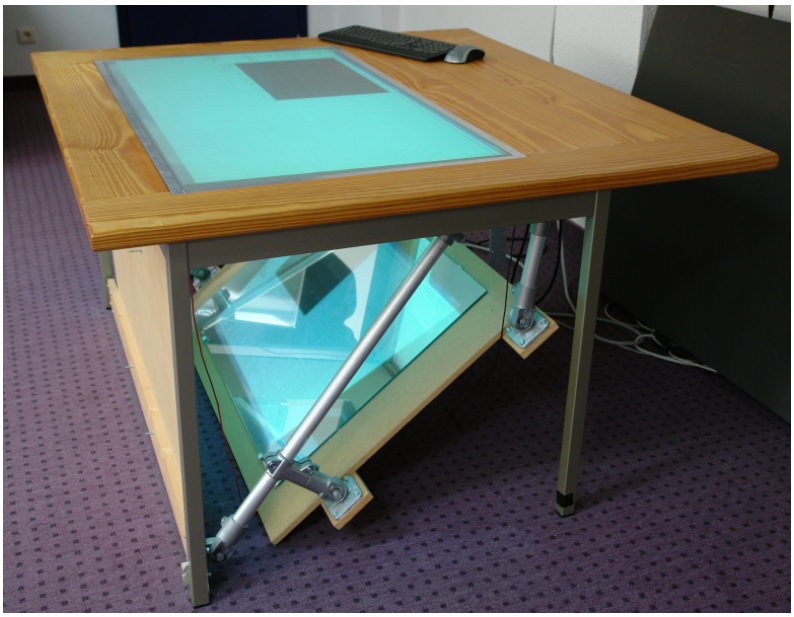


Figure 3.2: The custom-built FTIR tabletop used during the evaluation provides a 50-inch touch-sensitive surface, with a resolution of 1920 by 1080 pixels.

Microsoft Windows 7 touch events with Multi-Touch Vista[1], for example when running applications using the Microsoft .NET framework.

3.2.2 Tasks

For the experiment, we created three task sets, each connected to a different widget:

- The picture widget supports "add the picture to a picture pile", "show the gray values", and "adjust the overlay color of the picture".
- The picture pile widget (contains several pictures) supports "go to the next or previous picture", "browse through the entire pile at once by expanding it", and "scatter the pictures across the surface".
- The video player widget supports "play or pause the video", "modify the playing speed", and "change the audio volume".

[1] `http://multitouchvista.codeplex.com`

Each of these sets encloses an ordered sequence of tasks that the participant was asked to complete by performing the appropriate gestures. In addition to three gestures that the user had not yet encountered in a prior task, each widget also supported two common gestures, namely moving and resizing. The gestures ranged from very basic single-finger gestures to more complex gestures involving multiple fingers. A session ended as soon as the participant correctly performed all the gestures from the three task sets, which took about ten to fifteen minutes. To give an overview of the various gestures, we list the explanations we provided in the in-game textual help in Appendix A.1.

3.2.3 Experimental design

We used a balanced Latin Square within-participant design, with the three different visualizations as well as the three task sets, to counter-balance the order effects. To collect qualitative feedback, we used a post-experiment questionnaire, which can be found in Appendix A.2, to assess which visualization the participants liked and what features they liked best. A five-point Likert scale was used in the questionnaire, with 1 indicating strong disagreement and 5 indicating strong agreement. We also measured the discovery rate and observed the participants throughout the experiment to gather information on their behavior and the type of errors they made.

3.2.4 Procedure

Beforehand, the participants were asked to read a brief introduction, which described the experiment and some practical information. However, we did not explain the invocation or visualization strategies, since we wanted to simulate a walk-up-and-use environment. We did introduce the participants to the multi-touch hardware, as several of them had never used it. We wanted to avoid errors due to participants not maintaining a sufficient amount of pressure on the surface while moving, something that happened quite often during the pilot studies. The participants could try the surface in a very simple application that only visualized the detected pressure points.

We encouraged the think-aloud protocol in our introductory text. One observer took notes throughout each session about actions and things said by participants. Participants could invoke the help system at any time and as little or as often as they wanted, but they were not allowed to ask the observer for help of any kind. They were allowed to try a task without consulting the help, but an overabundance of repetitive guessing and fooling around was discouraged by the observer to ensure participants stayed focused on the task at

hand. After the three task sets, the participants were asked to fill in a questionnaire concerning the different strategies. Participants were encouraged to give as much feedback as possible.

3.2.5 Results

In this section we provide the analysis of the invocation and visualization strategies. The findings are mainly based on the subjective evaluation provided by participants through the questionnaire, and on the analysis of some of our observations during the experiment (more observations are reported in Section 3.2.6). For clarity, we concisely list the questions from the questionnaire, which can be found in its entirety in Appendix A.2:

- Q1: The help system was easy to activate.
- Q2: I understood the help system without any explanation.
- Q3: The help system explained the gesture clearly.
- Q4: After consulting the help system, I could easily replicate the gesture.
- Q5: The help system allowed me to discover the gesture quickly.
- Q6: Overall, I found this technique effective.
- Q7: I did not find reading the text bothersome.
- Q8: I would like to have more controls (pause, navigation bar, ...).
- Q9: The graphical feedback (red dot to indicate a press) was clear.
- Q10: I would like to have more controls (rewind, speed up, ...).

First, we will have a look at the invocation strategy. A few gestures were rather common (e.g. move, resize), but the majority of the gestures was very difficult to find out through "guessing", as several of them required the use of three or more fingers. As a result, each participant had to use the help at least once per task set. All the participants were able to invoke the help system without any explanation and ranked its ease-of-use highly in the questionnaire (Q1: $mean = 4.76$, $\sigma = 0.44$).

All participants managed to complete all the given tasks, so the discovery rate was 100% across the board. Some errors were made, but the participants were always able to correct themselves. We observed a few notable differences in the type of errors between the different visualization strategies, as we

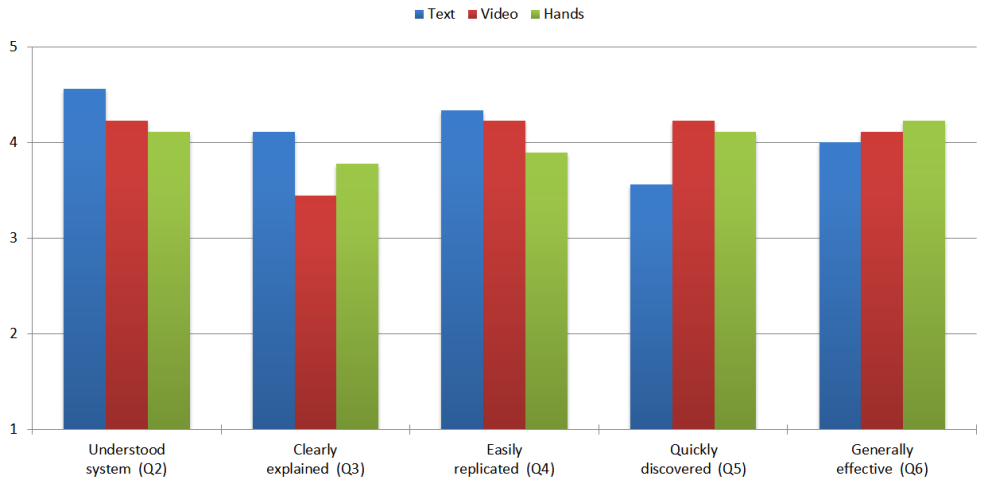


Figure 3.3: Means of the common questionnaire results for the three single-user visualization strategies we evaluated.

discuss throughout the subsequent paragraphs. The questionnaire results of the common questions related to the visualization strategies are summarized in Figure 3.3 (due to the limited number of participants, the differences are statistically insignificant).

When asked if they understood the workings of the visualization strategy without any explanation (Q2), they ranked text ($mean = 4.56$, $\sigma = 0.73$) slightly above video ($mean = 4.22$, $\sigma = 0.97$) and virtual hands ($mean = 4.11$, $\sigma = 1.05$). These results were expected, since the textual help is very straightforward and resembles traditional help systems the most, while participants reported that they needed a little time to get acquainted with the virtual hands and the way press and release events were visualized. On this topic, they also stated that the graphical feedback (Q9) was clear ($mean = 4.22$, $\sigma = 0.83$). Next, we evaluated if the help system explained the gesture clearly (Q3). As simultaneous actions of multiple fingers were not always recognized right away, videos explained gestures the least clearly ($mean = 3.44$, $\sigma = 1.24$) compared to textual help ($mean = 4.11$, $\sigma = 0.78$) and animated hands ($mean = 3.78$, $\sigma = 1.20$). Participants proposed adding visual clues to the videos, to draw people's attention.

Participants could reproduce the gestures a bit more easily (Q4) with the textual help ($mean = 4.33$, $\sigma = 0.71$) and video ($mean = 4.22$, $\sigma = 0.97$). The virtual hands ($mean = 3.89$, $\sigma = 1.45$) did not allow the participants to interact with the application during the animation. Text and video, on

the other hand, gave the participants the opportunity to perform the gestures while the help was shown on the screen. In case of the textual help, we observed that participants occasionally first read the entire text and then executed the gesture step-by-step, while consulting the textual instructions again. The text widget had to be closed manually, while the media player automatically disappeared after playing the video and showing a replay button for a few seconds. Some participants pressed the replay button and performed the gesture together with the video, because they wanted to be absolutely sure they did it right. The participants did not express a strong opinion on having more controls (pause, rewind, etc.) on the video player (Q8: $mean = 2.67$, $\sigma = 0.87$) or virtual hands (Q10: $mean = 3.33$, $\sigma = 1.00$).

The virtual hands ($mean = 4.11$, $\sigma = 0.93$) and video ($mean = 4.22$, $\sigma = 0.83$) allowed participants to discover gestures the quickest (Q5), thanks to their visual nature. Participants had to interpret the textual help ($mean = 3.56$, $\sigma = 1.24$) and map it to actual actions on the interface component. For instance, the text contained no information on which hand or finger to use, which led to users trying to perform some gestures in very uncomfortable or even physically impossible manners. A few participants suggested speeding up the animations of the virtual hands, since they had to wait for the animation to finish before they could perform the task. However, the majority of participants simply canceled the visual guide as soon as the gesture was clear to them. One participant suggested using sketches as a visualization strategy or adding iconic representations of the gestures to the pie menu. This would offer a quicker view of the gesture and would improve the system's usability for the more experienced users.

Overall, the participants found all the visualization strategies to be effective (Q6), with no clear preference for one or the other (hands: $mean = 4.22$, $\sigma = 0.83$; video: $mean = 4.11$, $\sigma = 0.93$; text: $mean = 4.00$, $\sigma = 0.71$), although some participants did note the animations were more fun. When asked if they did not found reading the text bothersome (Q7), participants tended to disagree ($mean = 2.67$, $\sigma = 0.71$), stating that they did find it somewhat bothersome.

3.2.6 Other observations and discussion

Our experiment shows that each participant succeeded in executing all tasks, which indicates that the visualization strategies provided sufficient and appropriate information to discover and learn the gestures. However, our observations, the think-aloud remarks and the feedback given in the questionnaire

form provided some additional insights and uncovered a number of potential improvements.

When the participants invoked the textual help or videos, some of them did not notice the help widget at first. It was positioned at the bottom right of the application and since the screen is quite large, participants would not see it if they were focusing on the pictures in the middle of the screen. In the case of videos this led to people missing the first part, meaning they had to restart the video once it was finished. We deliberately placed the widget in one of the corners, so users would have plenty of space to carry out the gestures. However, we should either explicitly draw people's attention to the widget when it appears on the screen, or we should position it so that it would be noticed even when focusing on items in the middle of the screen.

On several occasions, we noticed users accidentally performing a gesture. Often, they had difficulties replicating the unintentional gesture or figuring out what effect it had on the interface. Nielsen [Nielsen 10, Nielsen 11] reported similar issues with accidental activation occurring when users touch things by mistake or make a gesture that unexpectedly initiates a feature. In that case, a help system that allows users to ask why something happened would be preferable. Myers et al. [Myers 06] propose such an applications framework, which allows users to ask questions about why things did or did not happen and how the related features of the application can be used.

The textual help can clearly convey information such as "put two fingers close to each other, on top of the element". The video or animated hands, on the other hand, show the two fingers on top of the picture and in close proximity. Some participants have a tendency to generalize this to "put two fingers down somewhere", which results in the gesture not being recognized by the system, or even being recognized as one of the other gestures supported by the system. In contrast, a few participants tried to reproduce every gesture as accurately as possible, without making any generalization. The video shows, for example, how to resize a picture by putting the left index finger on the left-hand side of a picture and the right index finger on the right-hand side of a picture. While those actions could be generalized as "put any two fingers on the picture", some participants tried to replicate the gesture exactly like that, even if it was far from the most comfortable solution in their particular situation.

Animations and videos take a certain amount of time to watch, and the speed of the actions that are being performed needs to be carefully balanced so that novice users have no problems to interpret what is happening. Experienced users, however, want a quicker way of discovering gestures. One of

the options is showing a storyboard of the actions instead of an animation or video, since different users can process that kind of information at different speeds. Help must serve the needs of both novice and experienced users, so ideally the help system would be able to adjust to a particular user. Ryall et al. [Ryall 06a] give a similar suggestion to customize the level of hints delivered in an educational tabletop application on a per-user basis, reflecting each user's current level of mastery. It is also possible to take into account a user's history, for example earlier answers and interactions.

The current implementation of the virtual hands does not allow users to interact with the application for the duration of the animation. Some users, however, wanted to imitate the hands during the demonstration, to practice the gesture. One solution is to provide a copy of the actual interface component in a sand-boxed environment, to allow users to safely experiment with the gesture, much like the practice area of the GestureBar [Bragdon 09] for example. The virtual hands could also demonstrate the gestures in several discrete steps. Once the user successfully reproduces a step, the hands show the next step. However, most gestures are fairly simple, and thus difficult to partition in discrete steps.

In summary, the results of this first experiment indicate that users do not have a distinct preference for one method or another. However, our observations revealed some important consequences of existing limitations, such as not being able to interact during an animated sequence. Furthermore, we noticed textual help being excellent at clearly conveying some particularities of an interaction. In the next section, we describe our second experiment, which takes into account these conclusions.

3.3 Evaluation of multi-user strategies

In Section 2.5.3 of the previous chapter we put forward some considerations on multi-user TouchGhost strategies, and we proposed a possible solution by demonstrating interaction techniques in a separate overlay window, containing a scaled down copy of the related interface components. In this section, we present the setup and results of our evaluation of such a multi-user strategy, and we discuss how the results from the first evaluation influenced the design of this multi-user approach. This research was done in collaboration with dr. Anastasiia Beznosyk, who focused on the effects of help on the cooperative and communicative characteristics of casual games.

In this second experiment, we compared two different conditions: animated help in a separate overlay window and textual help. To this end, we developed

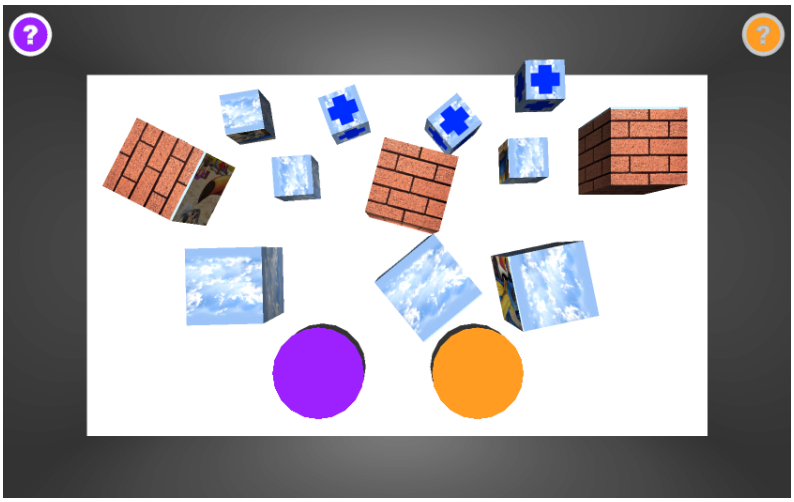


Figure 3.4: The puzzle game with light, heavy and small puzzle pieces, special "enlargement" cubes, and two avatars (the purple and orange "disks" at the bottom of the screen).

a collaborative 3D puzzle game in Unity[2], an integrated game development tool for creating 3D games. We favored Unity over the Microsoft .NET architecture we used in our first experiment because it enabled us to quickly create an interactive 3D environment with rigid-body physics. Figure 3.4 shows the casual two-player puzzle game.

3.3.1 Participants and apparatus

We recruited twenty-eight volunteers to participate in the experiment, three female and twenty-five male, ranging in age from twenty-two to forty-five. All participants were recruited among university staff and students, and either have a computer science background, or have experience working on computer science projects in one way or another. Most participants indicated to have very little experience with a multi-touch tabletop and six participants had no experience with tabletops at all. All except two participants had some experience with playing any type of multi-player games. According to the self-evaluations on the questionnaire, the average player experience with multi-player games was 2.9 on a scale from one (never played) to five (playing almost every day).

[2] `http://unity3d.com`

The hardware setup we used during this evaluation, a custom-built multi-touch tabletop, is identical to the one described in Section 3.2.1. To avoid any issues with orientation within the user interface, we asked all participants of a session to stand on the same side of the tabletop. A video camera recorded the sessions for later analysis.

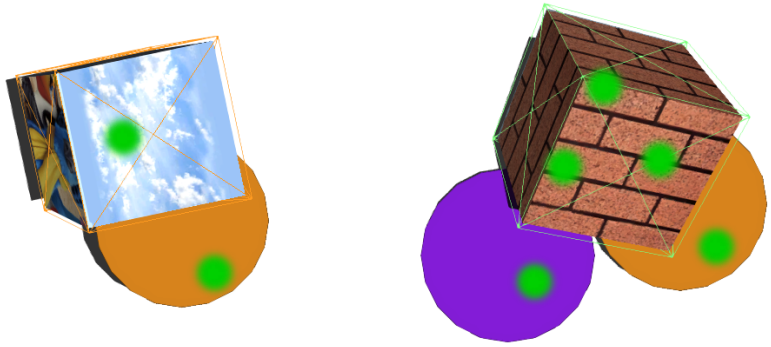
3.3.2 Tasks

The participants were divided in pairs, and each pair had to complete a puzzle. In collaborative applications, interacting in parallel often is the most common form of "multi-user" interaction [Peltonen 08, Jacucci 10]. However, our game is designed in such a way that it encourages and sometimes requires two users to collaborate tightly by means of simultaneous input [Ryall 06a] to complete the puzzle, as demonstrated in Figure 3.5. In this section, we first have a look at the puzzle game and the various ways of manipulating the puzzle pieces, and next we discuss the two different help systems the application provides in support of these tasks.

Figure 3.5: Two users helping each other to solve the puzzle.

Solving the puzzle

The puzzle consists of nine puzzle pieces, scattered across the tabletop's surface. Each puzzle piece is represented as a cube, with a picture on one of its sides. However, not all puzzle pieces act in the same way, as we include three different kinds of pieces: three light puzzle pieces, three heavy puzzle pieces, and three small puzzle pieces. In addition, three special "enlargement" cubes are present. Those three cubes serve a special purpose and are not part of the actual puzzle that needs to be assembled, as they do not have a picture on one of their sides. The type of a cube is visually indicated by the cube's texture, as illustrated in Figure 3.4: light and small pieces have a sky texture, heavy pieces a brick texture, and the enlargement cubes have a sky texture with a blue cross on top.



(a) Selected by one user, the bounding box has the same color as the avatar. This puzzle piece is being moved.

(b) Selected by two users simultaneously, the bounding box is green. This puzzle piece is being rotated in 3D.

Figure 3.6: Double tapping the avatar selects/deselects the nearest colliding cube. Selection is indicated by the bounding box. The green dots are visual feedback provided by the application to indicate successful recognition of touch points.

In a regular tabletop application users would be able to interact directly with the puzzle pieces. However, our tabletop setup does not support the identification of user input, like the DiamondTouch [Dietz 01] for example, which means the application cannot support straightforward identity-differentiating group input and logging [Ryall 06a]. To enforce actions to be executed by a particular user and to log the necessary data about the actions of individuals, each user is assigned an avatar (the purple and orange "disks" at the bottom of

the playing field in Figure 3.4) that is used to interact with the cubes. In order to manipulate a cube, the user first moves his or her avatar to the cube and then selects it by double tapping the avatar. Double tapping the avatar for a second time deselects the cube. Selection is indicated by showing the bounding box of the cube, in the same color as the involved avatar, as demonstrated in Figure 3.6a. Because users frequently forget to sustain enough pressure on the multi-touch surface for their input to be accepted, green dots provide visual feedback to indicate successful recognition of touch points.

By keeping one finger on the avatar, the selected cube is slowly lifted up in the air. To provide extra feedback, a heavy cube is lifted slower and less high than a light or small cube. While lifted, any type of cube can be moved or rotated. However, heavy cubes will move and rotate at a greatly reduced speed. To overcome this slowdown, heavy cubes can be selected and lifted by two avatars simultaneously, restoring the normal manipulation speeds and lifting height. To indicate that two users selected a cube, the bounding box is colored green, as can be seen in Figure 3.6b.

To solve the puzzle, users need to perform four types of actions: moving puzzle pieces, rotating puzzle pieces in 2D, rotating puzzle pieces in 3D, and enlarging small puzzle pieces by using the special enlargement cubes. To give a brief overview of the different manipulations, we list the explanation provided in the in-game textual help. The textual explanation is accompanied by Figure 3.7, which visually represents the different elements of the game.

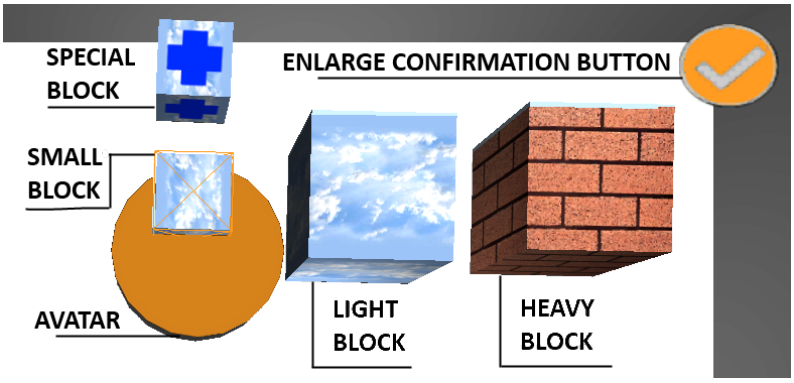


Figure 3.7: The image that accompanies the in-game textual help for the orange avatar, showing the various types of cubes, the avatar, and the confirmation button for enlarging small puzzle pieces.

- **Select** *To select a block, drag your avatar to it. When the avatar and block collide, tap the avatar twice in quick succession. To deselect, double tap the avatar again.*

- **Move** *To move a block, keep pressing your avatar with one finger to lift the block off the floor and at the same time drag the block around with another finger.*

- **Rotate** *To rotate a block, first lift it off the floor. To rotate in 2D, put two fingers on the block and move one of them. To rotate in 3D, put two fingers on the block and then spin the block around by moving over it with a third finger.*

- **Enlarge** *To enlarge a block, move a small block and special block close to each other. While both blocks are selected, simultaneously press the two confirmation buttons located at the top of the screen.*

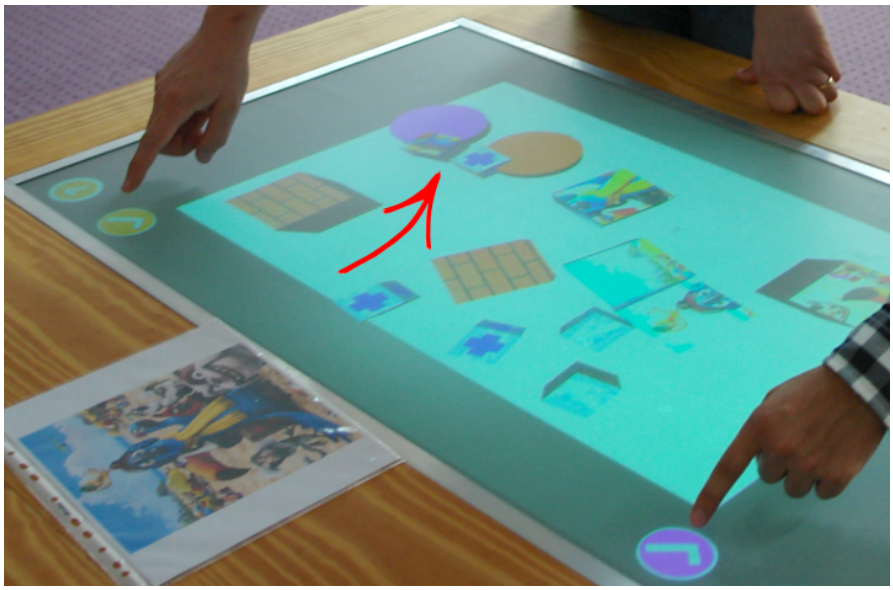
- **Heavy block** *Heavy blocks move and rotate slower than light blocks. However, a heavy block can be selected and lifted by two avatars simultaneously to increase the speed of those actions.*

Figure 3.6a illustrates the movement of a puzzle piece, by lifting and dragging the cube. Figure 3.5 and Figure 3.6b depict the rotation in 3D, again lifting the cube and then using three fingers to spin it in a particular direction. Two users are collaborating to manipulate the heavy puzzle piece at an increased speed, by lifting the cube with two avatars simultaneously. Finally, Figure 3.8 shows two users working together to enlarge a small puzzle piece, by moving the small cube and a special enlargement cube close to each other and pressing the two confirmation buttons at the same time.

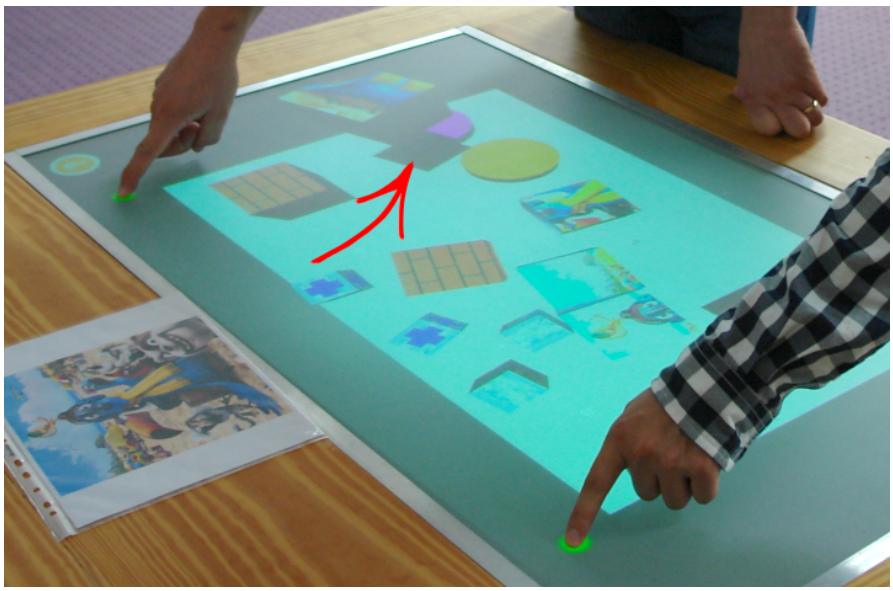
The puzzle game carries on until all the small puzzle pieces are enlarged, and the puzzle pieces are (more or less) in the correct orientation and position, as in Figure 3.9. The application does not check for correctness automatically, this is simply done visually.

Help

Similar to the first experiment, users have to discover and learn all the different actions by themselves, using the TouchGhost interface that is present in the application. Half of the pairs play the puzzle game with the textual help



(a) Move a small cube and special cube close to each other (indicated by red arrow) and simultaneously press the two confirmation buttons.



(b) The small cube and special cube are then transformed into one normal-sized, heavy cube (indicated by red arrow).

Figure 3.8: Small puzzle pieces can be enlarged by two people, with the help of special enlargement cubes.

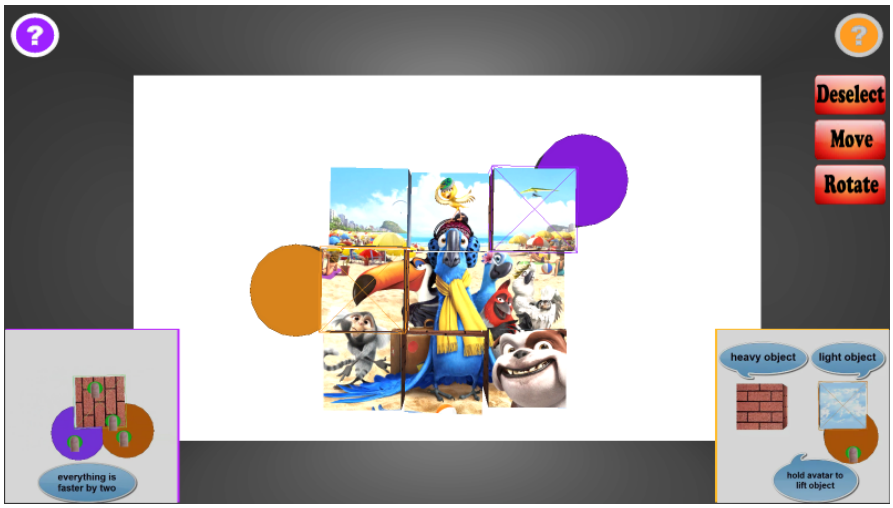
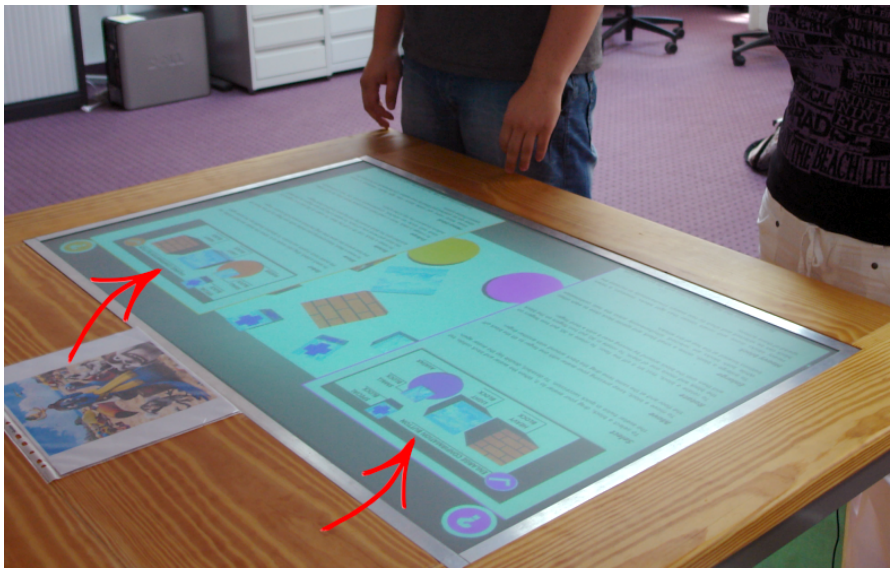


Figure 3.9: The completed puzzle, with all the pieces more or less in the correct orientation and position. Animated help is being shown in both corners of the screen.

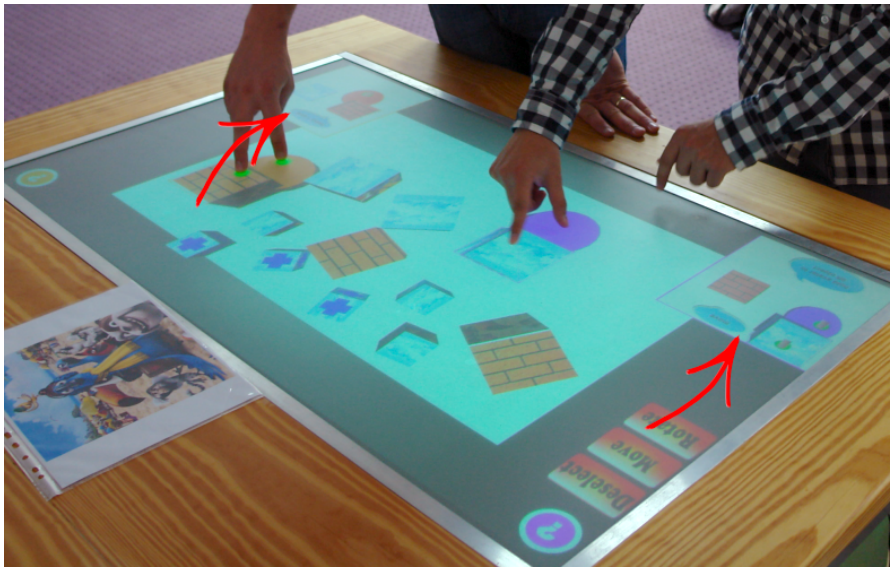
we listed in the previous section, the other participants get animated help as a visualization strategy, demonstrating interactions in a separate overlay window. In both cases, users are able to interact with the application while consulting the help, so they can try actions while reading text or watching an animation. This solves the problem revealed in Section 3.2.6, where the virtual hands temporarily prohibited users from interacting with the application.

The textual help has only one manner of invocation: pressing the button with the question mark, located at the top of the interface. When the game is started, the textual help automatically opens and users can close it by pressing that same button, as indicated in the help ("To open or close this help, tap the button with the question mark"). Each user has an individual panel that displays the help on his or her side of the tabletop, as illustrated by Figure 3.10a. The textual help is accompanied by an image, Figure 3.7, that indicates the various objects in the environment.

The animated help visualizes the interaction techniques in a separate overlay window, containing a scaled down copy of the relevant parts of the user interface (i.e. the avatars, cubes, and buttons), as described in Section 2.5.3 and shown in Figure 3.9 and Figure 3.10b. Virtual fingers are animated on top of the cubes and avatars, and to indicate a press during an animated sequence, we use the green dots that are used as visual feedback when the user touches the surface. To help with the problems of users not generalizing particular as-



(a) Textual help. An image of the various components is followed by descriptions of all the actions, as reported in the previous section.



(b) Animated help. The most important aspects of an animated sequence are annotated, as depicted in detail in Figure 3.11.

Figure 3.10: Two users consulting the help (indicated by red arrows) at the beginning of the game.

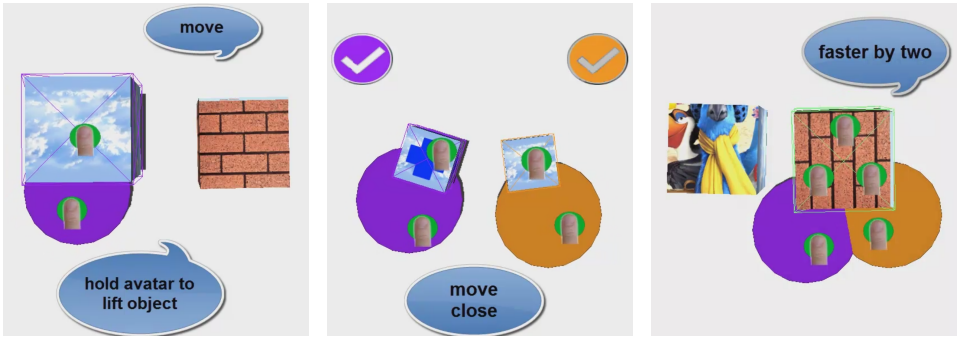


Figure 3.11: A few examples of the use of virtual fingers, with green dots indicating presses. Important aspects of the animated sequences are highlighted through text bubbles annotations.

pects of a demonstration or generalizing some aspects too quickly, as reported in Section 3.2.6, we annotate the animated sequences with a minimal amount of text. The annotations point out important aspects of the interaction, such as the type of cube that is involved, so users know that they have to pay attention to this. Figure 3.11 shows three examples of this visualization strategy, including the use of virtual fingers, green dots to represent presses, and text bubbles annotations.

As the animated sequences are annotated, they were prerecorded rather than generated on the fly like in the Microsoft .NET application we evaluated in Section 3.2. However, both visualizations, textual help and animated help, are individualized to some degree. Figure 3.7 contains an orange avatar, for example, while the other user is shown an image with a purple avatar. The same is true for the animations, which will always show the avatar of the corresponding user. The advantage of prerecorded animations is that they grant us a lot more flexibility as to what an animation can include. Animations that are generated on the fly only show one particular action in one specific situation, and it is up to the user to decide if that action can be generalized or not when encountering a comparable situation. When a user learned how to rotate a light puzzle piece, she might just replicate that action when trying to rotate a heavy piece, without ever finding out that heavy pieces can be manipulated faster with two users. The prerecorded animations not only show how to rotate a light puzzle piece, but also quickly point out that heavy pieces behave in a slightly different way.

In contrast to the textual help that offers the explanations of all the possible actions at once, animations are invoked in a step by step manner. When the game starts, an implicit invocation strategy triggers the animation that shows users how to select a puzzle piece with their avatar. Once the user successfully selects one of the cubes, the application automatically demonstrates how to move it, and after a certain amount of time how to rotate it in 2D and 3D. If the user selects a small cube, the application also explains how to enlarge it a short while after all the basic actions (e.g. movement and rotation) have been shown. If a single user selects and manipulates a heavy cube, the invocation strategy displays an animation as a reminder after a while, to indicate that this type of cube can be manipulated faster with the help of the other user.

Initially, we intended to show one animated sequence that demonstrates all the possible actions one after another at the start of the game, comparable to the textual help. However, pilot studies revealed that users had quite some difficulties coping with the long animation. In contrast, reading the text was not a problem, as users could process it at their own speed and pause in between to try out some of the actions. We also used these pilot studies to optimize the speed and length of the animated sequences, to make sure that people are able to read the annotations comfortably and interpret all the actions. This resulted in animations that take around twenty to twenty-five seconds.

Although the implicit invocation strategy triggers all the animations automatically at one moment or another, users can also activate a specific animation through an explicit strategy, in the form of a simple menu. This menu is context-sensitive, as it only lists actions that are currently available to the user. If the user has nothing selected, only “Select” will be available in the menu, and once the user has selected a cube, all the actions that are available on that type of cube will be listed in the menu, as illustrated in Figure 3.12. Users can watch animations as often as they like, and are always in control of the system: animations that are explicitly invoked by the user through the menu will never be interrupted by the implicit invocation strategy, while animations that are automatically shown will be interrupted if the user initiates an animation through the menu.

The overlay windows that show the animations are located in the bottom corners of the application, to avoid any overlap. However, during the pilot studies, we observed issues similar to the ones reported in Section 3.2.6, as users did not always notice an animation immediately. The problem is less distinct, as we now have two users working on the tabletop instead of just

Figure 3.12: The context-sensitive menu of the animated help. The purple avatar has nothing selected, so only "Select" is available in the corresponding menu on the left. The orange avatar selected a small puzzle piece, so all related actions are available in the menu on the right.

one, which results in users standing closer to the corners and thus noticing something popping up in their corner more easily. To avoid users missing the first few seconds of an animated sequence, we added an "introduction" to the sequence, showing the name of the interaction technique for a few moments to draw people's attention. We also experimented with other methods, such as displaying an arrow that points in the direction of the animation near the user's avatar, but this only led to clutter and confusion, as some users for instance assumed that they had to move their avatar in the direction of the arrow.

3.3.3 Experimental design

For this experiment, we used a between-subjects design. The twenty-eight participants were randomly divided in fourteen pairs, with seven pairs testing the animations and the other seven pairs getting the textual help. The condition was randomly assigned to the pairs. Subjective data was collected through a post-experiment questionnaire (Appendix A.2), using a visual analogue scale: participants were asked to give a rating by placing a mark at the appropriate position on a continuous ten-centimeter line, representing the point between "not at all" and "very much" that they felt represented their perception best.

We also observed the participants throughout the experiment to gather information on their individual and collaborative behavior, the use of the help and the type of errors they made.

We measured and logged task completion time, the amount of time players worked individually (i.e. selected a cube on their own) or collaboratively (i.e. selected a cube together), and the number of times help was accessed. Afterwards, the video recordings of all the sessions were analyzed to determine the amount of time players spent watching or reading help together, the first time and amount of time players communicated, and the first time players collaborated. The logged amount of individual and collaborative work was also revised during this analysis, to provide a more comprehensive assessment of those aspects.

3.3.4 Procedure

Similar to the procedure in Section 3.2.4, the participants were asked to read a brief introduction beforehand, describing the experiment and some practical information about the hardware. Again, we did not explain the invocation or visualization strategies, since we wanted to simulate a walk-up-and-use environment. During the experiment, we encouraged the think-aloud protocol and we filmed each session, recording all the participants' actions and communication. Two observers also took notes about actions and things said by participants. Participants could invoke the help system at any time and as little or as often as they wanted, but they were not allowed to ask the observers any questions.

The participants had a printed image of the finished puzzle at their disposal, as seen in Figure 3.10. Since there was no snapping mechanism in place to position puzzle pieces very precisely against each other, it was not required to achieve a perfect result. Upon completion of the puzzle task, the participants were asked to fill in a questionnaire about their former experiences and their findings regarding the puzzle task, the collaboration with their partner, and the help system. Participants were encouraged to give as much feedback as possible.

3.3.5 Results

In this section, the use of the textual help and animations in the puzzle game are analyzed. The findings are based on performance logs, the subjective evaluation provided by participants through the questionnaire and analysis of the videos recorded during the experiment.

The help system

Just as in the first experiment, all participants managed to complete the given task. It took approximately thirty minutes for each pair to complete the puzzle. The results of the subjective evaluation regarding the help, provided by participants through the questionnaire, are summarized in Figure 3.13 (the differences are again statistically insignificant). These results are clearly comparable with the results of the single-user experiment, reported in Section 3.2.5. Here are the related questions from the questionnaire, which can be found in its entirety in Appendix A.2:

- Q11: It was clear how to use the help without any explanation.
- Q12: The help explained the required actions clearly.
- Q13: After consulting the help, I could easily replicate the actions.
- Q14: It was clear how to perform cooperative tasks.
- Q15: The help allowed me to discover the required actions quickly.
- Q20: Help took me out of the game experience.

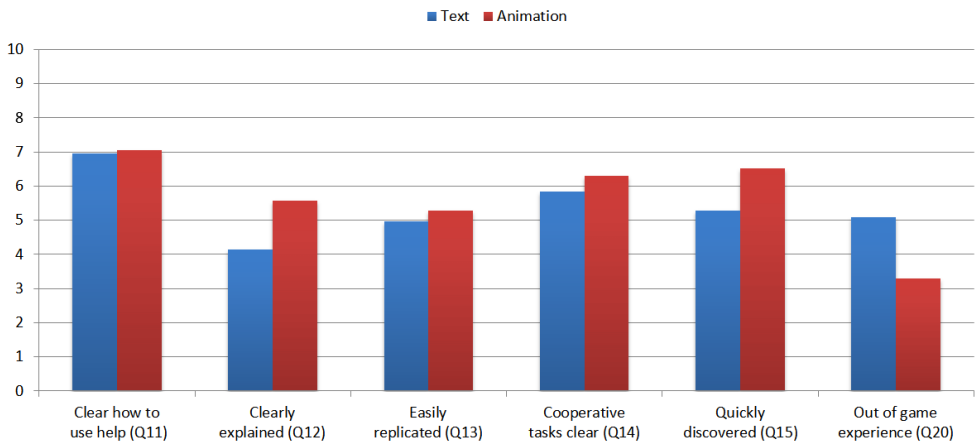


Figure 3.13: Means of the questionnaire results regarding the help for the two multi-user visualization strategies we evaluated.

As we used avatars to interact with the cubes in the puzzle game, it was very difficult to find out the required actions through trial-and-error. All the

participants were able to invoke the help system without any explanation and ranked its clarity highly in the questionnaire (Q11) in both the text ($mean = 7.0$, $\sigma = 2.6$) and animation ($mean = 7.1$, $\sigma = 2.6$) condition. With the textual help, some participants first read the complete text and then started to try the various actions, while others only read the first part and then tried that particular action before reading and trying the next part. In some cases, one participant read part of the text out loud, while the other participant performed the actions.

All participants had to reopen the textual help at least a handful of times at a later stage. To be precise, each pair reopened the textual help 9.6 times on average, or more or less five times per participant. In case of animated help, the animations that were automatically shown by the implicit invocation strategy were mostly disregarded, except for the one that was shown when the application started. Once a participant successfully selected a cube, for example, the invocation strategy automatically triggered the animation on how to move, but that animation was typically ignored as the participant first tried deselecting and selecting other cubes. Afterwards, the participant explicitly initiated the animation through the menu. Pairs used the menu about 9.4 times on average to trigger a specific animation.

When asked if the visualization strategy explained the gesture clearly (Q12), animations ($mean = 5.6$, $\sigma = 1.9$) were rated higher than text ($mean = 4.1$, $\sigma = 2.4$). This is in contrast with the first test, but there are some notable differences between both experiments that explain this outcome. The animations are now annotated to highlight the most important aspects, thereby combining the advantage of text and animations to a certain degree. The text is also "enhanced" by including one image that shows all the essential components, but a graphical representation of each action may further improve the clarity of the textual descriptions. The disadvantage of not being able to interact with the application during animations is no longer present, and as a result there is almost no difference between animations and text when it comes to replicating the actions easily (Q13), with animations ($mean = 5.3$, $\sigma = 2.3$) scoring even marginally better than text ($mean = 5.0$, $\sigma = 2.5$) this time.

The more advanced actions in the second experiment were more complex, as some of them required two users to cooperate. If we consider the enlargement of a small puzzle piece, the animation clearly visualizes two avatars being used, while the textual description merely implies the use of two avatars by stating that both cubes need to be selected. Results from the questionnaire indicate that the textual help ($mean = 5.9$, $\sigma = 2.0$) performed only slightly

worse than animations ($mean = 6.3$, $\sigma = 1.7$) in making clear how to perform cooperative tasks. However, in our observations we clearly noticed quite a few participants struggling with enlargement in case of textual help. It took them a while to figure out that they needed to cooperate to enlarge an object, as some first tried, for instance, to select two cubes with only one avatar, which is not possible.

As indicated in Section 3.3.2, the textual description of a particular interaction technique was limited to a few short sentences, while the annotated animations took about twenty to twenty-five seconds. The animated help ($mean = 6.5$, $\sigma = 2.1$) allowed participants to discover gestures quicker (Q15) than the textual help ($mean = 5.3$, $\sigma = 2.5$), thanks to the visual nature that makes it easier to interpret the help. We regularly noticed users trying a gesture while reading the textual help out loud, and actually counting the number of fingers they put down on the tabletop. Compared to the first experiment, the length and speed of animation were less of an issue, since participants could interact while the animation was playing. However, a few participants expressed their annoyance when they wanted to watch the last part of an animated sequence again, but they had no means of skipping the first part.

An interesting conclusion concerning textual and animated help in the context of gaming is the reasonably large difference regarding the game experience (Q20). Participants indicate that textual help ($mean = 5.1$, $\sigma = 3.2$) took them out of the game experience, which is less of an issue with animated help ($mean = 3.3$, $\sigma = 1.8$).

Learning from different sources

The questionnaire also included some questions related to the way participants learned the necessary actions:

- Q16: I learned most of the actions through the help.
- Q17: I learned a lot by talking to my partner.
- Q18: I learned a lot by watching my partner's actions.
- Q19: I had to consult help multiple times for a particular action.

The results are shown in Figure 3.14. The differences between text and animation are negligible, but it is interesting to see how participants rate the different ways of gathering information. Participants learned the most from

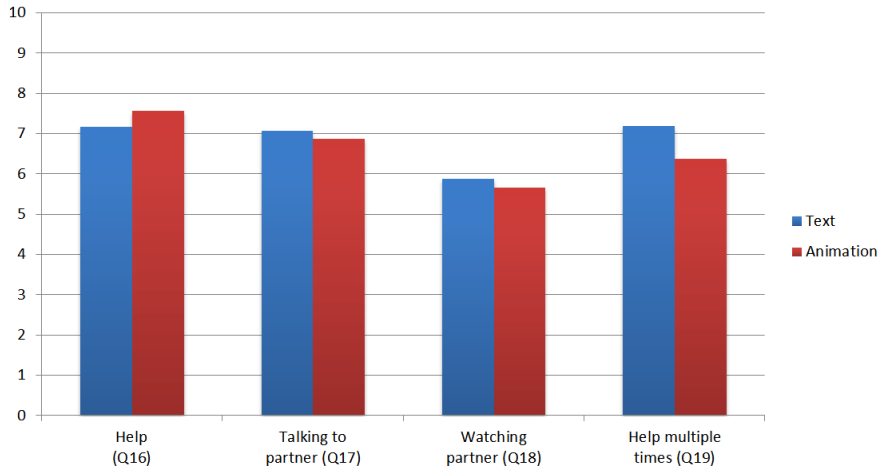


Figure 3.14: Means of the questionnaire results regarding how participants learned the necessary actions.

consulting the help (Q16) and from talking to their partner (Q17). Watching their partner's actions (Q18) is a close third, which we found somewhat surprising, as we expected participants to learn more by watching each other than by talking. However, our observations indicate two possible reasons for this. First of all, explaining something verbally allows you to continue your current task, a behavior that we observed repeatedly. In addition, participants regularly explained an interaction technique when seeing the other person doing it wrong. They often tried to correct the mistakes of the other participant verbally at first, and only if that did not succeeded, they demonstrated how to do it. These results highlight that supportive collaboration and social learning are important aspects in a multi-user environment.

We already reported that almost all users had to consult the help multiple times, which is confirmed by the questionnaire (Q19). Although the logs show that participants opened the textual help and animations roughly the same amount of times, there is a slight difference noticeable in Figure 3.14. This inconsistency is actually very logical, as animations only explain one action and then automatically close. Textual help, on the other hand, contains the descriptions of all the actions, so the user can open the help once and then read about as many actions as he or she wants, which only counts as one access to the help in our logs. In addition, since the textual help needs to be closed manually, it was occasionally left open for some time. The difference was even more pronounced during our observations, as we regularly noticed

participants opening an animation, but then being interrupted by the other participant and thus having to activate it again afterwards.

Level of collaboration

A secondary goal of our study was to investigate how in-game help can influence the level of collaboration during the game. In addition to the results from the questionnaire, we analyzed the video recordings of all the sessions and logged the amount of time participants worked individually or in tight collaboration, the amount of time they spent consulting help together, the first time they collaborated and communicated, and the total amount of communication. However, this part of the study primarily concerns dr. Anastasiia Beznosyk's research [Beznosyk 12] and is mostly beyond the scope of this dissertation. It will therefore not be discussed in detail, as we only look into the results that are directly related to the help system.

For the video analysis, we considered activities to be completely individual if the participants really focus on two separate tasks and do not communicate in any way. Tight collaboration, on the other hand, includes two participants either working closely together to accomplish a task (e.g. enlarging a small puzzle piece, manipulating a heavy piece together), communicating with each other (e.g. discussing a strategy, explaining actions), or consulting the help together (e.g. watching an animation together, one participant reading a piece of text to the other one). The sum of the individual and collaborative time is not necessarily equal to the total amount of time it took to complete the task, as some activities were considered to be neither completely individual nor in tight collaboration (e.g. two participants each moving a different cube with the goal of clearing a pathway).

