TIDE: Using Device Composition on Tabletop Computers to Extend the Smartphone Experience.

Master Thesis
by
Leo Sicard
(091181-lnsi)

Supervisor:
Professor Jakob E Bardram

March 1st, 2012
IT University of Copenhagen, Denmark

IT University of Copenhagen
Rued Langgaards Vej 7, DK-2300 København S, Denmark
Phone +45 72185000
itu@itu.dk
www.itu.dk

Abstract

Tabletop displays are now appearing in public spaces. They present a horizontal interactive surface that is ideal in social situations, and they support a touch-based experience which appeals to all users. Modern smartphones are able to support most users' daily computing tasks, and their mobility bring spontaneity to the computer interaction. However, the screen size of smartphones is a limitation when viewing graphically intense content and in situations with multiple users.

This thesis focuses on extending the smartphone experience to tabletop displays. It investigates the use of *device composition* to allow smartphone users to spontaneously annex the display space made available by tabletops. In particular, this report presents the design, implementation and evaluation of *TIDE (Tabletop Interactive Display Extension)*: a composite device that integrates a smartphone to a tabletop by way of UI replication, with focus on *spontaneous user interaction*.

The design process shows that by following a human-centered approach and using interaction techniques that are familiar to most users, the system can be designed to be easy to learn and easy to use. The implementation of TIDE demonstrates that it is feasible to build this type of system on standard hardware and many standard platforms. Moreover, TIDE supports multiple simultaneous smartphones, and uses computer vision to track the location of the devices on the display. The evaluation of TIDE, in the form of a participant-based usability study, shows that the system is highly learnable, easy to use, and useful in context.

Acknowledgements

I give my sincere thanks to:
- Professor Jakob E. Bardram, for supervising this thesis project,
- Aurélien Tabard for the good advice and continuous support throughout this process,
- Juan David Hincapié Ramos for the original idea, for advising me and giving me confidence,
- Sebastian Büttrich and the PITLab (Pervasive Interaction Technology Lab) for providing the hardware and working space,
- Mads Frost for taking and fixing pictures,
- Marie Choe Tarp for drawing,
- Ane Tarp and Jakob Borg for proofreading,
- the researchers and students associated with the PITLab for the support and helpful comments,
- and last but not least, all the people who gracefully volunteered to participate in the user experiments.

Contents

Chapter 1

Introduction

Modern smartphones are able to support most users' daily computing tasks. They fit in a pocket, which makes them ultra mobile, and they offer good storage capacities as well as all-around connectivity. This tendency implies that users have access to personal data and applications at all times. Smartphones give rise to a new type of computer interaction which is unplanned, spontaneous and on-the-go. They bring computing to situations where laptops do not fit, such as standing in a crowded train, or walking in the street. Furthermore, smartphones make it possible to get the most out of unforeseen opportunities, and in particular, they seem to be the ideal tool to support these chance meetings that suddenly turn into constructive collaboration. However, the size of the smartphone can be a limitation to improvised computer interaction, especially in situations with multiple users.

Tabletop displays, on the other hand, are ideal in social contexts. They bring computing to a simple piece of furniture, the table. As such, they present a horizontal interactive surface which multiple users can regroup around, and share a common experience, or conduct parallel activities. They have been used extensively in museums and galleries, as documented by Geller [2006], who notes that tabletops encourage a collaborative atmosphere, and provide a tactile experience that reaches even the less computer literate. As shown in figure 1.1, tabletops provide a touch-based experience that is fundamentally different from the traditional desktop metaphor. A number of HCI studies show that this experience is more natural for the non-technical user, because it removes the mouse-and-keyboard abstraction layer and generates a more direct interaction.

The first interactive tabletop displays were designed for individual use. Systems such as the DigitalDesk [Wellner 1993] aimed at incarnating the desktop metaphor of the personal computer to the common office desk. However tabletops have a huge potential for collaboration. Thus, a research branch emerged that focused on *smart rooms*; spaces equipped with various cutting-edge computing devices that can interoperate, to support collaborative work for multiple users. Tabletops are essential elements of smart rooms, as shown with the InteracTable [Streitz et al. 1999] and the iTable [Johanson et al. 2002].

With the emergence of commercial items such as the Microsoft Surface [2012a], table-

Figure 1.1: People using an interactive tabletop exhibit at the Asia Society Museum. Source: T. Geller. Interactive tabletop exhibits in museums and galleries. *Computer Graphics and Applications*, 2006.

tops are gradually appearing in public environments such as museums, meeting rooms, public lobbies, bars and restaurants. They are an ideal platform for spontaneous use and multi-user interactions, and provide a touch-based experience that is similar to the one on most smartphones, which makes them easily accessible to the public.

State-of-the-art smartphones boast screen resolutions up to 1280 by 720 pixels, which exceed the naked eye's ability to distinguish separate pixels. However, a smartphone screen is too small to provide a satisfactory user experience in the presence of multiple users, and when viewing dense content such as text or high-quality images. Screen sizes for ultra mobile devices only go up to five inches. Reading a text, consulting a map and viewing images are examples of situations in which small displays present limitations.