When relating this data to the total amount of time it took to complete the task, Figure 3.15 clearly shows a statistically significant increase ($t_{12} = 2.44$, $p = 0.031$) of completely individual work in case of textual help ($mean = 31.4\%$, $\sigma = 13.8$) compared to animated help ($mean = 16.9\%$, $\sigma = 7.5$), and a statistically significant decrease ($t_{12} = -2.39$, $p = 0.034$) in tight collaboration ($mean = 29.0\%$, $\sigma = 9.2$) compared to animated help ($mean = 41.1\%$, $\sigma = 9.6$). One cause of these differences is the moment participants started their first collaboration, as it took them significantly longer ($t_{12} = 2.21$, $p = 0.047$) with textual help ($mean = 511$ seconds, $\sigma = 316$ seconds) than with animations ($mean = 214$ seconds, $\sigma = 164$ seconds).

The fact that it took longer for the first collaboration to happen can be logically explained, seeing that some pairs read the whole text before start-

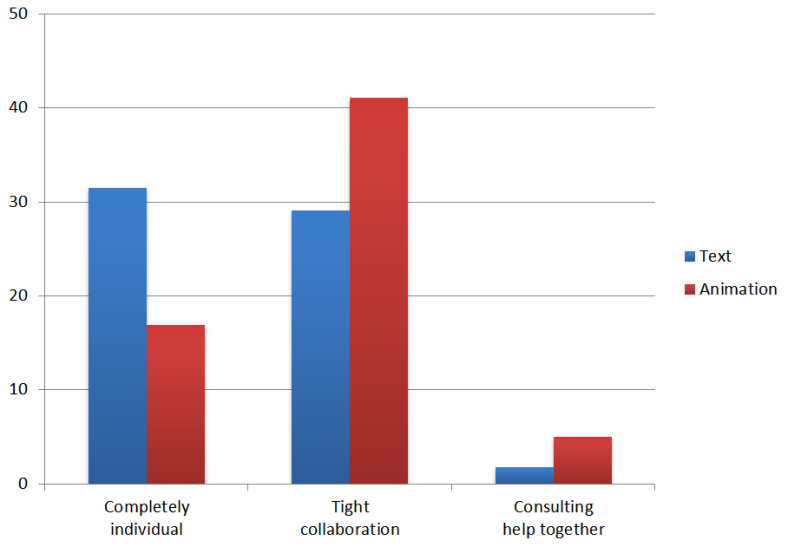


Figure 3.15: Percentages of completely individual work, tight collaboration and consulting help together in regard to the total amount of time it took to complete the task, as observed in the video recordings.

ing to interact. Nonetheless, it does not completely explain the difference in amount of individual activities and tight collaboration, as the other pairs also had to spend some time watching other animations once they figured out the basic interactions. There are, however, a few other factors to take into account. First of all, if we look specifically at how much time participants spent consulting help together (thus reading the same text or watching one animation together), we see a statistically significant difference ($t_{12} = -3.28$, $p = 0.007$) between textual help ($mean = 1.8\%$, $\sigma = 1.3$) and animated help ($mean = 5.0\%$, $\sigma = 2.3$), which accounts for part of the aforementioned difference in amount of individual and collaborative activities. Secondly, if a user is manipulating a heavy cube, the implicit invocation strategy displays an animation after a while to remind the user that the cube can be manipulated faster with two. The animated help thus invites users to collaborate more.

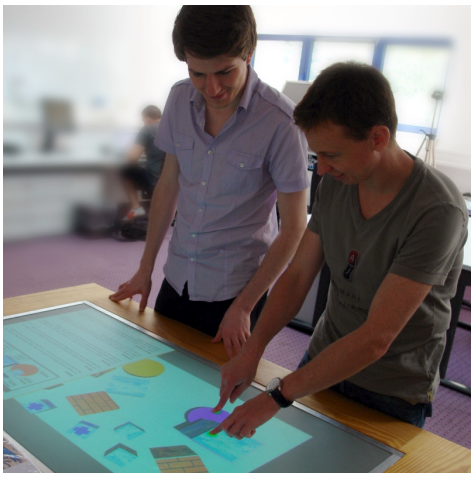
3.3.6 Other observations and discussion

We took a minimalistic approach with the annotations that are used to point out important aspects in the animated sequences, to avoid users having to read too much and animations becoming lengthier as a result. Based on the pilot

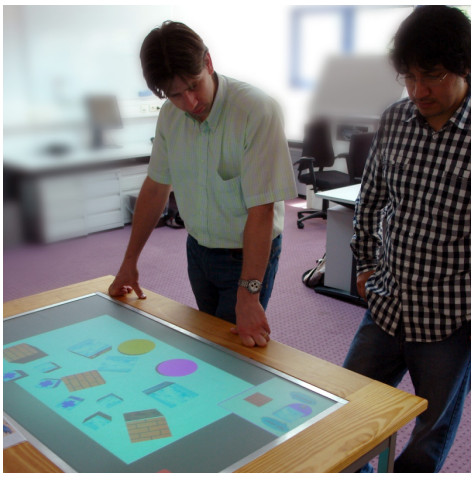
studies, we tried to identify the most essential aspects that needed highlighting. However, several participants still made errors because they overlooked some important information that was not included in the annotations. The most common mistake was putting fingers on the wrong component. To select a cube, for example, the user had to double tap on the avatar, but occasionally participants tried to double tap the cube instead. The accompanying annotation only includes the fact that the user needs to double tap, but not that it has to be on the avatar. We wrongfully presumed that the location would be sufficiently clear from the animation itself. This issue could have been easily avoided by extending the annotation.

Another issue we observed frequently, was participants accidentally performing an action without truly knowing what happened or how they did it. Additionally, some participants executed actions in an uncomfortable or uncontrolled manner, for example by using the wrong fingers or by momentarily losing contact with the surface of the tabletop during movements. Our approaches, both textual descriptions and annotated animations, only "explain" actions to the user, but do not offer a lot of feedback during or after the execution of an action, except for the green dots to confirm a successful press and the actual effect on the cube when an action is performed correctly. An approach like Gesture Play [Bragdon 10] provides both feedback and feedforward during the execution of a gesture, and clearly indicates whether the gesture is being performed appropriately. This kind of feedback improves a user's ability to successfully execute an action the way it was intended to be, and would be an excellent addition to our current approach.

While the menu to invoke animations was very easy to use, the animations shown automatically by the implicit invocation strategy were less effective. Although one pair never used the menu and learned all actions through those automatically triggered animations, other participants mostly ignored them. We expected that once a user successfully completed an action, he or she would (more or less) immediately proceed to the next action. However, this appeared to be untrue, as users repeated the same action a number of times to confirm that they understood its workings completely. This kind of unpredictable behavior makes it difficult to automatically show help at appropriate times, although simply waiting for the user to be completely inactive for a while before triggering a new animation may somewhat lessen this problem. The animation that suggests manipulating heavy puzzle pieces with two users instead of one did seem to be noticed more often, since it only starts to appear after participants learned all the basic actions and it is repeated each time a single user manipulates a heavy cube for a while.



(a) Textual help was mostly consulted individually, but occasionally one user tried to explain a description.



(b) Animated help was more frequently watched together, as indicated by the results in Section 3.3.5.

Figure 3.16: Two users consulting the help together during the game.

In Section 3.3.5 we reported that participants spent much more time consulting help together in case of animations. Our observations confirm this, as we noticed participants consulting the textual help mostly on their own, although occasionally one user tried to explain a description to the other user, as illustrated by Figure 3.16a. Animations, on the other hand, were frequently watched together, as seen in Figure 3.16b. This seems rational, as animations are easier to watch from different angles, while text is more difficult to read when not standing more or less in front of it. These insights indicate that it may be preferable to provide only one shared help panel somewhere in the middle of the screen, which would invite more collaboration and social learning. This approach has the added benefit of being more flexible regarding the number of users that the application supports, as the current approach is optimized for two users. Further research is needed, however, as we may see different results if users have to share one help panel.

In this experiment, we observed only two participants who worked together throughout each session. With different group sizes, people develop different work strategies in achieving the same collaborative goal [Ryall 04]. In other words, as the number of participants increases, we may see different behaviors,

such as a group splitting in smaller subgroups, which in turn can influence how participants deal with the help provided by the application and how they support one another. To get more insights into the behavior of a larger group of users in this particular context, additional studies are required. Furthermore, when targeting walk-up-and-use scenarios in public places, we also need to evaluate the various approaches with a more diverse selection of users, including people with very limited computer skills, children and elderly people, and so on.

In both experiments presented in this chapter, we evaluated complete gesture sets and we never investigated individual gestures. It is conceivable that particular visualizations suit specific types of gestures better than others. Virtual hands may, for instance, be more effective for multi-user gestures, while text may be better for very complex single-user gestures (or the other way around). In addition, care has to be taken with generalizing our results to any walk-up-and-use application, since both experiments were conducted in our research lab. In a real walk-up-and-use environment, users will be less committed to completing their task, and factors such as social embarrassment come into play.

3.4 Conclusion

We conducted an initial evaluation of textual help, demonstration videos and animated virtual hands. The questionnaire results indicate that users do not have a distinct preference for one approach or another, but our observations revealed some important consequences of existing limitations, such as not being able to interact during an animated sequence. Based on our experiences with the first experiment, we built a puzzle game and investigated textual and animated help in a collaborative setting. The second study shows that animated help allows users to quickly discover the available interaction possibilities, with less of a negative impact on the game experience. Animated help also has a positive effect on collaboration, as users work together to explore and learn the application.

When comparing different strategies, we should not only consider the user, as we did in our studies, but also the developer. Once the help system is in place, it is rather easy to generate the gestural information required for the virtual hands in the Microsoft .NET architecture. The process of constructing gestures can even be automated to some extent, for example by extracting the necessary data from a gesture's description in the recognition software, or by recording the events during a demonstration of the gesture (i.e. learning by

example). The text, videos or annotated animations require more attention, since they have to be manually updated each time a gesture changes or a new gesture is added.

This chapter concludes the first part of the dissertation, on self-explanatory interfaces. In the next part, we continue with our "multi-user" research, as we investigate various ways of enhancing collaboration in multi-user environments.

Part II

Enhancing collaboration in multi-user environments

Chapter 4

Focus+Roles: socio-organizational conflict resolution and access control in collaborative user interfaces

Contents

4.1 Introduction

Along with the advantages of supporting and enhancing group productivity [Galaczy 99], collaborative applications introduce a number of challenges. In particular, allowing multiple people, either co-located or remote, to interact in a shared workspace simultaneously, gives rise to uncertainty and unpredictability. This brings about several types of conflicts and intrusions, as we previously described in the scenario of Section 1.3.2. Consequentially, collaborative applications call for some interaction management, but to fully exploit the advantages of group productivity, the regular work flow should be interrupted as little as possible.

Providing concurrent, multi-user interaction allows users to take full advantage of group dynamics and different interaction styles [Scott 03]. However, concurrency can lead to conflicts, and if no synchronization mechanism is in place, the system may end up in an inconsistent state. A very common type of conflict is, for example, caused by several people trying to manipulate a shared resource at the same time. To counter this problem, numerous strategies are available: execute activities sequentially, try to merge both actions or only allow one of the users to manipulate the object at the same time, among others. In such circumstances, the floor control policy of an application determines how conflicts are handled [Dommel 97]. In addition, data sharing has to be treated carefully in a multi-user environment, since others can easily access the data and may have dishonest intentions.

In this chapter, we explore possible uses of a user's roles and focus to provide conflict handling and access control, and we present our Focus+Roles approach [Vanacken 07a]. Roles have already been applied widely, for instance in behavioral and social sciences, psychology, business and software project management, and agent system modeling [Zhu 06]. Furthermore, through observations, natural roles have been identified in CSCW (Computer-Supported Cooperative Work) systems. CoWeb [Guzdial 00] is, for instance, a tool that does not enforce or explicitly support specific roles. However, as the collaborative tool matured, several well-defined roles emerged over time. Providing new features and tools to address the concerns and activities related to those particular roles has allowed more users to become actively involved. This illustrates the potential of using role-based mechanisms in a collaborative environment.

In addition to roles, we take the users' focus into consideration. In real-life circumstances, a person's physical presence provides numerous indications relating to that person's center of attention. We notice if someone is looking at one item in particular (primarily based on head movements and eye gaze),

we often point our finger at things or use other hand gestures, distinct facial expressions may reveal a person's intentions, and so on. As a result of such a rich awareness of one another, many conflicts are avoided in a natural way. We customarily refrain from moving a document while we are aware that others are reading it, for instance. In contrast, a lack of awareness of one another's activities can cause confusion and unintentional side-effects in a collaborative environment. We therefore investigate how the users' focus can be used to handle certain consequences of a lacking awareness.

We apply our approach to a collaborative meeting environment, *iConnect*, resulting in graceful (e.g. correct in a socio-organizational context) conflict handling and access to shared data. After the related work in the next section, we present our interpretation of roles and focus, and our take on access control in Section 4.3. In Section 4.4, we give an overview of the iConnect system, and we describe how we integrated our Focus+Roles approach as a proof of concept in Section 4.5. Finally, in Section 4.6, we end this chapter with our conclusions.

4.2 Related work

In this section, we explore the existing work relating to various kinds of conflict handling and access control, the use of roles within those mechanisms, and mutual awareness in multi-user environments.

Reaching beyond social protocols. An intuitive approach to floor control is to assume that "social protocols", such as polite behavior and social standards, are adequately observed and suffice to coordinate the actions of a collaborating group of users in CSCW systems [Johanson 02]. Even though social protocols perform well in some cases, they cannot prevent or resolve numerous types of conflicts which may lead to an inconsistent system state or hinder users in their activities. Users often fail to realize the side-effects of their actions, or become confused when other users in a collaborative session operate on a shared resource simultaneously [Greenberg 94].

Numerous issues have been reported over the years, as multi-user environments that rely on social protocols were explored. In Dynamo [Izadi 03], an interactive surface that supports media sharing, users described inconveniences with conflicting interaction, such as one user closing a document belonging to someone else. Moreover, the high degree of freedom concerned users, because it may easily lead to malpractices, such as stealing other people's work. MultiSpace [Everitt 06] explores the use of an interactive table to support ad-hoc

collaboration in a multi-device environment. Problems were reported with regard to overlapping documents, as people liked to maximize their current document in the shared space, regardless of other documents that were actively being edited.

Coordination policies reaching beyond social protocols, employing direct manipulation mechanisms to avoid and resolve conflicts, can improve multi-user interfaces [Morris 04]. Solutions include the use of rank to factor in differences in privilege among users, or privatising strategies to restrict a user's access to a subset of documents. While one such policy may suffice in a constrained environment, multifaceted systems such as iConnect, described in Section 4.4, have need of a more extensive strategy that combines several approaches.

Roles as a set of privileges. A different class of policies is described in terms of roles, which are used to associate specific categories of users (e.g. authors and book editors) with particular policies (e.g. an author and editor policy). In Quilt [Leland 88], a tool for collaborative document production, different document views are provided based on the user's position in a permission hierarchy that reflects a number of social roles and communication types. Quilt includes roles such as writers, readers and commentators, which control the degree of access that individuals have to the document.

Turoff [Turoff 91] outlines a set of primitive privileges (such as read, modify, append, use and assign) for computer-mediated communication systems and defines roles as a subset of these privileges. Turoff also adds the concept of a ticket to handle unpredictable needs. A ticket allows users to pass a specific privilege that they possess to another user. Tickets can be made conditional, so they can only be used a limited number of times or during a limited time interval, and usually notify the issuer when they are used.

Kansas [Smith 98], a shared application environment, augments social protocols by considering a user's role. The role determines the amount of visual output, to avoid users being overwhelmed by too much irrelevant information, and can limit the user's input capabilities. Although this fairly restricted approach to roles is capable of preventing some conflicts, users still had to resort to verbal communication to coordinate tightly-coupled collaborative activities. Furthermore, problems arose from unintentional user actions, which may have consequences that are hard to undo.

As shown in the related work, assigning sets of privileges to specific categories of users allows a floor control policy to prevent a number of conflicts, since it basically prohibits particular users to perform certain activities. In

addition, this method can easily prevent unauthorized access to shared data. For those reasons, we also define our roles as a set of privileges, as presented in Section 4.3.1.

Roles as a set of responsibilities. Another tactic is to define a role through a set of responsibilities, specifying how different categories of users have different duties. The system can then enforce a user to perform these duties according to particular policies or protocols [Simone 95]. Such a policy is expressed as a set of rules in the organizational context, for instance a set of rules on how to handle a software bug (e.g. registering, diagnosing, and correcting). Additional rules may state that a role can be exercised only by someone with, for example, a certain skill or rank (e.g. verifying a bug correction must be done by a senior designer).

Lupu and Sloman [Lupu 97] similarly propose roles that are defined in terms of authorization and obligation policies for a particular manager position, specifying what actions that manager is permitted or obliged to perform. Their framework also makes it possible to specify interaction protocols, and how different managers should coordinate and synchronize their activities. Our approach currently limits roles to a set of privileges, and does not explicitly include responsibilities and protocols. Providing this kind of additional information to a floor control policy makes it more cumbersome, but also enables more factors to be taken into account while detecting and handling conflicts.

Access control through role-based approaches. We already discussed access control to some extent throughout the previous examples. However, the most prevalent method of authorization management is role-based access control (RBAC) [Sandhu 96, Sandhu 99, Osborn 00, Park 01, Ferraiolo 01], which associates access control rights with roles. When users are assigned the appropriate roles, they simply acquire the associated access control rights. These roles are typically created based on the various job functions in an organization, and can be easily reassigned to users. Organizational roles are fairly static, however, and transitory roles, which can alter dynamically throughout a collaborative session, are not supported as such.

Temporal-RBAC [Bertino 01] is an extension of the RBAC model that allows some form of transitory roles, through periodic enabling and disabling of roles, and temporal dependencies among such actions. In addition, Generalized Temporal-RBAC [Joshi 05] also allows expressing duration constraints on roles, user-role assignments and role-rights assignments. Other extensions, such as GEO-RBAC [Bertino 05] and Ex-RBAC [Cui 07], take into account

spatial and location-based information to activate roles. The numerous RBAC approaches mainly handle authorization management, but do not address, for instance, issues related to a lack of mutual awareness.

Edwards [Edwards 96] extended Intermezzo, a coordination infrastructure, with roles that represent not only statically-defined collections of users, but also dynamic descriptions that are evaluated at runtime. Whereas static roles are implemented by associating a list of users with a set of access control rights, dynamic roles are implemented by associating a predicate function with such a set of rights. When a user requests access, Intermezzo first applies the static roles and then evaluates the associated predicate functions to see if access may be granted. Edwards' specification language is powerful, but rather complex, which makes it difficult for end-users to add new or adjust existing policies and roles.

Coordination on interactive surfaces. The results of a study on multi-touch input versus multiple mice indicate that interactive surfaces enable fluid interaction and switching of roles between co-located users [Hornecker 08b]. Moreover, higher levels of awareness are reported in case of multi-touch input, accompanied by significantly more actions that interfere with one another. Analysis shows that interference was quickly resolved, and therefore suggests the possibility of increasing resources for dealing with interference instead of trying to eliminate conflicts. However, the study only takes into account co-located users gathered around an interactive surface.

Two customs that people uphold on a tabletop are to use orientation of objects to coordinate actions and mediate communication [Kruger 03], and to informally divide the space into areas that support shared and individual work (e.g. personal, group, and storage areas) [Scott 04]. Several studies show that people maintain a personal territory on a shared surface, and that people spend most of their time interacting in their personal territories [Ryall 04, Nacenta 07]. Using indirect techniques to interact with shared workspaces reduces territorial behavior and awareness, resulting in poor coordination. New coordination techniques, designed specifically for tabletops, can reduce conflicts arising from indirect input by allowing users to protect objects when they work near their personal areas, or by allowing users to set certain protection levels dynamically [Pinelle 09].

Building on the idea of distinct areas, the UbiTable [Shen 03] combines an interactive table with personal laptops and PDAs to support face-to-face collaboration. The workspace allows users to conveniently share documents through private, personal and public spaces. Private data is not visible or

accessible to others, and the owner always retains control over the distribution and duplication of documents, even after they have been transferred to the shared space. In iConnect, a similar notion of personal and shared spaces exists, as described in Section 4.4.1, but documents do not explicitly retain knowledge of ownership once they are transferred to a shared space, so additional access control is needed.

If we can relate the contact points on the interactive surface to a particular user, for instance by using a DiamondTouch [Dietz 01] or the identification technique we propose in Chapter 6, privileged access to widgets or documents can be accomplished through identity-differentiating widgets [Ryall 06a] or virtual lenses that allow for personalized input and output [Schmidt 10b]. Furthermore, providing visual cues on ownership, in the form of distinct borders of territories or through distinct coloring of the interactive objects, can help to maintain social protocols on a shared tabletop, reducing the need for policies in those circumstances [Fetter 11].

4.3 Focus+Roles

In the next sections, we examine the user's roles and focus: we define the terms in the context of our work and discuss their significance during collaborative meetings. In addition, we consider some useful properties concerning access control, which we relate to the collaborative items in our environment.

4.3.1 Roles in an organizational and meeting context

In our approach of using roles to provide conflict handling and access control, users can not only adopt various long-lasting roles within the organizational hierarchy, but may also switch dynamically from one transitory role to another throughout a collaborative meeting. By bringing this kind of information about users into the system, our floor control policy is capable of dynamically adjusting to those users, their characteristics, and their ongoing activities, thereby preventing a lot of common conflicts.

A user's role in an organizational context is determined by a job description, defined as a collection of responsibilities, privileges and associated organizational tasks. We call such an organizational role *static*, since it changes rather infrequently. In our approach, static roles are defined as a set of privileges, and represent lines of work such as department heads, project managers, software engineers, developers, accountants, technical and administrative support, and so on. Likewise, a person's proficiencies can be considered reasonably

fixed, as they do not change on a regular basis. Therefore, our static roles can also be based on proficiencies, and thus include C++ experts, network specialists, hardware professionals, and so on. Users may adopt several static roles at once, since people sometimes fulfill more than one role in an organization and often have several proficiencies.

In the context of a collaborative meeting, a role is determined by a user's current activities within the group, defined as a second collection of privileges. Since a user switches from one activity to another on a regular basis during a meeting [Bergqvist 99], we call this role *dynamic*. Dynamic roles represent presenters and attendees, chairpersons, brainstorm participants, facilitators, and so on. Users may adopt only one dynamic role at once, yet they can switch roles whenever the need arises. If the environment is able to detect the current activities of its users, the dynamic roles of these users can also be updated automatically.

In a collaborative environment, possibly intrusive activities such as browsing through a set of slides or altering the contents of a slide are mostly carried out by a rather narrow subgroup of users. This subgroup consists of those users taking on a dynamic role that is inherently related to the activity. One can expect, for instance, a speaker to browse through a slideshow or alter a slide, whereas an attendee will not. Users outside the subgroup generally perform non-intrusive actions such as observing the ongoing activities or taking notes. We exploit these characteristics of collaboration to prevent a range of conflicts.

To illustrate this approach, we look at our scenario from Section 1.3.2. Jane, the department head, presents an outline of what the meeting is about. In her role of speaker, Jane can navigate through the slideshow unhindered, since her dynamic role is closely related to that activity. In contrast, to prevent conflicts with other users during Jane's presentation, attendees are limited to non-intrusive actions such as observing the slideshow on a shared screen. As a result, Jake's accidental attempt to close Jane's Microsoft PowerPoint presentation is prohibited during the talk, thereby averting an inconvenient disruption.

Another example of a conflict includes two users trying to move the same object simultaneously. If Jane and Jake both try to move a particular document during Jane's presentation, Jane's action will be given priority, given her dynamic role as a presenter compared to Jake's role as an attendee. However, when several users adopt one and the same dynamic role, collaborative activities may result in a power struggle. In such circumstances, the organizational hierarchy is of overriding importance, or, in other words, the users' static roles.

In our scenario, all users take on an identical dynamic role for the duration of the open discussion toward the end of the meeting. If Jane and Jake now try to move a document simultaneously, Jane's action will be given priority because she is the department head.

4.3.2 Passive and active focus

Awareness of individual and group activities is essential to successful collaboration [Dourish 92, Gutwin 99], and in a collaborative environment that allows users to participate in both a co-located and remote manner, the natural sense of mutual awareness is inadequate. As already stated in Section 2.5.3, typical collaborative work involves periods of tightly coupled group activities, alternated with more loosely coupled individual work [Bergqvist 99, Tang 06]. A group of users frequently begins to collaborate on one topic, and eventually evolves into several smaller subgroups, working separately in multiple threads. Such threads close, split off and merge repeatedly, making it difficult for users to stay informed about all ongoing parallel activities, including the other group members' current state and goals.

The inevitable lack of mutual awareness brings about many conflicts, which have to be resolved by instructing or requesting someone to take certain actions or even through negotiation. However, such engagements disrupt the work flow. A sense of mutual awareness should be attained in the course of doing the work, without having to resort to deliberate actions. We therefore provide a method to counter the lack of mutual awareness, by taking advantage of a user's focus. We define two different kinds of focus: *passive* focus and *active* focus. A user's passive focus is determined by perceptual activities such as reading a document, watching a video or listening to an audio stream. The user's active focus, on the other hand, involves interactions such as manipulating a spreadsheet, moving or resizing an item, and leafing through a text document. In other words, active focus requires some form of user input, while passive focus implicates the perception of output.

Our scenario from Section 1.3.2 illustrates a few useful applications of focus. While people are still discussing Jane's and Jake's brain scans, Jacob wants to show another scan of a patient suffering from a similar disorder. However, Jacob is not permitted to disturb the others by overlapping their brain scans, since a large number of users are centering their attention on those scans. Therefore, the system rules out unfavorable occlusions when showing new contents on the whiteboard. Later during the meeting, Jake and Lucie are comparing the approaches of two hospitals, and Jane starts discussing a

financial report with Jacob. Jane thereby loses track of all nearby activities, and, not aware of the ongoing discussion between Jake and Lucie, may try to resize the financial report so that it overlaps some of the other documents. However, Jane is not permitted to disturb Jake and Lucie by overlapping their documents, since they are centering their attention on them.

4.3.3 Access control

In a collaborative meeting room that includes shared surfaces, such as a multi-touch tabletop, access control can be a very sensitive issue, particularly in an organizational context where some information has to be treated as confidential. Although all documents that are discussed during the meeting are available on a single surface, some belong to individuals who may wish to prohibit certain types of access by their colleagues. Access restrictions can limit a user's ability to manipulate a document, for example opening, copying, printing, editing or removing the document.

From an organizational perspective, access rights and restrictions are related to the user's static roles. A financial report, for example, mainly concerns users with an administrative role, while developers are generally not qualified to understand or handle such information. Moreover, the financial document may contain sensitive information, which should not be accessible to everyone. Instead of the traditional operating system's approach of being able to set certain permissions on a file, we allow a user's access rights to be inferred from the associated roles, in combination with the document's content type (e.g. financial report, patient file, brain scan) and sensitivity (e.g. highly confidential, internal).

To illustrate this process, we refer to our scenario from Section 1.3.2 once more. Jane shows several images of a brain scan to inform the others of a particular case. Since the patient files are accessible to all neurologists working at the same hospital, everyone is allowed to make a personal copy of Jane's slideshow, except for Lucie and the members of the remote team, who are employed at another hospital. Later, Jane opens a private spreadsheet, containing some financial prognoses. Given that this financial information is highly confidential, nobody else is authorized to distribute it, and as a result copying or printing the spreadsheet is prohibited.

Controlling the access of users is not only applicable to documents in a shared workspace. The collaborative group dynamics can also benefit from restricted access to other types of objects in the application, including menu items or toolbar buttons. Functionalities such as closing the entire workspace

or automatically tiling all items should be used with care in a multi-user environment. The dynamic roles of a user pose a solution, since the set of current activities often determines which users should be in control of the system. The speaker should, for example, be able to tile the workspace, as opposed to the attendees.

4.3.4 Overview of the Focus+Roles process

All the abovementioned approaches are incorporated into a Focus+Roles process, which can be summarized as follows:

1. When a user performs an action, we first check that user's dynamic role. If the user lacks the required privileges, the action is prohibited. Otherwise, we continue to the next step.

2. If the user's action involves accessing a document (e.g. open, copy, print, edit), we check all the user's static roles and the meta-data associated with the document to decide if that type of access may be granted.

3. To resolve if the action can be executed safely, we also take into account other users' passive and active focus, if applicable (i.e. the action involves something others may be focusing on, such as opened documents).

4. Finally, we check for any conflicting activities. If the user's action conflicts with another user's action, we decide on the outcome of this conflict by comparing dynamic roles. If both users have the same dynamic role, their static roles are the decisive factor.

In this way, the Focus+Roles policy provides effective conflict handling and access control, with minimal intrusions into the normal work flow. Throughout the next sections, we describe how this policy was implemented in an existing multi-user environment.

4.4 The iConnect environment

As a proof of concept, we added a Focus+Roles policy to the collaborative iConnect [Cardinaels 06] environment, which we already mentioned briefly in Section 1.3.2. In this section, we introduce the iConnect environment in more detail.

The IBBT GBO project *iConnect* aimed at the creation of a computer-supported collaborative meeting room, and is in some respects similar to systems such as i-LAND [Streitz 99], iRoom [Johanson 02], Dynamo [Izadi 03],

MultiSpace [Everitt 06] or WeSpace [Wigdor 09]. The iConnect environment meets a lot of the same requirements as, for instance, the WeSpace project: provide a shared display, allow the use of personal laptops, support native applications on the system, maintain interactivity of existing applications, retain user control over own data, support equitable and visible input, and provide an interactive table. These requirements were derived for a collaborative tool to support scientists conducting collaborative research across multiple disciplines [Jiang 08]. In this section, we clarify how the iConnect environment meets those requirements, and how it supports both co-located and remote participants.

4.4.1 Personal and shared workspaces

As described in the scenario in Section 1.3.2, the iConnect environment facilitates simultaneous engagement of multiple users, sharing arbitrary data and employing heterogeneous input and output devices. The key feature to accomplish such an environment is an architecture to support a seamless integration of the shared devices already present in the meeting room and personal devices users may bring along [Cardinaels 06]. Such a distributed system results in the coordinated existence of both shared and personal workspaces [Streitz 94, Rekimoto 98]. Shared workspaces allow several users to interact with shared data simultaneously, while personal workspaces are only accessible by a single user and contain private data, which is not shared with the other users unless explicitly stated.

An iConnect meeting room can be equipped with a multitude of shared devices, such as interactive whiteboards (e.g. SMART board[1]) and tabletops (e.g. DiamondTouch [Dietz 01]). Using a shared device, co-located as well as remote users may perform actions that are reflected on the shared workspace. The users' personal devices, such as laptops and PDAs, act as a personal workspace, which can be used in conjunction with the shared workspace (tablets and smartphones were less widespread when the iConnect project started in 2005, and are therefore not explicitly supported, but those portable devices have similar characteristics to PDAs). Figure 4.1 shows a typical iConnect setup during a meeting.

To implement the abovementioned functionality, the iConnect environment includes a workspace server and a number of clients, all interconnected by a network. Users not able to physically attend the meeting can interact with a duplicated workspace remotely. The workspace server hosts a workspace that

[1] `http://www.smarttech.com/smartboard`

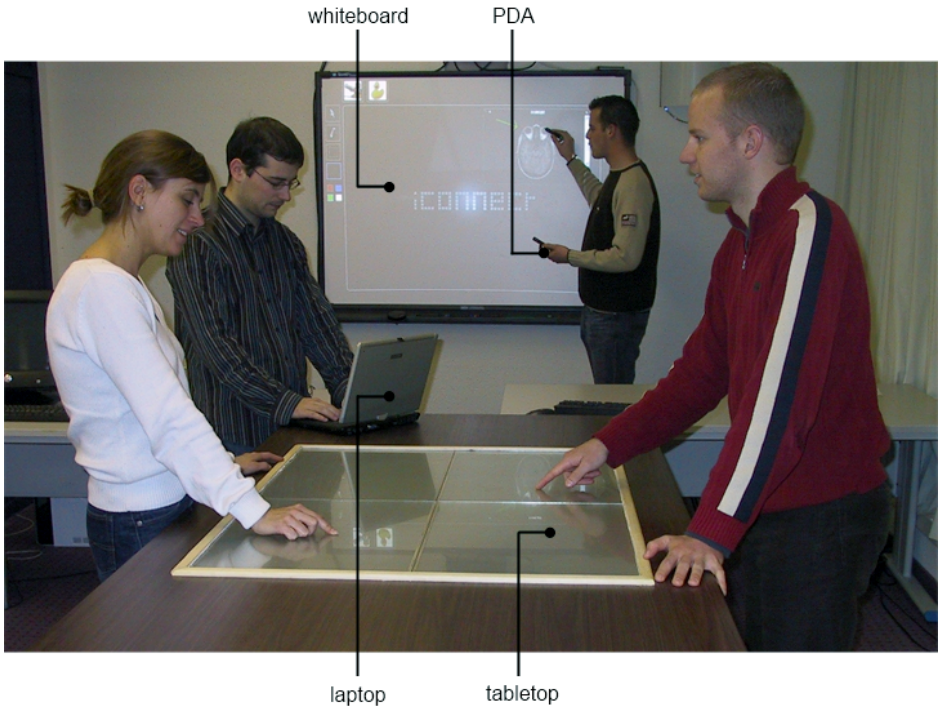


Figure 4.1: The iConnect environment. Users interact with the environment using touch-sensitive shared displays, or personal devices such as laptops and PDAs.

is shared across iConnect clients using Virtual Network Computing (VNC)[2], a graphical desktop sharing system that allows a user to remotely control another computer. Since VNC's standard input behavior is single user-oriented, cursors are managed by and rendered on the server.

4.4.2 Embedding native applications in containers

The shared workspace is capable of embedding native applications which may be useful during a meeting, such as the typical office applications (e.g. Microsoft Word, PowerPoint and Excel) or viewers, to support various data formats (e.g. PDF documents using Adobe Acrobat Reader). An ActiveX container acts as a bridge between native Microsoft Windows oriented applications and the iConnect software. All functionality of the application is accessible, and interacting with the application through iConnect does not differ

[2] `http://www.realvnc.com`

from interacting with the application launched outside the iConnect system. If an appropriate application is available, the associated data formats can be supported. In addition, containers can embed custom applications, such as a video streamer or a special-purpose painting system [Van Laerhoven 06].

The shared workspace offers a generic set of operations to manipulate its containers. Since a shared workspace can be mirrored on various shared devices (clarified in the next section), container operations are represented in suitable widgets, such as a toolbar at the edge of the screen. Standard window operations (e.g. moving, resizing, minimizing ...) are a classic subset of the container operations. However, more specific operations may be needed, such as rotating a container on a horizontal display. Containers also support annotations: a widget provides basic tools for drawing scribbles and primitive shapes on top of a container and its embedded contents, as shown in Figure 4.2c.

4.4.3 User representation and data sharing

Figure 4.2a illustrates how each user is represented by an avatar. During an iConnect meeting, each individual can be identified through this avatar, whether that person is working co-located or remote. Each user also commands an individual cursor in a shared view. One of the complications regarding multiple cursors on a single display is determining which cursor belongs to a particular user. We therefore attach a miniature representation of the user's avatar to each cursor. The personalized cursor gives an idea about the user's current action and state, allowing users to distinguish one another's actions. Other types of virtual embodiments, such as virtual arms or pantographs, could further improve the awareness of others' actions and locations on tabletops [Pinelle 08].

The personalized cursor allows a user to manipulate (e.g. open, close, move, resize) objects, write down annotations, transfer data to and from a personal space, and so on. A user's personal cursor can be operated either at the shared display itself, or from a distance, using a personal device such as a PDA. In that case, the screen of the personal device is transformed into a touchpad, as depicted in Figure 4.3. Cursors disappear after being inactive for a while, to avoid cluttering the workspace.

By dragging and dropping a container on an avatar, the container's embedded data source (e.g. a document) is copied to the personal device of the corresponding user. The client software on the personal device decides how to process the arriving data (e.g. open the document). When a user is associated

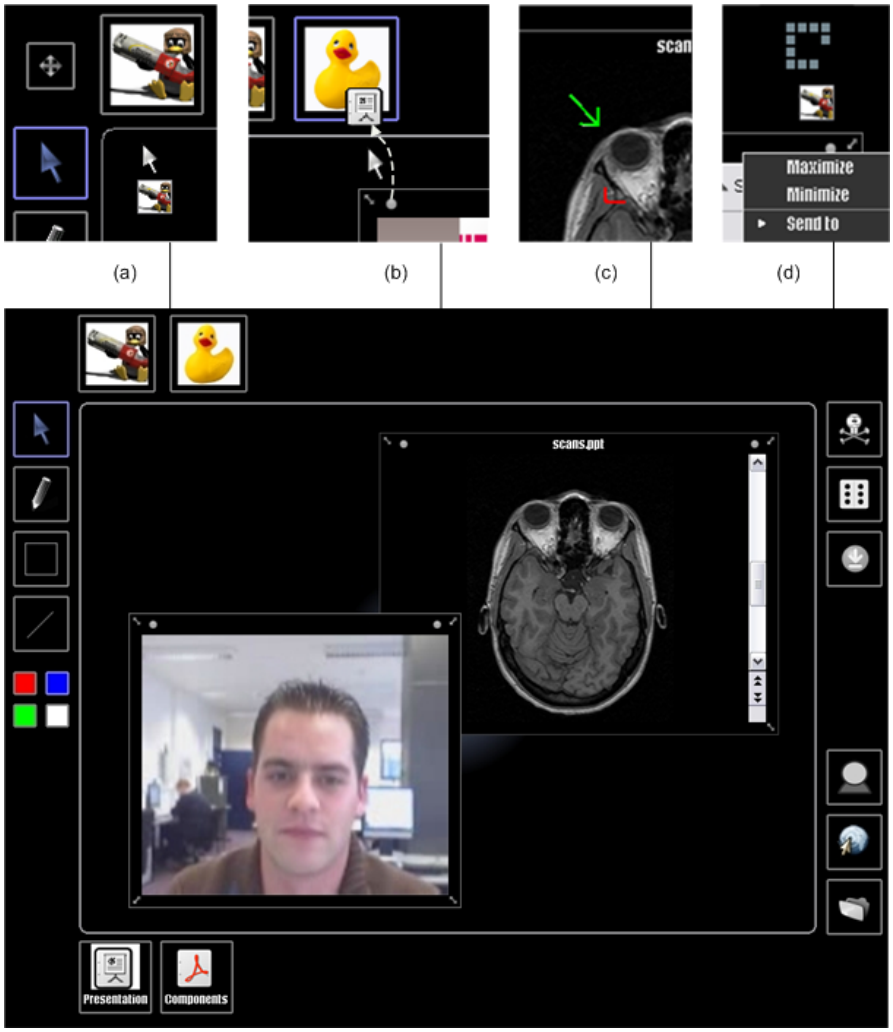


Figure 4.2: SMART board interface: (a) each user is represented by an avatar and controls a personal cursor; (b) data can be exchanged by drag-and-drop operations; (c) annotations clarify a presentation; (d) pop-up menus reveal the set of possible actions on a container.

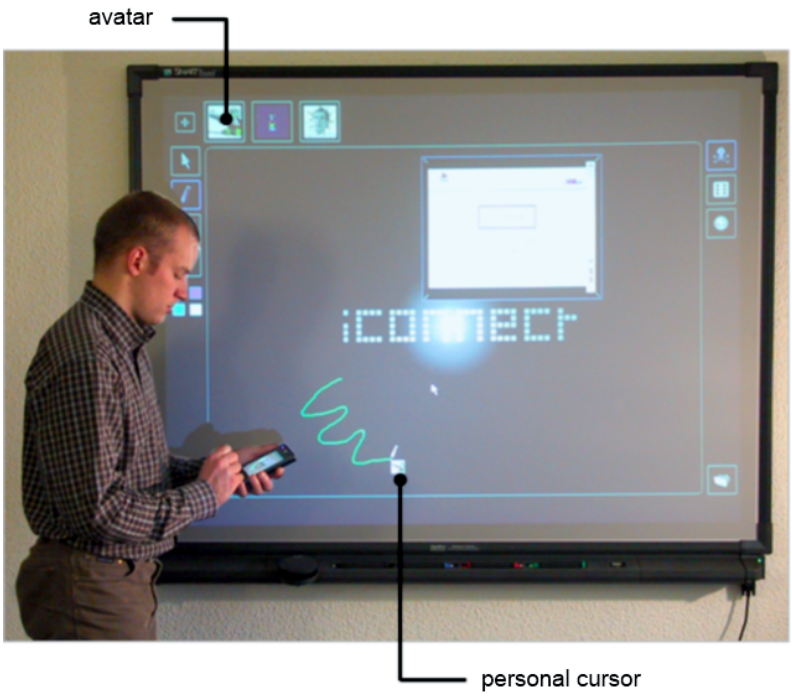


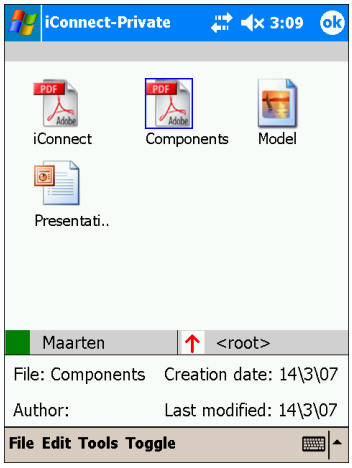
Figure 4.3: On a shared workspace, users are represented by an avatar and a personal cursor. A user's personal cursor can be operated using a personal device such as a PDA.

with multiple personal devices, a pop-up menu appears, and the target device can be selected, as illustrated in Figure 4.6c. In Section 4.5.3, we discuss how users can retain control over their own data, by preventing, for example, unauthorized users from copying private documents from others to their own personal devices.

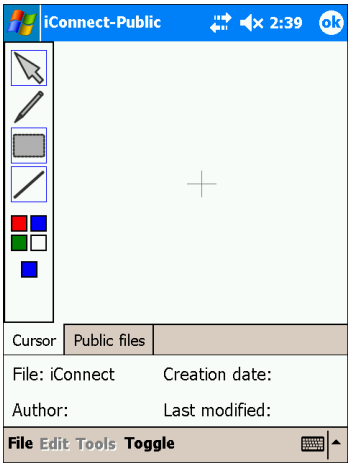
When a user drags a container outside the workspace's boundaries and into one of the "toolbars", as seen in Figure 4.2, the widget is replaced by a small icon representing the container's embedded application. This icon can be dropped on an avatar or a special-purpose button, such as a "kill" button which closes the container (adding a close button to each container tends to cause accidental closures). Since moving large widgets on a shared workspace can be very annoying to other users, we provide an alternate approach to initiate a drag-and-drop. Containers are equipped with a "drag bullet", shown in Figure 4.2b, enabling users to drag the container to a target, such as an

avatar, by only moving the outline of the container. This avoids disturbing other users.

4.4.4 Integrating personal devices



(a) Private space allows users to locally organize, view and manipulate documents.



(b) Public space allows users to operate their personal cursor and serves as a tool palette.

Figure 4.4: Interface of the user's personal workspace on a mobile device. The personal workspace is divided into a private and a public space.

The user's personal workspace, for instance on a mobile device such as a PDA, is divided into two subspaces: a private and a public space. Figure 4.4 shows both spaces. The private space mimics the shared interface, allowing users to locally organize, view and manipulate documents. Due to screen space constraints on small mobile devices, documents are represented by icons, although it is also possible to view and edit documents. The public space serves as a coupling between the shared and the personal workspaces, since users can operate their personal cursor with it. In that case, the screen of the mobile device is transformed into a touchpad, with movements and clicks being mapped onto the user's cursor.

Backed by a dynamic environment model that is updated at runtime, iConnect can migrate parts of a user interface to personal devices in order to facilitate the interaction with shared data and other devices. To allow the user to

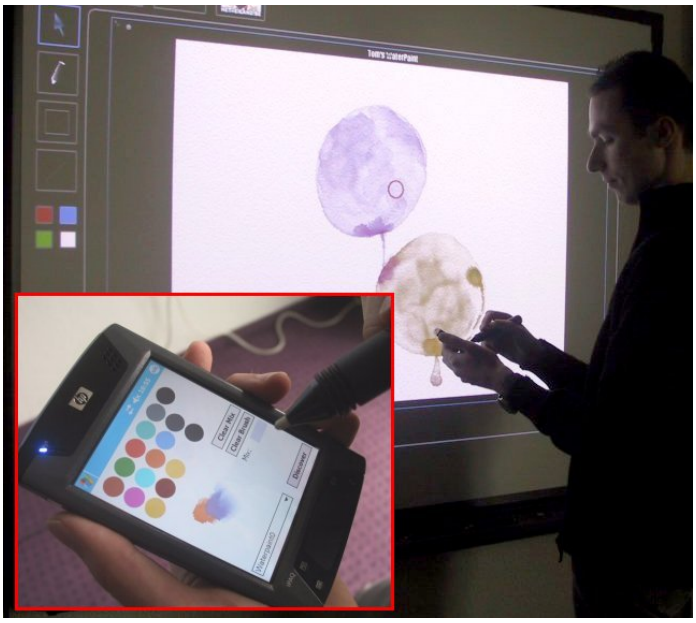


Figure 4.5: A painting application is running in an iConnect container on the whiteboard. The color chooser is distributed to the mobile device of the user.

change the cursor's function when using the mobile device as a remote touchpad, we distributed a drawing toolbar to the mobile device, as illustrated in Figure 4.4b. Figure 4.5 displays a second example, with a special-purpose painting application [Van Laerhoven 06] running on the whiteboard and the associated color chooser distributed to the mobile device of the user. This way, the mobile device may serve as a tool palette for the whiteboard [Rekimoto 98]. A complete description of iConnect's dynamic environment model and user interface migration is, however, beyond the scope of this work.

4.4.5 A collaborative tabletop

Large displays and in particular tabletops are one of the most common settings for human collaboration [Pinelle 06]. They naturally support co-located collaborative work: people can display and manipulate task artifacts, they enable verbal and gestural communication and allow people to be aware of co-participants' actions. Furthermore, a multi-touch tabletop promotes more equitable participation in co-located group settings [Rogers 09, Wigdor 09]. Although occasionally integrated into meeting rooms, we believe the user in-

terface of a tabletop environment should not merely be an extension of desktop systems, and the unique affordances of tables should be taken into account [Shen 04].

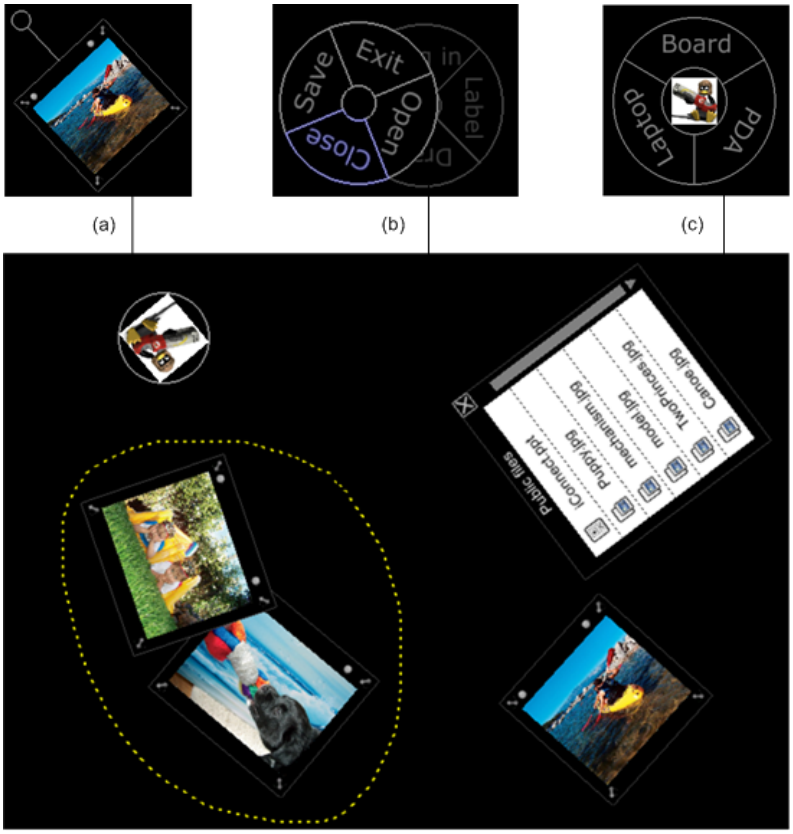


Figure 4.6: Tabletop client interface: (a) widgets can be rotated, moving the handle rotates the widget around its center; (b) hierarchical pie menus allow users to trigger actions on objects in the workspace; (c) user representations are embedded inside pie menus.

Horizontal displays, where users can sit or stand at different sides, suffer from the fact that documents oriented toward one user may be hard to read by others. To overcome this issue, we have included containers that can be rotated in the tabletop interface, as shown in Figure 4.6. Users can pass a container to others (the container will automatically rotate in an ellipse around the centre of the table) or duplicate it to provide a similar view in the desired angle. In addition, Figure 4.6a illustrates how containers can be rotated around their

own centre, so they can be oriented manually as well.

To accommodate interaction with existing applications on interactive surfaces, we simulate the typical "right click" of a mouse by pressing and holding a pen or finger stationary over a small period of time. This action triggers a short feedback animation, illustrated in Figure 4.2d, followed by a pop-up menu. In the tabletop interface we opted for pie menus instead of linear menus, as shown in Figure 4.6b. Pie menus arrange all items in a circle around the cursor, thus making all items equidistant [Balakrishnan 04]. As our system does not recognize where a user is sitting or standing around the table, pie menus have the added benefit of improving readability. This results in a very convenient way of triggering actions on large interactive displays.

The current tabletop interface could be extended to further enhance its integration in an environment that contains portable devices and ancillary displays such as a whiteboard. First of all, additional visualization mechanisms can improve the visual connectivity between the various displays or personal devices [Wigdor 06, Everitt 06]. The interaction can also be expanded by adding proxy objects to the tabletop interface, allowing some control over the other displays in the environment. Such a proxy can, for instance, take the form of a virtual camera to alter the viewpoint in a geospatial application shown on one of the displays [Forlines 06]. To easily move objects from one display to another, the proxy can implement a technique such as worlds in miniature [Wigdor 06], or radar views and pick-and-drop, which have been found to be the most efficient when comparing the performance of various techniques for multi-display reaching [Nacenta 05]. Furthermore, mechanisms for sharing documents in a co-located tabletop environment can be used to support transitions between periods of active collaboration and periods of individual activity [Ringel 04].

4.5 Illustrative Focus+Roles implementation

Simultaneous interactions in a highly collaborative environment such as iConnect cause various kinds of conflicts on a regular basis, which disturb the dynamic work flow and group coordination. Since iConnect is primarily targeted at common daily meetings, in which users typically take on a variety of roles and share data frequently, it provides an appropriate test case. In this section, we provide some of the particulars regarding our Focus+Roles implementation in the iConnect environment. Throughout the section, we illustrate our approach using the scenario about a team of neurologists from Section 1.3.2.

4.5.1 Roles as a set of privileges

As described in the overview of the Focus+Roles process in Section 4.3.4, a user's static roles are primarily employed to provide the necessary access control. A user's dynamic role, on the other hand, is put to use to prevent intrusive activities or resolve conflicts. In the iConnect environment, static and dynamic roles associate categories of users (e.g. department heads) with specific sets of privilege levels (e.g. full privilege to access financial records). The roles are written down in an XML format, as shown in Listing 4.1 and Listing 4.2.

```
<ContentPrivilege role="DEPARTMENTHEAD" content="PATIENT" level="100"/>
<ContentPrivilege role="DEPARTMENTHEAD" content="FINANCIAL" level="100"/>

<ContentPrivilege role="SENIORPHYSICIAN" content="PATIENT" level="100"/>
<ContentPrivilege role="SENIORPHYSICIAN" content="FINANCIAL" level="30"/>

<ContentPrivilege role="PHYSICIAN" content="PATIENT" level="90"/>
<ContentPrivilege role="PHYSICIAN" content="FINANCIAL" level="0"/>
```

Listing 4.1: Partial description of static roles. This particular example shows, for instance, physicians with no access to financial documents and department heads with full access to such documents.

To provide access control in the iConnect environment, we associate each static role with a set of privilege levels for specific types of content (e.g. patient files, financial records), as illustrated in Listing 4.1. In that example, physicians have, for instance, no access to financial documents, while department heads have full access. Instead of implementing a privilege as a binary "allowed" or "not allowed", which is the traditional operating system's approach, privilege levels can range from zero to one hundred, indicating to what extent a user is suited to handle a particular type of content (more on this in Section 4.5.3, where we discuss access control in detail).

A dynamic role is expressed in terms of privilege levels for specific actions (e.g. move or close a document), as seen in Listing 4.2. Every time a user performs an action in the iConnect environment, the privileges of that user's dynamic role are checked. The privilege level specifies whether a user may perform a particular action or not, and to what extent a user's action is given priority in case of conflicts. A privilege level of zero basically stands for "not allowed", meaning the user is not able to perform that type of action. An attendee of a presentation is, for instance, typically not allowed to close documents. Rather than also prohibiting attendees to move documents, their very

restricted privilege level permits moving new information to an open space on the whiteboard, but without overlapping other documents, as we explain in the next section.

```
<ActivityPrivilege role="ATTENDEE" activity="MOVE" level="10"/>
<ActivityPrivilege role="ATTENDEE" activity="CLOSE" level="0"/>

<ActivityPrivilege role="PRESENTER" activity="MOVE" level="100"/>
<ActivityPrivilege role="PRESENTER" activity="CLOSE" level="100"/>

<ActivityPrivilege role="BRAINSTORM" activity="MOVE" level="50"/>
<ActivityPrivilege role="BRAINSTORM" activity="CLOSE" level="30"/>

<ActivityPrivilege role="FACILITATOR" activity="MOVE" level="100"/>
<ActivityPrivilege role="FACILITATOR" activity="CLOSE" level="100"/>
```

Listing 4.2: Partial description of dynamic roles. This particular example shows, for instance, attendees who are not allowed to close documents and presenters with full privileges.

In case a conflict occurs in the iConnect environment, for example when two users try to move the same document simultaneously, the user with the highest privilege level for that action gets precedence over the other user. When both users have the same dynamic role, their static roles are taken into account and the outcome is based on the privilege levels for the type of content that is being manipulated. If, for instance, a brainstorm participant and a facilitator simultaneously try to move a patient file, the facilitator "wins" because of the higher privilege level for moving documents. If both users are brainstorm participants, on the other hand, privileges for moving documents are the same. Assuming that one user is the department head and the other a physician, the department head gets priority because of the higher privilege level with regard to patient files.

```
<ActivityPrivilege role="BRAINSTORM" activity="MANIPULATE" level="50"/>
<ActivityPrivilege role="FACILITATOR" activity="MANIPULATE" level="100"/>
```

Listing 4.3: To make the process of describing dynamic roles less tedious, privileges can also be associated with categories of actions.

Instead of linking privileges of a dynamic role to very specific actions such as "open" and "close" in the XML file, privileges can also be associated with categories of actions, as shown in Listing 4.3. The "manipulate" category bundles a number of similar actions, such as moving and resizing documents,

to make it less tedious to describe dynamic roles. The current XML format is a very straightforward approach, which is somewhat limited in expressiveness, but very easy to handle, as it provides an accessible way to adjust existing roles or add new ones. A specification language such as the one used in RBAC [Sandhu 96] or Intermezzo [Edwards 96] can be used as a more expressive alternative.

While a user's static roles are fixed from the beginning of a collaborative meeting, the environment must allow the user to easily switch between dynamic roles during the course of the collaboration. Since iConnect requires each user to log on to the system, static roles are simply recovered from a database. The dynamic role, however, must be selected by the user when connecting to the environment, and may be changed at any time, for example by means of a personal device such as a PDA.

4.5.2 Focus as an amount of attention

As stated in Section 4.3.2, we consider two kinds of focus: a user's passive focus is centered around the perception of output, while a user's active focus requires some form of input. Based on both passive and active focus, each container (e.g. a text file, spreadsheet, picture, video) in the iConnect environment is attributed an "amount of attention". This amount is simply an accumulative value that represents to what degree users are currently focusing their attention on that container, and is used to overcome conflicts that are caused by a lack of awareness.

Each user commands a personal cursor on a shared surface in the iConnect environment, as described in Section 4.4.3. To detect a user's passive focus, we analyze that user's cursor movements: when a user starts hovering over a container, the system assumes that the user's center of attention shifted to that container. Therefore, the amount of attention associated with the container is increased. When the user's cursor leaves the container, the value is decreased. Of course, this approach is merely an estimate of the user's passive focus, as it is likely that a user sometimes places the cursor on top of a container without really focusing on that container, or vice versa. Tracking a user's eye gaze, head orientation or posture can definitely improve the passive focus recognition.

Active focus is determined by explicit actions on a container, such as opening a document, browsing through slides, or editing a spreadsheet. Those actions can simply be detected by filtering all the input events on the container. Each time an explicit action is performed, the amount of attention associated

with the container is slightly increased. If a user is, for instance, editing a spreadsheet, the value increases with each alteration to reflect the current attention that the document is receiving. However, if the user stops editing the spreadsheet, the amount of attention has to decrease accordingly. One option is to remove the increase in attention as soon as the user's cursor leaves the container, but that approach does not take into account someone working on several documents at once. Therefore, we progressively diminish the amount of attention over time, so the value slowly returns to zero if users are no longer involved with the document.

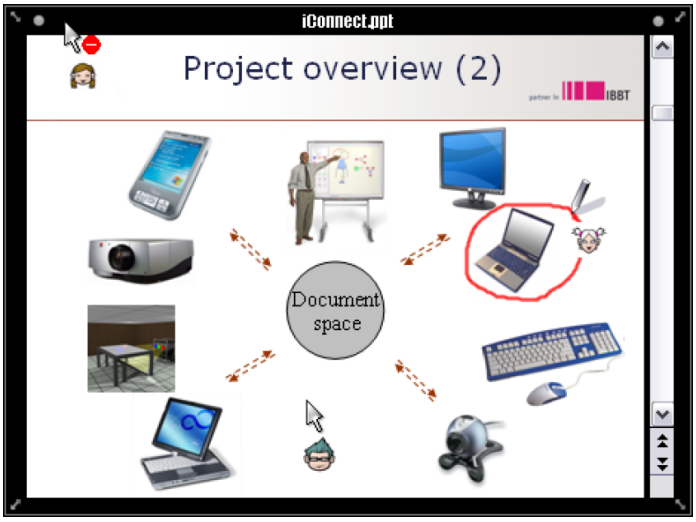


Figure 4.7: Three users simultaneously interacting with a document using their personal cursors. One user tries to move it while two others center their attention on it. The system discreetly notifies the users of the issue by means of a miniature stop sign and does not allow the action to take place.

If a user performs an action that influences a shared container, the container's current amount of attention is compared to the user's privilege level for that particular action. The outcome of this comparison determines whether the system carries out the action or notifies the users of the intrusion. It is important that users are notified discreetly, to avoid disrupting the others. Consider, for example, two brainstorm participants discussing a slide. If a third participant suddenly tries to move the slideshow, that action is prohibited due to the slideshow's high amount of attention. A miniature stop sign is displayed next to the user's personal cursor, as illustrated in Figure 4.7. If the third user was a brainstorm facilitator, the action would have been allowed,

since a facilitator has the maximum privilege level for moving documents, as seen in Listing 4.2.

4.5.3 Access control, content type and sensitivity

A practical workspace in a collaborative meeting environment requires document formats such as text files, slideshows, spreadsheets, pictures, video files, and so on. However, the document format not always provides a sufficient indication of the actual content: a text file may contain an internal memo or a simple note, and spreadsheets can hold confidential information about finances or a commonly available planning. Therefore, we employ a secondary classification, based on meta-data such as content type and sensitivity.

When uploading a document to the iConnect environment, the user has to state the type of content (e.g. financial report, patient file, brain scan) and the document's sensitivity (e.g. highly confidential, internal). In some cases, it is possible for the system to determine one or both properties automatically, although the owner of the document will always have the final say in this matter. Ordinarily, a C++ file always contains programming code, for instance, and a bank statement is usually considered to be delicate information.

If a user tries to access a document in any way (e.g. actions such as open, delete, copy, print), the privileges of that user's static roles are compared to the meta-data associated with the document. If the user lacks the necessary privilege level for that particular type of content and degree of sensitivity, access to the document is denied. A regular physician will, for instance, never be able to make a personal copy of a financial document. The physician will have to abandon this intent, or ask someone with the necessary privilege level for assistance. A senior physician, however, is allowed to copy a financial document if it is not marked as highly confidential through its sensitivity, since that role has limited access to financial information, as seen in Listing 4.1.

4.5.4 Limitations and possible extensions

Incorporating role management and focus tracking into floor control policies is only a start, and further refinements and additional features are needed to improve the usability of the system. First of all, users need more feedback when a conflict or intrusive action is prevented. Currently, they are discreetly notified of the prevention, but without any additional information, confusion will occasionally arise, since some users might have no idea why their action was prohibited by the system. However, the feedback should not disrupt the work flow. One possible solution is to show additional feedback on the user's

personal device, so he or she can access this feedback when necessary, and ignore it otherwise. In addition, the current roles of a user could be visualized, for example by adding some iconic representations to the user's personal avatar and cursor, since it may give an indication as to why an action is currently not allowed.

The reliability of the existing approach to adopting dynamic roles depends largely on the goodwill of the users. Currently, users are responsible for manually updating their role, and this approach can easily fail if users forget to do it, or if they select the wrong role on purpose, to circumvent the system's restrictions. Depending on the available hardware setup in the meeting rooms, it might be possible to automatically deduce a user's current dynamic role, ensuring smoother and more reliable role transitions. If the system supports, for example, tracking of a user's location in the meeting room, it is safe to assume that a user standing next to a projection screen is currently practicing the role of presenter, and the ones sitting in front of the screen the role of attendee.

By storing users' static roles in a database and not making those users responsible for updating their own roles, it is harder to bypass the system's access control. However, users are not in direct control of the access to their documents, as the system handles access control autonomously. This lack of control causes a degree of uncertainty, and will prohibit a lot of users from sharing very sensitive or confidential data. To overcome this uncertainty, the system should enable a user to set certain access rights when sharing documents, similar to IMPROMPTU [Biehl 08], stating if other members of the group are allowed to modify or only view the data, for example.

Currently, our approach handles conflicting actions on the global level of documents, so two users cannot edit the same document simultaneously. However, in a highly collaborative environment, multiple users may want to edit a document together, seeing one another's changes in real-time. A potential solution is to embed a Web browser in a container, and rely on a Web-based office suite such as Google Docs[3], so text files, spreadsheets and presentations can be opened and edited by multiple users concurrently. Another possibility is using an approach like CoWord [Xia 04] to add real-time collaboration to single-user applications such as Microsoft Word or PowerPoint.

To further refine our floor control mechanism, we can track a user's interaction history. By considering all past and present actions on an object, we can define a degree of ownership for each user, which is useful when deciding

[3] `http://docs.google.com`

on a particular action, such as deleting a document. If others were far more active on a document, they should be in control of the most sensitive activities, and not a user who barely paid any attention to it until now. In addition, an "honesty" policy can benefit less assertive people, whose actions will frequently be suppressed by more prevailing users. An algorithm can determine the activity rate of each individual and subsequently prioritize a less active user's actions to work toward balanced contributions.

To increase the overall flexibility and to avoid some users being left out from time to time because of a lack of privileges, the system could offer users more autonomy by providing a means for one user to grant temporary privileges to another user, for example through a ticket mechanism [Turoff 91]. Further flexibility can be achieved by taking a mixed-initiative approach to the conflict handling, by merely warning a user of a possible conflict and giving that user the opportunity to complete the action anyway. In this manner, the system blends a floor control policy with social protocols.

4.6 Conclusion

Social protocols represent a popular floor control choice in present-day collaborative environments. However, such solutions are not always infallible. Our Focus+Roles approach effectively provides both conflict handling and access control, by introducing roles and focus. As a proof of concept, we added a Focus+Roles policy to the iConnect environment.

As a result, iConnect users can adopt a variety of static roles, representing their organizational function and proficiencies. In addition, a dynamic role specifies the user's current activities throughout a collaborative iConnect meeting, and a user may switch from one such role to another at any time. Our floor control policy dynamically adjusts to the users' roles, while focus tracking attempts to overcome the typical lack of mutual awareness, thereby reducing confusion and unintentional effects. Furthermore, we allocate properties such as content type and sensitivity to the documents in the iConnect environment, allowing for effective access control when combined with the users' roles.

It should prove interesting to include other properties in the floor control policy, such as the specific properties of different devices, since users will behave differently when seated around a tabletop compared to standing in front of a vertical display [Rogers 04b, Everitt 06], and their behavior might change depending on the type of personal devices they brought along [Pering 10]. Behavior will also be governed by the various roles that are taken on by the different members of the group. However, further studies are needed to assess

the behavior of a group of users in those particular circumstances.

In the next chapter, we explore some aspects of collaborative meetings in more detail, by conducting an in-depth investigation of one case in particular, namely storyboarding in co-located, multidisciplinary teams.

Chapter 5

An observational study on collaborative storyboarding in multidisciplinary teams

Contents

5.1 Introduction

In the previous chapter, we presented the use of the roles and focus of users as a way to achieve conflict resolution and access control in a collaborative environment. One of the devices in that environment was an interactive tabletop,

and although our general Focus+Roles approach is applicable to this kind of shared surface, we never fully explored the collaborative potential of the tabletop in particular. We also made some assumptions on the behavior and needs of users, such as the way they organize a shared workspace, or the influence of different roles during the collaboration. To explore these aspects in more detail, we conduct an in-depth investigation of one specific case: storyboarding in co-located, multidisciplinary teams.

In this introduction, we first answer the questions: why do we consider multidisciplinary teams, and why in the context of storyboarding? User-centered design and development of software systems typically involves design teams that include multidisciplinary skills and perspectives, which is beneficial for the overall user experience of the resulting system. These teams generally involve team members having expertise in human-computer interaction, user interface design and systems engineering, as well as end-users and application domain specialists [Int 10]. Several participatory design techniques support the collaboration within these teams. However, since the team members often have different expectations of the representation and transformation of end-user needs and concepts for the future software system, it is a challenging task to ensure that they all reach a common understanding in the first stages of user-centered approaches.

An accessible and very useful participatory design technique for user-centered design and development of software systems is storyboarding. A storyboard is a narrative that uses rich, iconic pictures to visually represent systems' scenarios of use, and is well suited for creating a common understanding of the application domain. Based on the framework for the organization of participatory design tools and techniques of Sanders et al. [Sanders 10], storyboarding can be considered as a technique that supports the generation of ideas or design concepts for the future. More specifically, in user-centered approaches, storyboarding takes place during the requirements elicitation of a future system. The storyboard notation is very suitable to depict the functional and non-functional requirements of the system in different contexts of use. Furthermore, annotated storyboards can be used throughout the design and development process [Haesen 11b]. Storyboards help in keeping focus on the end-user and enable design teams to share concepts with a broad audience [Cilella 11].

Several efforts are presented toward the creation of digital tools that focus on individual or collaborative storyboarding as part of a software development process [Truong 06, Atasoy 11, Haesen 11b]. These tools concentrate on the (re)use of storyboards, which is interesting to support adaptations and consis-

tency checks with the system's requirements. However, none of the studies and tools consider the respective contributions of team members having different skills, perspectives and goals in the creation of storyboards. Because the existing tools have proven the benefits of digitizing storyboards for user-centered design and development purposes, we are investigating the requirements for a digital tool that supports the collaborative creation of storyboards within a multidisciplinary team.

As stated in Section 1.3.3, we believe that a digital tool for collaborative storyboarding should balance creative aspects, the needs of several disciplines represented by the multidisciplinary team, and support for equitable contributions. In this chapter, we present an observational study on storyboarding by multidisciplinary teams. In Section 5.3, we describe the setup of this study and we present an overview of the observations and results. Our insights lead to concrete recommendations on how a digital storyboarding tool can effectively facilitate this type of group work on an interactive tabletop, as reported in Section 5.4. Section 5.5 summarizes our conclusions.

5.2 Related work

In this section, we explore the existing work relating to digital storyboarding tools that are targeted toward individual designers, the suitability of interactive tabletop systems to support collaborative storyboarding in multidisciplinary teams, and present-day storyboarding tools for tabletops.

Digital tools for individual designers. SILK [Landay 95] is a digital tool that enables designers to quickly sketch a user interface by using an electronic pad and stylus. Unlike paper sketches, electronic sketches can easily be annotated and modified using gestures. Furthermore, the sketches are interactive, so end-users can experience the envisioned interaction before the interface becomes finalized. The tool allows the designer to define interface behaviors by sketching storyboards, which specify how a screen should change in response to user actions [Landay 96]. DENIM [Lin 00] is a closely related tool for Web interface design, enabling designers to quickly sketch and link pages, and interact with them. The tool's use of storyboards for behaviors is similar to SILK. The zoomable canvas enables a designer to quickly move among various ways of viewing a website, such as a sitemap and storyboard.

Anecdote [Harada 96] is a tool to support the design of multimedia applications. It allows a designer to create a set of annotated storyboards, which are organized in a hypermedia structure. Different views, such as a timeline

that enables linear editing of the hypermedia structure, facilitate the use of various design styles. Anecdote supports simulating the execution and transforming the prototype into the final application. DEMAIS [Bailey 01] is another sketch-based multimedia design tool, intended for a pen-based tablet. The tool uses a designer's ink strokes and textual annotations to transform a static storyboard into a working example, supporting experience-based exploration, similar to the other tools.

The suitability of interactive tabletops. The abovementioned tools are targeted toward individual designers, and the storyboards do not take into account the larger context of use of an application. We already suggested some high-level requirements for collaborative storyboarding in the introduction: support creativity, the needs of several disciplines represented by the multidisciplinary team, and equitable contributions. Digital meeting systems can be designed to support such requirements, since they enable a structured decision-making process that encourages full participation by all team members [Galaczy 99]. Furthermore, such systems promote creativity by allowing all team members to generate ideas simultaneously and by allowing them to immediately respond to the ideas of others.

In an exploration of the creative storyboarding process to determine best practices and guidelines, Truong et al. [Truong 06] highlight the importance of being able to share and be inspired by artifacts created by other members of the design team. A design tool should therefore allow users to show what they are creating and allow members of the same team to observe and easily use artifacts from other members. Pinelle et al. [Pinelle 06] confirm this, as they observed storyboarding on a regular table to explore how highly-integrated collaboration can be supported by digital tabletops. Although the study focuses on collaboration, not on storyboarding, the proposed design principles also include "make artifacts and actions visible to others". Taking this into account, shared surfaces such as interactive tabletops look like a proper candidate to facilitate collaborative storyboarding.

To reinforce this statement, we consider the requirement of supporting equitable contributions. Avila-Garcia et al. [Avila-Garcia 10] identify active participation and involvement of all the team members as an important challenge to support decision making in multidisciplinary team meetings. A study [Rogers 04a] on how novel fingertip interactions can support collaborative decision-making on a tabletop reveals that team members communicate a lot with their fingers. This "finger talk" results in much discussion, sharing of ideas and invitations to others to take a turn, respond, confirm or partic-

ipate. This reinforcement of balanced participation is confirmed by a later study on equal opportunities [Rogers 09], which shows that tabletops produce more equitable participation in terms of physical actions in co-located group settings. Moreover, extending the tabletop with a set of tangible artifacts (e.g. physical 3D models) leads to more equitable participation in terms of verbal contributions.

Digital storyboarding tools for tabletops. Interactive tabletops are already used to support collaborative storyboarding. Storify [Atasoy 11] is, for instance, a tool to assist design teams in incorporating user experiences into the design process. It provides assistance in organizing the elements of an experience similar to the elements of a story, such as characters, settings, plot and theme. The design process consists of a number of stages, such as team members organizing parts of their individual collection of inspirational materials (such as sketches, images, text, Web links, sound tracks and videos) that are needed for the specific context they have in mind, and composing a storyboard with the gathered materials to clarify the experience. In one of the final stages, the generated concepts are evaluated by designers, users and other stakeholders. To accommodate different audiences, Storify can present the outcome in different presentation formats, which can be annotated during discussions. The development of Storify is still in progress, but the requirements include an individual mode that lets designers work with their personal devices and a collaborative mode in which the team works together on a shared interactive table and wall projection display.

Coeno-Storyboard [Haller 05] is a face-to-face presentation environment for storyboard discussions using tabletop technology in combination with a wall projection and portable computers such as laptops and tablets. Users can create artifacts (e.g. scenario sequences, scribbles) on their personal computer, move them to the shared tabletop for discussion, and then to a timeline on the wall-sized projection for final organization. Artifacts can only be moved, rotated or scaled on the table. In a pilot study, users indicated that, instead of always having to edit or create artifacts on a personal computer, the table should allow more actions. To move artifacts from the table to the wall, users either use virtual keypads on the table, or one person assumes the role of a coordinator and organizes the artifacts by moving them with a wireless mouse. Although Coeno-Storyboard aims to allow participants from different domains (e.g. designers, modellers, project managers) to interact easily with the system, no specific attention is given to the multidisciplinary aspect. In addition, several assumptions are made that we intend to explore in our obser-

vational study, such as having one person assuming the role of a coordinator and organizing artifacts on a timeline.

StoryCrate [Bartindale 12] is a tangible tabletop interface to support live film production during a shoot. The interface uses a storyboard as a shared data representation, enabling a greater awareness of current progress for the entire crew and facilitating team creativity. The work mainly focuses on film production, but observations revealed a few interesting findings that we want to investigate in more detail: a designated operator was appointed to maintain data and the tangible objects facilitated the transfer and holding of control over the interface when users gathered around it.

5.3 Observational study

In this section, we present the details of our observational study, which provides us with valuable information on how multidisciplinary teams organize their tasks, what kind of artifacts they produce and how they collaborate in storyboarding sessions.

5.3.1 Participants and apparatus

Twelve participants, four female and eight male, ranging in age from twenty-two to forty-six, were divided in three groups of four people. All of them were experienced researchers, who have expertise in the design or development of interactive systems from HCI and/or UI design perspective. Participants' backgrounds varied from computer science to history, visual arts and product design.

Each participant was instructed to take on a particular role during the study, so that each team consisted of an HCI specialist, a UI designer, a systems analyst, and a stakeholder (end-user or application domain specialist). We assigned roles based on the participants' skills and expertise. The results from the post-study questionnaire indicate that most participants felt "comfortable" to "very comfortable" in their role. Three participants felt "neutral" toward the attributed role, and nobody felt uncomfortable. All but one participant had at least one year of experience in being part of a multidisciplinary team. All participants in the role of HCI specialist and approximately one-third of the others had experience in creating storyboards.

We acknowledge that assembling multidisciplinary teams in this way may yield different results compared to actual teams that have been working together for some time. However, we consider this to be a preliminary study to

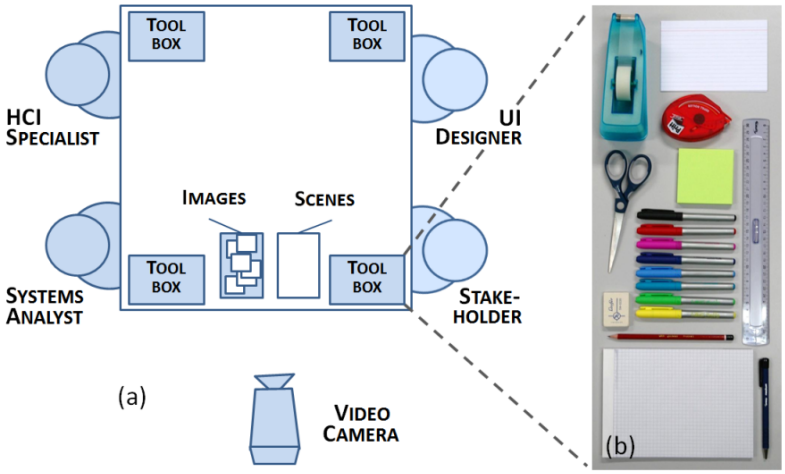


Figure 5.1: Setup of the observational study: (a) each participant positioned at a different corner; (b) contents of the toolbox that was provided to each participant.

aid us in outlining a number of initial requirements, and we intend to observe multidisciplinary teams in a real-life setting at a later stage in order to confirm and refine these requirements.

The storyboarding sessions were carried out in a room equipped with a regular table (160 by 160 centimeters) and four chairs positioned around it so that each group member would be seated at a different corner (Figure 5.1a). A video camera recorded the sessions for later analysis. On top of the table, participants could find a stack of A4 sheets to create storyboard scenes and a box with images representing personas and items from the scenario used for the study. Each participant also had a personal box of "tools": a pencil and ballpoint pen, colored pens and highlighters, an eraser, a ruler, scissors, glue, adhesive tape, a notebook, post-it notes, and index cards (Figure 5.1b).

5.3.2 Tasks and experimental design

Instructions for the task and a description of the participant's role and responsibilities were provided to each participant. We decided to use personas and a scenario as identical starting points for all storyboarding sessions, because these documents describe the use of a future software system and can be related to storyboards [Haesen 11b]. The personas (Appendix B.1) and scenario (Appendix B.2) that were provided to the participants revolved around a home automation system to control the heating and lighting, which can as-

sist a household in saving money on energy consumption. The system can be controlled by different family members (differing in age and technological aptitude), using different devices (e.g. touchscreen, laptop, smartphone). Although not explicitly mentioned, the scenario suggested that the system should take into account settings related to personal profiles and activities, that settings of profiles should be merged in certain situations, and that the system should be able to detect people's presence in particular rooms. In order to fine-tune our setup and procedure, we first conducted a pilot study.

5.3.3 Procedure

Each group was asked to create a storyboard that represents the given scenario. Once they read and understood all instructions, including personas and scenario, each participant had fifteen minutes to prepare individually. They were asked to write down or sketch anything considered to be important, bearing in mind their specific role and goals. After preparation, the participants were asked to start the storyboarding task. Each group was told they had sixty minutes, and we stopped groups after the allotted time. Two observers took notes throughout each session about actions and things said by participants. Upon completion of the storyboarding task, the participants were asked to fill in a questionnaire about their former experiences, their findings regarding the storyboarding task and collaboration within a multidisciplinary team, and opportunities for future storyboarding tool support. The questionnaire can be found in Appendix B.3.

5.3.4 Observations and results

In this section, we present the findings of our study, based on the results of the questionnaire and the observations made throughout the three storyboarding sessions. Figure 5.2 visualizes the physical activity on the table throughout the sessions. If a region has a darker color, it means there was more frequent physical activity in that region, and vice versa. We automatically generated this visualization from the video recordings, based on the movements of participants. Because our participants were seated at a prearranged location around the table and tended to stay at that location throughout the sessions, we can more or less associate physical activity on the table with particular participants. This approach obviously has its limits, since it does not consider participants reaching across the table, for instance. Therefore, we have to rely on our observations to interpret the physical activity correctly.

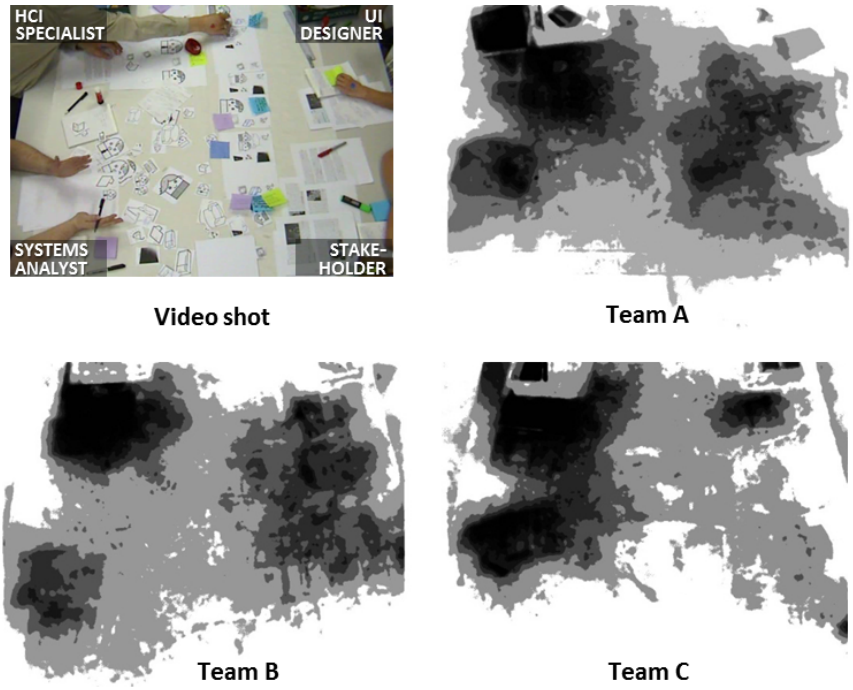


Figure 5.2: A shot of a video that recorded the table during a storyboarding session and a visualization of the participants' physical activity on the table throughout each of the three sessions, automatically generated from the videos (darker color means more frequent activity).

Individual preparation

During the individual preparation prior to the collaborative storyboarding, several participants highlighted phrases in the provided text. All participants structured the information in a certain manner: a few used bulleted lists, while others represented it by means of graphical artifacts, ranging from diagrams to sketches. Materials used during preparation include the available images, colored pens and highlighters, notebooks, post-it notes and index cards.

In terms of content, the roles of the participants were clearly expressed in the artifacts they prepared. The HCI specialists mainly focused on the relationship between personas, devices and tasks, the designers focused on UI designs and accompanying requirements, the systems analysts focused on the devices and their connections, and the stakeholders focused on general requirements and the needs of the personas. In all sessions, participants began to explain their prepared artifacts to the others once the cooperation started,

but in two out of three sessions, not all members presented their preparation. Artifacts were rarely included explicitly in the storyboard, but participants did use them during discussions.

Storyboarding task

The approach to the storyboarding task differed in the three teams. Team A started by shortly discussing their strategy for storyboarding and decided to first depict the equipment and users in the different rooms of the house. Almost immediately after this decision, they started creating the first scene collaboratively, in a shared space in the middle of the table. Next, the team implicitly split in two to prepare other scenes. Awareness was maintained, since participants frequently switched between cooperating with their neighbor and cooperating with the entire team, and a lot of the work was done in the middle of the table. The HCI specialist maintained the relationship between the storyboard and the scenario. For team A, Figure 5.2 clearly shows the high degree of activity of the HCI specialist, the cooperation between neighbors, and cooperation with the entire team.

Team B first discussed the system based on the requirements mentioned by the stakeholder. After a discussion of approximately fifteen minutes, in which some decisions regarding the system were already made, the HCI specialist reminded the team of the storyboard and took the lead in creating the scenes. The other team members were actively involved in the discussion and handed required images to the HCI specialist. Once the HCI specialist started creating a new scene, the stakeholder and designer finalized the former scene together. Figure 5.2 shows the high activity corresponding to the leading role of team B's HCI specialist, as well as the stakeholder and designer collaborating to complete scenes.

Not unlike team B, team C first discussed the system based on the requirements presented by the stakeholder. This discussion lasted nearly thirty minutes before a first scene was created. While discussing the devices for the system, the available images were put in the middle of the table to debate the different options and to decide which devices should be used. Again, it was the HCI specialist who reminded the team of the storyboard and who, based on the discussion, started creating the different scenes. Team C shows the least amount of cooperation between participants in Figure 5.2. The seemingly high activity of the systems analysts was actually caused by the HCI specialist, who was active in that region while creating scenes.

When considering the storyboarding sessions altogether, system features

that were implicitly described in the scenario led to a lot of discussions. Sometimes it was just one person who noticed a particular requirement, but in many cases, visually representing situations sparked these discussions. Participants used the part of the table in front of them as personal workspace, and the available images were scattered across the middle or side of the table to give everyone an overview, somewhat similar to the findings of Scott et al. [Scott 04].

Resulting storyboard

The resulting storyboards consisted of seven to ten scenes, and can be seen in Figure 5.3. All storyboards contained scenes representing personas and devices, and showed the status of particular devices (e.g. a light that was switched on or off). Two storyboards contained text to indicate the location or general situation of scenes. One team added post-it notes to the storyboard that would remind team members of particular features, difficulties and decisions.

Structuring the scenes of the storyboard was done in different ways. Team A created a visual representation of all rooms and their equipment, and consequently for each room the situation was depicted in different scenes. Teams B and C tried to put the scenes in a chronological order, based on the flow of events in the scenario. Extra scenes were inserted into the storyboard sequence when considered necessary. Scenes were labeled with numbers, and in team C, titles were added to each scene as well.

Multidisciplinary team

HCI specialists rated their direct contribution to the storyboard highest on average. Most systems analysts, designers and stakeholders rated their direct contribution considerably lower: analysts and designers prepared artifacts that were used to a lesser extent during the session, and stakeholders were more verbally involved. We can clearly relate these ratings to the amount of physical activity seen in Figure 5.2 (e.g. team B's systems analyst and team C's designer and stakeholder ranked direct contribution lowest). Stakeholders and systems analysts rated their general influence on the storyboard notably higher than their direct contribution. Frequent discussions between stakeholders and analysts about feasibility and costs of particular approaches account for this difference.

The participants confirmed in the questionnaire that being part of a multidisciplinary team had a positive impact on the storyboarding session, since

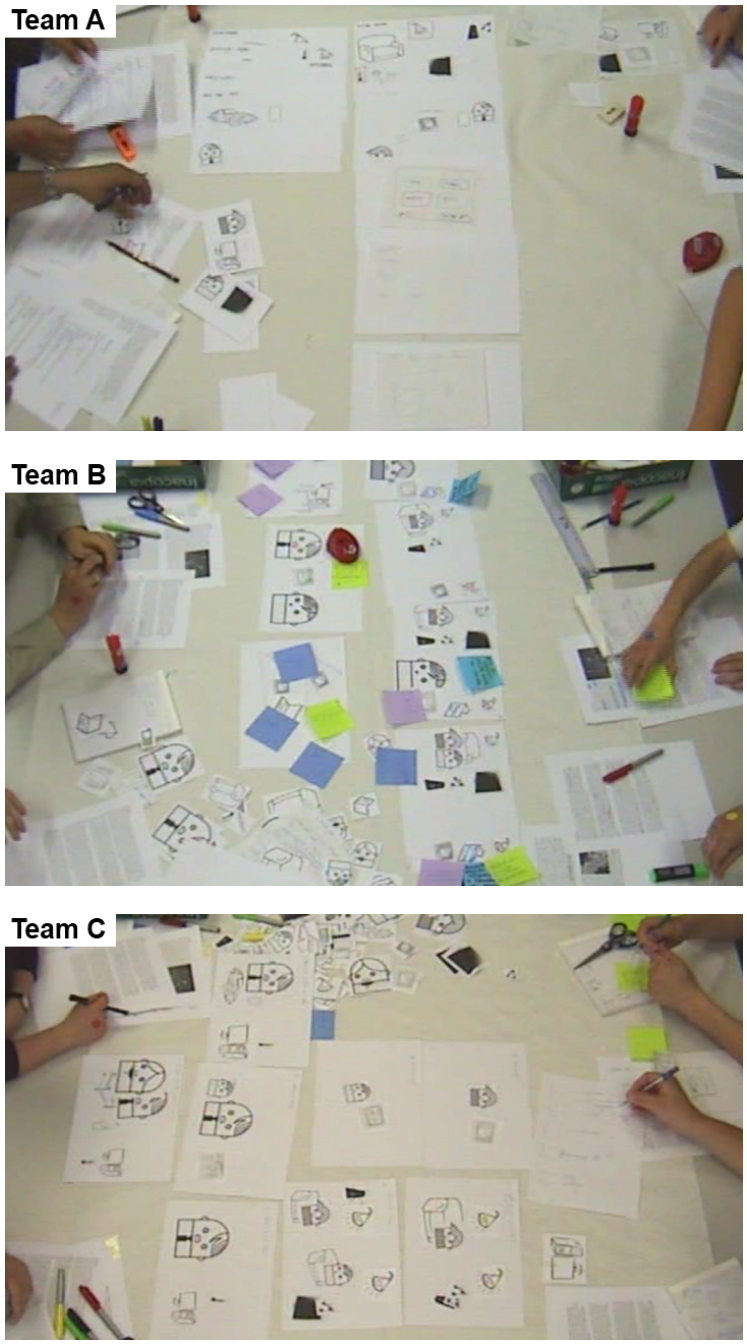


Figure 5.3: Frames from the videos that were recorded during each session, showing the final storyboard of each group.

it resulted in different perspectives, ideas and considerations. All participants responded positively when asked about the usefulness of a digital storyboarding tool. Features that were often reported as being essential point toward the traditional benefits of digital systems: the ability to locate tools or images more easily, editing operations such as moving or scaling items in a scene, cloning or reorganizing scenes and other artifacts in the storyboard, and saving scenes and artifacts for later reuse.

5.4 Lessons learned

The observations of the teams indicate that findings of a study on a creative activity such as storyboarding are not easily generalizable. Although teams created storyboards in different ways, we could identify some emerging work practices. We summarize the main requirements to support the work practices that were most valuable during storyboarding, and provide concrete suggestions on what type of design constructs can support these requirements in a digital tool. We relate our requirements to design patterns for collaborative tabletop applications described by Remy et al. [Remy 10]. Table 5.1 gives a structured overview of the associated design patterns.

5.4.1 Allow for differences, support agreements

One of the first things to bear in mind is the individual preparation of a storyboarding session. Atasoy and Martens [Atasoy 11] mention for example that most designers continuously accumulate graphical material. They frequently use this material as a source of reference and inspiration, browsing through their collection to see if anything might be useful for the project at hand. A digital tool should not only be able to import all kinds of digital artifacts, but tangible artifacts such as paper documents should be considered as well, since paper is still a very ubiquitous medium. Prior studies indicate, for instance, that a designer still prefers pencil and paper early in the design process [Bailey 01]. This maps to the pattern *Input Tangibles*, or *Replace Physical Paperwork* in case hardware support for tangibles is missing.

In our study, preparation resulted in many different artifacts, including device or task descriptions, UI designs, and requirements. Relations between artifacts were also considered during preparation. As the representation style and viewpoints differed greatly and the members of a multidisciplinary team are already accustomed to specific tools and devices, we should not enforce one particular way of preparing artifacts. The accustomed tools and devices can

Requirements	Pattern categories: Ergonomics / Special	Interface	Usability and Collaboration	Extending Input
Allow for differences, support agreements			*Balanced Participation, Replace Physical Paperwork*	*Embedding Electronic Devices, Input Tangibles*
Facilitate different approaches in structuring		*Zoomable Interface*	*Hand Gestures*	
Maintain the design rationale		*User Identification*		
Favor shared over personal space	*Large Collaboration Table, Physical Object Storage Bin*	*Zoomable Interface*	*(Private Space)*	*Embedding Electronic Devices*
Support visible and direct physical interaction			*Replace Physical Paperwork, Hand Gestures*	*Pen Input Device, Physical Keyboard, Input Tangibles*

Table 5.1: Overview of the mapping of requirements on collaborative tabletop design patterns of Remy et al. [Remy 10].

be supported by integrating personal devices through the pattern *Embedding Electronic Devices*, and by allowing an easy exchange of data between devices and the storyboarding tool.

Given the differences in the prepared artifacts and viewpoints, the team members had to come to an agreement on several occasions. All teams, for example, had to agree on the devices that would be used in the home automation system. They also had to agree on a visual representation, so they scattered the available images across the table and chose which picture would represent a particular person, room or device. Since involving all team members in the decision making process results in more comprehensive storyboards, the tool should enable everyone to contribute in the same way, preventing more quiet users from being less involved and facilitating balanced decisions supported by the entire team. We already mentioned that interactive tabletops produce more equitable participation in the related work section, which can be enhanced by the pattern *Balanced Participation*. To enforce decisions supported by the entire team, a voting widget [Ryall 05] may require all users (or a quorum) to agree.

5.4.2 Facilitate different approaches in structuring

Structuring the storyboard happened in two ways: one team preferred a spatial arrangement, connecting scenes to a particular location, while the others employed a temporal arrangement by organizing their scenes chronologically. Since there is no "one best solution", a digital tool should not be restricted to one particular arrangement and should allow teams to create (different alternatives of) scenes and connections between them freely. In Storify [Atasoy 11], for instance, a team can add multiple alternatives per storyboard frame with the purpose of discussing various user experiences. The Anecdote [Harada 96] tool, on the other hand, allows various design styles by providing several views of the design, including an outline view, timeline view, and scene view.

Creating multiple alternatives of a scene and switching between multiple arrangements of scenes can offer different perspectives. However, teams will not take advantage of such features if the actions are too time-consuming. Existing tools like SILK [Landay 95], DENIM [Lin 00] and DEMAIS [Bailey 01] support quick creation and editing of sketches by means of gestures, which can be achieved on a tabletop through the pattern *Hand Gestures*. To quickly move between various views of a design, DENIM relies on zooming, which can be achieved through the pattern *Zoomable Interface*.

5.4.3 Maintain the design rationale

Visually representing the future home automation system stimulated the teams to discuss some unclear and challenging features. Despite the interesting discussions, almost none of those considerations or decisions were included in the storyboard. Since the design rationale is often valuable for later stages of user-centered design and development, a digital tool should include features to record the rationale. After investigating the relationship between imagery and design rationale, Wahid et al. [Wahid 10] state the importance of presenting the rationale in a designer-digestible format. This format depends on factors such as the homogeneity of the team and the familiarity of the team with the problem.

To maintain the design rationale, the digital tool can monitor all the artifacts and (encourage team members to) connect those artifacts to the storyboard (e.g. connect a designer's user interface sketches to a particular scene). SILK [Landay 95] also enables, for instance, designers to examine, annotate and edit a complete history of the design. Maintaining the design rationale, in combination with the balanced participation we mentioned earlier, may lessen the dissatisfaction some participants reported regarding the extent of their contribution, because their artifacts did not end up in the actual storyboard. To keep track of vocal discussions, audio or video annotations can be connected to the storyboard.

To monitor all storyboard artifacts, the tool can track their ownership or origin by applying the pattern *User Identification*. Haller et al. [Haller 05] state the importance of clearly identifying who is manipulating each data object in Coeno-Storyboard. Similarly, Avila-Garcia et al. [Avila-Garcia 10] use a DiamondTouch [Dietz 01] tabletop to identify the input of up to four different users, because identifying, saving and tracking contributions made by team members can be relevant in a decision making scenario. To support easy logging and audit trail creation, identity-differentiating widgets [Schmidt 10b] or lenses [Schmidt 10b] can be incorporated. The subject of user identification is discussed in more detail in Chapter 6.

5.4.4 Favor shared over personal space

While preparing, participants each created a personal workspace. Within the boundaries of our observational study, privacy was never an issue when participants shared data. In a real-life setting, however, privacy might come into play from time to time [Shoemaker 01], although further studies are required to investigate this aspect in the context of storyboarding in multidisciplinary

teams. In case privacy becomes a concern, the pattern *Private Space* is applicable.

During the cooperative storyboarding sessions, almost all work was done in the shared space between two participants or toward the middle of the table, even when multiple scenes were being created in parallel. Personal workspaces were still used sporadically, for actions such as writing on a post-it note or consulting the instructions or preparation. The sides of the table were mainly used for storage (e.g. toolboxes, available images, finished scenes). Since space is often at a premium, care has to be taken with personal workspaces or toolboxes taking up lots of space, leaving too little shared space to support a clear overview (as requested by participants in the questionnaire) and effective collaboration.

A straightforward, but not always feasible approach is to use a tabletop that offers enough space for all team members and their expected tasks, as stated in the pattern *Large Collaboration Table*. Integrating personal devices through the pattern *Embedding Electronic Devices* can reduce the problem of limited space, since they can act as personal workspaces. The pattern *Physical Object Storage Bin* can be applied to store unused physical objects, and the digital space can be extended by making space-demanding components zoomable through the pattern *Zoomable Interface*. Another solution is to extend the environment with additional displays. Ryall et al. [Ryall 04] state that for larger groups it might be necessary to add additional vertical displays for shared information. Avila-Garcia et al. [Avila-Garcia 10] also suggest the addition of one or more vertical displays. However, their goal is to accommodate passive team members, as one of the displays could show the interactions that are taking place on the tabletop. In our case, we want to avoid members being passive, and adding more displays may have a detrimental effect on balanced participation.

5.4.5 Support visible and direct physical interaction

All participants favored a shared device such as an interactive tabletop when asked about their preference for a digital system, because it makes collaborating easier and it encourages involvement and discussion. They commented that physical interactions on a shared surface emphasize what is being done and make participants explicitly aware of the progress and contributions. Some participants voiced concerns over the fluency of sketching and text entry on a digital tabletop. These concerns can be alleviated to some extent by incorporating additional input devices, such as a physical pen (the pattern *Pen*

Input Device) and keyboard (the pattern *On-Screen Keyboard* or *Physical Keyboard*), or by supporting paper (the pattern *Input Tangibles* or *Replace Physical Paperwork*). About half of the participants also suggested including a personal device to consult preparations or take notes, with the ability to easily share items with others (achievable through the aforementioned pattern *Embedding Electronic Devices*). A point of attention, however, is the possible decrease of mutual awareness and involvement when personal devices are being used extensively.

5.5 Conclusion

Digitizing the collaborative storyboarding process, and with it the resulting artifacts, broadens the opportunities to reuse them at later design and development stages. To enable an informed design of collaborative storyboarding tools, we identified the following basic requirements through an observational user study that includes an analysis of group interaction in multidisciplinary teams: allow for differences and support agreements, facilitate different approaches in structuring, maintain the design rationale, favor shared over personal space, and support visible and direct physical interaction. By taking into account these lessons learned, resulting tools will be more likely to engage all team members, while respecting individual contribution and creativity. We strive for a balance in participation among members of a multidisciplinary team because involvement of all members results in more complete storyboards. The result of a storyboarding session should reflect all opinions and artifacts, also those of more reserved team members. Incorporating viewpoints of multiple disciplines remains a challenge, however, and cannot be entirely accounted on the storyboarding tool.

Our study also uncovered a limitation in analyzing video recordings. Figure 5.2 visualizes the physical activity of participants on the table. As stated in Section 5.3.4, this visualization is automatically generated from the videos, but only represents the physical activity on the whole and not the activity of each user individually (for instance by visualizing each user's movements in a different color). Consequently, analysis was time-consuming, as it required us to rely on our observations to interpret the activity of individuals. In the next chapter, we propose a technique for non-intrusive identification of the different users around a table, which can, among other things, help with future analyses of a user's physical activity.

Chapter 6

Carpus: a non-intrusive user identification technique for interactive surfaces

Contents

6.1 Introduction

Interactive surfaces such as tabletops are well suited to support co-located collaboration because of their ability to track multiple inputs simultaneously, and we already presented various uses of such systems in collaborative environments throughout the previous chapters. However, the multi-user experience on these devices can be enriched significantly if the inputs on the surface are identified by associating them with particular users. That way, the system is able to differentiate between the actions of multiple users.

New opportunities are, for example, widgets that are customized on a per-user basis [Ryall 05] or virtual lenses that allow for personalized input and output in a multi-user application [Schmidt 10b]. In addition, we also covered a number of opportunities in the preceding chapters:

- Based on our evaluation of various strategies to invoke and visualize help, we mentioned in Section 3.2.6 that help must serve the needs of both novice and experienced users, so ideally the help system would be able to adjust to a particular user.

- In the collaborative puzzle game we presented in Section 3.3.2, participants of our study had to use a personal avatar to interact with the puzzle pieces, because we wanted to log data regarding individual users.

- Whenever the rules of that puzzle game require two people to collaborate, players can easily "cheat". Since the game cannot prohibit interacting with someone else's avatar, one player is able to control both avatars at once, for instance to move heavy objects at a faster speed.

- In Chapter 4, we discussed interaction management and access control. When the user's identity is readily available with every touch, it is possible for multi-user applications to enforce such protocols more strictly on tabletops [Morris 04, Piper 06].

- In the previous chapter, we pointed out the usefulness of user identification in different circumstances, for instance to keep track of an object's ownership and origin during a storyboarding session, and to help with the analysis of users' activities during an observational study.

Although several approaches exist for identifying users of a touch display, each has important limitations. Some techniques require additional instrumentation of the users at the start of each session [Marquardt 11, Meyer 10,

Roth 10] or the use of a mobile device [Schöning 08]. Others are error-prone because they identify the shoes that the user is currently wearing [Richter 12], which requires additional methods to associate those identified shoes with the actual touch points on an interactive surface. DiamondTouch [Dietz 01] and Medusa [Annett 11] assign an identity to fixed positions around an interactive table, making them unsuitable for free-flow environments. In contrast to these approaches, Schmidt et al. [Schmidt 10a] presented a system that uses a biometric technique. However, very specific hand postures are required each time the user's identity is needed, which impedes the "naturalness" of the interaction.

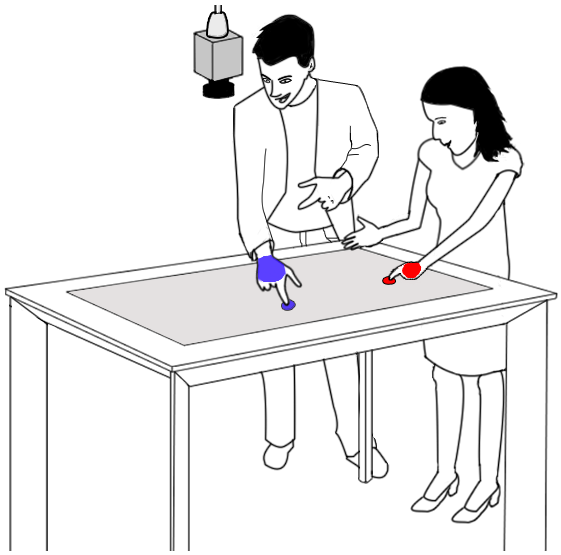


Figure 6.1: Carpus recognizes users by observing the dorsal region of their hands with a high-resolution camera mounted above an interactive surface.

Because of these limitations, we decided to explore user identification by looking at the user's hands by means of a high-resolution camera. Mounting a camera above a tabletop is a fairly inexpensive solution, and more importantly, non-intrusive from a user's point of view. We started from the very rudimentary idea of using the color model of hands to distinguish multiple users. As initial results were poor, we continued to explore new ways of analyzing the camera images. Our exploration finally led to a novel technique to relate touch points on an interactive surface to a user with high accuracy. We named our technique *Carpus*.

Carpus [Ramakers 12] supports walk-up-and-use scenarios in which each touch can be identified transparently once users have been registered. By mounting a high-resolution camera above an interactive surface, as sketched in Figure 6.1, we are able to extract identifying information from the back of the human hand (also known as the *dorsal* region) during traditional multi-touch interactions. As a result, our technique is non-intrusive and can be used in combination with a large range of touch technologies, including sensor-based, optical-based and vision-based hardware. Carpus supports user identification for setups intended for ad-hoc or informal collaboration [Gutwin 08], with users collaborating in unplanned fashion.

As a proof of concept, we developed a prototype of the application for a mobile phone retail environment that we described in our scenario in Section 1.3.1. The system can differentiate between actions of customers searching together for information about new products, a task that frequently involves collaboration [Morris 08]. In such a situation, it is possible for the system to interpret two-handed gestures unambiguously while multiple users are interacting simultaneously. In addition, a profile can be created for each customer, making it possible for the system to track a user's interests in order to recommend other relevant products.

Our main contribution is a new technique to *identify touch points on a surface non-intrusively by observing the back of the users' hands.* By applying this identification technique, the actions of multiple users of an interactive surface can be differentiated. Our approach is robust for hand postures that are common during the practical use of multi-touch systems. For traditional pointing, which is most common in touch-based interaction [Ryall 06a], Carpus uniquely identifies both hands of a user with more than 97% accuracy, even when up to twenty users are registered. As a result, it even becomes possible to differentiate between actions performed by a user's right and left hand in order to support non-symmetric division of labor [Guiard 87]. For more complex hand postures, which are less common, the recognition rates drop to 82% in the worst case scenario. A smaller group size is recommended to achieve higher recognition rates in these situations.

In the next section, we present more of the related work on identifying users. We briefly explain our Carpus approach in Section 6.3 and its benefits and limitations in Section 6.4. Next, we go through the details of the different steps of the algorithm in Section 6.5, followed by a summary of the algorithm's performance in Section 6.6. In Section 6.7, we investigate how unique the dorsal hand region is and we study the robustness of Carpus against posture variations. We also consider the benefits of adding a tracking algorithm to

Carpus, in Section 6.8. Finally, we illustrate the usefulness of Carpus by means of a scenario in Section 6.9, we discuss the overall results in Section 6.10, and we present our conclusions in Section 6.11.

6.2 Related work

Identifying users is a challenging and prominent issue in surface computing research. In this section, we review the previous efforts in more depth.

Limitations of existing solutions. As summarized in Table 6.1, existing approaches to user identification on interactive surfaces have at least one of the following limitations:

1. The naturalness of the interaction is impeded due to users being required to wear additional instrumentation or because of limitations on the supported hand postures.
2. Users need to stay at a fixed position or area around the interactive surface to preserve their identity. Changing position or leaving and reentering the area results in a new identity being assigned.
3. Associating an identity to a touch point on the surface can be difficult, resulting in ambiguities and accuracy problems.

Using additional instrumentation or specific hand postures. Visual tags [Marquardt 11] can be easily used to identify users. Electronic tags such as the IdWristband [Meyer 10] or the IR-Ring [Roth 10] are also able to identify users reliably, in this case by flashing a unique sequence in the infrared spectrum to the interactive surface. However, these techniques require users to wear equipment, and if users want to preserve their identity over multiple sessions, they always have to wear the same tag or they have to "log in" each time to associate their current tag with their identity. This overhead can be a burden, especially when the system is used for short, spontaneous and unplanned interactions.

Schöning et al. [Schöning 08], on the other hand, require the use of a mobile device to authenticate with the system, while Schmidt et al. [Schmidt 10a] extract biometric geometry features from the human hand. In order to measure these metrics reliably, however, a hand needs to be placed flat on the interactive surface. These approaches require either additional equipment or unusual hand postures, interrupting the "naturalness" of multi-touch interaction.

	Non-intrusive	Location invariant	Unambiguous association
IdWristbands [Meyer 10]	–	√	–
IR-Ring [Roth 10]	–	√	–
Tagged gloves [Marquardt 11]	–	√	√
HandsDown [Schmidt 10a]	–	√	√
Mobile Phone [Schöning 08]	–	√	–
DiamondTouch [Dietz 01]	√	–	√
Medusa [Annett 11]	√	–	–
Interaction workspaces [Kim 09]	√	–	–
Hand tracking [Dohse 08]	√	–	√
User tracking [Thelen 12]	√	–	√
Bootstrapper [Richter 12]	√	√	–
Carpus	√	√	√

Table 6.1: A summary of the strengths and weaknesses of related user identification techniques for interactive surfaces.

Making use of positional data. Other researchers have presented non-intrusive user identification by associating touches to users' positions around an interactive tabletop. DiamondTouch [Dietz 01] leverages a specialized capacitive surface to transmit an electrical charge through a user's body when touching it. A receiver unit inside each user's chair is used to register all touch events of the user who is currently sitting on that chair. If users swap chairs, their "identity" will change.

Annett et al. [Annett 11] instrumented a tabletop with arrays of proximity sensors to relate each touch to a user's position. Although this technique can track users' bodies that are moving near the tabletop, the system cannot preserve identities when users move further away, unless it requires user to log in each time they return to the tabletop. The same holds true for assigning movable and resizable regions to individual users, or using heuristics to divide the interactive surface into discrete territories [Kim 09]. When a touch point is located in a certain region, the system assumes that it belongs to the user associated with that region.

Tracking (the hands of) users with an overhead camera [Dohse 08] or depth sensor [Thelen 12] results in similar limitations. It enables the tabletop to associate each touch with a user, but if a user leaves the area that is being observed by the overhead camera or depth sensor, that user will be assigned

a new identity upon return. These identification techniques can be practical in situations where the users' positions around the table are fixed or where it is unnecessary to preserve identities when users leave, but they cannot be used in free-flow environments where the interaction is interrupted frequently by other activities and the users' positions around the table are likely to vary.

Identifying the user's shoes or gait. A few projects have demonstrated the potential of revealing identities by observing shoes or gait. Bootstrapper [Richter 12] extracts features from the top of a user's shoes, whereas Multitoe [Augsten 10] observes shoe sole patterns. Orr et al. [Orr 00] took another approach, identifying users by analyzing the forces and timings while walking on custom-built floor plates. However, these extracted features are not unique, as users can wear the same shoes or walk with an abnormal gait.

In addition, identification techniques such as shoe recognition, or for instance face recognition [Abate 07], suffer from a large space between the touch point and the region where the identity is extracted. Relating touches to these identified regions requires additional tracking, and is therefore more challenging and error-prone. This unreliable gap can be eliminated entirely by extracting the user's identity directly from the touch, using for example fingerprints [Holz 10]. However, more research is needed to integrate fingerprint scanning into multi-touch hardware.

6.3 Carpus

Our approach uses a high-resolution overhead camera, as shown in Figure 6.1, and works as follows:

1. Carpus continuously captures frames from above the interactive surface and extracts the visible dorsal hand regions.

2. Only when a user performs a "touch down" event, unique features are extracted from the hand region visible at the location of the touch point in the last captured frame.

3. These unique features are matched against a database of feature-user pairs that was constructed beforehand during a short training session.

4. The resulting user identity associated with that touch becomes available to the application running on the interactive surface. The surface then tracks the touch point to automatically identify subsequent finger movements.

The dorsal hand region, which is marked in Figure 6.2, is well suited to user identification in a collaborative environment for several reasons. To begin with, each person's dorsal hand regions have many strongly identifying characteristics, such as lines, grooves, folds and furrows [Napier 93]. Furthermore, the dorsal hand region is quite large and is also the part of the hand that is the least flexible [Napier 93], making it possible to capture consistent patterns in the skin over many frames. The dorsal hand region is visible to an overhead camera throughout the majority of hand postures encountered during interaction with a tabletop, something that is not true of fingers (e.g. when performing vertical touches [Wang 09]).

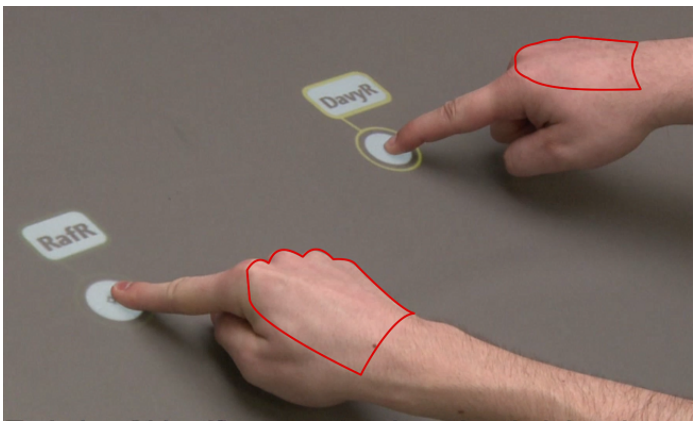


Figure 6.2: Unique features are extracted from the dorsal hand region. Fingers are excluded from the region.

6.4 Benefits and limitations

Carpus overcomes some drawbacks of current identification approaches (Table 6.1). Our technique provides transparent identification of users by extracting unique features directly from the back of the hand. Once users are registered, Carpus supports real walk-up-and-use scenarios. Furthermore, the camera can be mounted above any type of surface and captures the entire interaction. As a result, each touch point can be unambiguously related to an identified skin region, enabling identification of touches even when arms of users overlap. Carpus is also able to differentiate between the two hands of a user, which enables non-symmetric devision of labor [Guiard 87].

Carpus is, however, subject to a few limitations. First, the dorsal hand region needs to be clearly captured by the camera. When performing very fast movements, it is possible that the camera cannot capture a sharp image of the hand. This can easily be addressed by using a camera with an appropriate shutter speed. Furthermore, while Carpus can handle most common hand postures encountered during typical multi-touch interactions, some techniques [Morris 06, Harrison 11b] require the user's hand to be in such a position that the back is not visible to an overhead camera. In order to get a clear image of the dorsal region in those circumstances, we can add multiple cameras to our setup, capturing the hand from different points of view.

Secondly, Carpus has not been designed to offer authentication for privacy or security reasons. The current version of our algorithm can differentiate between users working simultaneously on an interactive surface, but has difficulties eliminating users who are not registered. As a result, the automatic detection of a "new" unregistered user is unreliable, and thus the users' hands need to be registered beforehand to achieve higher recognition rates. However, registration is only a one-time cost compared to techniques that require configuration at the start of each session [Meyer 10, Roth 10].

Lastly, the reported recognition rates are only representative for groups of people with Caucasian skin. We recruited people with similar skin color for the evaluation of Carpus, because the similarity makes it harder to find unique features. Further research is needed to investigate the effect of different skin types on the accuracy of Carpus.

6.5 Skin region and identity extraction

In order to identify a hand that is visible in a captured frame, Carpus performs the following four steps:

Step 1: Extraction of the dorsal hand region.

Step 2: Feature extraction.

Step 3: Feature matching.

Step 4: Relating touches to identified regions.

6.5.1 Step 1: Extraction of the dorsal hand region

First, the dorsal hand region needs to be detected in a captured frame. When a touch event occurs, Carpus does this in three sub-steps:

A. Extraction of the skin region.

B. Detection of visible fingers.

C. Detection of the position of the wrist.

Two iterations of this algorithm are executed for each detected skin region. During the second run, a more accurate segmentation is used in order to detect the dorsal region of the hand more precisely in particular situations, as we explain below.

A. Extraction of the skin region

Skin segmentation [Phung 05] involves finding ranges of intensity values for which most skin pixels fall in a given color space. The image is first converted to the YCrCb color space [Phung 02] to obtain a decision rule that is robust under varying illumination conditions. To support various types of human skin, all pixels in a relatively large intensity range are classified as skin:

$$Y > 20, 85 < Cb < 135, 135 < Cr < 180 \tag{6.1}$$

Figure 6.3-A1 shows areas of pixels that were selected using this static segmentation rule. Because of the large intensity range, shadowed regions between fingers are erroneously classified as skin pixels. Therefore, a second, more dynamic skin segmentation rule is used after the position of the wrist and the fingers are detected. This dynamic segmentation rule is based on the average (μ) and standard deviation (σ) of the intensity of the pixels in the detected dorsal region:

$$\begin{aligned} \mu_Y - 2\sigma_Y < Y < \mu_Y + 2\sigma_Y \\ \mu_{Cr} - 2\sigma_{Cr} < Cr < \mu_{Cr} + 2\sigma_{Cr} \\ \mu_{Cb} - 2\sigma_{Cb} < Cb < \mu_{Cb} + 2\sigma_{Cb} \end{aligned} \tag{6.2}$$

In Figure 6.3-A2 the shadowed regions between the fingers are correctly excluded with dynamic segmentation. Parts of an arm or finger are sometimes erroneously classified as non-skin regions because of variations in the observed skin color. However, this does not influence the outcome of our algorithm, since only the back of the hand is used.

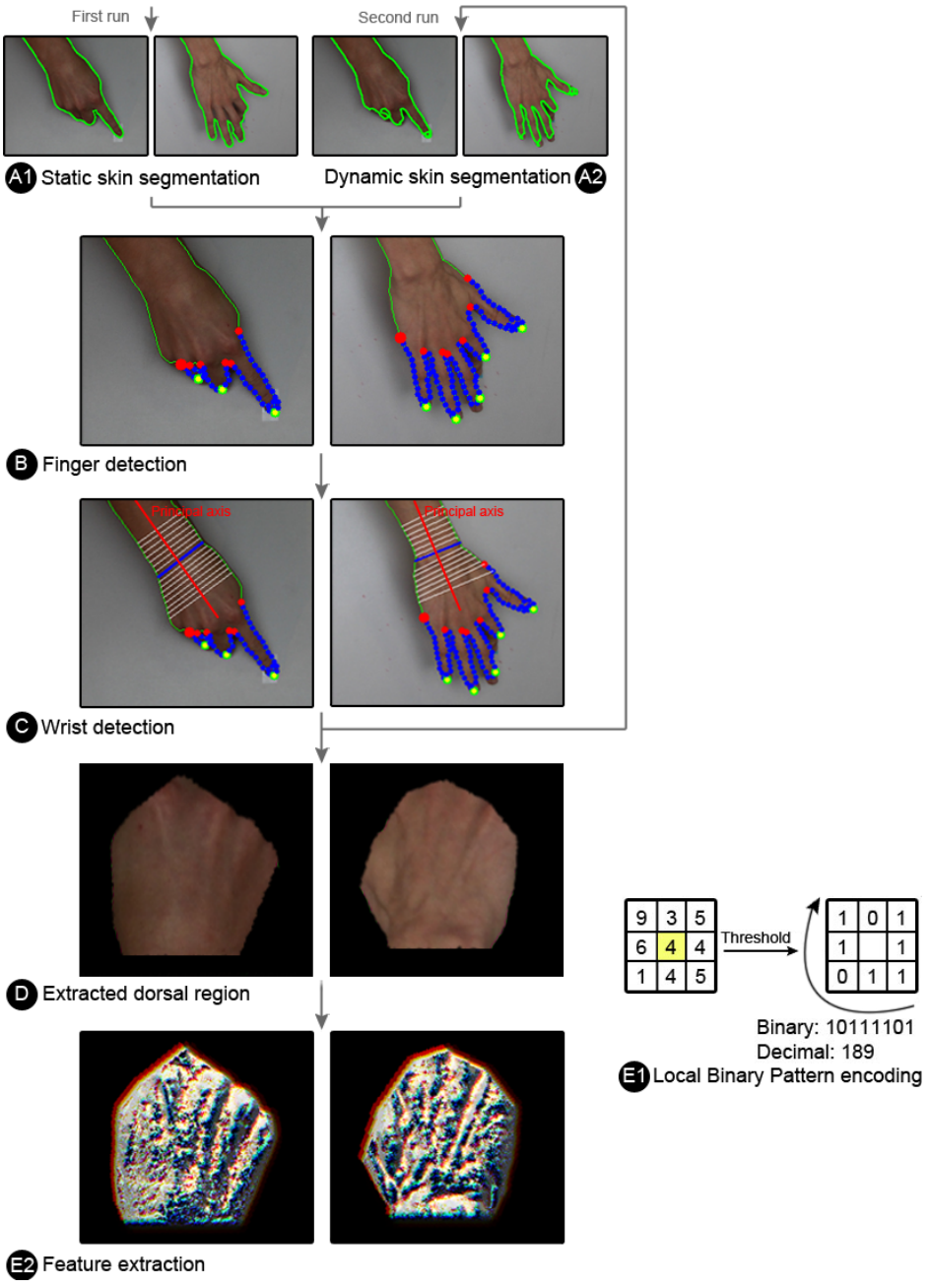


Figure 6.3: Extracting details from the dorsal hand region (step 1 and 2): (A) detect skin region in the image; (B) detect fingers; (C) detect wrist direction and position; (D) extract dorsal hand region; (E) encode fine-grained details using LBP.

B. Detection of visible fingers

Once the skin regions are extracted from a captured frame, fingers need to be detected in order to exclude these regions from the final contour. We first detect the tips and then the phalanges of the fingers, as depicted in Figure 6.3-B. To detect fingertips, we use a curvature-based approach similar to Malik and Laszlo [Malik 04] and Segen and Kumar [Segen 98]. The vectors from each contour point k to $k+n$ and $k-n$ are computed (n is a fixed value and depends on the distance between two contour points). If the angle between the two vectors is below some threshold, then the contour point is marked as a fingertip.

We use a relatively high angular threshold value of 60 degrees to make the fingertip detection algorithm more "greedy". Our algorithm even classifies the knuckles of folded fingers as fingertips, because we want to exclude all visible finger regions from our contour and are not interested in the exact position of the real fingertips. The misclassification of finger valleys is avoided by taking into account the sign of the angle between two vectors while processing the contour in counterclockwise direction. Finally, non-maximal suppression is used to avoid detecting fingertips too close to one another.

In order to extract entire finger regions, the phalanges of the fingers also need to be detected. Boreki and Zimmer [Boreki 05] do this by finding the position of finger valleys between two consecutive fingertips. However, we need to support all hand postures, even when only one finger is visible (e.g. pointing). Therefore, we take a different approach. Our algorithm starts roaming the skin contour from each detected fingertip to the left and right. Each contour point is then classified as part of a finger phalange until the length of the finger has reached a maximum value or the angle between two phalanges exceeds an angular threshold value. This threshold is inversely proportional to the length of the finger and ranges between 40 and 60 degrees in our algorithm.

C. Detection of the position of the wrist

Before we are able to detect the position of the wrist, we need to determine its orientation. The orientation of the wrist is based on the orientation of the arm, which is given as the principal axis of inertia of the extracted skin region. The orientation of the principal axis, shown in Figure 6.3-C, can be computed from the image moments up to the second order, as described by Freeman et al. [Freeman 98]. When the aspect ratio of the skin region along the direction of the principal axis is almost equal to 1:1 (e.g. a user with long

sleeves performing a vertical touch), this approach is less reliable. In that case, we use the principal axis of the entire arm region, obtained by doing an additional background subtraction step.

Once the orientation of the wrist is known, the precise position of the wrist can be determined. Kioke et al. [Koike 01] assume that this position is always located at a fixed distance from the top of the hand. However, we observed that the distance from the wrist to the top of the hand can vary extensively, even when fingers are detected, because hands are free to move in three dimensions when working on an interactive surface. Therefore, we take an approach similar to Choi et al. [Choi 09], and evaluate the width of the contour along the direction of the principal axis from the finger tips to the arm region. The width of the contour has very specific characteristics at the position of the wrist. In the direction of the fingertips, the width will vary significantly, but in the direction of the arm, the width will be fairly constant. We consider the wrist to be located outside the finger regions, at the position where the width ratio reaches its maximum value, as illustrated in Figure 6.3-C.

6.5.2 Step 2: Feature extraction

Once the dorsal hand region is detected using the algorithm described in the previous step, as shown in Figure 6.3-D, unique features need to be extracted from this region. There are several interest point detectors, such as SIFT [Lowe 99] and SURF [Bay 08]. However, the skin of the hand is flat and does not contain a lot of these interest points. Therefore, we use the pattern of the entire skin region as a feature. Before we can compare skin patterns in different images (step 3), a texture descriptor is needed to quantize these patterns. Color histograms [Swain 91] are the most straightforward image descriptors, but this technique is very sensitive to changes in illumination. During preliminary studies, we noticed that color histograms are not able to capture unique information consistently over time, because the illumination of the hand changes significantly due to shadows cast by the user's body.

Local Binary Patterns (LBP) [Ojala 96] is a technique to describe very fine-grained details of textures in images. In contrast to color histograms, LBP is invariant to any monotonic change in intensity values. The LBP-operator labels the pixels of an image by thresholding the three by three neighborhood of each pixel with the center value. The decimal representation of the number formed by the concatenation of all binary digits in the neighborhood is the intensity value of the central pixel in the local pattern, as explained in Fig-

ure 6.3-E1. Carpus uses Circular Local Binary Patterns (CLBP) [Ojala 02], an extension to the basic LBP. The CLBP-operator samples the neighborhood circularly with a variable radius. If a point on the circle does not correspond to image coordinates, the point gets interpolated. The CLBP-operator is thus able to capture patterns at different scales: a small radius captures local details, whereas a larger radius captures more global information.

In our preliminary exploration, we noticed that the luminance component of the hand's skin texture (i.e. the hairs, lines, grooves, folds and furrows) contains most of the discriminative information. The color information was not very discriminative. Therefore, we only capture information in the luma (Y) component of the extracted skin regions. When capturing details at radii two, five and ten using the CLBP-operator, we can describe most of the unique features of the dorsal region of the human hand, as seen in Figure 6.3-E2.

6.5.3 Step 3: Feature matching

To identify users, Carpus matches descriptions of textures captured using the CLBP-operator with a database of previously captured features. Before features of different images can be compared, we need to ensure that these features are captured at the same scale. Therefore, we only process a detected dorsal hand region when a "touch down" event is performed.

A naive approach to comparing descriptions of textures would be a simple histogram comparison. When using this approach, however, all spatial information is lost. Through experiments, we discovered that without this spatial information, the extracted features are not very discriminative. Therefore, we use a technique similar to Ahonen et al. [Ahonen 04]. First, the skin region is divided into square patches of equal size, as illustrated in Figure 6.4. We currently use patches of twenty by twenty pixels. Next, histograms of these local micropatterns are computed. Histograms of the same patch, which contain information of fine-grained details at different scales, are concatenated to form a single description of the patch.

Opposed to the approach of Ahonen et al. [Ahonen 04], the local histograms of all patches in an image are not concatenated. Concatenation of these local histograms implies the integration of the hand's shape as a feature. Since the shape of the hand can vary significantly under different postures, we take another approach. Carpus compares two images by finding the best match for each patch in a predefined neighborhood around the corresponding position in the other image, as shown in Figure 6.4. We use a neighborhood of two times the patch size.

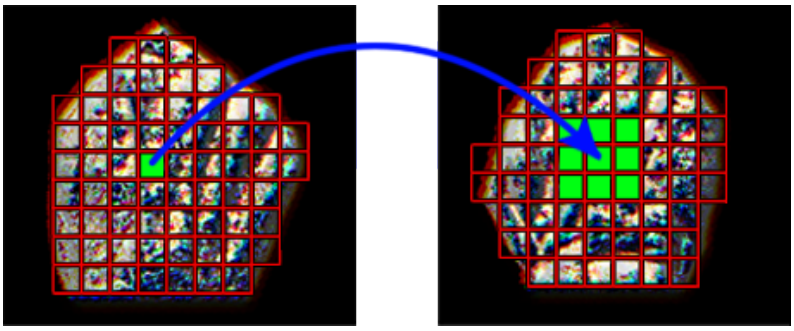


Figure 6.4: Feature matching (step 3): finding similar patches in a predefined neighborhood.

Experimenting with different histogram comparison metrics, we found that the histogram intersection technique provides the best matching of patches. As a result, the similarity between features extracted from two dorsal hand regions is the sum of the number of similar pixels for each of the patch's best matches. An unknown hand is assigned the identity of the dorsal hand region in the training set with the highest number of similar pixels.

6.5.4 Step 4: Relating touches to identified regions

Once the identities of all extracted dorsal hand regions are known, each touch point on the interactive surface needs to be related to an identified skin region. In some touch identification techniques [Richter 12, Meyer 10], bridging the spatial "gap" between the region where the identity was extracted (e.g. shoes, wristband) and the actual touch point may cause errors. Carpus is able to relate touch points to identified skin regions unambiguously, since each touch is directly related to a single dorsal hand region by the contour of the fingers detected in step 1.

The resulting user identity associated with that touch is made available to the application running on the interactive surface through UDP messages. The surface then tracks the touch point (for instance through its recognition software, which in our case is FTIRCap [Cuypers 08]) to automatically identify subsequent finger movements. Only identifying a hand on a "touch down" event, and not during subsequent movements, enhances the performance of our system significantly.

6.6 System specifications and performance

For our experiments, we use a Philips BDT4225EM/06 42-inch (107 centimeters) multi-touch display, which can detect up to six simultaneous touches through an infrared grid on top of the screen (Carpus was also informally tested in combination with other technologies, such as an FTIR tabletop). The display is mounted horizontally on a table, and a Point Grey Grasshopper2 camera with a resolution of 1624 by 1224 pixels is mounted above the surface. The overhead camera transmits 22 frames per second at 45 dots per inch via its network interface to an off-the-shelf laptop computer (2.1 GHz Intel Core 2, 4 GB RAM) running Carpus. Artificial light sources are used to ensure that over different sessions of our experiment, the lighting conditions remain unchanged.

Carpus is able to identify users in real-time and scales well to higher resolutions. Moreover, we did not focus on the performance of our implementation of the algorithm, so many optimizations are possible. The extraction of the dorsal hand region (step 1) takes 28 milliseconds on our system if only one hand is visible in a single frame with a resolution of 1100 by 790 pixels, 41 milliseconds if two hands are visible and 52 milliseconds if three hands are visible. When using a higher resolution camera, as in our setup, this step can still be performed at the specified lower resolution without reducing the accuracy of our identification technique.

After the position of the dorsal hand region is detected, features can be extracted from the original high-resolution image. The feature extraction and matching step is only executed on a hand that is detected at the position of the "touch down" event in the last frame, to ensure that all extracted features have the same scale. This process takes on average 43 milliseconds when the hand is captured at 45 dots per inch. When sixteen samples are used for each registered hand, matching features of a single hand takes on average 68 milliseconds if four hands are registered, and 155 milliseconds if eight hands are registered.

6.7 Evaluation of Carpus

In this section, we present two experiments with regard to Carpus: we first investigate the uniqueness of the dorsal hand region to validate our overall approach, and next, we evaluate to what extent Carpus can handle posture variations.

6.7.1 Uniqueness of the dorsal hand region

In this first experiment, we evaluate the uniqueness of the dorsal hand region over different users, as well as over different hands of the same user. Here, we only consider the hands-down posture [Schmidt 10a], because in this posture the overhead camera can capture clear images of the dorsal region. The robustness of Carpus with regard to posture changes will be tested in a second experiment, which we discuss in the next section.

Tasks

We recruited twenty-two participants, five female and seventeen male, between twenty-two and fifty years old. We instructed them to take off all jewelry (jewelry could actually increase the recognition rates, because it often provides very unique features). Each participant sat down at an interactive tabletop and placed her/his left and right hand flat on the surface with fingers spread. They did this at fifteen predefined positions that were evenly distributed over the entire surface area. In each position, our camera captured a single image. These images are used to train and test our system.

Procedure

From a set of 660 images, we simulated six scenarios that differed in the number of hands registered with the system (four, ten and twenty users, each registering one or two hands). For each scenario, we generated twenty-five sets of randomly drawn groups. For each set, a ten-fold cross-validation with a stratified random selection of training images is performed for each hand, resulting in a total of 34000 trials (for each scenario twenty-five sets $\times$ ten trials $\times$ two hands $\times$ number of users). A hand region is correctly identified if the system can not only match it to the correct user, but also to the correct hand of that user.

Results

Table 6.2 lists the recognition rates of our technique for the six scenarios. The accuracy is very high in all scenarios, although it slightly decreases when larger groups of users are registered with the system. These results demonstrate that our extracted features are unique, even for fairly large groups of users. In addition, note that if both hands of a user are registered, our system can distinguish between each hand the vast majority of the time. This enables identification of both hands and thus non-symmetric devision of labor [Guiard 87].

	Group size		
	4	10	20
One hand registered	99.5%	99.4%	99.1%
Two hands registered	99.4%	99.4%	99.0%

Table 6.2: Recognition rates for the hands-down posture when one or both hands of four, ten and twenty users are registered.

6.7.2 Robustness against posture variations

In the first study, we showed that the dorsal hand region can be used to reliably map hands to users. The goal of this second study is to show that Carpus is sufficiently robust for postures that are common during the practical use of multi-touch systems. When interacting on a tabletop, the overhead camera often gets a clear view of the dorsal hand region, e.g. when clicking a button, as shown in Figure 6.5-A. When scaling or rotating an object, as depicted in Figure 6.5-B and Figure 6.5-C, however, the captured dorsal hand region can be skewed, potentially influencing the accuracy of the recognition. To evaluate to what extent Carpus can handle posture variations, we ran a second experiment.

Tasks

We asked our twenty-two participants to perform four additional tasks. During these tasks, we did not give any instructions regarding hand postures with the purpose of capturing natural interaction.

1. In the first task, participants were asked to color the contours of a flower in a painting application using only their right hand (Figure 6.5-A). The application showed a preview of the flower, with the colors we expected the participants to use.

2. In the second task, participants scaled and rotated ten images (evenly spread out over the surface area) to fit them in a box using traditional free transformation gestures (Figure 6.5-B). This task was first performed with the right hand and afterwards repeated with the left hand. The center points of the images were fixed, so that there was no need to move them.

3. In the third task, those same images were displayed and participants

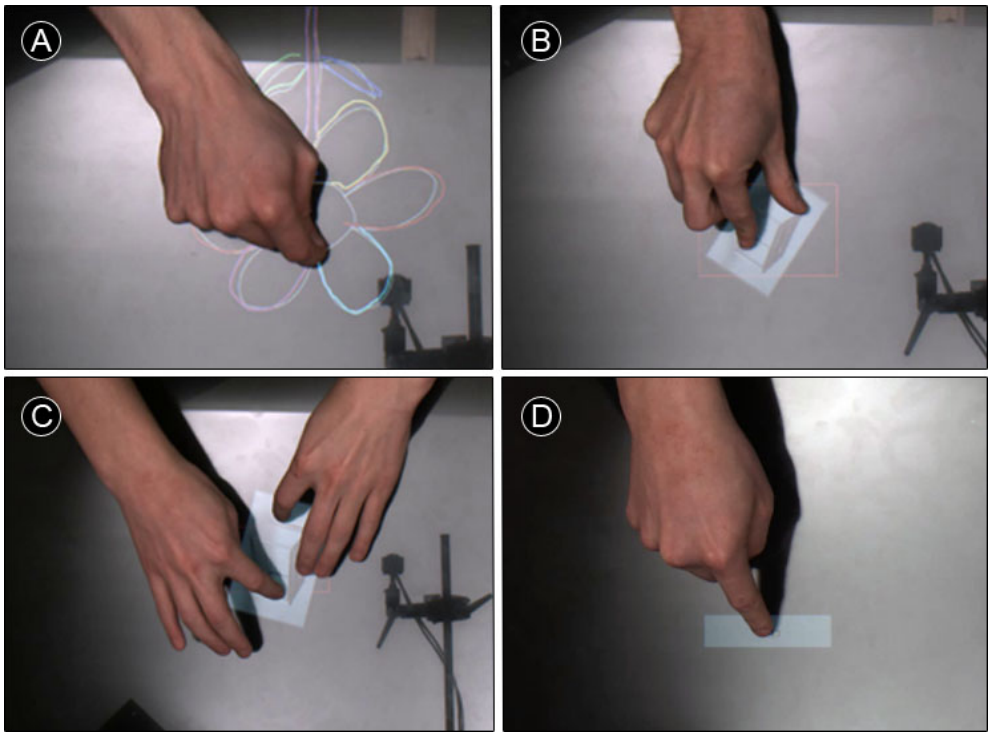


Figure 6.5: The four tasks in our second experiment: (A) painting a flower; (B) scaling and rotating images using one hand; (C) scaling and rotating images using two hands; (D) pressing buttons.

were asked to scale and rotate them using both their left and right hand together (Figure 6.5-C).

4. Finally, each participant was asked to click fifteen buttons that were spread out over the surface area, first with their right hand and afterwards with their left hand (Figure 6.5-D).

Data collection

During this experiment, all interaction was captured by our overhead camera and streamed to a PC, together with all touch events ("down", "move" and "up"). Afterwards, all frames were extracted in which one or more "down" events were registered, and these frames were used in simulations as training and test data. For the first, second and third task, we collected on average

35.5, 82.8 and 39.7 images per participant, respectively. For the last task, 30 images per participant were captured. In total, we collected 5009 images. For the third task, in which two hands are visible at the same time, we provided the position of the left and right hand to the system by processing those images by hand. With this data, the correctness of the result of our hand identification algorithm can be verified.

Procedure

Using our collection of captured frames, we simulated six scenarios that differed in the number of hands registered with the system (four, ten and twenty users, each registering one or two hands). For each scenario, we generated twenty-five sets of randomly drawn groups. We trained the system with randomly drawn samples for each hand that needed to be registered. This training set consisted of four samples of hands in the hands-down posture from the previous experiment, and four samples of the pointing task (task 2) and the free transformation task with one (task 3) and both hands (task 4). We determined that this collection of sixteen training samples per hand is needed if a large range of postures is to be supported.

After the training phase, we tested our system with ten randomly drawn samples for each hand in each of the four tasks (the randomly drawn samples were always different from the training samples), resulting in a total of 136000 trials (for each scenario twenty-five groups $\times$ ten trials $\times$ two hands $\times$ four tasks $\times$ number of users). As in the first experiment, a hand region is correctly identified if the system can relate it to the correct user and the correct hand of that user.

Results

Figure 6.6 summarizes our findings of the second experiment. These results show that hands in a pointing posture can be identified very accurately, because the overhead camera has a clear view on the dorsal hand region.

Carpus had the greatest difficulty identifying a single hand in a free transformation gesture, such as the one in Figure 6.5-B. We observed that users often rotated their hand in various directions while performing such a gesture. As a result, the dorsal hand region was highly skewed in many of the captured frames. In that case, matching features is much more difficult. We also noticed that, because of tilting of the hand, the space between two fingers was sometimes almost entirely occluded. That makes it harder for our algorithm to detect all fingers correctly, in order to exclude them. Parts of

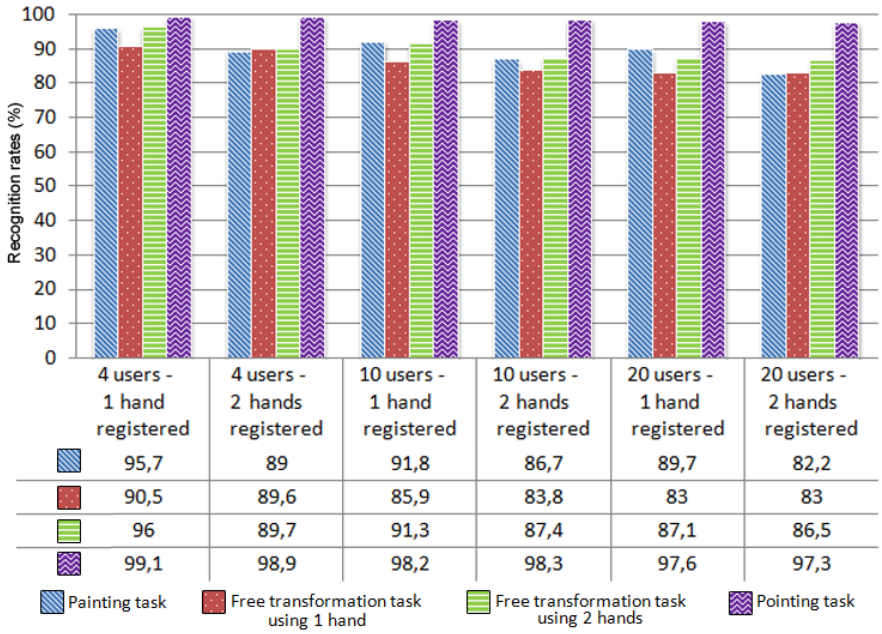


Figure 6.6: The recognition rates for different group sizes for all four tasks of our second experiment.

fingers still present in the feature extraction step will produce unreliable data, which can influence the accuracy of our technique. However, even for these challenging hand postures, the recognition rate for a smaller group of users is still relatively high.

The accuracy for the painting task was somewhat surprising, as we anticipated that this recognition rate would be almost identical to the pointing task because of the similarity of the expected hand posture. After analyzing our captured data, we noticed that some users tilted their hand during this task to reduce the occlusion of the active painting area, in order to paint the contour more precisely. This behavior causes the extracted dorsal hand region to be skewed, and makes feature matching more difficult. However, the results of this experiment show that, even when no instructions are given and the user can naturally interact with the surface, Carpus provides an accurate technique for identifying touches of a small group of users.

We reproduced our entire experiment, after adding four randomly drawn images of the painting task to the training set. The recognition rate of the painting task was significantly higher (97%, 96.5%, 97.8%, 96.2%, 96.1% and

95.5% for the six scenarios). The accuracy for the other tasks remained almost unchanged. This suggests that the recognition rate can be improved by training the system with samples of hand postures that are more similar to the hand postures that will occur in the application.

One can imagine many other hand postures that can be used on interactive surfaces. However, our tasks produced a wide range of postures that are very common during the practical use of multi-touch systems.

6.8 Extending Carpus with tracking

As discussed in Section 6.3, Carpus captures frames and extracts the visible hand regions continuously. However, unique features are only extracted from the hand region when a user performs a "touch down" event. In other words, the algorithm only considers the last captured frame when identifying a user. In this section, we consider tracking and identifying a user's hand as it moves across the screen, gathering data about that user's identity over time. If we can relate a user's hand to measurements from previous frames, we may be able to prevent certain errors during the identification, for instance when the dorsal hand region is not clearly visible to the overhead camera (e.g. the user's hand is tilted sideways or upside down).

To implement a suitable tracking algorithm, we first have to consider a few restrictions of Carpus. First of all, Carpus assumes that all the dorsal hand regions that need to be identified are located at approximately the same distance from the overhead camera. In our current approach, Carpus only identifies a hand when it touches the surface, so this requirement is always satisfied. A tracking algorithm can take this constraint into account in a similar manner, by limiting the identification of a tracked hand to frames during and immediately following touches. Alternatively, we could scale the dorsal hand regions to the correct size, but the payoff of this added complexity would be minimal, as the user's hand is less likely to be in a reliable posture when it is floating in the air.

Moreover, we do not have a reliable method of automatically assessing the trustworthiness of an identity provided by Carpus. Every now and then, a mismatch of identities occurs even if the feature matching step results in a considerable similarity. Therefore, we cannot simply use tracking as a fallback option in case Carpus is incapable of determining the correct identity, but we need to come up with another way of using the temporal data. Bearing in mind these different factors, we propose the following approach:

- We accumulate data by tracking a hand over time, using Carpus to identify that hand in all frames during and immediately after a touch. This results in a dataset that contains the tracking ID of the hand, the user identities that Carpus associates with that hand over time, and the number of times that a certain identity occurs, as shown in Table 6.3.
- When a sufficient amount of data is available, we analyze it to determine if we should rely on the tracking rather than on the identification provided by Carpus. In the dataset in Table 6.3, it is safe to assume that the hand with ID 0 belongs to the user with ID 1. In such a case, the tracking data can be used to "override" the identity provided by Carpus.
- If the accumulated data is not conclusive, as is the case with hand ID 1, both the tracking data and the results provided by Carpus are not very dependable. We keep relying on Carpus and add new data to the dataset, which will hopefully become more conclusive over time. In addition, we can report to the application that the provided identity is not very reliable.

hand ID	user ID	number of occurrences
0	1	28
0	5	2
1	4	3
1	3	2

Table 6.3: Data accumulated by tracking a hand over time, using Carpus to identify that hand in all frames during and immediately after a touch.

Although this approach can help to improve the overall recognition rate of Carpus, it also has some limitations. First of all, the algorithm has no immediate effect, as we have to accumulate a sufficient amount of data before tracking comes into play. Possible errors made by Carpus will therefore not be prevented during that period. Furthermore, if a user's first interactions do not involve reliable postures, tracking may take longer to establish the correct identity, as is the case with hand ID 1 in Table 6.3. Finally, if a hand disappears from the camera's field of view and then reappears, it will be assigned a new hand ID, and the data has to be accumulated again.

Initial tests indicate that tracking the users' hands, using OpenCV's[1] pyra-

[1] `http://opencv.org`

midal implementation of the Lucas-Kanade algorithm [Lucas 81], can prevent a number of errors. However, further research is needed to find the optimal parameters (e.g. the various thresholds used by the tracking algorithm) and to assess the exact effect on the recognition rates. Note that tracking was not considered in the aforementioned studies, because it would bias the results depending on the reliability of the tracking and the type of user interface (e.g. how frequently a user's hands leave and enter the surface area).

Tracking can also be useful in other ways. We found, for instance, that the recognition rate is higher when training the system with samples of hand postures that are more similar to the postures that occur in the application. This suggests that the accuracy of our technique could be improved for some hand postures by refining the training set over time with samples of hand postures that are performed in the actual application. The tracking algorithm can be used to accomplish this, as samples for which no good match is found in the training set can be added to the training data after a reliable sample of the same hand has been identified.

6.9 Usage scenario

Carpus enables non-intrusive and transparent identification of users on interactive surfaces. This allows an entire range of new applications in which people collaborate on a shared surface, such as buying new products [Morris 08]. We developed an interactive application that can be deployed in, for example, a mobile phone retail environment. Families or friends shopping together for a new mobile phone can collaborate on a multi-touch tabletop located inside the store to find more information about products and compare specifications.

Figure 6.7 illustrates a typical usage scenario, which we already described in detail in Section 1.3.1. John and Jane are looking for a new mobile phone. A nice looking phone catches their attention and they walk to the interactive display to get more information about this product. (A) Because it is the first time that they have visited the store, a simple widget allows them to very quickly register both hands with the system. (B) Jane then puts the phone on the interactive display to get more information (the device is currently recognized by a visual tag). (C) Identity information from Carpus is used to create a temporary profile for each customer. This makes it possible to track the user's interests in order to provide product recommendations. (D) While John reads through the specifications of recommended phones, Jane finds another nice mobile phone and compares two products using a two-handed gesture. (E) Carpus unambiguously recognizes this gesture because the system can

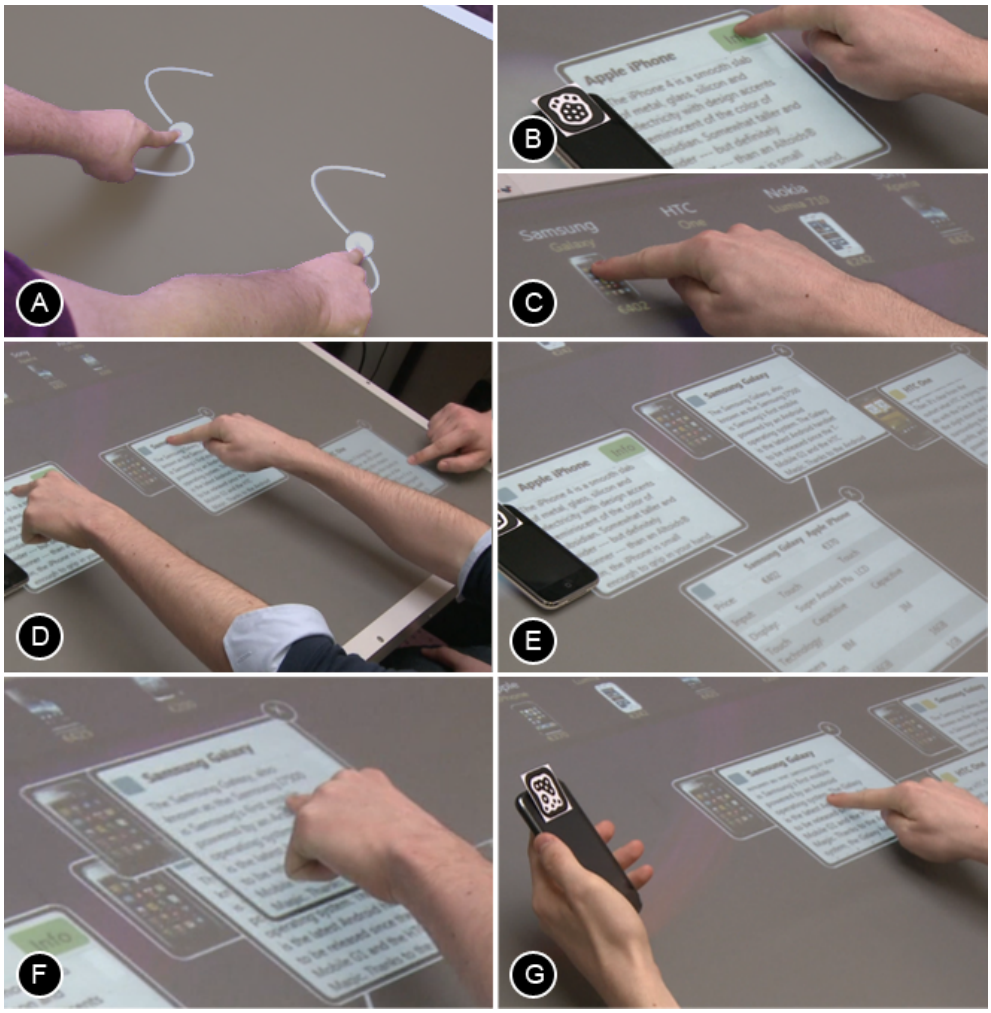


Figure 6.7: Carpus enables non-intrusive identification of (both hands of) users, for example in a mobile phone retail environment, allowing users to find more information about products and compare specifications.

distinguish between the input of both users. (F) John notices Jane's actions and also wants to take a look at the product in which she is interested. Jane then makes a copy of the information by moving it with her left hand to John. (G) John and Jane are now both interested in the phone suggested by the system and go find it in the shop. When returning back to the interactive display, they notice that other customers started a new session. However, when they touch the display, the system uses the identify information to restore their session.

6.10 Discussion

Carpus extracts identity features from the back of the human hand. However, as mentioned in the benefits and limitations section, the detection of this region is difficult in two particular situations. The first situation is when identification is needed while performing a very fast gesture, such as a swipe-gesture. It is possible that the camera cannot capture a sharp image of the dorsal region. This problem can be addressed by using a camera with an appropriate shutter speed, or by adding tracking capabilities to the algorithm in order to identify the dorsal region during slow movements, and track the arm during fast movements.

Secondly, researchers have investigated new types of interactions in which the dorsal hand region is not always clearly visible to the overhead camera [Morris 06, Harrison 11b]. Tracking the hands, as described in Section 6.8, can only partially solve this problem, because the system needs to gather reliable data before the tracking algorithm can take over. Using multiple cameras that observe the dorsal hand region from different points of view may alleviate this issue, making it much more likely that the system can capture a clear image of the dorsal hand region. Such a multi-camera setup can also be used to reduce occlusion problems that are inherent to the use of overhead cameras. This is especially the case when vertical displays or tilted tables are used in combination with Carpus.

In our first study, we found that the dorsal hand region is unique over a large group of users. In a second study, we demonstrated the robustness of our technique for a range of hand postures. Carpus has the highest recognition rate (97.3% when twenty users register both hands) when a user clicks buttons, which is a very common action in interfaces. Since we only identify a user when a "touch down" event occurs, a lot of other interactions are actually reduced to clicking (e.g. moving sliders, scrollbars and objects).

The recognition rate was the lowest when identifying scale and rotate ac-

tions performed by a single hand, as in Figure 6.5-B. This is expected, because users rotate their hands in all directions when performing such gestures. As a result, the camera captures a skewed view of the dorsal region that is more difficult to match with the training set. However, user identification during these actions is only required in very specific circumstances and the majority of applications only need the user's identity during pointing actions [Ryall 06a]. In addition, a tracking algorithm can be used to improve the recognition in such circumstances, as discussed in Section 6.8.

The recognition rate of Carpus is most likely sufficiently accurate for all postures when a small group of users are registered with the system. However, even in these situations, Carpus is currently not intended for applications that require real security (e.g. accessing emails, online banking). As the dorsal hand region contains no papillary ridges, our extracted features are not as persistent and immutable as fingerprints. It is, for example, possible for the skin to get a tan, and wounds can result in permanent scars. However, more long-term research is needed in this area to investigate the effects of these changes on the recognition rate of Carpus.

As already mentioned in the benefits and limitations section, our technique has some difficulties eliminating users who are not registered. In an additional study, we tested our system with four known and four unknown users, which resulted in an acceptance rate of 87% and a false acceptance rate of 7%. These results indicate that explicit registration is needed when higher reliability is required. However, registering with the system is only a one-time cost compared to systems that require configuration at the start of each session.

When a reduced reliability is acceptable, for example in case of very short interactions with non-crucial applications, spontaneous registration is possible. We already demonstrated the potential use of interactive widgets as a means for transparent registration in our usage scenario. We are exploring several interactive widgets for registration purposes that can be easily integrated in applications. Registration can be as simple as performing a single gesture before first usage (e.g. similar to the interactions required to unlock a smartphone). However, more experiments are needed to make sure that a sufficient number of different hand postures are captured during the interaction with this widget to get the best possible accuracy rates.

6.11 Conclusion

In this chapter, we presented Carpus, a non-intrusive technique for identifying users of interactive surfaces. Our approach relies on the extraction of unique

information from the back of the human hand. This hand region is extracted from high-quality images captured by an overhead camera. Fine-grained features are encoded using a special pattern description technique. For traditional pointing tasks, Carpus can uniquely identify both hands of a user with 97.3% accuracy, even when large groups of users are registered. For more complex gestures that occur less often, smaller group sizes are recommended to achieve a higher recognition rate.

Carpus enables touch identification for non-crucial applications in walk-up-and-use scenarios in which users interact frequently and in an unplanned fashion. In addition, our presented technique is able to differentiate between the two hands of a user, opening up even more interaction possibilities. Carpus is also easy to deploy and can be used in combination with all existing touch technologies. This makes Carpus suitable in many situations in which touch identification can enrich the multi-user experience, for example by using approaches such as iDwidgets [Ryall 05] and IdLenses [Schmidt 10b].

Pertaining to the research presented in this dissertation, possible applications of Carpus include customizing help on a per-user basis to better suit the needs of particular users, enforcing game rules that require several players to cooperate, upholding floor control policies and access control to documents on shared surfaces, and tracking the origin and ownership of artifacts during a collaborative storyboarding session. We also expect that Carpus can satisfy the demand for easy to deploy techniques to identify users in groupware studies, such as the observational study we presented in Chapter 5, or studies regarding territoriality [Scott 04], orientation [Barnkow 12], and so on. With Carpus, analysis of users' behavior will be significantly less time consuming, as each action is automatically associated with a particular user.

Part III

An engineering perspective

Chapter 7

NiMMiT: a graphical notation for modeling touch-based and multi-user interaction techniques?

Contents

7.1 Introduction

In the preceding chapters, we discussed touch-based and multi-user interaction mostly from the perspective of the users of a system, although we also proposed a number of software approaches and lessons learned in support of the designer and developer. This chapter focuses entirely on the design and development of interactive systems, as we present NiMMiT, a graphical notation for modeling multimodal interaction techniques. Since the notation was designed with 3D virtual environments in mind, we reflect on the possibilities of using NiMMiT for modeling touch-based and multi-user interaction. Throughout this chapter, we not only explain NiMMiT, but also a number of extension that others have made over the years. My personal contributions to NiMMiT were mainly situated at the beginning of my PhD, as I developed the original notation together with dr. Joan De Boeck.

Designing and developing an intuitive and easy to use application is almost never a straightforward undertaking, in spite of the extensive knowledge and tools that we have at our disposal nowadays. When designing user interfaces, we are presented with a large number of possibilities: choosing, combining and adapting existing interaction techniques (e.g. particular multi-touch or multi-user gestures), or developing new custom-made solutions. As the acceptance of an interaction technique typically depends on the actual application setup and the target users, the most appropriate method to evaluate a particular solution is by testing it in a user experiment. However, testing several alternatives implies that each one must be fully implemented. This situation often results in an iterative process in which a solution is designed, implemented and evaluated multiple times.

To shorten the development cycle and to enable quick prototyping of several alternatives, a model-based user interface design (MBUID) approach can be adopted [Da Silva 00]. A MBUID process facilitates the development of a user interface by defining it at different levels of abstraction. An overview of different model-based processes reveals several common properties [Clerckx 04, Meixner 11]: nearly all processes start with a task model abstraction and gradually evolve toward the final user interface through an incremental approach. During each increment, the model is converted into a new one by means of an automatic transformation (through certain mapping rules) or manual adaptation by the developer.

In the IWT SBO project *VR-DeMo* [Coninx 06b] that was carried out in our research lab from 2003 until 2008, the use of a MBUID process to develop interactive virtual environments [Cuppens 05] pointed out the need to

also specify interaction in a high-level manner, rather than having to manually code it. However, until recent work on multi-touch and gestural interfaces [Paternò 09, Spano 12], MBUID approaches lacked the means to easily describe rich interaction techniques (i.e. not just interacting with buttons and menus, but with the actual objects in the environment through direct manipulation or multimodal interaction).

In this chapter, we propose NiMMiT [Vanacken 06, De Boeck 07], together with the CoGenIVE tool [De Boeck 08, De Boeck 09]. They were developed in the context of the VR-DeMo project, in order to facilitate the design and prototyping of multimodal interaction techniques with a minimum of coding effort. NiMMiT allows a designer or developer to quickly test different alternatives or adjust existing interaction techniques according to the findings of an evaluation, thereby shortening the development cycle significantly. Moreover, the high-level description introduces a way to easily communicate about and reuse solutions. While the CoGenIVE tool facilitates the creation of NiMMiT diagrams, the VR-DeMo application framework supports automatic execution of the interaction techniques by interpreting the graphical representation.

In the next section, we summarize the overall model-based process of VR-DeMo and CoGenIVE, and how NiMMiT fits in this approach. In Section 7.3, we describe existing graphical notations for modeling interaction techniques. Based on some initial experiments, we put forward a number of requirements to describe interaction in Section 7.4. We also explain the basic primitives of our NiMMiT notation, and we show how NiMMiT diagrams are created and automatically executed in VR-DeMo. We illustrate the use of NiMMiT by means of an extensive example in Section 7.5. In Section 7.6, we summarize the approach of NiMMiT extensions to support evaluation, and integrate contextual and semantic information. Finally, in Section 7.7, we reflect on the use of NiMMiT for multimodal, touch-based and multi-user interaction, and we present our conclusions in Section 7.8.

7.2 VR-DeMo and CoGenIVE

It is beyond the scope of this chapter to fully explain the overall model-based framework for the development of virtual environments in which NiMMiT fits. However, since this framework is responsible for the interpretation and automatic execution of NiMMiT diagrams, we summarize some of its important aspects.

The goal of the VR-DeMo project is to facilitate the development of interactive virtual environments by defining all aspects of the application at a

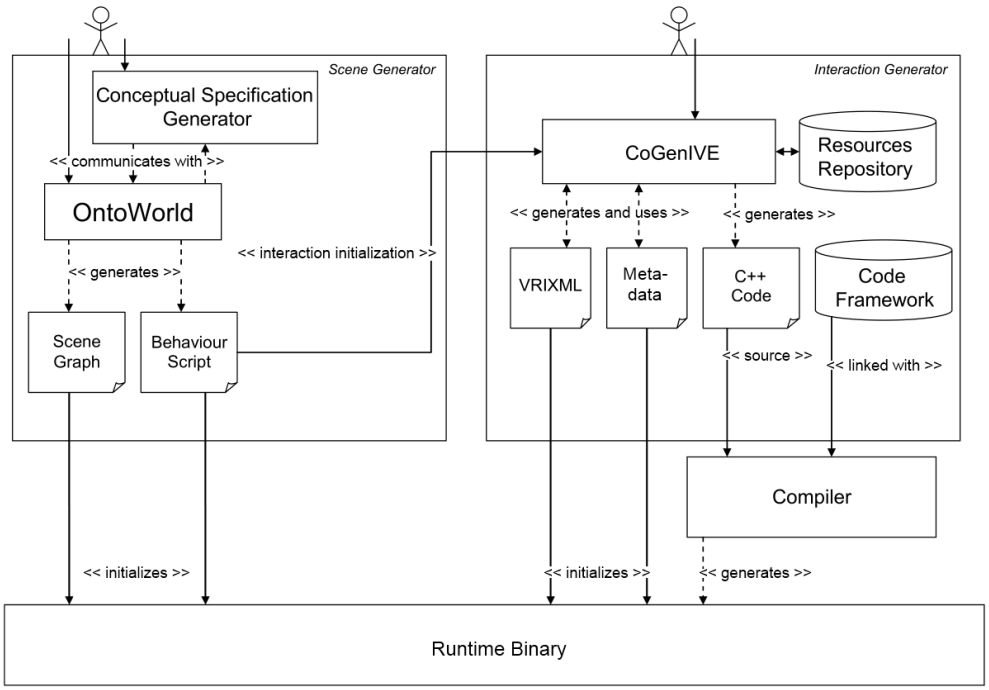


Figure 7.1: Overview of the overall VR-DeMo framework [Coninx 06b].

high level of abstraction. To this end, conceptual models are transformed into source code or interpreted by the application framework at runtime, as seen in Figure 7.1. The process is divided in two main parts: the scene generator relies on an ontology to design a virtual scene and the behavior of the scene's objects, and the interaction generator defines the user interaction. The scene generator is part of the research that was executed by our partners in the VR-DeMo project, the VUB-WISE research lab. However, this scene generator is not mandatory and will therefore not be discussed in detail, as our interaction generator can also use existing virtual scenes, or scenes that are generated in another way.

The process of the interaction generator, shown in Figure 7.2, is supported by the CoGenIVE (Code Generation for Interactive Virtual Environments) tool. This process usually starts from a task model, describing the possible tasks that can be performed by both the user and the system. We use the ConcurTaskTrees (CTT) notation [Paternò 00], which structures various types of tasks (i.e. user, abstract, interaction, and application tasks) in a hierarchical tree with temporal relationships between subtasks at the same abstraction

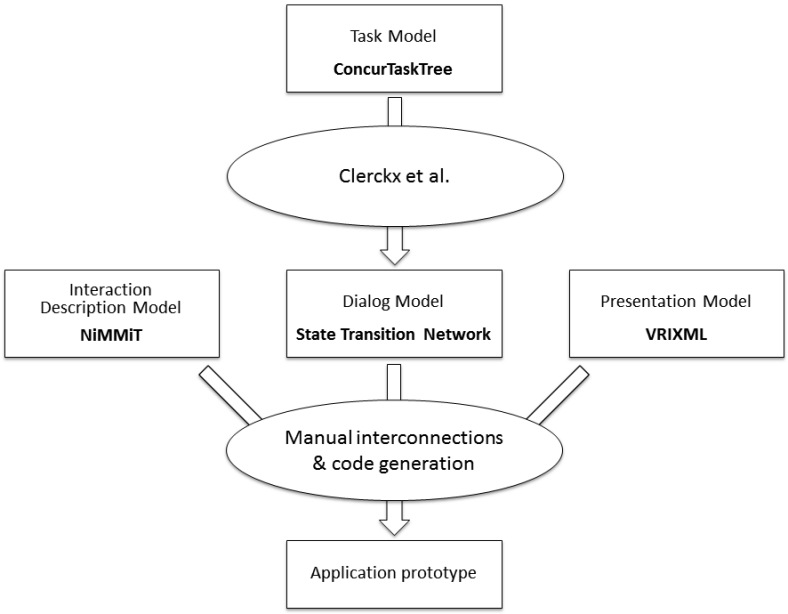


Figure 7.2: The model-based user interface design process used in VR-DeMo and CoGenIVE [De Boeck 06a].

level, indicating which tasks should be performed in sequence, which tasks can be performed in parallel, and so on. Figure 7.3 illustrates the use of the CTT notation.

From this task model, a dialog model is automatically derived using the approach of Clerckx et al. [Clerckx 04]. The dialog model is implemented as a state transition diagram, in which each state represents a set of tasks that can be performed at a given moment (e.g. after selecting an object, that object can be moved and rotated). In other words, the different states in the dialog model correspond to the enabled task sets of the task model [Paternò 00]. Some tasks will cause a transition from one state to another (e.g. selecting or deselecting an object), thereby enabling another set of tasks. Instead of automatically deriving the dialog model, it can also be created manually in the CoGenIVE tool, so the task model is optional.

The dialog model can be "annotated" with a presentation and interaction description model. The presentation model describes the user interface widgets, such as menus and dialogs, while the interaction description model specifies the user interaction that relies on direct manipulation or multimodal input. The task tree in Figure 7.3 contains several interaction tasks that typ-

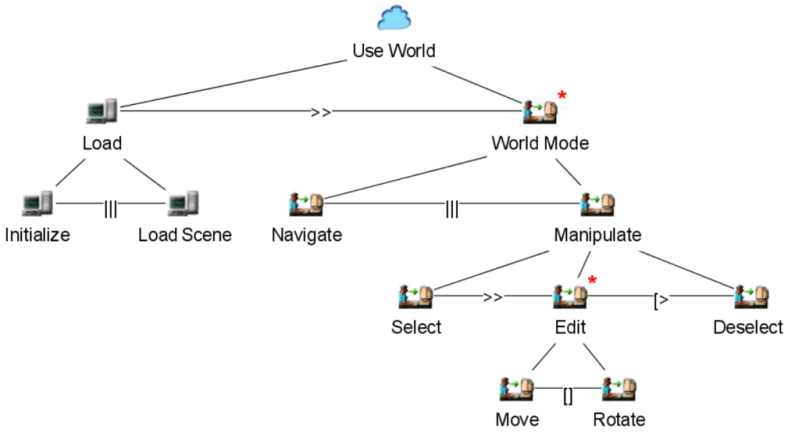


Figure 7.3: The task model structures various types of tasks in a hierarchical tree using the ConcurTaskTrees notation [Paternò 00].

ically fit into the interaction description model, such as navigate, select, move and rotate. However, such tasks may also end up in the presentation model (e.g. instead of using direct manipulation, an object can be rotated by setting angular values in a dialog widget), or may not need further modeling because they correspond to an atomic action of the framework (e.g. deselect an object).

As in most MBUID approaches, the presentation model describes the user interface widgets at an abstract level, which means that the model does not define the exact looks of the interface, but only what functionality is provided through the widgets. CoGenIVE saves the presentation model as a set of files in the VRIXML format [Cuppens 04], an XML-based user interface description language intended for 2D/3D hybrid widgets. In the final user interface, concrete widgets are instantiated, such as 2D menus and dialogs that are positioned in 3D space [Coninx 97, Raymaekers 01].

The interaction description model defines interaction tasks that rely on direct manipulation or multimodal input. One such example is selecting an object, which is a very common task that can be carried out in various ways, such as the traditional virtual hand selection, ray casting, or even speech [Bowman 04]. The technique to select an object could be decomposed into its atomic parts in the task model, but this approach is not very practical, as it results in unwieldy models. Furthermore, the framework has to be able to automatically execute the modeled interaction techniques. We therefore propose another notation, NiMMiT, which is discussed in the next section.

The dialog, interaction description and presentation model can all be cre-

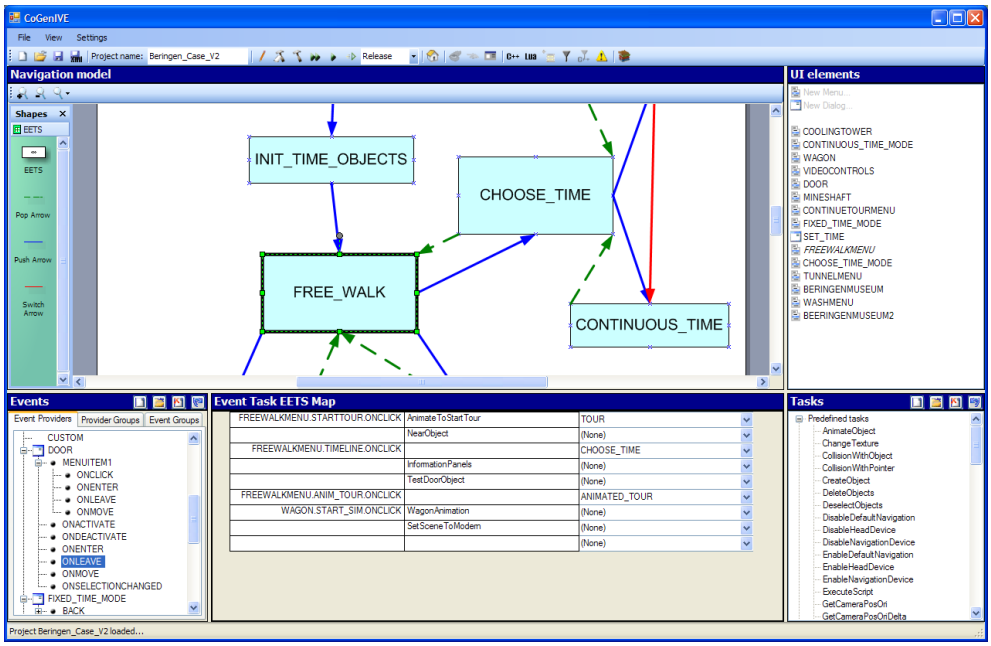


Figure 7.4: The CoGenIVE tool, being used to interconnect the various models of the user interface design process.

ated with the CoGenIVE tool. Once created, the different models have to be interconnected. This is a manual process, in which the designer or developer indicates which events (either originating from an input device, or from a user interface widget) correspond to a particular task. Such a task can be an atomic interaction task, or an interaction that is described in NiMMiT. Figure 7.4 shows the CoGenIVE tool being used to interconnect the models. The pane at the top contains the dialog model, where a specific state can be selected. The bottom pane in the middle shows how events are associated with tasks for that specific state. All the available events are shown in the bottom left pane, while all the task are listed in the bottom right pane.

Once the entire design process is completed, the specified models are used to automatically generate a runtime binary of the application. CoGenIVE also produces repository files that are interpreted by the application at runtime. These XML-based files typically contain descriptions of the user interface widgets and interaction techniques. To increase flexibility, the generated source code of the application is made available. This enables a programmer to modify the source code if necessary, for instance to add specialized features. To

support an iterative prototyping process, manual modifications to the source code need to be limited to designated regions. If the interaction description model and presentation model are modified, changes in those regions are preserved in a subsequent iteration.

7.3 Related work and early experiments

In this section, we explore existing notations for modeling interaction techniques and we present a few of our initial experiments on modeling interaction in 3D virtual environments. The graphical notations described in literature can roughly be divided into two families: data-driven approaches and state-driven approaches.

Data-driven approaches. Notations such as Labview[1], a graphical programming language, and UML [Ambler 04] focus specifically on the data or object flow. Their basic element is "activity", supporting data input and output, and the execution of such an activity is driven by the presence of valid input data.

InTml [Figueroa 02], ICon [Dragicevic 04], which is used in the MaggLite user interface toolkit [Huot 04], the OpenInterface Platform [Lawson 09] and Squidy [König 10] are examples of data-driven notations for user interaction modeling. Such diagrams consist of connected building blocks, sometimes called filters, which are composed of input ports, output ports and possibly some state information. An input port is used to receive information of a particular type and an output port sends the generated information to all filters that follow. In such a case, the control flow of the diagram is directed by the data, since a filter is executed when it receives a legal value on all of its input ports.

State-driven approaches. Notations such as statecharts [Harel 87] and Petri nets [Petri 62] are based on the formal mechanisms of finite state machines. A basic element of such a notation is a state transition, which has the general form "when event a occurs in state A, the system transfers to state B, if condition C is true at that time". We noticed many of the existing interaction techniques are of this nature. For instance, when the button is pressed while an object is selected, we start dragging the object.

[1] `http://www.ni.com/labview`

Some state-driven notations are extended to support data flow, or add actions to state transitions, which are executed each time the transition happens. Examples are Colored Petri nets [Jensen 94] and Object Petri nets [Valk 98]. In the domain of user interaction modeling, notations are for the most part derived from finite state machine solutions, but optimized for a specific purpose. IOG [Carr 97], ICO [Palanque 94, Navarre 09], HsmTk [Blanch 06], Chasm [Wingrave 09], Malai [Blouin 10], statechart modeling of interactive gesture-based applications [Deshayes 11] and SMUIML [Dumas 11] are mainly state-driven approaches, for instance.

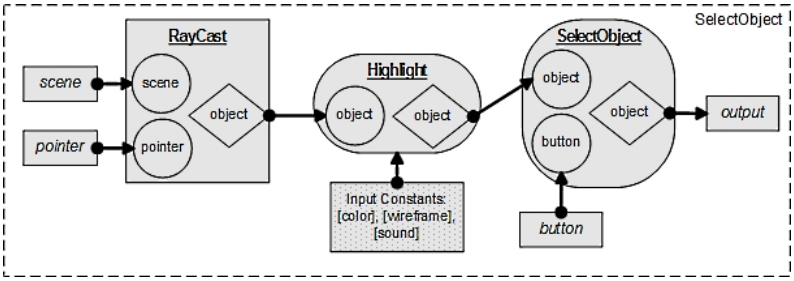


Figure 7.5: Early experiment with a rudimentary data-driven approach to model interaction in virtual environments. This example represents object selection.

Early data-driven experiments. In a preparatory study, we conducted several experiments, describing existing interaction metaphors using different notations. Figure 7.5 shows one of our early experiments with a straightforward data-driven approach, based on InTml and ICon. The big rectangles represent filters, the circles are input ports and the diamonds are output ports. The arrows model the data flow, while the small rectangles indicate input from or output to a device or the application.

The example in Figure 7.5 represents the selection of an object. The first filter handles the collision detection between the pointer and the objects. If the pointer collides with an object, the filter outputs this object. When the second filter receives an object as input, this object is highlighted. The third filter requires two inputs before it can be executed: an object and a button click. If both inputs are present, the highlighted object is selected.

We found data-driven approaches to be very intuitive and easy to comprehend. Unfortunately, a lot of interaction techniques are difficult to model due to the absence of states, and rather complex structures have to be designed in order to enable or disable certain parts of the diagram. Assume we want to

extend the example in Figure 7.5, so users can move a selected object. If the object is deselected, a user should no longer be able to manipulate it, which means we have to temporarily disable that part of the diagram. This kind of behavior is typically much easier to model with a state-driven solution.

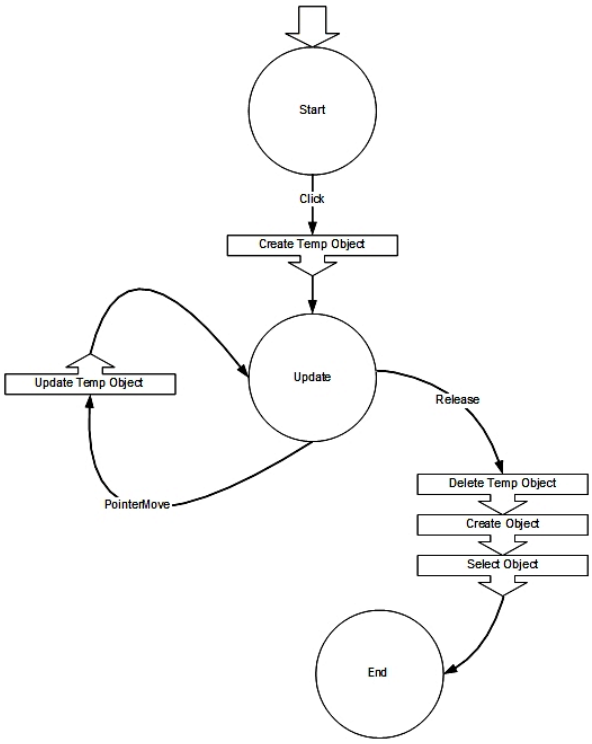


Figure 7.6: Early experiment with a basic state-driven approach to model interaction in virtual environments. This example represents the creation of a new object.

Early state-driven experiments. We also conducted a number of experiments with state-driven approaches, as illustrated in Figure 7.6. The approach in this example is based on Interaction Object Graphs and Interactive Cooperative Objects. The circles depict states and arrows represent state transitions, with events as labels. The rectangles are actions that will be executed whenever the corresponding state transition occurs. Each action may contain one or more conditions, which have to be fulfilled before the action can be executed.

The example in Figure 7.6 represents the creation of a new object. When the user presses the button of the input device, a temporary object is created.

	Pro	Contra
State-driven	• fits the intuitive feeling of different "phases" during an interaction • fairly easy to read by humans	• lacks the data flow necessary for automatic execution of a diagram • risk of state explosion
Data-driven	• easy to understand principle, and therefore easy to learn notation • limited number of primitives	• difficult to enable or disable different parts of a diagram • can be more difficult to edit

Table 7.1: Partial summary of the pros and contras of state-driven and data-driven notations. A more detailed comparison is given by De Boeck et al. [De Boeck 06b].

As long as the button is pressed, the object can be moved around. When the user releases the button, the temporary object is replaced with a definitive object. Before finishing the interaction, this definitive object is selected.

We noticed that, while describing a technique using a state-driven notation, the lack of data handling can be restricting at times, especially when automatic execution of the diagram is required. In the example in Figure 7.6, for instance, we create a new object that is selected immediately after its creation. However, it is impossible to explicitly indicate that we want to select the newly created object, and not some other object. Moreover, diagrams such as Petri nets quickly become very complex, even when describing a reasonably simple interaction.

Overview of our findings. A partial overview of the pros and contras of state-driven and data-driven notations is summarized in Table 7.1. For a more detailed comparison, we refer to the work of De Boeck et al. [De Boeck 06b], in which an in-depth evaluation of the different notations is performed, using cognitive dimensions [Green 89].

Based upon these findings, we concluded that a data-driven approach will benefit from the inclusion of states, and vice versa. Therefore, we developed NiMMiT, a notation for modeling multimodal interaction techniques that is both state-driven and data-driven. The combination of both concepts allows us to maintain data flow, while inheriting the formalism of state transition diagrams, which is necessary for interpretation and execution of diagrams. As a testament to its strengths, a number of recent solutions also combine state and data flow concepts, such as StateStream [de Haan 09], VITAL [Csisinko 10] and the heterogeneous modeling environment ModHel'X [Deshayes 12]. The

main strength of NiMMiT lies in the easy-to-learn and easy-to-read diagrams, which provide an unambiguous description of the interaction and can be interpreted by an application at runtime.

7.4 NiMMiT

In this section, we first formulate a set of requirements, based on the shortcomings and strengths we identified in the previous section. Next, we describe how NiMMiT meets these requirements by briefly explaining the syntax and semantics of our notation. We also show how NiMMiT diagrams can be created and executed in the VR-DeMo framework.

7.4.1 Requirements for describing user interaction

In our opinion, based on the findings of our aforementioned experiments, a notation to describe interaction techniques should meet the following requirements: it should be *event-driven*, *state-driven*, *data-driven*, and it should support *encapsulation for hierarchical reuse*. In the next subsections, we motivate the importance of these requirements in the context of interaction techniques.

Event-driven

Interaction techniques are inherently driven by user-initiated actions, which we define as events. Since human interaction is multimodal by nature, it can be interpreted as a combination of unimodal events (e.g. pointer movement, click, speech command). An event has the following properties:

- a source, indicating the modality and/or abstract device that caused it (pointing device, speech, etc.),
- an identification, defining the event itself (button click, a particular speech command, etc.),
- parameters, giving additional information about the event (e.g. the position of the pointer).

Events can be seen as "the initiators" of different parts of the interaction.

State-driven

While interacting with it, the system not always has to respond to all available events. Most of the time, specific events must have occurred before other events are enabled. For instance, the user first needs to click the pointing device's button before being able to drag an object. Therefore, we perceive an interaction technique as a finite state machine, in which each state defines to which set of events the system will respond. The occurrence of an event also initiates a state transition. In our example, the dragging technique consists of two states. In a first state, the interaction technique awaits a click, before moving to the second state. The second state responds to pointer movements.

Data-driven

Limiting our vision on interaction techniques to a finite state machine would be too restrictive. After analysing several interaction techniques in 3D virtual environments, it became clear that throughout the execution of an interaction some indispensable data flow occurs. A common sequence of actions is first selecting an object and afterwards moving that object around. Obviously, certain data must be transferred between the different tasks of the interaction technique. Therefore, a notation to describe interaction techniques should support data flow.

Encapsulation for hierarchical reuse

Some subtasks of interaction techniques recur rather frequently. Selecting objects is an example of a very common component. When modeling a new interaction technique, the designer or developer should be able to reuse descriptions that were created earlier. That way, recurring components do not have to be modelled repeatedly. In other words, the notation should support encapsulation of entire diagrams. In this way, existing diagrams can be reused as a subtask of a new description. Using such building blocks contributes significantly to more efficient development.

7.4.2 NiMMiT's basic primitives

In this section, we briefly describe the basic syntax and semantics of NiMMiT. We aimed for a high-level graphical notation that not only meets the abovementioned requirements, but also allows:

- designers and developers to *communicate* about the functionality of an interaction technique, using an easy-to-learn and easy-to-read notation,
- an application framework to interpret a diagram, so it can be *executed* and *evaluated* without having to write additional programming code.

The notation must provide adequate low-level information for a software framework to execute the diagrams automatically, but it also needs to be sufficiently high-level and easily readable for people to reason about the interaction technique.

States and events

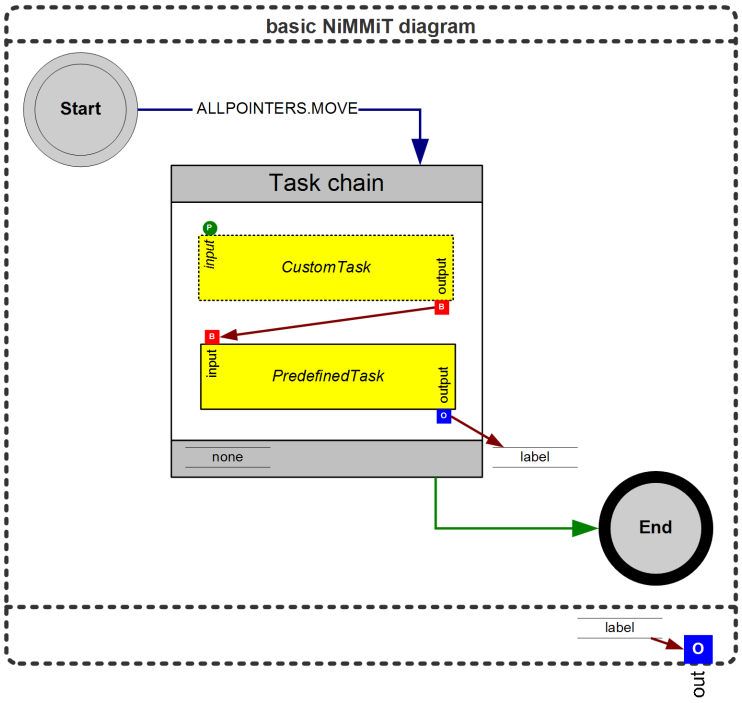


Figure 7.7: A NiMMiT diagram showing the basic building blocks.

The NiMMiT notation is state-driven and event-driven, so a diagram can basically be considered a statechart. When activated, an interaction technique goes to its initial state, the "start" state, and simply waits for events. Each state responds to a limited set of events, such as a speech command, a pointer movement, a button click, and so on. The recognition of an event causes a

chain of tasks to be executed, as shown in Figure 7.7. The blue arrows indicate the firing of a task chain, their labels are the triggering events.

Since the number of possible device setups is immense, the VR-DeMo framework uses VRPN [Taylor 01] to make an abstraction of concrete devices. NiMMiT provides a similar device abstraction: devices are grouped in "categories" according to the events they generate (e.g. pointing, navigation, speech, gestures). Devices within a category generate interchangeable events. As a result of this abstraction, switching between devices in the same category does not affect the interaction, and hence the NiMMiT diagram does not require any changes. However, when a pointing device is replaced by speech input, at least a small change to the diagram (changing the event arrow) is necessary.

Events are always noted in the form of "provider.event". The provider represents a device or modality, and can be a single component (e.g. "pointer1") or a group (e.g. "allpointers"). A group is expanded to its components by an "or" relation (e.g. "allpointers" becomes "pointer1 or pointer2"). Different NiMMiT events can also be combined with each other (e.g. two events in an "and" relation), which is explained in detail in Section 7.7.1.

Task chain and tasks

A task chain is a strictly linear succession of tasks. Figure 7.7 shows a task chain (big white rectangle with grey border) containing two tasks (smaller yellow rectangles). The next task in the chain is executed if and only if the previous task has been completed successfully. The execution of a task often results in an adjustment of the application's internal state. For instance, an object is selected or deselected, its position or orientation is altered, and so forth.

The main actions of an interaction technique are obviously situated in the tasks. The set of tasks that are essential for modeling a variety of interaction techniques largely depends on the application domain, and for each particular domain, the most common tasks have to be predefined. The VR-DeMo framework focuses on interaction in 3D environments, and therefore predefines tasks such as selecting, moving and deleting objects, doing rudimentary collision detection, and so on. Since it is impossible to predefine every conceivable task, we provide the possibility to define application specific tasks, called custom tasks, by means of the Lua scripting language [Lerusalimschy 96]. Another option is to model a task as a separate NiMMiT diagram, and assemble diagrams hierarchically, as we will discuss shortly.

Data flow, data types and labels

In a task chain, tasks can produce output and pass it from one task to another. Therefore, each task provides input and output ports. The port's data type is reflected by a color and a letter within the shape, and only ports of the same type can be linked to each other. As one of our primary concerns is to keep NiMMiT as accessible as possible, we prefer a limited set of data types, as shown in Figure 7.8: booleans, numbers, strings, objects, and a few domain specific types, such as 3D positions and rotations in case of 3D virtual environments.

Figure 7.8: The data type of input and output ports is reflected by a color and a letter within the shape.

An output port of a preceding task is typically connected to an input port of a next task. These input ports are either required (depicted as a square) or optional (a round shape). To share data between tasks in different task chains, or to store data for later reuse, we provide high-level variables in the form of labels. The content of a label remains available as long as the NiMMiT diagram is operational, and its scope is the entire diagram. The tasks in Figure 7.7 each have one output port, and the output port of the first task is connected to the required input port of the second task by a red arrow. The first task also has an optional input port. The output of the second task is stored in a label (two small parallel lines with text in between) for later reuse.

State transitions and pass-through states

After a task chain has been successfully executed, a state transition takes place. The green arrow in Figure 7.7 represents a state transition. The interaction technique moves either to a new state or back to the current state (in a loop). In a new state, the interaction technique may respond to another set of events. An interaction technique is terminated when it reaches the "end" state.

A task chain can also be associated with more than one state transition, in which case the value of the chain's conditional label is taken into account. Figure 7.9 shows a task chain with a conditional label "var" at the bottom, and two possible state transitions. If the value of the label is zero, for instance, the transition on the left is executed. This construction allows the designer

or developer to define state transitions that are dependent on the result of a task, since the output of a task can be linked to the label of the task chain.

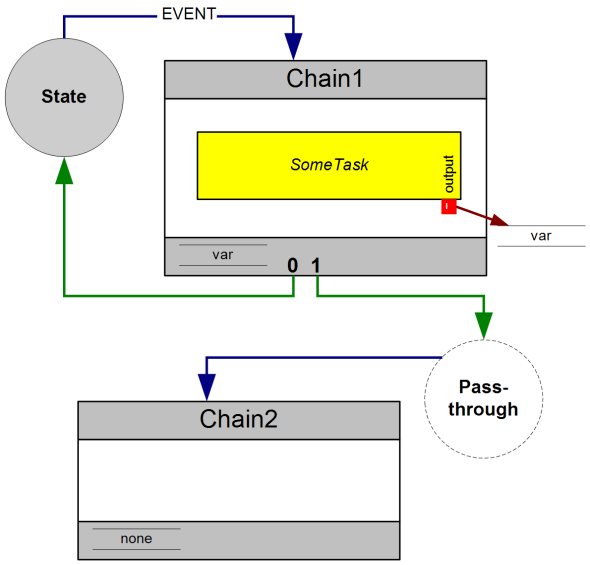


Figure 7.9: A pass-through state that splits a task chain in two separate branches by using conditional state transitions.

Sometimes, it is useful to create a task chain that immediately activates another task chain after its execution, especially in the case of conditional state transitions: in one condition, a simple state transition takes place, but in another condition, we first need to execute some additional tasks. Pass-through states are states that are left immediately after they are reached. It enables splitting a task chain in two separate branches, as illustrated in Figure 7.9. After the execution of the first task chain, we either return to the initial state, or we go to the pass-through state. In that case, the second task chain is immediately executed.

Preconditions

In NiMMiT, preconditions can be associated with tasks, task chains, or entire diagrams. Preconditions are functions, written in Lua scripting or C++, that return true when the conditions are met, and false otherwise. When a precondition is not met, the corresponding task, task chain or interaction technique is not executed. Preconditions are a way of countering the state explosion problem: if the execution of a task chain depends on a number of conditions,

often a lot of states need to be added to model all those conditions. By using preconditions, these extra states can be avoided.

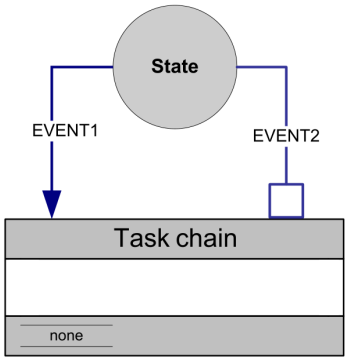


Figure 7.10: A task chain with preconditions, indicated by the square box on the event arrow.

Preconditions of tasks and diagrams are currently not visualized in a NiMMiT diagram. Preconditions of task chains, however, are visualized to some extent, as shown in Figure 7.10. A precondition of a task chain is connected to an incoming event arrow. In our example, the event arrow on the left has no precondition, while the one on the right does (indicated by the square box on the arrow). If "event2" occurs and the function that evaluates the precondition returns true, the task chain is executed. Otherwise, the event is ignored. Because preconditions may become quite complex, the condition itself is not visualized in the diagram. It can only be seen in the CoGenIVE tool.

Error handling

If an error occurs during the execution of a task chain (e.g. a required input value is missing, or a precondition is not met), the chain has to be aborted. In an early version of NiMMiT, this involved rolling back the values of the labels and returning to the previous state in the diagram. Using this approach, the application may end up in an inconsistent state. Assume, for instance, that a task chain selects an object and stores it in a label. If an error ensues in a subsequent task of that chain, the label's initial value is restored and the chain is aborted. As a result, the system is in an inconsistent state: the object is still selected in the underlying framework, but the NiMMiT label no longer reflects this.

To solve this problem, NiMMiT supports "non-transactional" and "trans-

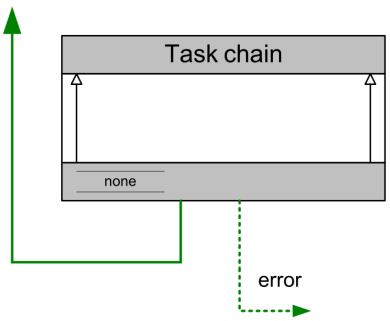


Figure 7.11: Error handling in NiMMiT by means of a transactional task chain and error arrow.

actional" task chains, which can be combined with "error arrows", as seen in Figure 7.11. If an error takes place in a transactional task chain, the labels and the tasks performed so far are rolled back one after the other. This means that each task must implement a procedure to undo its actions, if applicable. A non-transactional task chain, on the other hand, simply aborts if an error occurs, without rolling back the labels or tasks. In both cases, the system returns to the previous state in the diagram, unless an error arrow is present. Such an arrow takes the system to another state when an error happens. In case the error takes place in a task chain that was activated by a pass-through state, the task chain preceding that state is also aborted or rolled back.

Encapsulation for hierarchical use

Interaction techniques have an interface similar to the tasks of a task chain, i.e. a diagram can have input and output ports. The diagram in Figure 7.7 has one output port, for instance. As a result, not only predefined and custom tasks can be used to build a task chain, but it is also possible to include an entire diagram as a task, resulting in a hierarchical structure. When such a hierarchical task is activated, the execution of the current interaction technique is temporarily suspended and saved on a stack, waiting for the nested interaction technique to finish. In the next section, we show an example that makes use of this feature.

In addition to reusing a single task, entire task chains can also be "collapsed" in NiMMiT. From time to time, a task chain becomes quite large, containing more lower-level information than is desired in a high-level view of an interaction technique. As one of our goals is to create diagrams that are easy to understand and communicate about, the details of such a task chain

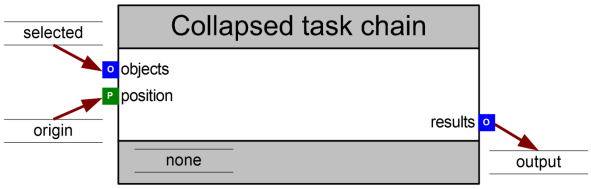


Figure 7.12: A collapsed task chain only shows the labels that are used as input and output in the chain.

can be hidden. A collapsed task chain is given a descriptive name, and only shows the labels that are used as input and output in the chain, as shown in Figure 7.12. The composing tasks are omitted, and the details of the task chain are modeled in a separate sub-diagram.

7.4.3 Creation and execution of a NiMMiT diagram

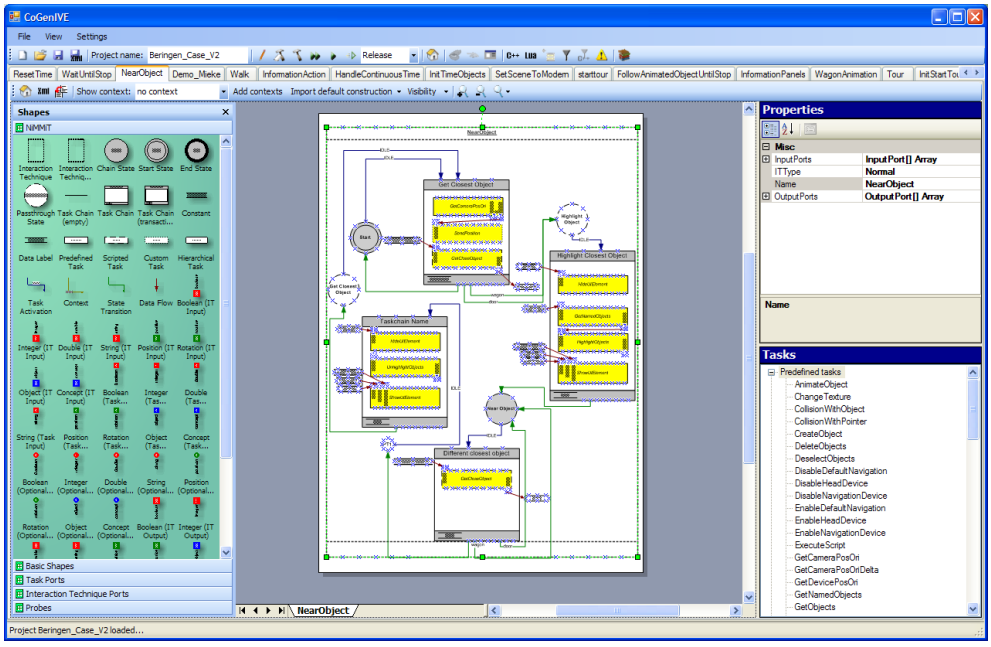


Figure 7.13: The CoGenIVE tool, being used to create a NiMMiT diagram.

Since we want to use the notation to prototype and evaluate interaction techniques, easy creation and automatic execution of diagrams are essential

requirements. CoGenIVE includes an editor to create NiMMiT diagrams, as seen in Figure 7.13. The basic building blocks can simply be dragged onto the diagram from the pane on the left. The properties of an element (e.g. the name of a state, the events associated with a task chain activation) can be changed in the top right pane, and all the available tasks can be selected from a list in the bottom right pane. As NiMMiT diagrams can be hierarchically reused, they also appear in the list of available tasks.

The NiMMiT editor ensures that created diagrams are syntactically correct. The editor checks, for instance, that each state transition arrow leads to a new state, and that data input is of the correct type when a label or output port is connected to an input port of a task. To allow a framework to execute an interaction technique, the graphical notation must be transformed into a format that can be parsed efficiently. Therefore, the CoGenIVE tool saves a NiMMiT diagram in an XML-based file format. The file can be loaded by the interpretation engine at runtime, as illustrated in Figure 7.14. An internal "manager" maintains the state, listens to events, executes task chains and keeps track of labels.

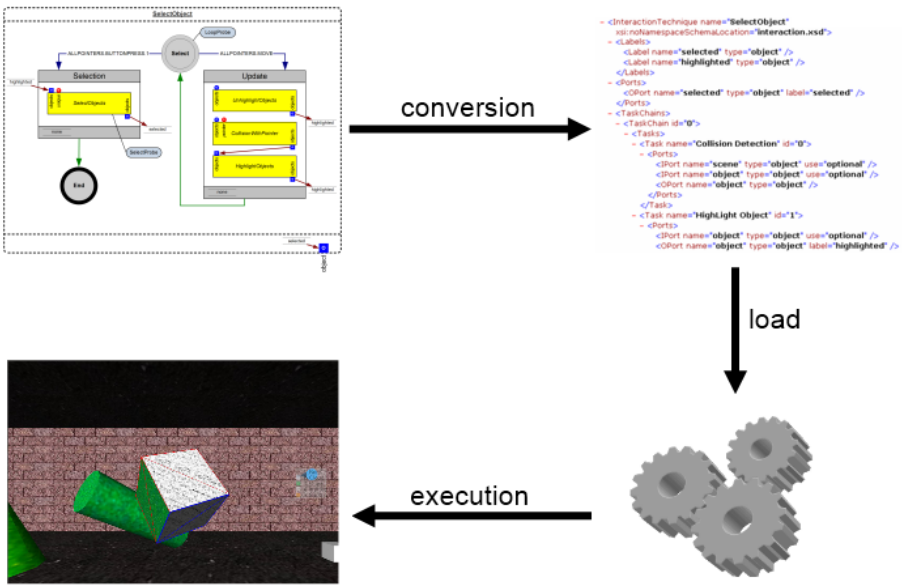
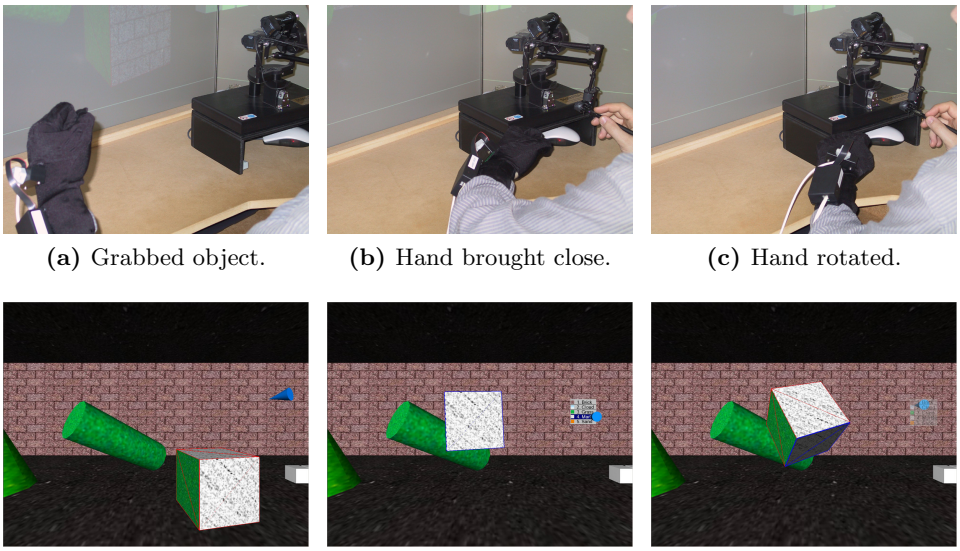


Figure 7.14: The execution process of a NiMMiT diagram. The diagram is converted to an XML file, which can be loaded and executed at runtime.

7.5 Case study: the Object-in-Hand metaphor

This section illustrates the use of NiMMiT by means of a practical example. We model a two-handed interaction technique that is called the Object-in-Hand metaphor [De Boeck 04]. After an object has been selected, the user can grab it by bringing the non-dominant hand's fist near the pointing device in the dominant hand. This causes the selected object to move toward the center of the screen, where it can be rotated by the non-dominant hand and manipulated by the dominant hand (e.g. change the texture of one of the cube's faces), as depicted in Figure 7.15. Since the interaction metaphor requires the user to utilize both hands, this example also illustrates a synchronization mechanism between different NiMMiT diagrams. To keep the diagrams readable, we do not include full error handling.



(a) Grabbed object. **(b)** Hand brought close. **(c)** Hand rotated.

(d) Selected object. **(e)** Object-in-Hand. **(f)** Rotated object.

Figure 7.15: The Object-in-Hand metaphor [De Boeck 04] allows the user to utilize both hands to manipulate an object.

7.5.1 Selecting an object

As a first step, the user is required to select an object. We chose a virtual hand metaphor: highlight the object by "touching" it with a virtual pointer

and confirm the selection by clicking a button. This interaction component can easily be expressed in the NiMMiT notation, as depicted in Figure 7.16.

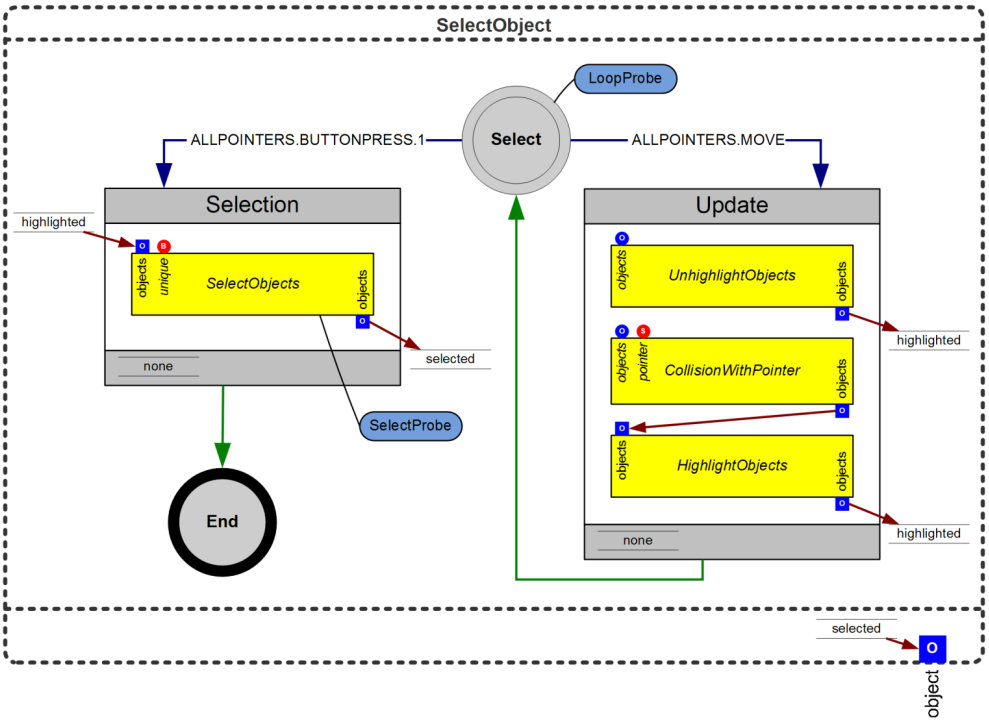


Figure 7.16: A NiMMiT diagram that models the interaction task of selecting an object.

The interaction technique starts in the state "Select", which responds to pointer movements and clicks. Each time the pointer moves (and the button is not clicked), the task chain on the right is executed. This chain contains three predefined tasks:

- **Unhighlight currently highlighted objects.** The task can be given a list of objects as an optional input. If not provided, the task simply takes into account all objects in the environment. The output always returns an empty list, which is used to clear the label "highlighted".

- **Check for collisions with the pointer.** The task has two optional input ports: the objects and the pointers that should be considered by the collision detection. If not provided, collisions are calculated between

all pointers and all objects in the environment. The colliding objects, if any, are obtainable through the output port.

- **Highlight the colliding objects.** The task is only executed when all required input ports receive an appropriate value. If no collisions were detected, this prerequisite is not satisfied and the chain is aborted. Consequently, the system returns to the state "Select" and awaits new events. If the task does receive a proper value, the colliding objects are highlighted and stored in the label. Finally, a task transition returns the system to the state "Select".

While the system resides in the state "Select", click events cause the task chain on the left to be executed. It contains only one task:

- **Select the highlighted objects.** The task receives the highlighted objects through the label "highlighted", and stores the results in a new label, "selected", which is used as output of the entire diagram. In case the label "highlighted" is empty (i.e. no object was highlighted), the chain is aborted and the system returns to the state "Select". If the task chain finishes successfully, a final state transition occurs and the system moves to the state "End". At that moment, the interaction technique is completed.

In the next section, we demonstrate how this entire diagram can be reused as a single, hierarchical task. The probes included in the diagram are explained in Section 7.6.1.

7.5.2 Non-dominant hand interaction

After an object has been selected using the aforementioned selection technique, it can be "grabbed" with the non-dominant hand. By bringing a closed non-dominant hand near the dominant hand, the object moves to the center of the screen, where it can be rotated by the non-dominant hand and manipulated by the dominant hand (e.g. change the object's texture).

Running the diagram in Figure 7.17 triggers the state "Start". As soon as an "idle" event occurs, which happens continuously while no other events are generated, the first task chain executes. The only task in this chain is our previously defined selection technique. While this hierarchical task is active, the execution of the current interaction technique is suspended. When the selection task ends, the current interaction technique resumes, and the yielded

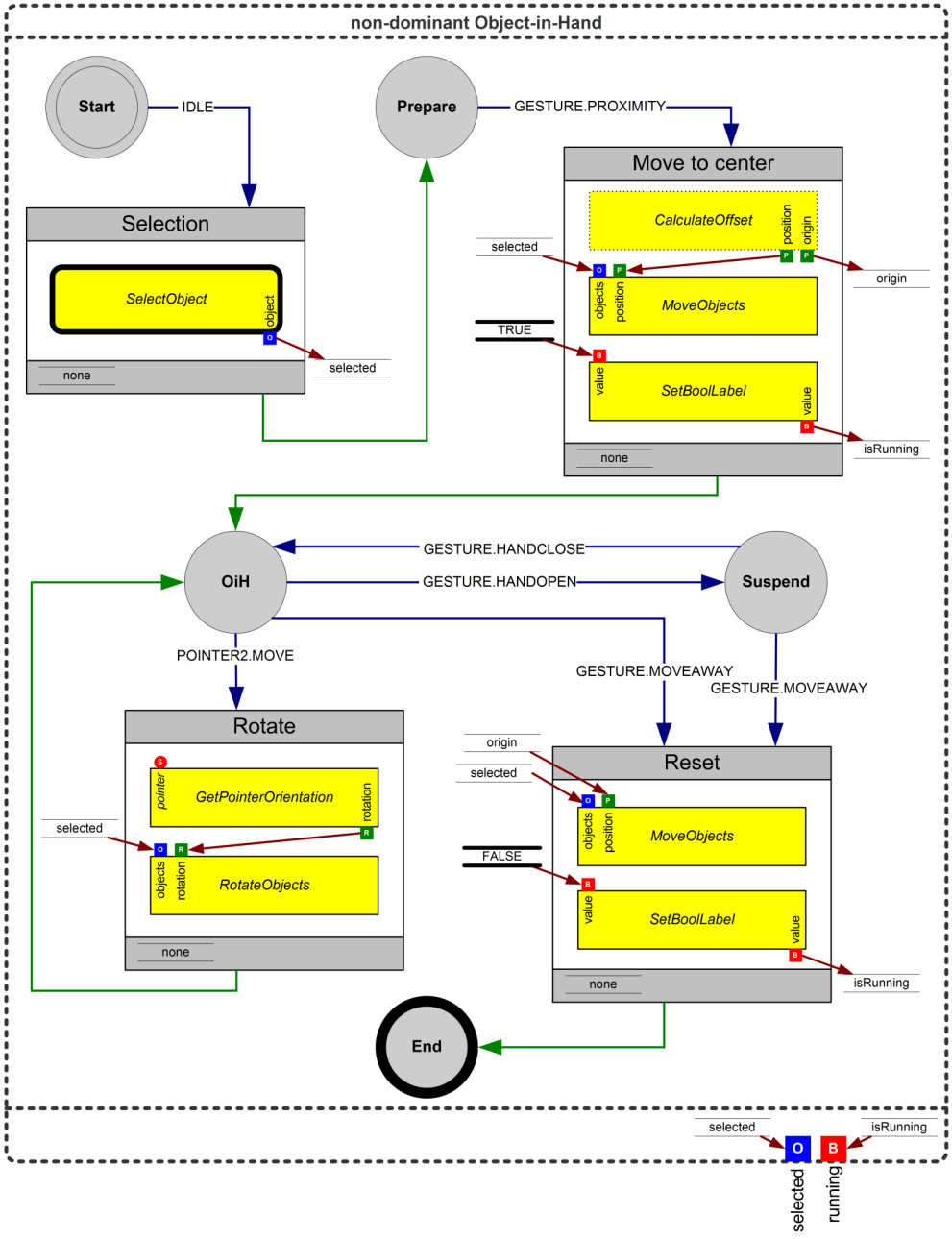


Figure 7.17: A NiMMiT diagram that models the interaction of the non-dominant hand in the Object-in-Hand metaphor.

result is stored in the label "selected". Next, a state transition to "Prepare" takes place.

"Prepare" awaits an event that is defined as "a closed non-dominant hand brought in the proximity of the dominant hand" and triggers the second task chain. The first task, a custom task, calculates the offset between the virtual position of the non-dominant hand and the center of the screen. The task requires the selected object as an input and returns the object's initial position and new position. The next task moves the object to its new position. The final task simply sets a label to "true", which is needed for synchronization, as we explain shortly.

Eventually, the system ends up in the state "OiH" (abbreviation of "Object-in-Hand"), awaiting an appropriate event. If the closed non-dominant hand is opened, a transition (without task chain) to "Suspend" occurs. When the hand closes again, the "OiH" state is reactivated. These state transitions im-

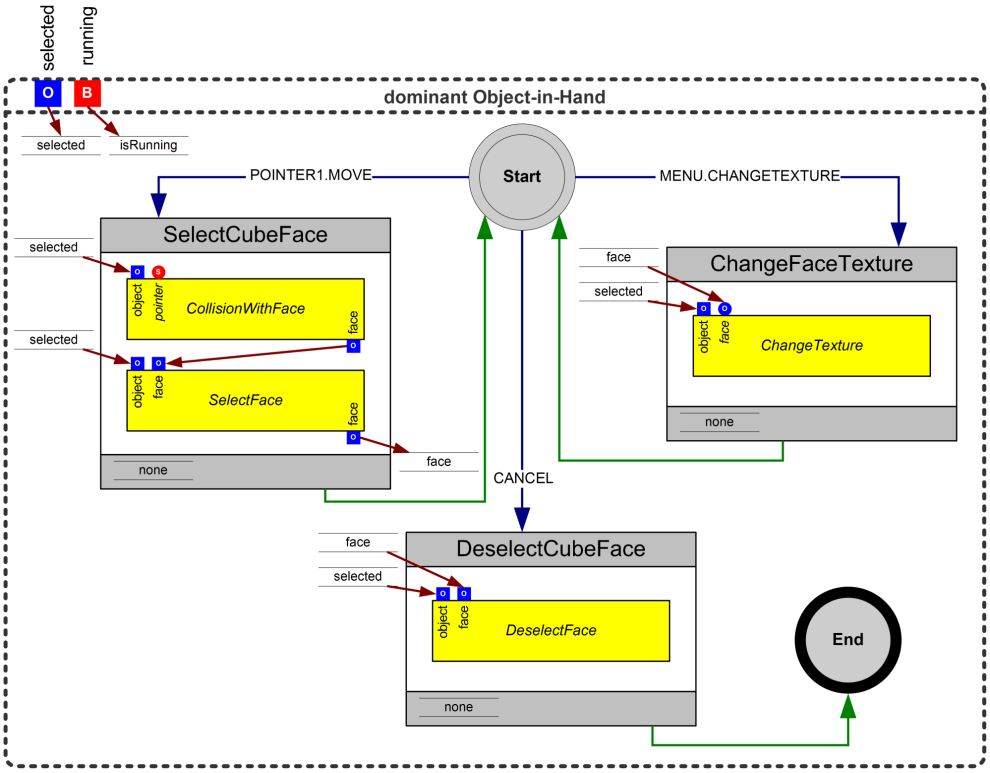


Figure 7.18: A NiMMiT diagram that models the interaction of the dominant hand in the Object-in-Hand metaphor.

plement clutching and declutching, allowing the rotation of an object beyond the physical constraints of the user's wrist. "OiH" also responds to movements of the non-dominant hand. In the corresponding task chain, these movements are mapped to the object's orientation. Finally, both states respond to the withdrawal of the non-dominant hand. The associated task chain restores the objects initial position and resets the label "isRunning".

7.5.3 Synchronization with the dominant hand

The interaction of the dominant hand, which is described in a new NiMMiT diagram, depends on the state of the non-dominant hand: when the non-dominant hand is not in the proximity of the dominant hand, the dominant hand's interaction is disabled. To this end, the label "isRunning" is updated by the "non-dominant Object-in-Hand" diagram and connected to an input port of the new diagram, as shown in Figure 7.18. Based on the value of this label, the execution of both diagrams is synchronized, as the "dominant Object-in-Hand" diagram takes this value into account in a precondition (not visualized in the diagram). The diagram of the dominant hand's interaction is not discussed in detail, as it offers no new insights into NiMMiT.

7.6 Extensions to NiMMiT

In this section, we summarize the most significant extensions that have been added to NiMMiT over the years. This includes support for evaluation, and the integration of contextual and semantic knowledge.

7.6.1 Adding support for evaluation

In the previous sections, we have shown how an interaction technique can be executed by feeding a NiMMiT diagram to an interpretation engine. This approach can save time that would otherwise have gone to writing programming code. To exploit this advantage even further, NiMMiT is extended to support automated data gathering and processing. It is well known that user tests often require ad-hoc adaptations to the application code, in order to log data for statistical analysis. Consequently, adding some level of automation to the evaluation process has many potential benefits, such as time efficiency and cost reduction [Ivory 01, Palanque 11]. In this section, we explain how NiMMiT is extended with three primitives: probes, filters and listeners [Coninx 06a, De Boeck 07].

Probes

A probe is a measurement tool, connected to a particular primitive in a NiMMiT diagram, much like an electrician placing a voltmeter on an electric circuit. Probes can be associated with either a state, a task chain, a task, or an input/output port. A probe returns relevant data regarding the primitive it is attached to:

- state probes return all events occurring while the state is active;
- task chain probes return the activation event(s) of the chain, its status (executed, interrupted, failed), and the value of the conditional label;
- task probes indicate whether or not the execution of the task succeeded;
- port probes return the current value of the input/output port.

If a probe is connected to a primitive that is inactive during the current phase of the interaction, it simply returns an empty value. The example given in Figure 7.16 shows probes connected to the state "Select" and to the task "SelectObjects".

NiMMiT's probes are a useful tool for debugging an interaction technique. By attaching a probe to all states of a diagram, one can for instance evaluate the correct order of execution or monitor the events that are being triggered. In addition, output of tasks can be verified using port probes. This approach leads to a significant reduction of the time necessary to discover logical errors in a diagram. The raw output of a probe is, however, not sufficient to collect information during an evaluation.

Filters and listeners

To "refine" the data coming from a probe, we define filters. Filters are meta-probes collecting and processing the output of one or more probes, and can be chained one after another. Filters can rearrange or summarize data, or wait until data arrives for the first time, and then start, stop or pause a timer. The latter technique is useful for measuring the time spent between two states of the interaction. Although the data necessary for an evaluation can be very divergent, often the same patterns return: counting the elapsed time, logging success or failure, measuring a distance, and so forth. Therefore, NiMMiT provides a standard set of commonly used filters, including (conditional) counting and distance or time measuring.

Filters and probes only collect and structure information. By connecting listeners to a probe or filter, output can be redirected to any output medium. The default listeners can write data to a file or console window, or even send it over a network to an external computer or a database. The data can then be used for the statistical analysis of the user experiment. If the need arises, our approach can be extended so that custom filters and listeners can be implemented using the Lua scripting language.

Example

The NiMMiT diagram in Figure 7.16 contains two probes: the state probe "LoopProbe", providing a list of recognized events, and the task probe "SelectProbe", returning an empty value while the right-hand task chain is being executed. When the user clicks the button, this second probe indicates whether or not the task "SelectObjects" has been successfully completed. The outputs can be redirected using a listener, providing a useful debugging tool.

In order to collect data for an evaluation, the standard "Timer" filter can be applied. Connected to both probes, this filter starts a timer as soon as "LoopProbe" generates its first output, and stops counting when "SelectProbe" outputs its first value. The filter calculates the time elapsed between the first and second trigger. Using an appropriate listener, the outputs can be redirected for statistical processing. Coninx et al. [Coninx 06a] provide a more extensive example, as they describe a user experiment conducted using probes and filters.

7.6.2 Integrating contextual and semantic knowledge

The incorporation of contextual information can be valuable when an application is used in a variety of contexts. For instance, an interaction task may be performed using a particular technique when the user is sitting, while the same task must be accomplished using another technique when the user is standing. Therefore, Lode Vanacken et al. [Vanacken 07b, Vanacken 08c, Vanacken 08d, Octavia 09] extended NiMMiT to support contextual interaction.

The process of context detection and context switching can be seen as an "event-condition-action" process: a certain event can signal a change in context, and if certain conditions are met, an action executes the context switch. For instance, while interacting with an application, the user may suddenly stand up (event). Before executing the context switch, the system must ensure that the tracking system is active (condition). If the condition

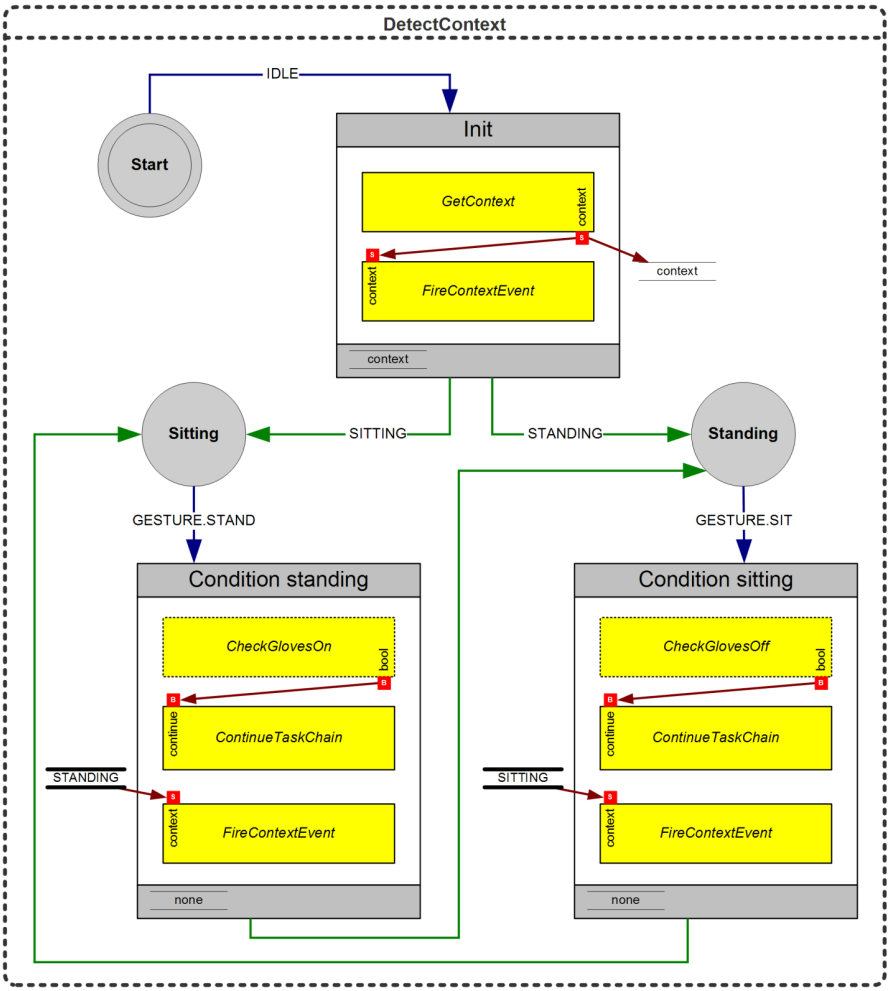


Figure 7.19: Modeling contextual interaction as an "event-condition-action" process in NiMMiT [Vanacken 08c].

is met, we disable the keyboard and mouse interaction, and we enable a new technique that relies on tracking the user's hands (action).

To accomplish this in NiMMiT, the framework is extended with events that represent changes in context. Figure 7.19 shows a NiMMiT diagram that handles the detection of a context switch. The events activate task chains that check the conditions and switch the context if these conditions are met. This NiMMiT diagram can be complemented with a second diagram to handle all the actions that need to be executed before a context switch occurs, such as enabling or disabling particular devices or user interface elements.

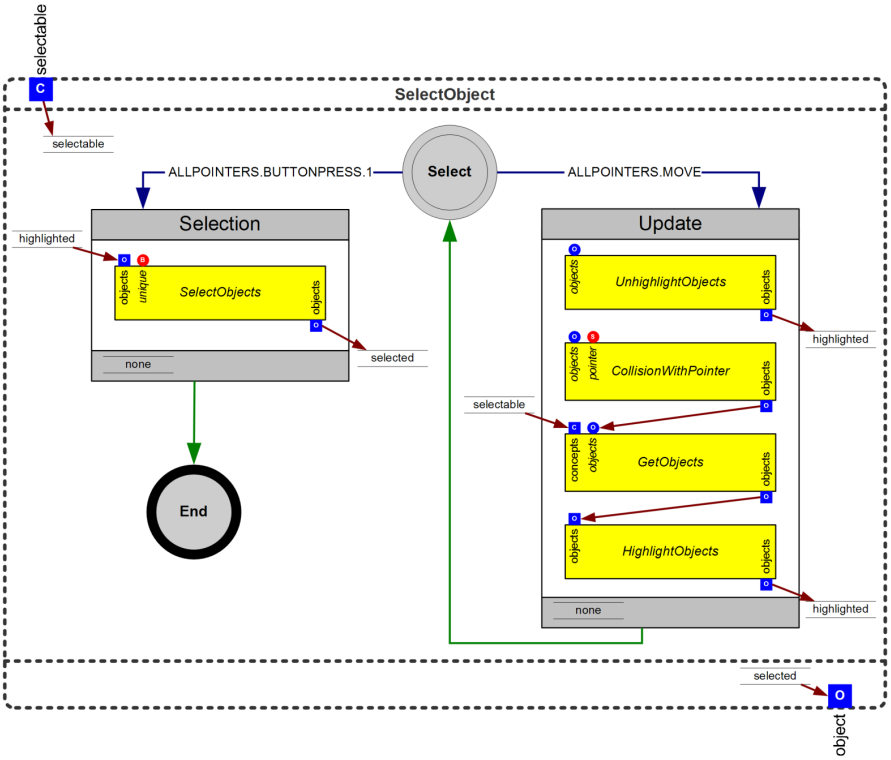


Figure 7.20: The use of concepts in a selection technique to only highlight "selectable" objects [Vanacken 09b].

A second NiMMiT extension of Lode Vanacken et al. [Vanacken 09b] involves the use of semantic knowledge. A recurring issue with modeling interaction is that part of the interaction depends on the application being developed. When modeling a selection technique, for example, often some objects should be selectable, while others should not. This usually results in the creation of

custom tasks, which are not reusable and increase the development time. As described in Section 7.2, the VR-DeMo framework also includes a scene generator. NiMMiT diagrams could make use of the semantic information from the scene generator to avoid the need for (some of) those custom tasks.

To take the semantic information into account, a "concept" data type was introduced to NiMMiT. Such a NiMMiT concept maps directly to a concept of the ontology that is used in the scene generator. In addition, two new tasks were defined: "GetObjects" and "IsOfConcept". Figure 7.20 shows an example of a selection technique that makes use of concepts and the task "GetObjects" to only allow highlighting of "selectable" objects. In contrast with an approach that uses a custom task to filter objects, this diagram is application independent and can therefore be reused as long as the semantic information is available.

7.7 Considerations on multimodal, touch-based, and multi-user interaction

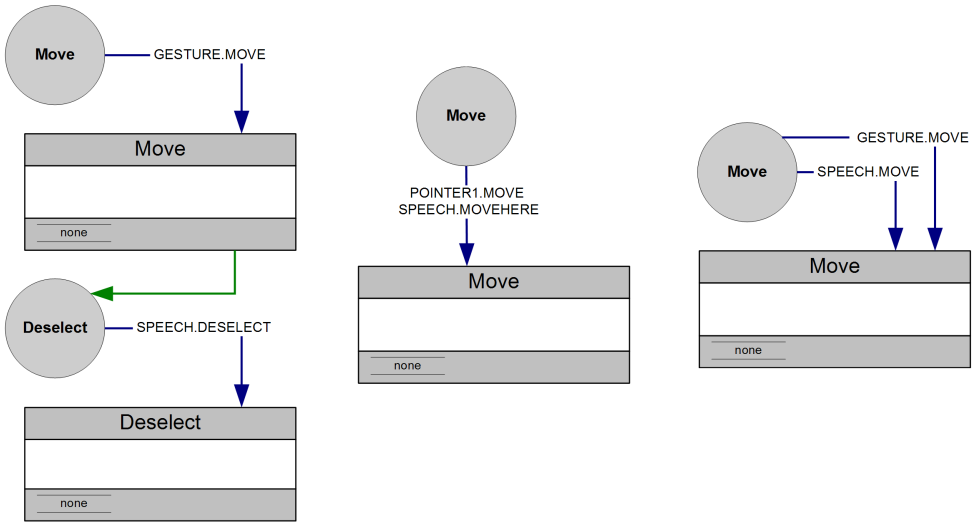
NiMMiT was initially developed with multimodal interaction in 3D virtual environments in mind, but we are currently exploring other types of interaction. In this section, we briefly discuss the use of NiMMiT to model multimodal interaction, and next, we investigate some of the limitations and opportunities with regard to modeling touch-based and multi-user interaction.

7.7.1 Modeling multimodality

Multimodal interfaces offer the user various "modalities" of input and output. A multimodal interface can combine, for example, the traditional modalities (e.g. input through keyboard and mouse, output through a display) with a voice modality (e.g. input through speech recognition, output through speech synthesis). This offers several advantages, as the weaknesses of one modality can be compensated by the strengths of another [Lemmelä 08], and a multimodal application can be more accessible to users with an impairment [Kane 11].

Multimodal input

Since we support several "categories" of devices, as described in the first part of Section 7.4.2, multimodal input is accomplished by combining events from different categories. Based on the notion that multimodal interaction is caused by



(a) Sequential. **(b)** Simultaneous. **(c)** Equivalence.

Figure 7.21: Multimodal support within NiMMiT.

several unimodal events, NiMMiT supports sequential and simultaneous multimodal interaction, as well as equivalence between the modalities [Nigay 93].

Sequential multimodality can be modeled by defining subsequent states that respond to events coming from different modalities. In Figure 7.21a, an object is first moved using a gesture and, in the next state, deselected using a speech command. Simultaneous multimodality is supported by combining events on one arrow. Figure 7.21b shows, for instance, how a user moves an object by combining a pointer movement and a speech command. Finally, equivalent modalities are supported by using two parallel arrows. Figure 7.21c illustrates a movement action, achieved by either a speech command or a gesture.

As events triggered by users never occur at exactly the same time, we use Nigay's melting pot principle [Nigay 95] as a fusion mechanism. It lets us combine input specified through different modalities, based on time and structural complementarity. Note that the various relations between events, combined with the melting pot principle, allow designers to implement all possible relationships expressed in the CARE properties [Coutaz 95], but it is beyond the scope of this chapter to discuss those properties in detail.

Multimodal interaction may also require the parallel execution of different

aspects of an interaction technique. One NiMMiT diagram on itself does not support concurrency. However, multiple diagrams may be executed at the same time. In that case, synchronization issues are solved by exchanging labels between the diagrams, as shown in Section 7.5.

Multimodal output

Although NiMMiT strongly supports multimodal input, the notation is somewhat lacking with regard to multimodal output. NiMMiT does not contain standard primitives for multimodal output, for example. Furthermore, a lot of output is generated automatically by the underlying framework, to maintain simplicity. Most visual feedback is, for instance, provided "by default", such as the virtual scene rendering. Unfortunately, multimodal output is not a straightforward feature to add, as such output can be very diverse (e.g. various forms of visual, tactile, and audio feedback).

Looking at the earlier example of a selection technique, presented in Figure 7.16, the diagram includes a predefined task to highlight objects. The goal of this task is to provide feedback on collisions between the virtual pointer and the objects in the environment. However, the task offers no control over the output modalities, and always shows the same predefined form of visual feedback. Likewise, the diagram includes a "SelectObjects" task that not only handles the actual selection, but also the visual feedback. As a result, designers and developers currently have no control over the feedback in a NiMMiT diagram, and by looking at a diagram, it is unclear which tasks offer output, especially when different modalities are being used.

An easy solution is to offer a number of predefined tasks for visual effects (e.g. highlighting objects), audio effects (e.g. confirmation sounds), tactile effects (e.g. vibrations), and so on. This is similar to the approach of HIT-PROTO [Panëels 10], which uses "action blocks" to provide haptic effects. As a fallback option, custom tasks can be written to achieve effects that are not available through the predefined tasks. However, to accomplish NiMMiT's goals, it is important to strike a balance between having more control over the output, and having easy-to-read diagrams that are not overburdened with tasks to handle all the required output. Splitting up all the diversified tasks such as "SelectObjects" in an execution and a feedback task might complicate diagrams too much. Therefore, further research is needed on how to deal with output, for example through the use of an additional "presentation model" on top of the current interaction description model.

7.7.2 Modeling touch-based and multi-user interaction

As stated in Section 7.4.2, we developed NiMMiT with two important requirements in mind: designers and developers should be able to communicate about the functionality of an interaction technique, using an easy-to-learn and easy-to-read notation, and an application framework should be able to interpret a diagram, so it can be executed and evaluated without having to write additional programming code. Since these two properties are very useful when researching and developing touch-based and multi-user interfaces, we investigate if NiMMiT can be used outside the boundaries of virtual environments.

NiMMiT uses an event system to handle input from the user and can easily support basic touch-based interaction (e.g. tapping, drag-and-drop). The VR-DeMo framework and CoGenIVE tool basically need to be extended with the typical touch events: "touch down", "touch move", and "touch up", which represent respectively a new touch (finger placed on the surface), a change in position of an existing touch (finger moved), and the end of an existing touch (finger lifted). However, as we discuss throughout the next sections, there are some issues when trying to model more complex interaction techniques, such as multi-user techniques or gestures.

Events and multi-touch setups

As discussed in Section 7.4.2, events in NiMMiT diagrams are always of the form "provider.event". The provider represents a device or modality, and is either a single component (e.g. "pointer1") or a group (e.g. "allpointers"). As a result, in case of multi-user interaction, actions of users can be distinguished if they each use a different device or modality. If two users interact with an application by using two mice, for instance, the NiMMiT diagram includes events with providers "pointer1" and "pointer2". The same holds true for other examples, such as two-handed techniques in which each hand uses a different device or modality, as in the Object-in-Hand example in Figure 7.15.

When considering touch-based interaction, often one or more users interact with both hands on the same device (e.g. a multi-touch tabletop). In that case, the event provider is just "touch", and it is difficult to distinguish between touches from different fingers, hands or users in a NiMMiT diagram. When modeling multi-user interaction, however, it is important to know the origin of an event, since we need to ensure, for instance, that events originating from one user do not inadvertently trigger state transitions in an interaction that is being performed by another user (unless, of course, the interaction technique requires two users to cooperate). In other words, it should be possible to

indicate in a NiMMiT diagram that all events have to come from the same user, or that some events have to come from a different user.

For single-user applications, this might seem less of an issue, because most interaction techniques simply require a number of touches (e.g. a pinch-to-zoom typically requires the use of two fingers), but the source of those touches is irrelevant (e.g. a user can pinch with any two fingers from the same hand, or one finger from each hand). Nonetheless, we need to be able to indicate in a NiMMiT diagram which events have to come from the same finger, and which have to come from another finger. Consider, for example, the three-finger rotation technique of the 3D puzzle game described in Chapter 3. The first two fingers that are put on a cube determine the axis of rotation, while the third finger spins the cube around that axis. When the first two fingers are moved, the axis changes accordingly. To model this behavior in a NiMMiT diagram, we need to be able to determine whether an event comes from one of those two fingers or not.

First, we should look at how this is handled when coding a touch-based application, instead of modeling it in NiMMiT. On a "touch down" event, each touch is normally assigned a unique identifier by the tracking software of the interactive surface. Subsequent "touch move" events from the same finger are assigned the same identifier, until a "touch up" event occurs. By considering this identifier, a developer knows which subsequent events originate from the same finger. In addition, if the hardware setup is capable of identifying the different users and hands, for example by means of the Carpus technique described in Chapter 6, a developer has access to even more information about each event. This kind of information also needs to be made available in NiMMiT.

A straightforward solution is to write custom tasks or provide more predefined tasks that offer access the data associated with the event(s) that trigger a task chain, such as "GetTouchID" or 'GetUserID" tasks. We use a similar tactic in Figure 7.17, as we retrieve the information of a pointer with the task "GetPointerOrientation" (the optional input port can be used to indicate from which pointer the data should be retrieved; if not provided, it takes the pointer associated with the event that activated the task chain). Although this method brings a lot of flexibility with it, the readability of diagrams will also decrease, because events are no longer limited to event arrows, but also play an important role inside task chains. Furthermore, the overall complexity of diagrams will increase, as additional states and task chains are needed to process the extra information of the events.

Another possibility is to extend the description of events, by adding more

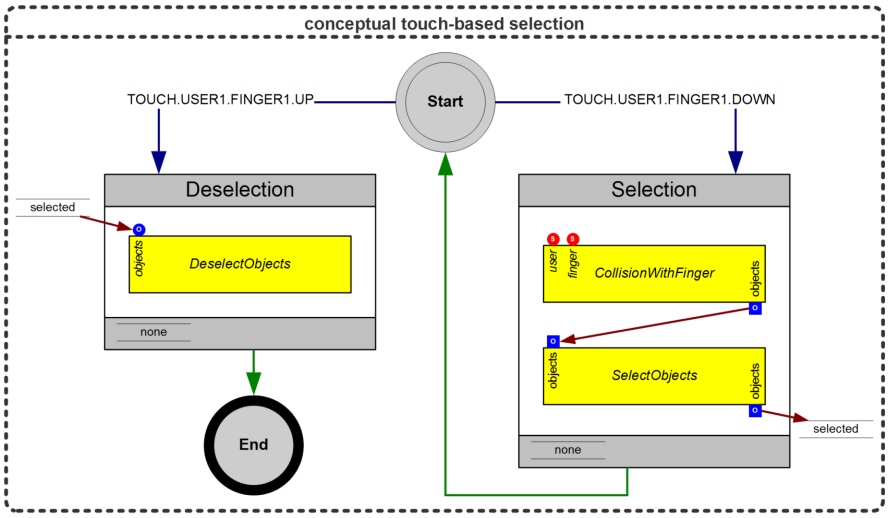


Figure 7.22: A conceptual NiMMiT diagram that illustrates the possible use of extended events to indicate that all events need to originate from the same user and finger.

information to the event provider. A fully extended event could look something like "touch.userid.fingerid.event". If we want to model a multi-user technique, we can listen to the events "touch.user1.move" and "touch.user2.move", for instance. The additional identifiers in these events do not represent specific individuals, but indicate that both "touch move" events should originate from two different users. Such an identifier is also useful to accomplish the exact opposite: we can reuse the same identifier at different places in a diagram to indicate that all those events need to come from the same user or finger, as illustrated in Figure 7.22. This method is somewhat similar to the identifiers suggested in the model-based approach for gesture interfaces by Spano [Spano 11], in order to assign the same touch to different building blocks.

Gestures as events or models

The example presented in Section 7.5 includes a number of gestures, such as closing or opening a hand and moving two hands in close proximity of each other. In this case, gestures are represented as atomic events in the NiMMiT diagram (e.g. "gesture.handclose"), and the details of those high-level gestures (e.g. the recognition, the visual feedback) are hidden in the VR-DeMo

framework. This very straightforward approach has a few advantages. First of all, external software can easily be used to do the gesture recognition, such as the "$1 recognizer" for user interface prototypes [Wobbrock 07]. Furthermore, adding such gestures to a NiMMiT diagram is very simple, and the atomic events provide an appropriate high-level view on the gestural interaction. Of course, this approach also has a major shortcoming: designers and developers have very little control over the gestures.

Representing a gesture as a single event provides a very narrow view on the gesture. Although atomic events are sufficient for traditional point-and-click interactions with a mouse, most gestures take a lot longer to execute than a simple click. The "closing of the hand" gesture, for instance, only fires an event when the hand is completely closed. As a result, a NiMMiT diagram has no information on the gesture at the start of or during the closing of the hand. This information can, however, be very useful, for example if we want the application to provide some kind of intermediate feedback during the performance of the gesture. An obvious example is the typical pinch-to-zoom interaction: users expect the zooming to take place while they move their fingers closer together or further apart, and not only at the moment that they end the gesture.

To this end, the single event that is fired after the completion of a gesture needs to be replaced by multiple events that represent the different stages of the gesture. Typically, a "start-update-end" arrangement is used, so the "closing of the hand" gesture would involve three separate events: "close start", "close delta", and "close end". These events are very similar to the basic touch events we discussed earlier: "touch down", "touch move", and "touch up". Using the "close delta" event, the application can provide continuous feedback on the gesture, as illustrated in Figure 7.23. Of course, depending on the circumstances, it might not always be necessary to make use of all three events. As an added benefit, this approach also facilitates the evaluation of gestures (e.g. speed of performance, error rate), since we can connect probes to the different stages of a gesture, as shown in Figure 7.23 and explained in Section 7.6.1.

Another solution is to model a gesture completely in a NiMMiT diagram, with the use of the basic touch events and some additional predefined tasks, such as tasks to calculate the distance or angle between points. This way, designers and developers have full control over the gesture, and they can create new gestures without new events having to be defined in the VR-DeMo framework. To make gesture models reusable, a diagram should only define how the gesture is executed, without already connecting an actual effect to

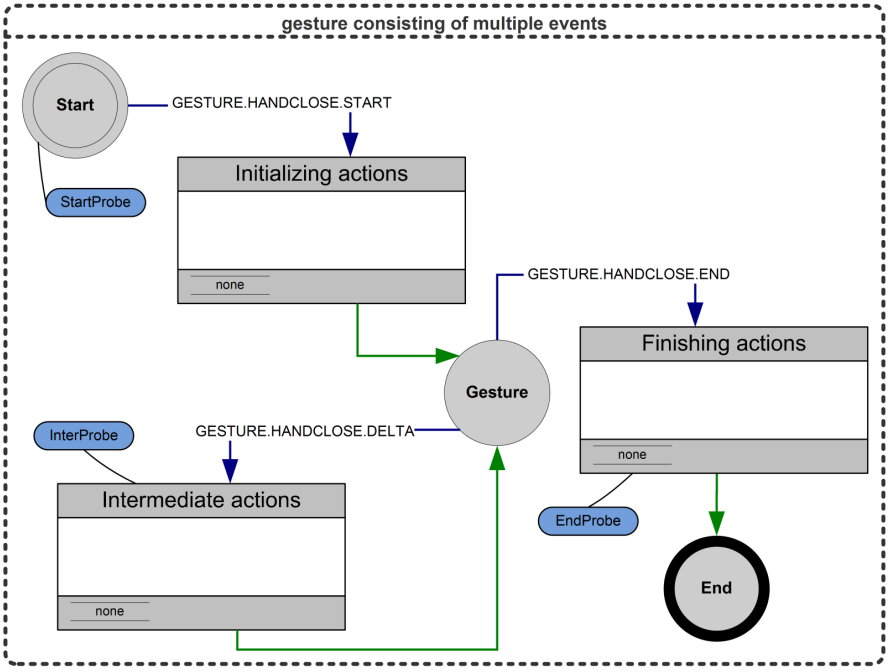


Figure 7.23: A conceptual NiMMiT diagram that illustrates the possible use of multiple events to represent the different stages of a gesture.

the gesture. Modeling that effect can be done in a separate diagram. This approach offers the most flexibility, but compared to the use of events to represent gestures, more low-level information is introduced to NiMMiT diagrams, which makes the notation less suitable for designers with a limited knowledge of programming.

Besides modeling gestures by means of basic touch events and predefined tasks, part of the gesture recognition can be done in a custom task. The Lua scripting that is currently used to define custom tasks could be complemented with a language that specifically targets gestures. GDL [Khandkar 10] and Midas [Scholliers 11] are, for instance, domain-specific languages designed to streamline the process of defining gestures, while GeForMT [Kammer 10] is a gesture formalization for multi-touch that is rooted in semiotics. Proton [Kin 12b] and its successor, Proton++ [Kin 12a], are declarative frameworks to describe multi-touch gestures as regular expressions of touch event symbols. Although an accompanying user study shows that tablature is an effective graphical representation to cope with the complexity of regular ex-

pressions to some degree, the trade-off between expressiveness and complexity of NiMMiT diagrams remains an important concern.

As a side note, we briefly want to discuss the performance aspect of the runtime interpretation of NiMMiT models, as explained in Section 7.4.3. In touch-based interaction, quite some attention goes to the performance of both the hardware, as well as the software. Since users interact with the virtual objects in a very direct manner, by using their fingers, any delay in the system's response is perceived to be a discomfort. The performance of NiMMiT's interpretation engine has never been of any concern in our virtual environments, but it might turn into an issue when all the available gestures are modeled as separate diagrams. This aspect must be kept in mind while exploring NiMMiT for touch-based interaction.

Execution of diagrams for multiple users

As stated in Section 7.4.3, the interpretation engine maintains an interaction technique's state, listens to events, executes task chains and keeps track of labels. Consider the case of a straightforward touch-based selection technique that selects an object when a user touches it, and deselects the object when the user lifts her finger. Suppose the technique is awaiting a "touch down" event. If two users are interacting simultaneously with the application, the user who touches the surface first will cause a state transition, and the other user's touch will be ignored. In most cases, this is not the result we want to achieve, for instance when two users want to select two different objects. In other words, we want two selection techniques to be executed in parallel, one for each user.

In Section 7.2, we explained how the task and dialog model determine which set of tasks can be performed at a certain moment in time. However, if a single selection task is added to the task tree, only one instance of the selection technique is allowed to run when this interaction task is enabled. To let two users simultaneously select objects, two parallel selection tasks need to be added to the task tree, so that the task ends up twice in the same enabled task set. This approach solves the problem in our example: when the first user touches the surface, the associated event is consumed by the first selection technique that is enabled, and the event of the other user will end up in the second enabled selection technique.

As a next step, we need to ensure that each of the two selection techniques process the correct events: the first technique should only react to events coming from the first user, and ignore the events of the second user, and vice

versa. Suppose, for instance, that each user selected an object. If one user lifts her finger, then her object should be deselected, and not the object of the other user. To accomplish this behavior, we can make use of the identifiers we proposed earlier (e.g. "touch.user1.down"). The interpretation engine has to compare the identifier of each new event to the identifiers of preceding events, to make sure that this event originates from the required source.

Although this approach works as expected, the overhead of having to duplicate all the interaction tasks at the task level for each user is bothersome. To find a solution, we reconsider the various steps in the model-based process. Since parallelism is possible at the task level, multiple diagrams can be executed at the same time. However, at the interaction description level, an interaction technique always has one active state, and does not support concurrency within one diagram. Since it must be possible for each user's selection technique to be in a different state at one moment in time (e.g. one user has already selected on object, while the other user has not), we cannot model the parallel selection techniques by only relying on our current notation (i.e. without making use of the parallelism at the task level).

One option to support this behavior at the interaction description level is extending NiMMiT. Instead of always having one active state during the execution of an interaction technique, we can adopt a token-based method: each active user is represented by a token, which starts in the first state and "travels" through the diagram based on the events that are triggered by that user. The labels of a diagram would be bound to a user, so each user can have different values stored in those labels. By applying this approach, each user can be in a different state, thereby solving the abovementioned limitation. However, the concept of tokens also entails new ways of synchronizing the actions of multiple users, and would increase the complexity of the notation. A more in-depth investigation is needed to ensure that such a method does not undermine our initial goals of having an easy-to-learn and easy-to-read notation.

Looking beyond the interaction description model, the ConcurTaskTrees of the task model could be replaced with cooperative ConcurTaskTrees [Mori 02]. These consist of multiple task trees: one task tree for each user role and one task tree that acts as a coordinator, specifying the collaboration and global interaction. This approach has some important limitations, however, as each user has to take on one particular role and multiple users cannot simultaneously fulfill the same role. Further research is ongoing to investigate to what extent a cooperative task model and an interaction description model based on (an extended version of) NiMMiT are suitable solutions.

Synchronization of interactions

Another factor to take into account is the synchronization of the actions of multiple users, for instance during collaborative interactions. The cooperative puzzle game described in Chapter 3 contains a clear example of such an interaction: in order to enlarge a small puzzle piece, two users are each required to select a specific type of puzzle piece, move those pieces close to each other, and simultaneously press two confirmation buttons. When pressing the buttons, the interactions of both users need to be synchronized, and we need to check if all the requirements are met.

At the task level, a task tree can handle such synchronization by means of the temporal relationships between the tasks. In the interaction description model, on the other hand, synchronization is achieved by using labels and exchanging the outputs of diagrams, as demonstrated in Section 7.5.3. The benefit of synchronization at the task level is that the task tree offers a structured overview of all the interactions. At the interaction description level, however, synchronization is not limited to temporal relationships, as NiMMiT diagrams can exchange any data through the labels and input or output ports.

One concern with using labels and data ports is that it may become error-prone when modeling complex multi-user interaction that involves a lot of synchronization, as the designer or developer needs to keep track of numerous synchronization values. An alternate approach that provides a clearer overview, is creating NiMMiT diagrams that are dedicated to synchronization, much like the aforementioned coordinator of the cooperative ConcurTaskTrees. However, further research is needed to assess the impact of such synchronization diagrams on the overall accessibility of the notation.

Concluding overview

Based on the Object-in-Hand example and our initial exploration of NiMMiT in the context of touch-based and multi-user interaction, we conclude that the notation in its current state can be used to model gestural interfaces to a limited extent. A gesture can either be represented by a single event, multiple events in a "start-update-end" arrangement, or a NiMMiT diagram that models the gesture. To this end, the underlying VR-DeMo framework needs to be extended with the required events and an adapted set of predefined tasks. By using NiMMiT, designers and developers can easily communicate about the touch-based technique, and the diagram can be executed and evaluated without having to write additional programming code.

To model more advanced interaction, our approach has to be enhanced in

a number of ways. First of all, NiMMiT needs a means of indicating that particular events have to come from the same source (e.g. when the same user has to perform several consecutive actions), or that events have to come from a different source (e.g. when two users have to collaborate). Additionally, the underlying framework should be able to concurrently execute an interaction technique for multiple users. When we model a selection technique, for instance, multiple users need to be able to select objects in parallel. This challenge can be approached from the task model, for instance by adding two parallel selection tasks to the task tree, or from the interaction description model, by adopting a token-based method that allows each user to be in a different state of a NiMMiT diagram.

The major downside of adding extensions to NiMMiT is that they often make it more complex, which goes against our desire of having an easy-to-learn and easy-to-read notation. Therefore, further research is required to investigate the aforementioned extensions on the subject of expressiveness and complexity. In this investigation, we also need to evaluate other modeling languages and tools that can be used to complement or enhance NiMMiT at the interaction description level. Throughout this chapter, we already discussed some interesting approaches that have been introduced since the VR-DeMo project ended in 2008, such as the gesture formalizations for multi-touch.

Improving the tool support is also an important factor to consider, as it can have a significant influence on the accessibility of a notation. A user evaluation of the Open Interface Development Environment [McGee-Lennon 09], a rapid prototyping platform for multimodal interaction, revealed that a suitable tool can enhance communication and exploration of alternate possibilities during the design phase. Furthermore, a tool can help its users to cope with complexity. In Squidy [König 10], for instance, users are able to adjust the complexity of the user interface through the concept of semantic zooming, which enables access to more advanced functionality on demand.

7.8 Conclusion

This chapter presents NiMMiT, a graphical notation that facilitates the design of multimodal interaction techniques with a minimum of coding effort. The notation allows a designer or developer to quickly test alternative solutions or easily adjust existing solutions according to the findings of an evaluation, shortening the development cycle significantly.

NiMMiT, based on both state-driven and data-driven primitives, provides device abstraction through the use of events and supports an hierarchical

build-up. The unambiguous modeling of an interaction technique allows diagrams to be interpreted and executed by our NiMMiT framework, which currently focuses on interactive 3D environments. We illustrated the expressiveness of NiMMiT by modeling the two-handed Object-in-Hand metaphor, but the interpretation engine has also been extensively tested on other well-known interaction techniques. Furthermore, we extended the notation to support automated data gathering and processing, which provides useful information for debugging and evaluation.

In the context of this dissertation, we explored the limitations of NiMMiT with regard to modeling touch-based and multi-user interaction. Although gestural interactions can already be modeled to some extent with the current notation, as shown in our Object-in-Hand example, additional extensions are needed to fully support this kind of interaction, for instance to enable intermediate feedback during the execution of a gesture, and to differentiate between the events of different users. Further research is needed to handle multiple users in both NiMMiT as well as the model-based VR-DeMo approach, as we need an easy way to execute interaction techniques for multiple users simultaneously.

Part IV

Conclusions

Chapter 8

Reflections, contributions and future work

Contents

8.1 Reflection on the research challenges

In Chapter 1, we put forward several research challenges, which we addressed throughout the different chapters of this dissertation. In this final chapter, we summarize the main conclusions of our dissertation with regard to those research challenges, we present our overall contributions and we discuss various opportunities for future work.

Our first research challenge revolved around *making touch-based interfaces in walk-up-and-use environments self-explanatory, for both single-user and multi-user settings*. Walk-up-and-use environments impose specific requirements on the accessibility of touch-based interfaces. There are few conventions regarding the use of gestures, and the limited interaction time and need for

immediate use of the system do not allow for much training or exploration. Furthermore, traditional help systems fail to express multi-touch interactions in a comprehensible and concise manner, because they often involve multiple synchronized actions. Therefore, we explored the concept of a self-explanatory interface in the first part of this dissertation.

With TouchGhosts, which we introduced in Chapter 2, we offer a system that demonstrates interaction techniques to the user through embedded visual "guides". The graphical nature of our approach lowers the threshold to consult the help system and allows a clear view on the synchronization of multiple inputs. We presented a number of strategies to invoke and visualize help and we discussed our COMETs and Microsoft .NET architectures, which allow easy integration of a self-explanatory TouchGhost interface in new or existing applications. In Chapter 3, we discussed the results of a few initial evaluations of different visualization strategies. While investigating textual help, demonstration videos and animated virtual hands in a single-user setting, participants did not express a distinct preference for one method or another. However, our observations revealed some important limitations, which we took into account while developing a collaborative puzzle game to investigate textual and animated help in a multi-user environment. The second study shows that animated help allows users to quickly discover the available interaction possibilities, with a positive effect on collaboration, as users worked together to learn the application.

In the second part of this dissertation, we delved deeper into the collaborative aspects of multi-user environments. The second research challenge dealt with *providing interaction management and enhanced mutual awareness in a multi-user environment without interrupting the dynamic work flow.* Allowing multiple people to interact simultaneously in a highly collaborative setting gives rise to several types of conflicts and possible misconducts, especially if the environment can include both co-located and remote participants. Collaborative systems are thus in need of floor control policies that resolve and prevent conflicts and misconducts gracefully. Our Focus+Roles approach, described in Chapter 4, employs the user's roles and focus to extend existing solutions. Roles define a user's privileges during a particular activity, while tracking the users' focus provides a means of countering some of the typical problems caused by a lack of mutual awareness. Furthermore, in combination with meta-data such as a document's content type and sensitivity, roles make up an effective access control system. We applied the approach to a group of users interacting simultaneously in a digital meeting system, iConnect, which results in elegant conflict handling and access to shared data.

Continuing our work on collaborative environments, our third research challenge was about *a storyboarding tool to support the various disciplines in a multidisciplinary team and maintain equitable contributions, without impacting the team's creativity.* To take advantage of the different viewpoints and approaches that members of a multidisciplinary team bring to the table, it is important that each member has the opportunity to contribute equally to the decision making process and that a degree of mutual engagement is established. In Chapter 5, we investigated how multidisciplinary teams create storyboards through an observational study, because storyboards are well suited to attain a common understanding in multidisciplinary teams involved in user-centered software design and development. We presented the lessons learned from this study regarding the storyboarding process, group interaction and design rationale. Based on these results, we formulated and concretized requirements to inform the design of a tabletop tool for collaborative storyboarding.

Multi-touch tabletops are well suited to co-located collaboration because of their ability to track multiple inputs simultaneously, but the majority of those tabletops are unable to associate contact points with particular users, a feature that can improve a multi-user interface in a number of ways. Existing approaches to user identification are intrusive, require users to stay in a fixed position, or suffer from poor accuracy. Therefore, our fourth and final research challenge dealt with *non-intrusive identification of the different users around a tabletop, independently of the hardware technology.* In Chapter 6, we presented a technique for mapping touches to their corresponding user in a collaborative environment. By mounting a high-resolution camera above the interactive surface, we are able to identify touches reliably without any extra instrumentation, while users can move around the surface at will. Our technique, which leverages the back of users' hands as identifiers, supports walk-up-and-use situations in which multiple people interact on a shared surface.

To complement our research on touch-based and multi-user interaction, we investigated the development of such interfaces in the third part of this dissertation. In one of the overall fields of research in our lab, we studied how model-based design can aid in the development process by designing environments through the use of high-level diagrams. In this context, we introduced NiMMiT, a graphical notation for expressing and evaluating multimodal user interaction. NiMMiT is focused on 3D virtual environments, with limited support for multiple users. In Chapter 7, we reflected on the current limitations of NiMMiT with regard to touch-based and multi-user interaction, and we explored various solutions to extend NiMMiT beyond these boundaries.

8.2 Summary of overall contributions

Based on our experiences reported throughout this dissertation, we can state that interactive surfaces such as tabletops offer great potential in walk-up-and-use and multi-user environments, but to fully exploit that potential, the user interface needs to meet a number of requirements. Touch-based interfaces have to be very accessible, for example, as users need to be able to quickly explore and learn the interface, especially in walk-up-and-use environments. Furthermore, in a multi-user setting, collaboration has to be explicitly supported in order to be effective, for instance by handling conflicts, facilitating decision making processes, and identifying the different users.

To situate our contributions to these challenges in the overall area of touch-based and multi-user interaction, we reconsider the various research fields listed in Section 1.4. The main contributions of this dissertation lie in the fields of enhancing the accessibility of touch-based user interfaces and improving the collaborative aspect of multi-user environments. In this context, we presented a number of approaches toward making single-user and multi-user interfaces self-explanatory in walk-up-and-use environments, and toward providing conflict-free multi-user environments, where we can non-intrusively identify the user behind each action. Although we primarily focus on touch-based interfaces in this dissertation, some of our approaches can also be adopted in other domains, such as deformable user interfaces and augmented reality, as discussed in the next section.

As we approached these topics from various angles, we also worked in some of the other research fields, in support of our primary goals. In our research on multi-user environments, we explored two application domains: a general meeting environment and collaborative storyboarding in a multidisciplinary team. In case of storyboarding, observational studies have shown to be very helpful in determining more specific requirements to inform the design of a digital tool. In addition, we encountered a need for non-intrusive user identification throughout our work on multi-user environments. With Carpus, we aimed for an identification technique that functions with any multi-touch technology, and as a result, we touched upon the field of hardware developments by extending typical tabletop setups with an overhead camera and recognition software.

From a design and development perspective, we put forward a number of useful insights into how users interact with a help system in single-user and multi-user settings, and how we can design a collaborative tool in an informed way. With regard to innovations in the field of development processes, we

reflected on the use of a graphical description language, NiMMiT, to model touch-based and multi-user interaction techniques.

8.3 Future work

Throughout the different chapters of this dissertation, we already discussed the main limitations of our work and possible directions for future research. In this section, we present an overview of the numerous challenges that are still ahead of us, and we elaborate on how our results can be used and extended in various ways, as we and other researchers tackle these challenges.

8.3.1 Refining (help for) touch-based interfaces

In Chapter 2, we mentioned a number of common issues regarding current touch-based interfaces. The user interface components are mostly hidden within the smooth aesthetic design of the user interface, with little to no perceived affordances. As a result, users have difficulties figuring out which of, and how, the various components respond to touches. In addition, there are very few conventions regarding the use of gestures, resulting in a lack of consistency across different applications and a low memorability of the gestures. We proposed a solution in the form of TouchGhosts, a help system that shows users the available gestures and how interface components respond to them.

One obvious line of future work is investigating other means of invoking and visualizing help. Other types of visualizations are, for instance, iconic representations or storyboards that show the different steps of an interaction technique. Instead of choosing one specific visualization to accommodate everyone, we can apply Carpus, our identification technique from Chapter 6, to customize the help on a per-user basis. When a particular user calls for help, the system simply selects a visualization strategy based on that user's profile, which can include a user's personal preferences, known languages, and level of experience. This offers numerous opportunities, such as automatically translating textual help or annotations into the user's native language, or showing animated help to beginners and iconic representations of gestures to experienced users.

Regarding the invocation of help, the results of our second experiment in Chapter 3 indicate that triggering help automatically during the puzzle game was not very useful. However, if the system uses Carpus to keep track of the interaction history of a user, this data can be analyzed to invoke help in a

"smarter" way. Analysis may reveal, for example, that a user repeatedly tried to rotate a puzzle piece without first selecting it, and the help system is then able to respond by pointing out that particular part of the interaction. Instead of showing the entire help animation on rotation, the user only gets the part of the animation that is related to the problem at hand, which in this case is the fact that you first need to select the puzzle piece. Splitting up animations in smaller segments will also improve their overall comprehensibility, as it allows users to process an explanation step by step, at their own pace.

In the long term, we also anticipate improvements to the accessibility of touch-based interfaces by working on the aforementioned problems of affordances and consistency, for instance through an in-depth exploration of touch affordances [Schöning 09] and the suitability of gesture sets [Wobbrock 09] in the context of walk-up-and-use environments. The objective of those investigations would be to provide a comprehensive set of design guidelines regarding the discoverability of touch-based interactions.

8.3.2 Group aspects and long-term effects of help

Based on the second experiment described in Chapter 3, we concluded that animated help has a positive effect on collaboration, as users work together to learn the application. However, we only observed groups of two, and both participants knew each other and had no experience with the application. As the diversity and size of the groups increases, we may see different behaviors. We expect, for example, groups splitting in smaller subgroups and users taking on particular roles, such as the role of a teacher, explaining the different gestures to the others, or the role of a facilitator, controlling the work flow to prevent conflicts. Furthermore, the characteristics of the environment influence certain behaviors, as people act differently in a work environment or public place.

We primarily concentrated on touch-based interfaces in walk-up-and-use scenarios, which implies that the time to learn an interaction technique is more important than remembering that technique over longer periods of time. In work environments, on the other hand, initial training may not always be an issue, but retention over time most likely is a looked-for characteristic. Some ways of providing help may allow users to remember instructions more efficiently than others [Palmiter 93]. Therefore, one of the long-term research challenges is to set up a number of longitudinal studies to investigate the effects of various visualizations on memory retention.

8.3.3 A storyboarding tool for multidisciplinary teams

In Chapter 5, we put forward a number of initial recommendations on how a digital storyboarding tool can effectively facilitate collaborative storyboarding in multidisciplinary teams on an interactive tabletop. The scope of the observational study that served as a basis for these recommendations was limited to a certain degree, as we assembled three teams that had to work on a particular topic during one session. In order to confirm and refine our recommendations, it is necessary to observe multidisciplinary teams in a real-life setting over an extended period of time. This will lead to further insights into aspects such as the individual preparation before a storyboarding session and the importance of maintaining the design rationale.

One of our recommendations highlights the importance of maintaining the design rationale. Once more, we can rely on Carpus to identify the different users, so it is possible to keep track of each user's interaction history and contributions. The system is then able to link those contributions to the different scenes in the storyboard. A user's interaction history and contributions are also valuable when pursuing equitable contributions to prevent reserved users from being less involved, and to facilitate balanced decisions supported by the entire team. After determining the activity rate of each individual, the system can prioritize the actions of less assertive users, whose actions may otherwise be suppressed by more prevailing users.

8.3.4 The future of touch-based interaction and beyond

So far, we mainly discussed future work that is immediately linked to the research challenges presented in this dissertation. Now, we look further into the future, as we speculate about how our research can interact and merge with other emerging technologies and domains. The recent evolution of touch-based interaction was driven by the very fast development and propagation of multi-touch hardware, on both a small (e.g. mobile phones) as well as a large scale (e.g. tabletops). Given the current competitiveness in the market of mobile devices, the development of innovative hardware will not come to an end anytime soon, and advancements will either create new opportunities to apply the research presented in this dissertation, or they will necessitate new approaches.

With regard to mobile devices, high-quality bendable displays are among the possible advancements, for instance to create flexible mobile phones with a deformable user interface. Several manufacturers have already demonstrated a concept device: interaction is no longer limited to the multi-touch screen,

as users can navigate through the user interface by flexing or twisting the whole body of the device in a particular direction. Again, novel interaction techniques will be introduced that users are not familiar with, and suitable ways of revealing and explaining these interactions will be required. This ties in with our research on TouchGhosts, even though it is currently focused on 2D gestures on flat surfaces and further studies are needed to assess the suitability of, for instance, 3D animations.

While mobile devices are typically used as personal devices, large interactive surfaces such as the Samsung SUR40 (the Microsoft PixelSense display, which measures 40 inches or about 102 centimeters) are often found in multi-user environments. Although we call it "large", such a surface is actually quite small to support effective collaboration, even for small groups of users. One solution is to introduce much bigger surfaces, the size of large meeting tables. Interactive surfaces of those proportions may invalidate some of the current guidelines on tabletop interaction, and solicit a reinvestigation of group behavior, for instance during collaborative storyboarding. Furthermore, as physical distances increase, new challenges will arise with regard to the readability and reachability of data, and thus also the mutual awareness and engagement of users.

Increasing the size of shared surfaces is only one possibility to support larger groups of users. Ideally, there would not always be a need for a specific infrastructure (e.g. a large multi-touch surface) when people want to collaborate. Ad-hoc collaboration can be achieved by using personal devices, such as tablets and mobile phones, with or without the inclusion of a shared display. An environment such as iConnect already combines PDAs with shared surfaces, but the omnipresence of feature-rich devices such as smartphones offers more opportunities regarding collaboration at any time or place. As stated in Chapter 5, however, the inclusion of a shared surface typically yields more equitable participation and only relying on personal devices may cause some people to be less engaged in the group activity.

When users collaborate by using personal devices, the size of the displays is a restraining factor as to how people are able to work with shared data. To counter this, a shared display can be created through a small mobile projector [Schöning 10], which is already integrated into some present-day mobile devices. Furthermore, instead of only relying on the touch-sensitive screen of the mobile device for input, any surface (e.g. a regular table or wall) could be transformed into an interactive surface by means of free-hand gesture recognition. OmniTouch [Harrison 11a] is a nice example of a wearable depth-sensing and projection system that enables interaction on everyday surfaces. Free-

hand gestures bring about some of the same challenges as touch-based interaction and flexible devices, as users have to be able to quickly discover and learn the interaction possibilities.

Although it is difficult to predict what kind of developments the future may bring, we envision that gestural interaction will gradually become the foremost interaction paradigm. However, it may not involve touch-sensitive displays, as personal devices could be replaced by a non-intrusive wearable computer, such as Google Glass[1]. This kind of head-mounted display can be used in conjunction with free-hand gestures to create an interactive augmented reality, somewhat similar to SixthSense [Mistry 09]. In addition to ensuring that such interfaces are accessible to their users, it will be an interesting challenge to explore them in the field of multi-user interaction.

8.4 Scientific contributions and publications

The research presented in this dissertation is published in several scientific articles. The following overview lists the most significant publications that contributed in a direct way to this dissertation:

- [Vanacken 08b] Davy Vanacken, Alexandre Demeure, Kris Luyten, and Karin Coninx. *Ghosts in the Interface: Meta-user Interface Visualizations as Guides for Multi-touch Interaction.* In Proceedings of the third IEEE international workshop on Horizontal interactive human computer systems, TABLETOP '08, pages 87-90, IEEE Computer Society, October 2008, Amsterdam, the Netherlands.

- [Vanacken 07a] Davy Vanacken, Chris Raymaekers, Kris Luyten, and Karin Coninx. *Focus+Roles: Socio-Organizational Conflict Resolution in Collaborative User Interfaces.* In Proceedings of the 12th international conference on Human-computer interaction, HCII '07, pages 788-796, Springer, July 2007, Beijing, P.R. China.

- [Ramakers 12] Raf Ramakers, Davy Vanacken, Kris Luyten, Karin Coninx, and Johannes Schöning. *Carpus: A Non-Intrusive User Identification Technique for Interactive Surfaces.* In Proceedings of the 25th ACM symposium on User interface software and technology, UIST '12, pages 35-44, ACM, October 2012, Cambridge, MA, USA.

[1] `http://plus.google.com/+projectglass`

- [Vanacken 06] Davy Vanacken, Joan De Boeck, Chris Raymaekers, and Karin Coninx. *NiMMiT: a Notation for Modeling Multimodal Interaction Techniques.* In Proceedings of the first international conference on Computer graphics theory and applications, GRAPP '06, pages 224-231, INSTICC, February 2006, Setúbal, Portugal.

In addition to the abovementioned works, we published a few other articles related to the work presented in this dissertation:

- [Vanacken 09a] Davy Vanacken, Kris Luyten, and Karin Coninx. *Touch-Ghosts: Guides for Improving Visibility of Multi-Touch Interaction.* In Multitouch and surface computing workshop at CHI '09, April 2009, Boston, MA, USA.

- [Vanacken 08a] Davy Vanacken. *Interactive Workspaces: Multi-user, Multi-touch, Multi-device.* In Electronic proceedings of the 2008 ACM conference on Computer supported cooperative work, CSCW '08 (doctoral colloquium), ACM, November 2008, San Diego, CA, USA.

- [De Boeck 07] Joan De Boeck, Davy Vanacken, Chris Raymaekers, and Karin Coninx. *High-Level Modeling of Multimodal Interaction Techniques Using NiMMiT.* In Journal of Virtual Reality and Broadcasting, JVRB '07, volume 4, number 2, September 2007.

We also organized two workshops on multi-touch interfaces:

- [Luyten 10] Kris Luyten, Davy Vanacken, Malte Weiss, Jan Borchers, Shahram Izadi, and Daniel Wigdor. *Engineering patterns for multi-touch interfaces.* In Proceedings of the second ACM SIGCHI symposium on Engineering interactive computing systems, EICS '10, pages 365-366, ACM, June 2010, Berlin, Germany.

- [Luyten 11] Kris Luyten, Davy Vanacken, Malte Weiss, Jan Borchers, and Miguel Nacenta. *Second workshop on engineering patterns for multi-touch interfaces.* In Proceedings of the third ACM SIGCHI symposium on Engineering interactive computing systems, EICS '11, pages 335-336, ACM, June 2011, Pisa, Italy.

Appendices

Appendix A

Documents of the single-user and multi-user evaluations of different TouchGhost strategies

This appendix contains the documents that were used during the evaluations of various TouchGhost strategies, as described in Chapter 3.

A.1 Textual help of the single-user evaluation

This section contains the list of explanations we provided in the in-game textual help during the evaluation of different single-user TouchGhost strategies, as reported in Section 3.2.

Move (common gesture)

1. Press with one finger on the element.
2. Move the finger to the desired position.
3. Release the finger.

Resize (common gesture)

1. Place two fingers in opposite corners of the element.
2. Move the two fingers to the inside or outside of the element.
3. Release the fingers.

Add to pile (picture)

1. Press with one finger on the element.
2. Move the element on top of a pile.
3. Release the finger.

Gray/color (picture)

1. Place two fingers next to each other on the element.
2. Place a third finger on the element.
3. Move the third finger diagonally (top left to bottom right or bottom right to top left).
4. Release the fingers.

Color overlay (picture)

1. Place two fingers next to each other on the element.
2. Place a third finger on the element.
3. Move the third finger diagonally (top right to bottom left or bottom left to top right).
4. Release the fingers.

Next/previous (picture pile)

1. Double tap with a finger on the right or left side of the element.

Expand (picture pile)

1. Place two fingers next to each other on the element.
2. Place a third finger on the element.
3. Move the third finger to the outside.
4. Release the third finger.
5. Press on another picture.

Scatter (picture pile)

1. Place four fingers on the element.
2. Release the fingers.

Play/pause (video player)

1. Double tap with a finger on the element.

Playing speed (video player)

1. Place two fingers next to each other on the element.
2. Place a third finger on the element.
3. Move the third finger horizontally.
4. Release the fingers.

Volume (video player)

1. Place two fingers next to each other on the element.
2. Place a third finger on the element.
3. Move the third finger vertically.
4. Release the fingers.

A.2 Questionnaires of the evaluations

This section contains the questionnaires that were part of the evaluation of the single-user TouchGhost strategies, as reported in Section 3.2, and of the multi-user strategies, as reported in Section 3.3.

Questionnaire of the single-user evaluation.

Test person:	**Feedback Experiment**

Sex: □ male □ female

Age: _____

How much experience do you have with multi-touch surfaces?

□ none □ very little □ some □ a lot

How much experience do you have with gesture-based interfaces?

□ none □ very little □ some □ a lot

Remarks:

_____

_____

_____

_____

_____

_____

_____

_____

This concludes the test. May we ask you not to give any information to others;
this could influence the results. Thank you for your cooperation!

General questions

The help system was easy to activate

Strongly disagree 1 2 3 4 5 Strongly agree

Help system - text

I understood the help system without any explanation

Strongly disagree 1 2 3 4 5 Strongly agree

The help system explained the gesture clearly

Strongly disagree 1 2 3 4 5 Strongly agree

After consulting the help system, I could easily replicate the gesture

Strongly disagree 1 2 3 4 5 Strongly agree

The help system allowed me to discover the gesture quickly

Strongly disagree 1 2 3 4 5 Strongly agree

I did not find reading the text bothersome

Strongly disagree 1 2 3 4 5 Strongly agree

Overall, I found this technique effective

Strongly disagree 1 2 3 4 5 Strongly agree

Help system - video

I understood the help system without any explanation

Strongly disagree 1 2 3 4 5 Strongly agree

The help system explained the gesture clearly

Strongly disagree 1 2 3 4 5 Strongly agree

After consulting the help system, I could easily replicate the gesture

Strongly disagree 1 2 3 4 5 Strongly agree

The help system allowed me to discover the gesture quickly

Strongly disagree 1 2 3 4 5 Strongly agree

I would like to have more controls (pause, navigation bar, …)

Strongly disagree 1 2 3 4 5 Strongly agree

Overall, I found this technique effective

Strongly disagree 1 2 3 4 5 Strongly agree

Help system – animated hands

I understood the help system without any explanation

Strongly disagree 1 2 3 4 5 Strongly agree

The help system explained the gesture clearly

Strongly disagree 1 2 3 4 5 Strongly agree

After consulting the help system, I could easily replicate the gesture

Strongly disagree 1 2 3 4 5 Strongly agree

The help system allowed me to discover the gesture quickly

Strongly disagree 1 2 3 4 5 Strongly agree

The graphical feedback (red dot to indicate a press) was clear

Strongly disagree 1 2 3 4 5 Strongly agree

I would like to have more controls (rewind, speed up, …)

Strongly disagree 1 2 3 4 5 Strongly agree

Overall, I found this technique effective

Strongly disagree 1 2 3 4 5 Strongly agree

Questionnaire of the multi-user evaluation.

TEST #_____ PARTICIPANT #_____

Gender: ☐ Male ☐ Female **Age**: _____

Please indicate your level of experience

	Never played	Playing/played seldom (once or twice a year)	Playing/played occasionally (once or twice a month)	Playing/played often (at least every week)	Playing/played a lot (almost every day)
with multiplayer video games	○	○	○	○	○

	Never used	Using/used seldom (once or twice a year)	Using/used occasionally (once or twice a month)	Using/used often (at least every week)	Using/used a lot (almost every day)
with multi-touch tabletop systems	○	○	○	○	○

Please mark with an 'x' the position on the line that corresponds the most to your gaming experience. If you want to explain your answer, please use the blank space after each question.

1. I enjoyed the puzzle game.
   Not at all ——————————— Very much

2. Solving the puzzle collaboratively increased my enjoyment.
   Not at all ——————————— Very much

3. I contributed a lot individually.
   Not at all ——————————— Very much

4. I experienced a high level of collaboration.
   Not at all ——————————— Very much

5. Everyone contributed equally to solving the task.

   Not at all ——————————— Very much

6. When a certain task could be performed both collaboratively and individually (e.g. the manipulation of heavy blocks), it was mostly accomplished together with my partner.

   Not at all ——————————— Very much

7. The provided help contributed to the level of collaboration.

   Not at all ——————————— Very much

8. It was clear when it was necessary to collaborate.

   Not at all ——————————— Very much

9. I was always aware of my partner's actions.

   Not at all ——————————— Very much

10. We had problems when performing collaborative tasks.

    Not at all ——————————— Very much

    Specify what problems:

11. It was clear how to use the help without any explanation.

    Not at all ——————————— Very much

12. The help explained the required actions clearly.

    Not at all ——————————— Very much

13. After consulting the help, I could easily replicate the actions.

Not at all ——————————— Very much

14. It was clear how to perform cooperative tasks.

Not at all ——————————— Very much

15. The help allowed me to discover the required actions quickly.

Not at all ——————————— Very much

16. I learned most of the actions through the help.

Not at all ——————————— Very much

17. I learned a lot by talking to my partner.

Not at all ——————————— Very much

18. I learned a lot by watching my partner's actions.

Not at all ——————————— Very much

19. I had to consult help multiple times for a particular action.

Not at all ——————————— Very much

20. Help took me out of the game experience.

Not at all ——————————— Very much

If you have any comments or remarks, please provide them below:

Appendix B

Documents of the observational study on collaborative storyboarding in multidisciplinary teams

This appendix contains the personas, scenario and questionnaire that were part of the observational study on collaborative storyboarding in multidisciplinary teams, as reported in Section 5.3.

B.1 Personas

Figure B.1: Photograph of Bob.

Bob is 45 years old, works as a banker, is married to Mary and is a father of two children: Kate and Benjamin. His working days are very long, and in the evenings he often has to do some work or has appointments with customers. Together with his wife and children, he lives in a nice house, which was constructed a few years ago. Bob likes gadgets and new technologies, and was the first one in his circle of friends that was using a smartphone. Since that moment, he can check his e-mails on any location and he likes the fact that he has a browser, a route planner and a camera in his pocket. Recently, he also uses his smartphone to control the home automation system. At home, it is easy to control the heating, lighting

and music from any place. He loves playing with the settings when he is relaxing in the living room. He also uses the control application on his smartphone at work. As a banker, he continuously checks how he can save money. Because of economic and environmental reasons, he also tries to do savings in expenses for energy and electricity. The home automation system helps him to follow up possible energy savings.

Figure B.2: Photograph of Mary.

Mary is 43 years old, works as a teacher, is married to Bob and is a mother of two children: Kate and Benjamin. She teaches the languages English and German to high school students. She loves her job, and the fact that she can spend long vacations together with the children. Since a few years, they are living in a newly constructed house with a big garden, and she loves cooking and gardening. Although she has a laptop for teaching activities, she is not very eager to learn new technologies and limits the use of computers to the necessary computer tasks she has to do for teaching and keeping in touch with a few friends. Every time the family buys new household appliances like a coffee machine, or a washing machine, it takes her several days to learn how to use the new system. Since a few months, a new home automation system is installed in their house. Mary loves the idea that some things can be controlled automatically, Bob installed the application on her laptop and because of its ease of use, Mary can easily control settings using her laptop. However, usually, she controls the settings at home, using the central displays that are available on each floor of their house.

Benjamin is 10 years old, he likes playing outside. Soccer is his favorite game, which he loves to play together with his friends. Two days a week, he practices soccer at the local soccer team. Sometimes, he helps his mother gardening and cooking. A few weeks ago, he surprised his parents and his sister with homemade desserts. He is really proud that he can prepare these recipes on his own. His sister is usu-

Figure B.3: Photograph of Benjamin.

ally teasing him, and is interested in the opposite things. However, lately she showed him some cool computer games and configured the home automation system so that he can easily select his favorite lighting settings and TV show, by selecting only one profile. Benjamin is worried about the future of the environment. The theme of a current project at school is the environment, and Benjamin tries to contribute by applying some guidelines at home.

Figure B.4: Photograph of Kate.

Kate is a 14 year old student. She is very interested in mathematics and her hobbies are reading and chess. Because her mother says it is important to do some sports, she tries jogging a few times a week, but she does not like that at all. She rather likes to read a book or to play chess or computer games. She is also interested in her father's smartphone and the different applications that are available for this device. She would love to have one of her own. Although she begged several times to have her own smartphone, her parents do not allow her to have one. Since her parents bought a home automation system, she likes to play with the settings of it. She loves the way that the system adapts to several user profiles and activities. For each activity at home, she has programmed settings in her profile.

B.2 Scenario

Recently, Bob and Mary decided to install a home automation system. They decided to do that in order to control heating and lighting easily and to save some costs on energy consumption. This system allows them to control settings and to adapt these settings according to their own profiles. In the past, it often happened that the children left the house without turning off the lights, or that the programmed heating system was heating the house, while no one was at home. Using the two displays installed in the house, settings of the system can be controlled. One display is available in the living room, and another one in the hallway on the first floor.

Today, Kate wants to read a book, while her mother and her brother are outside. In the living room, she loads her personal profile, and automatically

all lights are switched off, the light near the sofa is turned on and her favorite pop music starts playing. Thirty minutes later, her mother and Benjamin come in. Benjamin loads his personal profile using the home automation display in the living room. The system recognizes two profiles now. Based on the profiles Kate programmed before, the light above the sofa stays on, while the pop music stops and the TV starts playing Benjamin's favorite TV show.

Mary begins to prepare dinner. Five minutes later, the phone is ringing. Mary picks up the phone. It is Bob, who wants to notify her that he is stuck in traffic on his way home. Tonight he will have to work in the home office, and he already programmed the heating for that room using his smartphone. He also mentions that the family does not have to wait for him for dinner. One hour later, Bob arrives at home. After his dinner, he explores the home automation system together with Benjamin, using the central display in the living room. He teaches Benjamin how to interpret the statistics regarding the energy savings. Benjamin is impressed by the system's ability to record this kind of information and is already thinking about how the efficiency can be improved.

Later that evening, when the children are in bed, Bob shuts down his computer in the home office. He joins Mary, who is reading a book in the living room. Bob's smartphone reminds him that he left the home office, but the heating in this room is still on. Bob accepts the system's suggestion to switch off the heating in the home office. Bob and Mary discuss their day, and then they go to sleep. As programmed in the system, the heating is turned off automatically, and the lights are switched off by one press on a button of Bob's smartphone, when he gets into bed.

B.3 Questionnaire

This section contains the questionnaire of the observational study on collaborative storyboarding in multidisciplinary teams.

Questionnaire

Team: ……
Role: ……

This questionnaire can be answered in English or in Dutch.

ABOUT YOU

1. **Gender**: female / male
2. **Age**: ….. years
3. **Education**: ……
4. **Current job**: ……
5. **What role matches best with your current job (you can select multiple answers)?**
   - ❑ HCI specialist
   - ❑ Designer
   - ❑ Systems analyst / Programmer
   - ❑ Stakeholder (e.g. purchaser, application domain specialist)
   - ❑ None of the above
6. **How comfortable did you feel with the role that was assigned to you?**

Not comfortable at all	Not comfortable	Neutral	Comfortable	Very comfortable

EXPERIENCE

7. **Do you have any experience being part of a multi-disciplinary team?**
   - ○ Yes
   - ○ No

   If you answered yes, for how long did / do you work in a multi-disciplinary team?
   - ○ A few months
   - ○ More than 1 year
   - ○ More than 5 years
8. **Do you have any experience in creating this type of storyboards?**
   - ○ Yes
   - ○ No
9. **Do you have any experience in using this type of storyboards?**
   - ○ Yes
   - ○ No

THE STORYBOARDING WORKSHOP

10. How easy was it to create the storyboard, starting from the scenario and personas?

Very difficult	Difficult	Neutral	Easy	Very easy

11. Did you understand what your immediate goals were and what you needed to do to achieve them?

○ Yes
○ No

12. How satisfied are you with the resulting storyboard?

Not satisfied at all	Not satisfied	Neutral	Satisfied	Very satisfied

Why?_____
_____

13. How would you estimate your <u>influence</u> on the storyboard?

……% of the storyboard is based on my ideas.

14. How would you estimate your <u>direct contribution</u> to the storyboard?

……% of the storyboard contains my sketches, artifacts, etc.

15. How satisfied are you with the extent to which you could contribute to the storyboard?

Not satisfied at all	Not satisfied	Neutral	Satisfied	Very satisfied

16. How satisfied are you with the extent to which the team used your contributions?

Not satisfied at all	Not satisfied	Neutral	Satisfied	Very satisfied

17. Were there any missing tools during the storyboarding workshop? If yes, what tools should be added?

- ○ Yes, _____
- ○ No

18. Did you work on ideas in private (e.g. on an isolated piece of paper) before sharing them?

- ○ Yes
- ○ No

Why?_____
_____

19. Were you continually aware of what the others were doing throughout the workshop?

- ○ Yes
- ○ No

20. Did being part of a multi-disciplinary team influence the ideas contained by the storyboard?

- ○ Yes
- ○ No

Why?_____
_____

21. Did you understand the symbolism and sequence of the final storyboard and the related information that was visible on the table?

- ○ Yes
- ○ No

22. Given the role you had during the workshop, what aspects of the storyboard would be useful for your following tasks during the project?

_____
_____

MULTI-TOUCH STORYBOARDING

23. **Do you think that a tool for a multi-touch table that supports storyboarding workshops can be helpful? Explain your answer.**

_____
_____

24. **What kind of features would you expect to be part of this multi-touch tool?**

_____
_____
_____

25. **Would you be interested in storyboarding on one shared, collaborative device, or on your personal devices, or on a mix of both types of devices? Explain your answer.**

_____
_____

IN GENERAL

26. **Do you have any general remarks/comments regarding the user study or storyboarding workshop?**

_____
_____
_____

Thank you very much for participating in our user study!!!

Appendix C

Nederlandstalige samenvatting

Gebruikersinterfaces die aangestuurd worden door eenvoudige aanrakingen met de vingers zijn steeds prominenter aanwezig in onze dagdagelijkse omgeving. Zo zijn ze terug te vinden op allerlei hardware, gaande van mobiele telefoons tot grote publieke schermen. Bovendien laten ze diverse applicaties toe, van eenvoudige spelletjes voor één gebruiker tot meer zakelijke toepassingen die ondersteuning bieden voor brainstormen in groep. Hoewel dit soort interfaces verondersteld worden erg "natuurlijk" te zijn, moet er toch de nodige aandacht besteed worden aan hun toegankelijkheid. De specifieke bewegingen die dienen uitgevoerd te worden met de vingers kunnen immers moeilijk te ontdekken en leren zijn, mede door een gebrek aan algemeen aanvaarde conventies hieromtrent. Doordat de meeste hardware vandaag niet enkel "single-touch" maar ook "multi-touch" invoer ondersteunt, kunnen meerdere mensen bovendien gelijktijdig samenwerken op één gedeeld, interactief oppervlak. Ook dit brengt nieuwe onderzoeksuitdagingen met zich mee, met name rond hoe dit soort samenwerking op een effectieve manier ondersteund kan worden.

Vooral in "walk-up-and-use" omgevingen zoals publieke ruimten is de toegankelijkheid van de interface van erg groot belang, gezien gebruikers het systeem onmiddellijk moeten kunnen gebruiken en er heel weinig tijd is om de interface te verkennen en leren. Daarom onderzoeken we manieren waarop een interface snel en eenvoudig duidelijk kan maken hoe ermee om te gaan. Met TouchGhosts stellen we een helpsysteem voor dat de mogelijke interacties aan de gebruiker demonstreert binnen de interface zelf, bijvoorbeeld via geanimeerde handen die de acties tonen. De visuele aard van onze aanpak geeft een duidelijk beeld van de eventuele synchronisatie tussen meerdere vingers,

wat typisch is voor interacties zoals het inzoomen met twee vingers. Gebruikersstudies geven aan dat geanimeerde hulp gebruikers toelaat de beschikbare interactiemogelijkheden snel en makkelijk te ontdekken. Verder kunnen zulke animaties binnen "multi-user" toepassingen een positief effect hebben op de samenwerking, doordat ze de gebruikers de applicatie samen laten verkennen.

Wanneer meerdere gebruikers intensief samenwerken in een collaboratieve omgeving geeft dit aanleiding tot een aantal bijkomende uitdagingen, zeker indien de gebruikers zowel lokaal als vanop afstand kunnen deelnemen. Zulke samenwerking kan onder andere conflicten en wangedrag met zich meebrengen, waardoor de collaboratieve omgevingen nood hebben aan "floor control" mechanismen. Hiermee proberen we problemen op een elegante manier te voorkomen of op te lossen, zonder de dynamiek van het samenwerken te doorbreken. Wij stellen een Focus+Roles aanpak voor, die we toepassen binnen een digitaal vergadersysteem, iConnect. De rollen die de gebruikers uitoefenen tijdens de samenwerking definiëren hierbij de privileges van een gebruiker tijdens bepaalde activiteiten. Tevens houden we rekening met waar de aandacht van de gebruikers op gevestigd is, zodat we problemen kunnen vermijden die geassocieerd zijn met een gebrek aan "mutual awareness".

We zetten ons onderzoek rond collaboratieve systemen verder met een analyse van hoe een digitale toepassing storyboarding in een multidisciplinair team best kan ondersteunen. Storyboards zijn erg geschikt om een gemeenschappelijk begrip te verkrijgen tijdens "user-centered" software ontwerp en ontwikkeling, ongeacht de achtergronden en expertises van de teamleden. Ze laten elke teamlid toe deel te nemen aan het beslissingsproces, wat de wederzijdse betrokkenheid van de verschillende partijen bevordert. Om de verscheidenheid aan inzichten en aanpakken die de leden van een multidisciplinair team aanbrengen ten volle te kunnen benutten, gaan we met behulp van een studie na hoe zulke teams storyboards creëren. Op basis van onze observaties formuleren we een aantal richtlijnen om het ontwerp van een multi-touch tool voor storyboarding te vergemakkelijken.

In ons onderzoek rond collaboratieve systemen werden we meermaals geconfronteerd met de noodzaak om de verschillende gebruikers van een multi-touch opstelling te kunnen identificeren. Multi-touch hardware kan wel invoer van meerdere vingers tegelijk verwerken, maar de meeste systemen kunnen de contactpunten niet associëren met specifieke gebruikers. Daarom stellen we Carpus voor, een identificatietechniek waarmee we aanrakingen op een betrouwbare manier kunnen identificeren door de bovenkant van de handen te analyseren via een camera boven het interactief oppervlak. Dit laat toe een multi-user interface op een aantal manieren te verbeteren, bijvoorbeeld door

hulp te personaliseren per gebruiker, of door de activiteiten van een bepaalde gebruiker op te volgen om zo conflicten te voorkomen.

Een ander belangrijk aspect om in overweging te nemen is het ondersteunen van het ontwerpen, ontwikkelen en evalueren van multi-touch en multi-user interfaces. We onderzoeken daarom hoe model-gebaseerd ontwerp het ontwikkelingsproces kan vergemakkelijken, door applicaties te modeleren via "high-level" diagrammen in plaats van ze te implementeren via "low-level" programmeercode. In deze context bespreken we NiMMiT, een grafische notatie om multimodale interactietechnieken te modelleren en te evalueren. Omdat NiMMiT momenteel gericht is op virtuele omgevingen voor één gebruiker, reflecteren we over de huidige beperkingen van NiMMiT met betrekking tot multi-touch en multi-user interactie.

Bibliography

[Abate 07] Andrea F. Abate, Michele Nappi, Daniel Riccio & Gabriele Sabatino. *2D and 3D face recognition: a survey.* Pattern Recognition Letters, vol. 28, no. 14, pages 1885–1906, 2007.

[Ahonen 04] Timo Ahonen, Abdenour Hadid & Matti Pietikäinen. *Face recognition with local binary patterns.* In Proceedings of the 8th European conference on Computer vision, ECCV '04, pages 469–481. Springer, 2004.

[Ambler 04] Scott Ambler. Object primer, the agile model-driven development with UML 2.0. Cambridge University Press, 2004.

[Annett 11] Michelle Annett, Tovi Grossman, Daniel Wigdor & George Fitzmaurice. *Medusa: a proximity-aware multi-touch tabletop.* In Proceedings of the 24th ACM symposium on User interface software and technology, UIST '11, pages 337–346. ACM, 2011.

[Appert 09] Caroline Appert & Shumin Zhai. *Using strokes as command shortcuts: cognitive benefits and toolkit support.* In Proceedings of the 27th international conference on Human factors in computing systems, CHI '09, pages 2289–2298. ACM, 2009.

[Atasoy 11] Berke Atasoy & Jean-Bernard Martens. *STORIFY: a tool to assist design teams in envisioning and discussing user experience.* In Proceedings of the 2011 conference extended abstracts on Human factors in computing systems, CHI EA '11, pages 2263–2268. ACM, 2011.

[Augsten 10] Thomas Augsten, Konstantin Kaefer, René Meusel, Caroline Fetzer, Dorian Kanitz, Thomas Stoff, Torsten Becker, Christian Holz & Patrick Baudisch. *Multitoe: high-precision interaction with back-projected floors based on high-resolution multi-touch input.* In Proceedings of the 23rd ACM symposium on User interface software and technology, UIST '10, pages 209–218. ACM, 2010.

[Avila-Garcia 10] Maria Susana Avila-Garcia, Anne E. Trefethen, Mike Brady & Fergus Gleeson. *Using interactive and multitouch technology to support decision making in multidisciplinary team meetings.* In Proceedings of the 2010 IEEE 23rd international symposium on Computer-based medical systems, CBMS '10, pages 98–103. IEEE Computer Society, 2010.

[Baecker 90] Ron Baecker & Ian Small. *Animation at the interface.* In The Art of Human-Computer Interface Design, pages 251–267. Addison-Wesley Longman Publishing, 1990.

[Bailey 01] Brian P. Bailey, Joseph A. Konstan & John V. Carlis. *DEMAIS: designing multimedia applications with interactive storyboards.* In Proceedings of the ninth ACM international conference on Multimedia, MULTIMEDIA '01, pages 241–250. ACM, 2001.

[Bailly 08] Gilles Bailly, Alexandre Demeure, Eric Lecolinet & Laurence Nigay. *MultiTouch menu (MTM).* In Proceedings of the 20th international conference of the Association Francophone d'Interaction Homme-Machine, IHM '08, pages 165–168. ACM, 2008.

[Balakrishnan 04] Ravin Balakrishnan. *"Beating" Fitts' law: virtual enhancements for pointing facilitation.* International Journal of Human-Computer Studies, vol. 61, no. 6, pages 857–874, 2004.

[Barnkow 12] Lorenz Barnkow & Kai von Luck. *Semiautomatic and user-centered orientation of digital artifacts on multitouch tabletops.* In Proceedings of the 11th interna-

tional conference on Entertainment Computing, ICEC '12, pages 381–388. Springer, 2012.

[Bartindale 12] Tom Bartindale, Alia Sheikh, Nick Taylor, Peter Wright & Patrick Olivier. *StoryCrate: tabletop storyboarding for live film production.* In Proceedings of the 2012 ACM conference on Human factors in computing systems, CHI '12, pages 169–178. ACM, 2012.

[Bau 08] Olivier Bau & Wendy E. Mackay. *OctoPocus: a dynamic guide for learning gesture-based command sets.* In Proceedings of the 21st ACM symposium on User interface software and technology, UIST '08, pages 37–46. ACM, 2008.

[Bau 10] Olivier Bau, Emilien Ghomi & Wendy Mackay. *Arpege: design and learning of multi-finger chord gestures.* Rapport technique 1533, LRI, 2010.

[Bay 08] Herbert Bay, Andreas Ess, Tinne Tuytelaars & Luc Van Gool. *Speeded-up robust features (SURF).* Computer Vision and Image Understanding, vol. 110, no. 3, pages 346–359, 2008.

[Bergqvist 99] Jens Bergqvist, Per Dahlberg, Fredrik Ljungberg & Steinar Kristoffersen. *Moving out of the meeting room: exploring support for mobile meetings.* In Proceedings of the Sixth European Conference on Computer Supported Cooperative Work, ECSCW '99, pages 81–98. Kluwer Academic Publishers, 1999.

[Bertino 01] Elisa Bertino, Piero Andrea Bonatti & Elena Ferrari. *TRBAC: a temporal role-based access control model.* ACM Transactions on Information and System Security, vol. 4, no. 3, pages 191–233, 2001.

[Bertino 05] Elisa Bertino, Barbara Catania, Maria Luisa Damiani & Paolo Perlasca. *GEO-RBAC: a spatially aware RBAC.* In Proceedings of the tenth ACM symposium on Access control models and technologies, SACMAT '05, pages 29–37. ACM, 2005.

[Beznosyk 12] Anastasiia Beznosyk. *An experimental perspective on factors influencing collaborative user experience in virtual environments and games.* PhD thesis, Hasselt University, Diepenbeek, Belgium, November 2012.

[Biehl 08] Jacob T. Biehl, William T. Baker, Brian P. Bailey, Desney S. Tan, Kori M. Inkpen & Mary Czerwinski. *IMPROMPTU: a new interaction framework for supporting collaboration in multiple display environments and its field evaluation for co-located software development.* In Proceedings of the twenty-sixth SIGCHI conference on Human factors in computing systems, CHI '08, pages 939–948. ACM, 2008.

[Blanch 06] Renaud Blanch & Michel Beaudouin-Lafon. *Programming rich interactions using the hierarchical state machine toolkit.* In Proceedings of the working conference on Advanced visual interfaces, AVI '06, pages 51–58. ACM, 2006.

[Blouin 10] Arnaud Blouin & Olivier Beaudoux. *Improving modularity and usability of interactive systems with Malai.* In Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems, EICS '10, pages 115–124. ACM, 2010.

[Boreki 05] Guilherme Boreki & Alessandro Zimmer. *Hand geometry: a new approach for feature extraction.* In Proceedings of the 4th IEEE Workshop on Automatic Identification Advanced Technologies, AUTOID '05, pages 149–154. IEEE Computer Society, 2005.

[Bowman 04] Doug A. Bowman, Ernst Kruijff, Joseph J. Laviola Jr. & Ivan Poupyrev. 3d user interfaces: theory and practice. Addison Wesley Longman Publishing Co., Inc., 2004.

[Bragdon 09] Andrew Bragdon, Robert Zeleznik, Brian Williamson, Timothy Miller & Joseph J. Laviola Jr. *GestureBar: improving the approachability of gesture-based interfaces.* In Proceedings of the 27th international conference on Human factors in computing systems, CHI '09, pages 2269–2278. ACM, 2009.

[Bragdon 10] Andrew Bragdon, Arman Uguray, Daniel Wigdor, Stylianos Anagnostopoulos, Robert Zeleznik & Rutledge Feman. *Gesture play: motivating online gesture learning with fun, positive reinforcement and physical metaphors.* In Proceedings of the ACM international conference on Interactive Tabletops and Surfaces, ITS '10, pages 39–48. ACM, 2010.

[Brignull 03] Harry Brignull & Yvonne Rogers. *Enticing people to interact with large public displays in public spaces.* In Proceedings of the IFIP TC13 international conference on Human-computer interaction, INTERACT '03, pages 17–24. IOS Press, 2003.

[Buxton 12] Bill Buxton. *Multi-touch systems that I have known and loved.* http://www.billbuxton.com/multitouch Overview.html, 2012.

[Cao 08] Xiang Cao, Andrew D. Wilson, Ravin Balakrishnan, Ken Hinckley & Scott E. Hudson. *ShapeTouch: leveraging contact shape on interactive surfaces.* In Proceedings of the 3rd IEEE international workshop on Horizontal Interactive Human Computer Systems, TABLETOP '08, pages 139–146. IEEE Computer Society, 2008.

[Cardinaels 06] Maarten Cardinaels, Geert Vanderhulst, Maarten Wijnants, Chris Raymaekers, Kris Luyten & Karin Coninx. *Seamless interaction between multiple devices and meeting rooms.* In Proceedings of the CHI '06 workshop on Information visualization and interaction techniques for collaboration across multiple displays, CHI '06. ACM, 2006.

[Carr 97] David A. Carr. *Interaction Object Graphs: an executable graphical notation for specifying user interfaces.* In Formal methods for computer-human interaction, pages 141–156. Springer, 1997.

[Carroll 98] John M. Carroll. Minimalism beyond the nurnberg funnel. MIT Press, 1998.

[Choi 09] Junyeong Choi, Byung-Kuk Seo & Jong-Il Park. *Robust hand detection for augmented reality interface.* In Proceedings of the 8th international conference on Virtual Reality Continuum and its Applications in Industry, VRCAI '09, pages 319–321. ACM, 2009.

[Cilella 11] Sal Cilella. *Did you ever know that you're my hero?: the power of storytelling.* Interactions, vol. 18, pages 62–66, 2011.

[Clerckx 04] Tim Clerckx, Kris Luyten & Karin Coninx. *DynaMo-AID: A design process and a runtime architecture for dynamic model-based user interface development.* In Proceedings of the 2004 international conference on Engineering human-computer interaction and interactieve systems, EHCI-DSVIS '04, pages 77–95. Springer, 2004.

[Cleveringa 09] Writser Cleveringa, Maarten van Veen, Arnout de Vries, Arnoud de Jong & Tobias Isenberg. *Assisting gesture interaction on multi-touch screens.* In Multitouch and Surface Computing Workshop at the conference on Human factors in computing systems, CHI '09, 2009.

[Coninx 97] Karin Coninx, Frank Van Reeth & Eddy Flerackers. *A hybrid 2D/3D user interface for immersive object modeling.* In Proceedings of the 1997 conference on Computer Graphics International, CGI '97, pages 47–55. IEEE Computer Society, 1997.

[Coninx 06a] Karin Coninx, Erwin Cuppens, Joan De Boeck & Chris Raymaekers. *Integrating support for usability evaluation into high level interaction descriptions with NiMMiT.* In Proceedings of 13th international workshop on Design, Specification and Verification of Interactive Systems, DSVIS '06, pages 95–108. Springer, 2006.

[Coninx 06b] Karin Coninx, Olga De Troyer, Chris Raymaekers & Frederic Kleinermann. *VR-DeMo: a tool-supported approach facilitating flexible development of virtual environments using conceptual modelling.* In Proceedings of Virtual Concept 2006, pages 30–42. Springer, 2006.

[Coutaz 95] Joëlle Coutaz, Laurence Nigay, Daniel Salber, Ann Blandford, Jon May & Richard M. Young. *Four easy pieces for assessing the usability of multimodal interaction: the CARE properties.* In Proceedings of the fifth IFIP TC13 international conference on Human-computer interaction, INTERACT '95, pages 115–120. Chapman and Hall, 1995.

[Csisinko 10] Mathis Csisinko & Hannes Kaufmann. *VITAL the Virtual Environment Interaction Technique Abstraction Layer.* In Proceedings of the IEEE Virtual Reality 2010 workshop: Software engineering and architectures for realtime interactive systems, pages 77–86. Shaker Verlag, 2010.

[Cui 07] Xiutao Cui, Yuliang Chen & Junzhong Gu. *Ex-RBAC: An extended role based access control model for location-aware mobile collaboration system.* In Proceedings of the second international conference on Internet Monitoring and Protection, ICIMP '07, page 36. IEEE Computer Society, 2007.

[Cullingford 82] Richard E. Cullingford, Myron W. Krueger, Mallory Selfridge & Marie A. Bienkowski. *Automated explanations as a component of a computer-aided design system.* IEEE Transactions on System, Man and Cybernetics, vol. 12, no. 2, pages 168–181, 1982.

[Cuppens 04] Erwin Cuppens, Chris Raymaekers & Karin Coninx. *VRIXML: a user interface description language for virtual environments.* In Proceedings of the Advanced Visual Interfaces workshop on Developing User Interfaces with XML: Advances on User Interface Description Languages, pages 111–117, 2004.

[Cuppens 05] Erwin Cuppens, Chris Raymaekers & Karin Coninx. *A model-based design process for interactive virtual environments.* In Proceedings of 12th international workshop on Design, Specification and Verification of Interactive Systems, DSVIS '05, pages 225–236. Springer, 2005.

[Cuypers 08] Tom Cuypers, Jan Schneider-Barnes, Johannes Taelman, Kris Luyten & Philippe Bekaert. *Eunomia: toward a framework for multi-touch information displays in public spaces.* In Proceedings of the 22nd British CHI group conference on people and computers: culture, creativity, interaction, BCS-HCI '08, pages 31–34. British Computer Society, 2008.

[Da Silva 00] Paulo Da Silva. *User interface declarative models and development environments: a survey.* In Proceedings of the 7th international conference on Design, specification, and verification of interactive systems, DSV-IS '00, pages 207–226. Springer, 2000.

[De Boeck 04] Joan De Boeck, Erwin Cuppens, Tom De Weyer, Chris Raymaekers & Karin Coninx. *Multisensory interaction metaphors with haptics and proprioception in virtual environments.* In Proceedings of the third ACM Nordic Conference on Human-Computer Interaction, NordiCHI '04, pages 189–197. ACM, 2004.

[De Boeck 06a] Joan De Boeck, Juan Manuel Gonzalez Calleros, Karin Coninx & Jean Vanderdonckt. *Open issues for the development of 3D multimodal applications from an MDE perspective.* In Proceedings of the 2nd international workshop on Model Driven Development of Advanced User Interfaces, MDDAUI '06, pages 11–14, 2006.

[De Boeck 06b] Joan De Boeck, Chris Raymaekers & Karin Coninx. *Comparing NiMMiT and data-driven notations for describing multimodal interaction.* In Proceedings of the 5th international conference on Task models and diagrams for users interface design, TAMODIA '06, pages 217–229. Springer, 2006.

[De Boeck 07] Joan De Boeck, Davy Vanacken, Chris Raymaekers & Karin Coninx. *High-level modeling of multimodal interaction techniques using NiMMiT.* Journal of Virtual Reality and Broadcasting (JVRB), vol. 4, no. 2, 2007.

[De Boeck 08] Joan De Boeck, Chris Raymaekers & Karin Coninx. *A tool supporting model based user interface design in*

3D virtual environments. In Proceedings of the international conference on Computer graphics theory and applications, GRAPP '08, pages 367–375, 2008.

[De Boeck 09] Joan De Boeck, Chris Raymaekers & Karin Coninx. *CoGenIVE: building 3D virtual environments using a model based user interface design approach.* In Computer Vision and Computer Graphics, Theory and Applications, volume 24 of *Communications in Computer and Information Science*, pages 83–96. Springer, 2009.

[de Haan 09] Gerwin de Haan & Frits H. Post. *StateStream: a developer-centric approach towards unifying interaction models and architecture.* In Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems, EICS '09, pages 13–22. ACM, 2009.

[Demeure 08] Alexandre Demeure, Gaëlle Calvary & Karin Coninx. *COMET(s), a software architecture style and an interactors toolkit for plastic user interfaces.* In Proceedings of DSVIS '08, pages 225–237. Springer, 2008.

[Deshayes 11] Romuald Deshayes & Tom Mens. *Statechart modelling of interactive gesture-based applications.* In Proceedings of the INTERACT 2011 workshop on Combining design and engineering of interactive systems through models and tools, ComDeisMoto '11, 2011.

[Deshayes 12] Romuald Deshayes, Christophe Jacquet, Cécile Hardebolle, Frédéric Boulanger & Tom Mens. *Heterogeneous modeling of gesture-based 3D applications.* In Proceedings of the MODELS 2012 workshop on Multi-paradigm modeling, MPM '12, 2012.

[Dietz 01] Paul Dietz & Darren Leigh. *DiamondTouch: a multi-user touch technology.* In Proceedings of the 14th ACM symposium on User interface software and technology, UIST '01, pages 219–226. ACM, 2001.

[Dohse 08] K.C. Dohse, Thomas Dohse, Jeremiah D. Still & Derrick J. Parkhurst. *Enhancing multi-user interaction with multi-touch tabletop displays using hand tracking.*

In Proceedings of the first international conference on Advances in Computer-Human Interaction, ACHI '08, pages 297–302. IEEE Computer Society, 2008.

[Dommel 97] Hans-Peter Dommel & J.J. Garcia-Luna-Aceves. *Floor control for multimedia conferencing and collaboration.* Multimedia Systems, vol. 5, no. 1, pages 23–38, 1997.

[Döring 11] Tanja Döring, Dagmar Kern, Paul Marshall, Max Pfeiffer, Johannes Schöning, Volker Gruhn & Albrecht Schmidt. *Gestural interaction on the steering wheel: reducing the visual demand.* In Proceedings of the 2011 conference on Human factors in computing systems, CHI '11, pages 483–492. ACM, 2011.

[Dourish 92] Paul Dourish & Victoria Bellotti. *Awareness and coordination in shared workspaces.* In Proceedings of the 1992 ACM conference on Computer-supported cooperative work, CSCW '92, pages 107–114. ACM, 1992.

[Dragicevic 04] Pierre Dragicevic & Jean-Daniel Fekete. *Support for input adaptability in the ICON toolkit.* In Proceedings of the 6th international conference on Multimodal interfaces, ICMI '04, pages 212–219. ACM, 2004.

[Duffy 93] Thomas M. Duffy, James E. Palmer & Brad Mehlenbacher. Online help: design and evaluation. Norwood: Ablex Publishing Corp., 1993.

[Dumas 11] Signer Beat Dumas Bruno & Denis Lalanne. *A graphical UIDL editor for multimodal interaction design based on SMUIML.* In Proceedings of the workshop on Software support for user interface description language, UIDL '11, 2011.

[Dworman 04] Garett Dworman & Stephanie Rosenbaum. *Helping users to use help: improving interaction with help systems.* In Extended abstracts on Human factors in computing systems, CHI EA '04, pages 1717–1718. ACM, 2004.

[Edwards 96] Keith W. Edwards. *Policies and roles in collaborative applications.* In Proceedings of the 1996 ACM conference on Computer supported cooperative work, CSCW '96, pages 11–20. ACM, 1996.

[Everitt 06] Katherine Everitt, Chia Shen, Kathy Ryall & Clifton Forlines. *MultiSpace: enabling electronic document micro-mobility in table-centric, multi-device environments.* In Proceedings of the first IEEE international workshop on Horizontal interactive human-computer systems, TABLETOP '06, pages 27–34. IEEE Computer Society, 2006.

[Ferraiolo 01] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn & Ramaswamy Chandramouli. *Proposed NIST standard for role-based access control.* ACM Transactions on Information and System Security, vol. 4, no. 3, pages 224–274, 2001.

[Fetter 11] Mirko Fetter, Tom Gross & Maxi Hucke. *Supporting social protocols in tabletop interaction through visual cues.* In Proceedings of the 13th IFIP TC13 international conference on Human-computer interaction, INTERACT '11, pages 435–442. Springer, 2011.

[Figueroa 02] Pablo Figueroa, Mark Green & James H. Hoover. *InTml: a description language for VR applications.* In Proceedings of the seventh international conference on 3D Web technology, Web3D '02, pages 53–58. ACM, 2002.

[Forlines 06] Clifton Forlines, Alan Esenther, Chia Shen, Daniel Wigdor & Kathy Ryall. *Multi-user, multi-display interaction with a single-user, single-display geospatial application.* In Proceedings of the 19th ACM symposium on User interface software and technology, UIST '06, pages 273–276. ACM, 2006.

[Freeman 98] William T. Freeman, David B. Anderson, Paul A. Beardsley, Chris N. Dodge, Michal Roth, Craig D. Weissman, William S. Yerazunis, Hiroshi Kage, Kazuo

Kyuma, Yasunari Miyake & Ken-ichi Tanaka. *Computer vision for interactive computer graphics.* IEEE Computer Graphics and Applications, vol. 18, no. 3, pages 42–53, 1998.

[Freeman 09] Dustin Freeman, Hrvoje Benko, Meredith Ringel Morris & Daniel Wigdor. *ShadowGuides: visualizations for in-situ learning of multi-touch and whole-hand gestures.* In Proceedings of the ACM international conference on Interactive Tabletops and Surfaces, ITS '09, pages 183–190. ACM, 2009.

[Galaczy 99] Patricia Galaczy. Electronic meeting systems: win-win group decision making? IRC Press, 1999.

[Grayling 02] Trevor Grayling. *If we build it, will they come? A usability test of two browser-based embedded help systems.* Journal of the Society of Technical Communication, vol. 49, no. 2, pages 193–209, 2002.

[Green 89] Thomas Green. *Cognitive dimensions of notations.* In People and Computers, pages 443–460. Cambridge University Press, 1989.

[Greenberg 94] Saul Greenberg & David Marwood. *Real time groupware as a distributed system: concurrency control and its effect on the interface.* In Proceedings of the 1994 ACM conference on Computer supported cooperative work, CSCW '04, pages 207–217. ACM, 1994.

[Grossman 07] Tovi Grossman, Pierre Dragicevic & Ravin Balakrishnan. *Strategies for accelerating on-line learning of hotkeys.* In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '07, pages 1591–1600. ACM, 2007.

[Grossman 09] Tovi Grossman, George Fitzmaurice & Ramtin Attar. *A survey of software learnability: metrics, methodologies and guidelines.* In Proceedings of the 27th international conference on Human factors in computing systems, CHI '09, pages 649–658. ACM, 2009.

[Grossman 10] Tovi Grossman & George Fitzmaurice. *ToolClips: an investigation of contextual video assistance for functionality understanding.* In Proceedings of the 28th international conference on Human factors in computing systems, CHI '10, pages 1515–1524. ACM, 2010.

[Guiard 87] Yves Guiard. *Asymmetric division of labor in human skilled bimanual action: the kinematic chain as a model.* Journal of Motor Behavior, vol. 19, pages 486–517, 1987.

[Gutwin 99] Carl Gutwin & Saul Greenberg. *A framework of awareness for small groups in shared-workspace groupware.* Rapport technique 99-1, University of Saskatchewan, 1999.

[Gutwin 08] Carl Gutwin, Saul Greenberg, Roger Blum, Jeff Dyck, Kimberly Tee & Gregor McEwan. *Supporting informal collaboration in shared-workspace groupware.* Journal of Universal Computer Science, vol. 14, no. 9, pages 1411–1434, 2008.

[Guzdial 00] Mark Guzdial, Jochen Rick & Bolot Kerimbaev. *Recognizing and supporting roles in CSCW.* In Proceedings of the 2000 ACM conference on Computer supported cooperative work, CSCW '00, pages 261–268. ACM, 2000.

[Haesen 11a] Mieke Haesen. *User-centered process framework and techniques to support the realization of interactive systems by multi-disciplinary teams.* PhD thesis, Hasselt University, Diepenbeek, Belgium, December 2011.

[Haesen 11b] Mieke Haesen, Jan Van den Bergh, Jan Meskens, Kris Luyten, Sylvain Degrandsart, Serge Demeyer & Karin Coninx. *Using storyboards to integrate models and informal design knowledge.* In MDDAUI '11, pages 87–106. Springer, 2011.

[Haller 05] Michael Haller, Mark Billinghurst, Daniel Leithinger, Jakob Leitner & Thomas Seifried. *Coeno: enhancing face-to-face collaboration.* In Proceedings of the 2005 international conference on Augmented tele-existence, ICAT '05, pages 40–47. ACM, 2005.

[Han 05] Jefferson Y. Han. *Low-cost multi-touch sensing through frustrated total internal reflection.* In Proceedings of the 18th ACM symposium on User interface software and technology, UIST '05, pages 115–118. ACM, 2005.

[Harada 96] Komei Harada, Eiichiro Tanaka, Ryuichi Ogawa & Yoshinori Hara. *Anecdote: a multimedia storyboarding system with seamless authoring support.* In Proceedings of the fourth ACM international conference on Multimedia, MULTIMEDIA '96, pages 341–351. ACM, 1996.

[Harel 87] David Harel. *Statecharts: a visual formalism for complex systems.* Science of Computer Programming, vol. 8, no. 3, pages 321–274, 1987.

[Harrison 95] Susan M. Harrison. *A comparison of still, animated, or nonillustrated on-line help with written or spoken instructions in a graphical user interface.* In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '95, pages 82–89. ACM, 1995.

[Harrison 11a] Chris Harrison, Hrvoje Benko & Andrew D. Wilson. *OmniTouch: wearable multitouch interaction everywhere.* In Proceedings of the 24th ACM symposium on User interface software and technology, UIST '11, pages 441–450. ACM, 2011.

[Harrison 11b] Chris Harrison, Julia Schwarz & Scott E. Hudson. *TapSense: enhancing finger interaction on touch surfaces.* In Proceedings of the 24th ACM symposium on User interface software and technology, UIST '11, pages 627–636. ACM, 2011.

[Holz 10] Christian Holz & Patrick Baudisch. *The generalized perceived input point model and how to double touch accuracy by extracting fingerprints.* In Proceedings of the 28th international conference on Human factors in computing systems, CHI '10, pages 581–590. ACM, 2010.

[Hornecker 08a] Eva Hornecker. *"I dont understand it either, but it is cool" - visitor interactions with a multi-touch table in a museum.* In Proceedings of the 3rd IEEE international

workshop on Horizontal Interactive Human Computer Systems, TABLETOP '08, pages 113–120. IEEE Computer Society, 2008.

[Hornecker 08b] Eva Hornecker, Paul Marshall, Nick S. Dalton & Yvonne Rogers. *Collaboration and interference: awareness with mice or touch input.* In Proceedings of the 2008 ACM conference on Computer supported cooperative work, CSCW '08, pages 167–176. ACM, 2008.

[Huot 04] Stéphane Huot, Cédric Dumas, Pierre Dragicevic, Jean-Daniel Fekete & Gérard Hégron. *The MaggLite post-WIMP toolkit: draw it, connect it and run it.* In Proceedings of the 17th ACM Symposium on User interface software and technologies, UIST '04, pages 257–266. ACM, 2004.

[Int 10] International Standards Organization. *ISO 9241-210. Ergonomics of human-system interaction - Part 210: Human-centred design for interactive systems*, 2010.

[Ivory 01] Melody Y. Ivory & Marti A. Hearst. *The state of the art in automating usability evaluation of user interfaces.* ACM Computing Surveys, vol. 33, pages 470–516, 2001.

[Izadi 03] Shahram Izadi, Harry Brignull, Tom Rodden, Yvonne Rogers & Mia Underwood. *Dynamo: a public interactive surface supporting the cooperative sharing and exchange of media.* In Proceedings of the 16th ACM symposium on User interface software and technology, UIST '03, pages 159–168. ACM, 2003.

[Jacucci 10] Giulio Jacucci, Ann Morrison, Gabriela T. Richard, Jari Kleimola, Peter Peltonen, Lorenza Parisi & Toni Laitinen. *Worlds of information: designing for engagement at a public multi-touch display.* In Proceedings of the 28th international conference on Human factors in computing systems, CHI '10, pages 2267–2276. ACM, 2010.

[Jensen 94] Kurt Jensen. *An introduction to the theoretical aspects of coloured Petri Nets.* In A decade of concurrency, reflection and perspectives, pages 230–272. Springer, 1994.

[Jiang 08] Hao Jiang, Daniel Wigdor, Clifton Forlines & Chia Shen. *System design for the WeSpace: linking personal devices to a table-centered multi-user, multi-surface environment.* In Proceedings of the 3rd IEEE international workshop on Horizontal Interactive Human Computer Systems, TABLETOP '08, pages 105–112. IEEE Computer Society, 2008.

[Johanson 02] Brad Johanson, Armando Fox & Terry Winograd. *The Interactive Workspaces Project: experiences with ubiquitous computing rooms.* IEEE Pervasive Computing, vol. 1, no. 2, pages 67–74, 2002.

[Johnson 65] E.A. Johnson. *Touch display - a novel input/output device for computers.* Electronics Letters, vol. 1, no. 8, pages 219–220, 1965.

[Joshi 05] James B.D. Joshi, Elisa Bertino, Usman Latif & Arif Ghafoor. *A generalized temporal role-based access control model.* IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 1, pages 4–23, 2005.

[Kaltenbrunner 05] Martin Kaltenbrunner, Till Bovermann, Ross Bencina & Enrico Costanza. *TUIO - a protocol for table-top tangible user interfaces.* In Proceedings of the 6th international workshop on Gesture-based human-computer interaction and simulation, GW '05, 2005.

[Kammer 10] Dietrich Kammer, Jan Wojdziak, Mandy Keck, Rainer Groh & Severin Taranko. *Towards a formalization of multi-touch gestures.* In Proceedings of the ACM international conference on Interactive Tabletops and Surfaces, ITS '10, pages 49–58. ACM, 2010.

[Kane 11] Shaun K. Kane, Meredith Ringel Morris, Annuska Z. Perkins, Daniel Wigdor, Richard E. Ladner & Jacob O. Wobbrock. *Access overlays: improving non-visual access to large touch screens for blind users.* In Proceedings of the 24th ACM symposium on User interface software and technology, UIST '11, pages 273–282. ACM, 2011.

[Kang 03] Hyunmo Kang, Catherine Plaisant & Ben Shneiderman. *New approaches to help users get started with visual interfaces: multi-layered interfaces and integrated initial guidance.* In Proceedings of the 2003 national conference on Digital government research, dg.o '03, pages 1–6. Digital Government Society of North America, 2003.

[Kaviani 09] Nima Kaviani, Matthias Finke & Rodger Lea. *Encouraging crowd interaction with large displays using handheld devices.* In Crowd computing interactions workshop at the SIGCHI conference on Human factors in computing systems, CHI '09. ACM, 2009.

[Kelleher 05] Caitlin Kelleher & Randy Pausch. *Stencils-based tutorials: design and evaluation.* In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '05, pages 541–550. ACM, 2005.

[Khandkar 10] Shahedul Huq Khandkar & Frank Maurer. *A language to define multi-touch interactions.* In Proceedings of the ACM international conference on Interactive Tabletops and Surfaces, ITS '10, pages 269–270. ACM, 2010.

[Kim 09] Kyung Tae Kim, Tejas Kulkarni & Niklas Elmqvist. *Interaction workspaces: identity tracking for multi-user collaboration on camera-based multi-touch tabletops.* In Proceedings of the workshop on Collaborative Visualization on Interactive Surfaces, CoVIS '09, 2009.

[Kin 12a] Kenrick Kin, Björn Hartmann, Tony DeRose & Maneesh Agrawala. *Proton++: a customizable declarative multitouch framework.* In Proceedings of the 25th ACM symposium on User interface software and technology, UIST '12, pages 477–486. ACM, 2012.

[Kin 12b] Kenrick Kin, Björn Hartmann, Tony DeRose & Maneesh Agrawala. *Proton: multitouch gestures as regular expressions.* In Proceedings of the 2012 ACM conference on Human Factors in Computing Systems, CHI '12, pages 2885–2894. ACM, 2012.

[Knabe 95] Kevin Knabe. *Apple guide: a case study in user-aided design of online help.* In Conference companion on Human factors in computing systems, CHI '95, pages 286–287. ACM, 1995.

[Koike 01] Hideki Koike, Yoichi Sato & Yoshinori Kobayashi. *Integrating paper and digital information on EnhancedDesk: a method for realtime finger tracking on an augmented desk system.* ACM Transactions on Computer-Human Interaction, vol. 8, no. 4, pages 307–322, 2001.

[König 10] Werner König, Roman Rädle & Harald Reiterer. *Interactive design of multimodal user interfaces.* Journal on Multimodal User Interfaces, vol. 3, pages 197–213, 2010.

[Kruger 03] Russell Kruger, Sheelagh Carpendale, Stacey D. Scott & Saul Greenberg. *How people use orientation on tables: comprehension, coordination and communication.* In Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work, GROUP '03, pages 369–378. ACM, 2003.

[Krull 01] Robert Krull, Janet Friauf, Johel Brown-Grant & Angela Eaton. *Usability trends in an online help system: user testing on three releases of help for a visual programming language.* In Proceedings of IEEE International Professional Communication Conference, IPCC '01, pages 19–26. IEEE Computer Society, 2001.

[Kurtenbach 94] Gordon Kurtenbach, Thomas P. Moran & William Buxton. *Contextual animation of gestural commands.* Computer Graphics Forum, vol. 13, no. 5, pages 305–314, 1994.

[Landay 95] James A. Landay & Brad A. Myers. *Interactive sketching for the early stages of user interface design.* In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '95, pages 43–50. ACM/Addison-Wesley Publishing Co., 1995.

[Landay 96] James A. Landay & Brad A. Myers. *Sketching storyboards to illustrate interface behaviors.* In Conference

companion on Human factors in computing systems, CHI '96, pages 193–194. ACM, 1996.

[Lawson 09] Jean-Yves Lionel Lawson, Ahmad-Amr Al-Akkad, Jean Vanderdonckt & Benoit Macq. *An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components.* In Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems, EICS '09, pages 245–254. ACM, 2009.

[Ledo 12] David Ledo, Miguel A. Nacenta, Nicolai Marquardt, Sebastian Boring & Saul Greenberg. *The HapticTouch toolkit: enabling exploration of haptic interactions.* In Proceedings of the 6th international conference on Tangible, embedded and embodied interaction, TEI '12, pages 115–122. ACM, 2012.

[Leland 88] Mary D.P. Leland, Robert S. Fish & Robert E. Kraut. *Collaborative document production using quilt.* In Proceedings of the 1988 ACM conference on Computer-supported cooperative work, CSCW '88, pages 206–215. ACM, 1988.

[Lemmelä 08] Saija Lemmelä, Akos Vetek, Kaj Mäkelä & Dari Trendafilov. *Designing and evaluating multimodal interaction for mobile contexts.* In Proceedings of the 10th international conference on Multimodal interfaces, ICMI '08, pages 265–272. ACM, 2008.

[Lerusalimschy 96] Robert Lerusalimschy, Luiz Henrique de Figueiredo & Waldemar Celes Filho. *Lua - an extensible extension language.* Software: Practice and Experience, vol. 26, pages 635–652, 1996.

[Lin 00] James Lin, Mark W. Newman, Jason I. Hong & James A. Landay. *DENIM: finding a tighter fit between tools and practice for Web site design.* In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '00, pages 510–517. ACM, 2000.

[Lowe 99] David G. Lowe. *Object recognition from local scale-invariant features.* In Proceedings of the 1999 international conference on Computer vision, ICCV '99, pages 1150–1157. IEEE Computer Society, 1999.

[Lucas 81] Bruce D. Lucas & Takeo Kanade. *An iterative image registration technique with an application to stereo vision.* In Proceedings of the 7th international joint conference on Artificial intelligence, IJCAI '81, pages 674–679. Morgan Kaufmann Publishers Inc., 1981.

[Lupu 97] Emil C. Lupu & Morris Sloman. *Towards a role-based framework for distributed systems management.* Journal of Network and Systems Management, vol. 5, no. 1, pages 5–30, 1997.

[Luyten 10] Kris Luyten, Davy Vanacken, Malte Weiss, Jan Borchers, Shahram Izadi & Daniel Wigdor. *Engineering patterns for multi-touch interfaces.* In Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems, EICS '10, pages 365–366. ACM, 2010.

[Luyten 11] Kris Luyten, Davy Vanacken, Malte Weiss, Jan Borchers & Miguel Nacenta. *Second workshop on engineering patterns for multi-touch interfaces.* In Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems, EICS '11, pages 335–336. ACM, 2011.

[Lynch 11] Sean Lynch, Miguel A. Nacenta & Sheelagh Carpendale. *ToCoPlay: graphical multi-touch interaction for composing and playing music.* In Proceedings of the 13th IFIP TC13 international conference on Human-computer interaction, INTERACT '11, pages 306–322. Springer, 2011.

[Malik 04] Shahzad Malik & Joe Laszlo. *Visual touchpad: a two-handed gestural input device.* In Proceedings of the 6th international conference on Multimodal interfaces, ICMI '04, pages 289–296. ACM, 2004.

[Marquardt 09] Nicolai Marquardt, Miguel A. Nacenta, James E. Young, Sheelagh Carpendale, Saul Greenberg & Ehud Sharlin. *The Haptic Tabletop Puck: tactile feedback for interactive tabletops.* In Proceedings of the ACM international conference on Interactive Tabletops and Surfaces, ITS '09, pages 85–92. ACM, 2009.

[Marquardt 11] Nicolai Marquardt, Johannes Kiemer, David Ledo, Sebastian Boring & Saul Greenberg. *Designing user-, hand-, and handpart-aware tabletop interactions with the TouchID toolkit.* In Proceedings of the 2011 ACM international conference on Interactive Tabletops and Surfaces, ITS '11, pages 21–30. ACM, 2011.

[Marshall 11] Paul Marshall, Richard Morris, Yvonne Rogers, Stefan Kreitmayer & Matt Davies. *Rethinking 'multi-user': an in-the-wild study of how groups approach a walk-up-and-use tabletop interface.* In Proceedings of the 2011 conference on Human factors in computing systems, CHI '11, pages 3033–3042. ACM, 2011.

[Martin 05] Andrew P. Martin, Melody Y. Ivory, Rodrick Megraw & Beverly Slabosky. *Exploring the persistent problem of user assistance.* Rapport technique IS-TR-2005-08-01, Information School, University of Washington, 2005.

[Matejka 11] Justin Matejka, Tovi Grossman & George Fitzmaurice. *Ambient help.* In Proceedings of the 2011 conference on Human factors in computing systems, CHI '11, pages 2751–2760. ACM, 2011.

[McGee-Lennon 09] Marilyn Rose McGee-Lennon, Andrew Ramsay, David McGookin & Philip Gray. *User evaluation of OIDE: a rapid prototyping platform for multimodal interaction.* In Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems, EICS '09, pages 237–242. ACM, 2009.

[Meixner 11] Gerrit Meixner, Fabio Paternò & Jean Vanderdonckt. *Past, Present, and Future of Model-Based User Interface Development.* i-com, vol. 10, no. 3, pages 2–11, 2011.

[Meyer 10] Tobias Meyer & Dominik Schmidt. *IdWristbands: IR-based user identification on multi-touch surfaces.* In Proceedings of the 2010 ACM international conference on Interactive Tabletops and Surfaces, ITS '10, pages 277–278. ACM, 2010.

[Mistry 09] Pranav Mistry & Pattie Maes. *SixthSense: a wearable gestural interface.* In Proceedings of ACM SIGGRAPH Asia 2009 sketches, pages 11:1–11:1. ACM, 2009.

[Möllers 11] Max Möllers & Jan Borchers. *TaPS widgets: interacting with tangible private spaces.* In Proceedings of the ACM international conference on Interactive Tabletops and Surfaces, ITS '11, pages 75–78. ACM, 2011.

[Mori 02] Giulio Mori, Fabio Paternò & Carmen Santoro. *CTTE: support for developing and analyzing task models for interactive system design.* IEEE Transactions on Software Engineering, vol. 28, no. 8, pages 797–813, 2002.

[Morris 04] Meredith Ringel Morris, Kathy Ryall, Chia Shen, Clifton Forlines & Frédéric D. Vernier. *Beyond "social protocols": multi-user coordination policies for co-located groupware.* In Proceedings of the 2004 ACM conference on Computer supported cooperative work, CSCW '04, pages 262–265. ACM, 2004.

[Morris 06] Meredith Ringel Morris, Anqi Huang, Andreas Paepcke & Terry Winograd. *Cooperative gestures: multi-user gestural interactions for co-located groupware.* In Proceedings of the 24th SIGCHI conference on Human factors in computing systems, CHI '06, pages 1201–1210. ACM, 2006.

[Morris 08] Meredith Ringel Morris. *A survey of collaborative web search practices.* In Proceedings of the 26th SIGCHI conference on Human factors in computing systems, CHI '08, pages 1657–1660. ACM, 2008.

[Müller-Tomfelde 10] Christian Müller-Tomfelde. Tabletops - horizontal interactive displays. Springer, 2010.

[Müller 12] Jörg Müller, Robert Walter, Gilles Bailly, Michael Nischt & Florian Alt. *Looking glass: a field study on noticing interactivity of a shop window.* In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '12, pages 297–306. ACM, 2012.

[Myers 06] Brad A. Myers, David A. Weitzman, Andrew J. Ko & Duen Horng Chau. *Answering why and why not questions in user interfaces.* In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '06, pages 397–406. ACM, 2006.

[Nacenta 05] Miguel A. Nacenta, Dzmitry Aliakseyeu, Sriram Subramanian & Carl Gutwin. *A comparison of techniques for multi-display reaching.* In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '05, pages 371–380. ACM, 2005.

[Nacenta 07] Miguel A. Nacenta, David Pinelle, Dane Stuckel & Carl Gutwin. *The effects of interaction technique on coordination in tabletop groupware.* In Proceedings of Graphics Interface 2007, GI '07, pages 191–198. ACM, 2007.

[Nacenta 09] Miguel A. Nacenta, Patrick Baudisch, Hrvoje Benko & Andy Wilson. *Separability of spatial manipulations in multi-touch interfaces.* In Proceedings of Graphics Interface 2009, GI '09, pages 175–182. Canadian Information Processing Society, 2009.

[Nacenta 10] Miguel A. Nacenta, David Pinelle, Carl Gutwin & Regan Mandryk. *Individual and group support in tabletop interaction techniques.* In Proceedings of Tabletops - Horizontal Interactive Displays, pages 303–333. Springer, 2010.

[Napier 93] John Napier & Russell H. Tuttle. Hands. Princeton University Press, 1993.

[Navarre 09] David Navarre, Philippe Palanque, Jean-Francois Ladry & Eric Barboni. *ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability.* ACM

Transactions on Computer-Human Interaction, vol. 16, no. 4, pages 18:1–18:56, 2009.

[Nielsen 10] Jakob Nielsen. *iPad usability: first findings from user testing.* http://www.useit.com/alertbox/ipad-1st-study.html, 2010.

[Nielsen 11] Jakob Nielsen. *iPad usability: year one.* http://www.useit.com/alertbox/ipad.html, 2011.

[Nigay 93] Laurence Nigay & Joëlle Coutaz. *A design space for multimodal systems: concurrent processing and data fusion.* In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '93, pages 172–178. ACM, 1993.

[Nigay 95] Laurence Nigay & Joëlle Coutaz. *A generic platform for addressing the multimodal challenge.* In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '95, pages 98–105. ACM, 1995.

[Norman 88] Donald A. Norman. The design of everyday things. Doubleday, 1988.

[Norman 10a] Donald A. Norman. *Gestural interfaces: a step backwards in usability.* Interactions, vol. 17, no. 5, pages 46–49, 2010.

[Norman 10b] Donald A. Norman. *Natural user interfaces are not natural.* Interactions, vol. 17, no. 3, pages 6–10, 2010.

[Novick 06] David G. Novick & Karen Ward. *What users say they want in documentation.* In Proceedings of the 24th ACM international conference on Design of communication, SIGDOC '06, pages 84–91. ACM, 2006.

[Octavia 09] Johanna Renny Octavia, Lode Vanacken, Chris Raymaekers, Karin Coninx & Eddy Flerackers. *Facilitating adaptation in virtual environments using a context-aware model-based design process.* In Proceedings of the 8th international workshop on Task Models and Diagrams for User Interface Design, TAMODIA '09, pages 58–71. Springer, 2009.

[Ojala 96] Timo Ojala, Matti Pietikäinen & David Harwood. *A comparative study of texture measures with classification based on featured distributions.* Pattern Recognition, vol. 29, no. l, pages 51–59, 1996.

[Ojala 02] Timo Ojala, Matti Pietikäinen & Topi Mäenpää. *Multiresolution gray-scale and rotation invariant texture classification with local binary patterns.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pages 971–987, 2002.

[Orr 68] N.W. Orr & V.D. Hopkins. *The role of touch display in air traffic control.* The Controller, vol. 7, pages 7–9, 1968.

[Orr 00] Robert J. Orr & Gregory D. Abowd. *The Smart Floor: a mechanism for natural user identification and tracking.* In CHI '00 extended abstracts on Human factors in computing systems, pages 275–276. ACM, 2000.

[Osborn 00] Sylvia Osborn, Ravi Sandhu & Qamar Munawer. *Configuring role-based access control to enforce mandatory and discretionary access control policies.* ACM Transactions on Information and System Security, vol. 3, no. 2, pages 85–106, 2000.

[Palanque 94] Philippe Palanque & Remi Bastide. *Petri net based design of user-driven interfaces using the Interactive Cooperative Objects formalism.* In Proceedings of Interactive Systems: Design, Specification, and Verification, pages 383–400. Springer, 1994.

[Palanque 11] Philippe Palanque, Eric Barboni, Célia Martinie, David Navarre & Marco Winckler. *A model-based approach for supporting engineering usability evaluation of interaction techniques.* In Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems, EICS '11, pages 21–30. ACM, 2011.

[Palmiter 93] Susan Palmiter & Jay Elkerton. *Animated demonstrations for learning procedural computer-based tasks.*

Human-Computer Interaction, vol. 8, no. 3, pages 193–216, 1993.

[Panëels 10] Sabrina A. Panëels, Jonathan C. Roberts & Peter J. Rodgers. *HITPROTO: a tool for the rapid prototyping of haptic interactions for haptic data visualization.* In Proceedings of the 2010 IEEE Haptics Symposium, HAPTIC '10, pages 261–268. IEEE Computer Society, 2010.

[Park 01] Joon S. Park, Ravi Sandhu & Gail-Joon Ahn. *Role-based access control on the Web.* ACM Transactions on Information and System Security, vol. 4, no. 1, pages 37–71, 2001.

[Paternò 00] Fabio Paternò. Model-based design and evaluation of interactive applications. Springer, 2000.

[Paternò 09] Fabio Paternò, Carmen Santoro & Lucio Davide Spano. *MARIA: a universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments.* ACM Transactions on Computer-Human Interaction, vol. 16, no. 4, pages 19:1–19:30, 2009.

[Peltonen 08] Peter Peltonen, Esko Kurvinen, Antti Salovaara, Giulio Jacucci, Tommi Ilmonen, John Evans, Antti Oulasvirta & Petri Saarikko. *It's mine, don't touch!: interactions at a large multi-touch display in a city centre.* In Proceedings of the 26th SIGCHI conference on Human factors in computing systems, CHI '08, pages 1285–1294. ACM, 2008.

[Pering 10] Trevor Pering, Kent Lyons, Roy Want, Mary Murphy-Hoye, Mark Baloga, Paul Noll, Joe Branc & Nicolas De Benoist. *What do you bring to the table?: investigations of a collaborative workspace.* In Proceedings of the 12th ACM international conference on Ubiquitous computing, Ubicomp '10, pages 183–192. ACM, 2010.

[Petri 62] Carl A. Petri. *Fundamentals of a theory of asynchronous*

information flow. In IFIP Congress, pages 386–390, 1962.

[Phung 02] Son Lam Phung, Abdesselam Bouzerdoum & Douglas Chai. *A novel skin color model in YCbCr color space and its application to human face detection.* In Proceedings of the 2002 IEEE international conference on Image processing, ICIP '02, pages 289–292. IEEE Computer Society, 2002.

[Phung 05] Son Lam Phung, Abdesselam Bouzerdoum & Douglas Chai. *Skin segmentation using color pixel classification: analysis and comparison.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 1, pages 148–154, 2005.

[Pinelle 06] David Pinelle, Carl Gutwin & Sriram Subramanian. *Designing digital tables for highly integrated collaboration.* Rapport technique HCI-TR-06-02, University of Saskatchewan, 2006.

[Pinelle 08] David Pinelle, Miguel Nacenta, Carl Gutwin & Tadeusz Stach. *The effects of co-present embodiments on awareness and collaboration in tabletop groupware.* In Proceedings of graphics interface 2008, GI '08, pages 1–8. Canadian Information Processing Society, 2008.

[Pinelle 09] David Pinelle, Mutasem Barjawi, Miguel Nacenta & Regan Mandryk. *An evaluation of coordination techniques for protecting objects and territories in tabletop groupware.* In Proceedings of the 27th international conference on Human factors in computing systems, CHI '09, pages 2129–2138. ACM, 2009.

[Piper 06] Anne Marie Piper, Eileen O'Brien, Meredith Ringel Morris & Terry Winograd. *SIDES: a cooperative tabletop computer game for social skills development.* In Proceedings of the 2006 ACM conference on Computer supported cooperative work, CSCW '06, pages 1–10. ACM, 2006.

[Ramakers 12] Raf Ramakers, Davy Vanacken, Kris Luyten, Karin Coninx & Johannes Schöning. *Carpus: a non-intrusive user identification technique for interactive surfaces.* In Proceedings of the 25th ACM symposium on User interface software and technology, UIST '12, pages 35–44. ACM, 2012.

[Raymaekers 01] Chris Raymaekers & Karin Coninx. *Menu interactions in a desktop haptic environment.* In Proceedings of Eurohaptics 2001, pages 49–53, 2001.

[Reetz 06] Adrian Reetz, Carl Gutwin, Tadeusz Stach, Miguel Nacenta & Sriram Subramanian. *Superflick: a natural and efficient technique for long-distance object placement on digital tables.* In Proceedings of Graphics Interface 2006, GI '06, pages 163–170. Canadian Information Processing Society, 2006.

[Rekimoto 98] Jun Rekimoto. *A multiple device approach for supporting whiteboard-based interactions.* In Proceedings of the SIGCHI Conference on Human factors in computing systems, CHI '98, pages 344–351. ACM Press/Addison-Wesley Publishing Co., 1998.

[Remy 10] Christian Remy, Malte Weiss, Martina Ziefle & Jan Borchers. *A pattern language for interactive tabletops in collaborative workspaces.* In Proceedings of the 15th European conference on Pattern languages of programs, EuroPLoP '10. ACM, 2010.

[Richter 12] Stephan Richter, Christian Holz & Patrick Baudisch. *Bootstrapper: recognizing tabletop users by their shoes.* In Proceedings of the 30th conference on Human factors in computing systems, CHI '12, pages 1249–1252. ACM, 2012.

[Ringel 04] Meredith Ringel, Kathy Ryall, Chia Shen, Clifton Forlines & Frédéric D. Vernier. *Release, relocate, reorient, resize: fluid techniques for document sharing on multi-user interactive tables.* In CHI '04 extended abstracts on Human factors in computing systems, pages 1441–1444. ACM, 2004.

[Rogers 04a] Yvonne Rogers, William Hazlewood, Eli Blevis & Youn-Kyung Lim. *Finger talk: collaborative decision-making using talk and fingertip interaction around a tabletop display.* In CHI '04 extended abstracts on Human factors in computing systems, pages 1271–1274. ACM, 2004.

[Rogers 04b] Yvonne Rogers & Siân Lindley. *Collaborating around vertical and horizontal displays: which way is best?* Interacting With Computers, vol. 16, no. 6, pages 1133–1152, 2004.

[Rogers 09] Yvonne Rogers, Youn-Kyung Lim, William R. Hazlewood & Paul Marshall. *Equal opportunities: do shareable interfaces promote more group participation than single user displays?* Human-Computer Interaction, vol. 24, no. 1-2, pages 79–116, 2009.

[Roth 10] Volker Roth, Philipp Schmidt & Benjamin Güldenring. *The IR ring: authenticating users' touches on a multi-touch display.* In Proceedings of the 23nd ACM symposium on User interface software and technology, UIST '10, pages 259–262. ACM, 2010.

[Ryall 04] Kathy Ryall, Clifton Forlines, Chia Shen & Meredith Ringel Morris. *Exploring the effects of group size and table size on interactions with tabletop shared-display groupware.* In Proceedings of the 2004 ACM conference on Computer supported cooperative work, CSCW '04, pages 284–293. ACM, 2004.

[Ryall 05] Kathy Ryall, Alan Esenther, Katherine Everitt, Clifton Forlines, Meredith Ringel Morris, Chia Shen, Sam Shipman & Frédéric D. Vernier. *iDwidgets: parameterizing widgets by user identity.* In Proceedings of the 10th IFIP TC13 international conference on Human-Computer Interaction, 2005.

[Ryall 06a] Kathy Ryall, Alan Esenther, Clifton Forlines, Chia Shen, Sam Shipman, Meredith Ringel Morris, Katherine Everitt & Frédéric D. Vernier. *Identity-differentiating*

widgets for multiuser interactive surfaces. IEEE Computer Graphics and Applications, vol. 26, no. 5, pages 56–64, 2006.

[Ryall 06b] Kathy Ryall, Meredith Ringel Morris, Katherine Everitt, Clifton Forlines & Chia Shen. *Experiences with and observations of direct-touch tabletops.* In Proceedings of IEEE international workshop on Horizontal Interactive Human Computer Systems, TABLETOP '06, pages 89–96. IEEE Computer Society, 2006.

[Sanders 10] Elizabeth B.-N. Sanders, Eva Brandt & Thomas Binder. *A framework for organizing the tools and techniques of participatory design.* In Proceedings of the 11th Participatory design conference, PDC '10, pages 195–198. ACM, 2010.

[Sandhu 96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein & Charles E. Youman. *Role-based access control models.* IEEE Computer, vol. 29, no. 2, pages 38–47, 1996.

[Sandhu 99] Ravi S. Sandhu, Venkata Bhamidipati & Qamar Munawer. *The ARBAC97 model for role-based administration of roles.* ACM Transactions on Information and System Security, vol. 2, no. 1, pages 105–135, 1999.

[Schmidt 10a] Dominik Schmidt, Ming Ki Chong & Hans Gellersen. *HandsDown: hand-contour-based user identification for interactive surfaces.* In Proceedings of the 6th Nordic conference on Human-Computer Interaction, NordiCHI '10, pages 432–441. ACM, 2010.

[Schmidt 10b] Dominik Schmidt, Ming Ki Chong & Hans Gellersen. *IdLenses: dynamic personal areas on shared surfaces.* In Proceedings of the 2010 ACM international conference on Interactive Tabletops and Surfaces, ITS '10, pages 131–134. ACM, 2010.

[Schmidt 10c] Sebastian Schmidt, Miguel A. Nacenta, Raimund Dachselt & Sheelagh Carpendale. *A set of multi-touch graph interaction techniques.* In Proceedings of the ACM

international conference on Interactive Tabletops and Surfaces, ITS '10, pages 113–116. ACM, 2010.

[Schneider 10] Jan Schneider, Jan Derboven, Kris Luyten, Chris Vleugels, Stijn Bannier, Dries De Roeck & Mathijs Verstraete. *Multi-user multi-touch setups for collaborative learning in an educational setting.* In Proceedings of the 7th international conference on Cooperative design, visualization, and engineering, CDVE'10, pages 181–188. Springer, 2010.

[Scholliers 11] Christophe Scholliers, Lode Hoste, Beat Signer & Wolfgang De Meuter. *Midas: a declarative multi-touch interaction framework.* In Proceedings of the 5th international conference on Tangible, embedded, and embodied interaction, TEI '11, pages 49–56. ACM, 2011.

[Schöning 08] Johannes Schöning, Michael Rohs & Antonio Krüger. *Using mobile phones to spontaneously authenticate and interact with multi-touch surfaces.* PPD '08 workshop on designing multi-touch interaction techniques for coupled public and private displays, 2008.

[Schöning 09] Johannes Schöning. *Do we need further multi-touch affordances?* In Touch affordances workshop at the IFIP international conference on Human-computer interaction, INTERACT '09, 2009.

[Schöning 10] Johannes Schöning, Markus Löchtefeld, Michael Rohs & Antonio Krüger. *Projector phones: a new class of interfaces for augmented reality.* International Journal of Mobile Human Computer Interaction, vol. 2, no. 3, pages 1–14, 2010.

[Schwan 04] Stephan Schwan & Roland Riempp. *The cognitive benefits of interactive videos: learning to tie nautical knots.* Learning and Instruction, vol. 14, no. 3, pages 293–305, 2004.

[Scott 03] Stacey D. Scott, Karen D. Grant & Regan L. Mandryk. *System guidelines for co-located collaborative work on a tabletop display.* In Proceedings of the 2003 European

conference on Computer-supported cooperative work, ECSCW '03, pages 159–178. Springer, 2003.

[Scott 04] Stacey D. Scott, Sheelagh Carpendale & Kori M. Inkpen. *Territoriality in collaborative tabletop workspaces.* In Proceedings of the 2004 ACM conference on Computer supported cooperative work, CSCW '04, pages 294–303. ACM, 2004.

[Segen 98] Jakub Segen & Senthil Kumar. *Gesture VR: vision-based 3D hand interace for spatial interaction.* In Proceedings of the 6th ACM international conference on Multimedia, MULTIMEDIA '98, pages 455–464. ACM, 1998.

[Shen 03] Chia Shen, Katherine Everitt & Kathleen Ryall. *UbiTable: impromptu face-to-face collaboration on horizontal interactive surfaces.* In Proceedings of the 5th international conference on Ubiquitous computing, UbiComp '03, pages 281–288. Springer, 2003.

[Shen 04] Chia Shen, Frédéric D. Vernier, Clifton Forlines & Meredith Ringel. *DiamondSpin: an extensible toolkit for around-the-table interaction.* In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '04, pages 167–174. ACM, 2004.

[Shen 06] Chia Shen, Kathy Ryall, Clifton Forlines, Alan Esenther, Frédéric D. Vernier, Katherine Everitt, Mike Wu, Daniel Wigdor, Meredith Ringel Morris, Mark Hancock & Edward Tse. *Informing the design of direct-touch tabletops.* IEEE Computer Graphics and Applications, vol. 26, no. 5, pages 36–46, 2006.

[Shneiderman 91] Ben Shneiderman. *Touch screens now offer compelling uses.* IEEE Software, vol. 8, no. 2, pages 93–94, 1991.

[Shoemaker 01] Garth B.D. Shoemaker & Kori M. Inkpen. *Single display privacyware: augmenting public displays with private information.* In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '01, pages 522–529. ACM, 2001.

[Simone 95] Carla Simone, Monica Divitini & Kjeld Schmidt. *A notation for malleable and interoperable coordination mechanisms for CSCW systems.* In Proceedings of conference on Organizational computing systems, COCS '95, pages 44–54. ACM, 1995.

[Smith 98] Hixon Ranald Smith Randall B. & Bernard Horan. *Supporting flexible roles in a shared workspace.* In Proceedings of the 1998 ACM conference on Computer supported cooperative work, CSCW '98, pages 197–206. ACM, 1998.

[Spano 11] Lucio Davide Spano. *A model-based approach for gesture interfaces.* In Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems, EICS '11, pages 327–330. ACM, 2011.

[Spano 12] Lucio Spano, Antonio Cisternino & Fabio Paternò. *A compositional model for gesture definition.* In Proceedings of the 4th international conference on Human-Centered Software Engineering, HCSE '12, pages 34–52. Springer, 2012.

[Streitz 94] Norbert A. Streitz, Jörg Geißler, Jörg M. Haake & Jeroen Hol. *DOLPHIN: integrated meeting support across local and remote desktop environments and liveboards.* In Proceedings of the 1994 ACM conference on Computer supported cooperative work, CSCW '94, pages 345–358. ACM, 1994.

[Streitz 99] Norbert A. Streitz, Jörg Geißler, Torsten Holmer, Shin'ichi Konomi, Christian Müller-Tomfelde, Wolfgang Reischl, Petra Rexroth, Peter Seitz & Ralf Steinmetz. *i-LAND: an interactive landscape for creativity and innovation.* In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '99, pages 120–127. ACM, 1999.

[Sukaviriya 90] Piyawadee Sukaviriya & James D. Foley. *Coupling a UI framework with automatic generation of context-sensitive animated help.* In Proceedings of the 3rd ACM

symposium on User interface software and technology, UIST '90, pages 152–166. ACM, 1990.

[Swain 91] Michael J. Swain & Dana H. Ballard. *Color indexing.* International Journal of Computer Vision, vol. 7, no. 1, pages 11–32, 1991.

[Tang 06] Anthony Tang, Melanie Tory, Barry Po, Petra Neumann & Sheelagh Carpendale. *Collaborative coupling over tabletop displays.* In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '06, pages 1181–1190. ACM, 2006.

[Taylor 01] Russell M. Taylor II, Thomas C. Hudson, Adam Seeger, Hans Weber, Jeffrey Juliano & Aron T. Helser. *VRPN: a device-independent, network-transparent VR peripheral system.* In Proceedings of the ACM symposium on Virtual reality software and technology, VRST '01, pages 55–61. ACM, 2001.

[Thelen 12] Sebastian Thelen, Matthias Deller, Johannes Eichner & Achim Ebert. *Enhancing large display interaction with user tracking data.* In Proceedings of the international conference on Computer Graphics and Virtual Reality, CGVR '12, 2012.

[Truong 06] Khai N. Truong, Gillian R. Hayes & Gregory D. Abowd. *Storyboarding: an empirical determination of best practices and effective guidelines.* In Proceedings of the 6th ACM conference on Designing Interactive Systems, DIS '06, pages 12–21. ACM, 2006.

[Tse 11] Edward Tse, Johannes Schöning, Jochen Huber, Lynn Marentette, Richard Beckwith, Yvonne Rogers & Max Mühlhäuser. *Child computer interaction: workshop on UI technologies and educational pedagogy.* In Proceedings of the 2011 conference extended abstracts on Human factors in computing systems, CHI EA '11, pages 2445–2448. ACM, 2011.

[Tuck 90] Robin Tuck & Dan R. Olsen. *Help by guided tasks: utilizing UIMS knowledge.* In Proceedings of the SIGCHI

conference on Human factors in computing systems, CHI '90, pages 71–78. ACM, 1990.

[Turoff 91] Murray Turoff. *Computer-mediated communication requirements for group support.* Journal of Organizational Computing, vol. 1, no. 1, pages 85–113, 1991.

[Tversky 02] Barbara Tversky, Julie B. Morrison & Mireille Bétrancourt. *Animation: can it facilitate?* International Journal of HumanComputer Studies, vol. 57, no. 4, pages 247–262, 2002.

[Valk 98] Rudiger Valk. *Petri Nets as token objects: an introduction to elementary object nets.* In Proceedings of the 19th international conference on Application and Theory of Petri Nets, ICATPN '98. Springer, 1998.

[Van Laerhoven 06] Tom Van Laerhoven, Geert Vanderhulst, Kris Luyten & Frank Van Reeth. *Device federations for a creative process in distributed interaction environments.* Rapport technique TR-UH-EDM-0606, Hasselt University – Expertise Centre for Digital Media, 2006.

[Vanacken 06] Davy Vanacken, Joan De Boeck, Chris Raymaekers & Karin Coninx. *NiMMiT: a notation for modelling multimodal interaction techniques.* In Proceedings of the first international conference on Computer graphics theory and applications, GRAPP '06, pages 224–231. INSTICC, 2006.

[Vanacken 07a] Davy Vanacken, Chris Raymaekers, Kris Luyten & Karin Coninx. *Focus+Roles: socio-organizational conflict resolution in collaborative user interfaces.* In Proceedings of the 12th international conference on Human-computer interaction, HCII '07, pages 788–796. Springer, 2007.

[Vanacken 07b] Lode Vanacken, Erwin Cuppens, Tim Clerckx & Karin Coninx. *Extending a dialog model with contextual knowledge.* In Proceedings of the 6th international workshop on Task Models and Diagrams for User Interface Design, TAMODIA '07, pages 28–41. Springer, 2007.

[Vanacken 08a] Davy Vanacken. *Interactive workspaces: multi-user, multi-touch, multi-device.* In Electronic proceedings of the 2008 ACM conference on Computer supported cooperative work (doctoral colloquium), CSCW '08. ACM, 2008.

[Vanacken 08b] Davy Vanacken, Alexandre Demeure, Kris Luyten & Karin Coninx. *Ghosts in the interface: meta-user interface visualizations as guides for multi-touch interaction.* In Proceedings of the 3rd IEEE international workshop on Horizontal Interactive Human Computer Systems, TABLETOP '08, pages 81–84. IEEE Computer Society, 2008.

[Vanacken 08c] Lode Vanacken, Joan De Boeck, Chris Raymaekers & Karin Coninx. *Designing context-aware multimodal virtual environments.* In Proceedings of the 10th international conference on Multimodal Interfaces, ICMI '08, pages 129–136. ACM, 2008.

[Vanacken 08d] Lode Vanacken, Joan De Boeck, Chris Raymaekers & Karin Coninx. *An event-condition-action approach for contextual interaction in virtual environments.* In Proceedings of the 2nd conference on Human-Centered Software Engineering and 7th international workshop on Task Models and Diagrams, HCSE-TAMODIA '08, pages 126–133. Springer, 2008.

[Vanacken 09a] Davy Vanacken, Kris Luyten & Karin Coninx. *TouchGhosts: guides for improving visibility of multi-touch interaction.* In Multitouch and Surface Computing workshop at CHI '09, the 27th international conference on Human factors in computing systems, 2009.

[Vanacken 09b] Lode Vanacken. *Multimodal selection in virtual environments: enhancing the user experience and facilitating development.* PhD thesis, Hasselt University, Diepenbeek, Belgium, June 2009.

[Voelker 11] Simon Voelker, Malte Weiss, Chat Wacharamanotham & Jan Borchers. *Dynamic portals: a lightweight*

metaphor for fast object transfer on interactive surfaces. In Proceedings of the ACM international conference on Interactive Tabletops and Surfaces, ITS '11, pages 158–161. ACM, 2011.

[Vogel 04] Daniel Vogel & Ravin Balakrishnan. *Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users.* In Proceedings of the 17th ACM symposium on User interface software and technology, UIST '04, pages 137–146. ACM, 2004.

[von Zadow 10] Ulrich von Zadow, Florian Daiber, Johannes Schöning & Antonio Krüger. *GlobalData: multi-user interaction with geographic information systems on interactive surfaces.* In Proceedings of the ACM international conference on Interactive Tabletops and Surfaces, ITS '10, pages 318–318. ACM, 2010.

[Wacharamanotham 11] Chat Wacharamanotham, Jan Hurtmanns, Alexander Mertens, Martin Kronenbuerger, Christopher Schlick & Jan Borchers. *Evaluating swabbing: a touchscreen input method for elderly users with tremor.* In Proceedings of the 2011 conference on Human factors in computing systems, CHI '11, pages 623–626. ACM, 2011.

[Wahid 10] Shahtab Wahid, Stacy M. Branham, D. Scott McCrickard & Steve Harrison. *Investigating the relationship between imagery and rationale in design.* In Proceedings of the 8th ACM conference on Designing Interactive Systems, DIS '10, pages 75–84. ACM, 2010.

[Wang 09] Feng Wang, Xiang Cao, Xiangshi Ren & Pourang Irani. *Detecting and leveraging finger orientation for interaction with direct-touch surfaces.* In Proceedings of the 22nd ACM symposium on User interface software and technology, UIST '09, pages 23–32. ACM, 2009.

[Weiss 09] Malte Weiss, Julie Wagner, Yvonne Jansen, Roger Jennings, Ramsin Khoshabeh, James D. Hollan & Jan

Borchers. *SLAP widgets: bridging the gap between virtual and physical controls on tabletops.* In Proceedings of the 27th international conference on Human factors in computing systems, CHI '09, pages 481–490. ACM, 2009.

[Weiss 10] Malte Weiss, Florian Schwarz, Simon Jakubowski & Jan Borchers. *Madgets: actuating widgets on interactive tabletops.* In Proceedings of the 23nd ACM symposium on User interface software and technology, UIST '10, pages 293–302. ACM, 2010.

[Weiss 11] Malte Weiss, Chat Wacharamanotham, Simon Voelker & Jan Borchers. *FingerFlux: near-surface haptic feedback on tabletops.* In Proceedings of the 24th ACM symposium on User interface software and technology, UIST '11, pages 615–620. ACM, 2011.

[White 07] Sean White, Levi Lister & Steven Feiner. *Visual hints for tangible gestures in augmented reality.* In Proceedings of the 2007 6th IEEE and ACM international symposium on Mixed and augmented reality, ISMAR '07, pages 1–4. IEEE Computer Society, 2007.

[Wigdor 06] Daniel Wigdor, Chia Shen, Clifton Forlines & Ravin Balakrishnan. *Table-centric interactive spaces for real-time collaboration.* In Proceedings of the working conference on Advanced visual interfaces, AVI '06, pages 103–107. ACM, 2006.

[Wigdor 09] Daniel Wigdor, Hao Jiang, Clifton Forlines, Michelle Borkin & Chia Shen. *WeSpace: the design development and deployment of a walk-up and share multi-surface visual collaboration system.* In Proceedings of the 27th international conference on Human factors in computing systems, CHI '09, pages 1237–1246. ACM, 2009.

[Wingrave 09] Chadwick A. Wingrave, Joseph J. Laviola Jr. & Doug A. Bowman. *A natural, tiered and executable UIDL for 3D user interfaces based on concept-oriented design.* ACM Transactions on Computer-Human Interaction, vol. 16, no. 4, pages 21:1–21:36, 2009.

[Wobbrock 07] Jacob O. Wobbrock, Andrew D. Wilson & Yang Li. *Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes.* In Proceedings of the 20th ACM symposium on User interface software and technology, UIST '07, pages 159–168. ACM, 2007.

[Wobbrock 09] Jacob O. Wobbrock, Meredith Ringel Morris & Andrew D. Wilson. *User-defined gestures for surface computing.* In Proceedings of the 27th international conference on Human factors in computing systems, CHI '09, pages 1083–1092. ACM, 2009.

[Wu 03] Mike Wu & Ravin Balakrishnan. *Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays.* In Proceedings of the 16th ACM symposium on User interface software and technology, UIST '03, pages 193–202. ACM, 2003.

[Xia 04] Steven Xia, David Sun, Chengzheng Sun, David Chen & Haifeng Shen. *Leveraging single-user applications for multi-user collaboration: the CoWord approach.* In Proceedings of the 2004 ACM conference on Computer supported cooperative work, CSCW '04, pages 162–171. ACM, 2004.

[Yeh 11] Tom Yeh, Tsung-Hsiang Chang, Bo Xie, Greg Walsh, Ivan Watkins, Krist Wongsuphasawat, Man Huang, Larry S. Davis & Benjamin B. Bederson. *Creating contextual help for GUIs using screenshots.* In Proceedings of the 24th ACM symposium on User interface software and technology, UIST '11, pages 145–154. ACM, 2011.

[Zhang 06] Dongsong Zhang, Lina Zhou, Robert O. Briggs & Jay F. Nunamaker Jr. *Instructional video in e-learning: assessing the impact of interactive video on learning effectiveness.* Information & Management, vol. 43, no. 1, pages 15–27, 2006.

[Zhu 06] Haibin Zhu. *Role mechanisms in collaborative systems.* International Journal of Production Research, vol. 44, no. 1, pages 181–193, 2006.