An example of graphically dense content is a text of more than a few paragraphs. The default view of an online blog or a pdf document on a smartphone, is too small for a user to be able to discern single words. Depending on eyesight, it usually takes two to four zooming gestures to enlarge the text to a size that is conveniently readable. At this point, the user generally tilts the phone to landscape orientation, in order to give the paragraphs a more natural length. The screen provides only enough space for five to ten lines of text, which implies the use of successive pan gestures to update the content as the user reads on. Compared to the simple experience provided by a newspaper or a book, this seems troublesome.

Maps are graphically dense as well. They present a lot of different information, such as topography, street names and sights on limited space. To be able to read a street name on a smartphone map application, the user needs to zoom in on the relevant part of the map. The result is that the user can only view a very limited area. To be able to

relate this area to another location, the user must use a combination of pan and zoom out gestures that can be difficult. In certain cases, the area of interest is simply too vast to be viewed on a smartphone screen.

Viewing pictures on a smartphone is another experience made frustrating by the small screen size. Modern smartphones take pictures of high quality, and we carry them with us at all times. However, they are too small for us to conveniently view the pictures that we take, let alone to show them to others. Ideally, showing pictures to a friend implies both persons looking at, and commenting on the pictures simultaneously. But this is hardly possible on a smartphone. What typically happens is that the user finds a picture that he wants to show, then hands his phone to the friend, then the friend hands the phone back to the user, who chooses the next picture, and so on. This process is tiring at best.

Even though smartphones provide multiple opportunities, the aforementioned examples show that a small screen size can cause frustration. This thesis addresses situations where using a smartphone would be easier if it had a larger screen. The solution proposed is to make smartphones able to spontaneously integrate with larger displays that are available in the environment. In particular, the idea is to be able to walk up to a tabletop computer, and with a few easy steps, combine it with one's smartphone. The result, shown in figure 1.2, is that the screen of the smartphone is displayed on the tabletop, where it can be enlarged, and conveniently be viewed by multiple users simultaneously.

Figure 1.2: People viewing information on a smartphone.

To realize this vision, *device composition* can be used, a research approach that can be traced back to the work on smart space technologies and systems such as Augmented Surfaces [Rekimoto and Saitoh 1999], i-LAND [Streitz et al. 1999] and the Interactive Workspaces [Johanson et al. 2002]. Smart rooms were designed as closed computing environments, so recent efforts have been invested in developing for a more ad-hoc type

of device composition, with projects such as Obje [Edwards et al. 2009], Platform Composition [Pering et al. 2009] and CompUTE [Bardram et al. 2010].

Tabletops are a central device in the previously mentioned smart rooms, and a recurring element within device composition. They are an ideal platform for collocated collaboration, as shown with systems such as the MemTable [Hunter et al. 2011] and SketchTop [Clifton et al. 2011], that are designed to support the work of technical users. In recent years, however, tabletops are appearing in public spaces, aiming at engaging common users in a more relaxed form of interaction.

Within the context of designing for device composition between smartphones and tabletops, three fundamental challenges must be addressed.

The first one is the pairing procedure responsible for device discovery and connection. Pairing between unknown devices has been solved in many different ways. An example that is particularly relevant to this work is the BlueTable [Wilson and Sarin 2007], an interactive surface that uses Bluetooth to discover and connect to a mobile phone. The BlueTable improves the process by using computer vision to locate the phone on the table, and verify its identity via the phone blinking an infrared signal.

The second challenge is how to distribute the UI of the smartphone to the tabletop. Various stable technologies already exist, but recent researchers have chosen to build frameworks from scratch, to allow for a more flexible distribution and sharing of user interfaces between multiple types of devices. An example of such an approach is XICE (eXtending Interactive Computing Everywhere) [Arthur and Olsen 2011], a programming framework that is meant to support a new form of nomadic experience, where users can annex various display servers with their mobile personal devices.

The third challenge is to design a user interaction that suits the purpose of the system. Different interaction metaphors have been used to extend the UI of smartphones to larger display surfaces, including streaming, replication, expansion, projection and adaptation. In their recent work on Virtual Projection (VP), Baur et al. [2012] built a system where a smartphone can project its UI onto an available computer display.

1.1 Problem statement

This thesis addresses the problem of designing a composite device between a smartphone and a tabletop computer that allows for *spontaneous interaction*, by focusing on the learnability and ease of use of the system.

It does so by answering the following research questions:

- What are the requirements for such a system?
- Which interaction techniques are best suited for this type of system?
- Can this system be made to run on standard hardware and integrate different smartphone types?

1.2 Research methods

To solve the problem at hand, the following research methods were used.

Placing the present work within its research context, a comprehensive literary review was made of the prior work related to device composition and surface computing. In particular, the following aspects of device composition were examined; background, systems integrating smartphones to tabletops, pairing, UI distribution, user interaction. The results are presented in Chapter 2.

Following a user-centered approach, a solution design was completed. The process was divided into the following tasks. The application's context of use was analyzed based on scenarios and storyboards. This analysis lead to the definition of the solution requirements. A series of design options were derived from existing approaches to software design for surface computing. The design decisions were based on an experiment where end users engaged with low-fidelity prototypes of the system. The design process, together with the final system design, are presented in Chapter 3.

In order to demonstrate the feasibility of the suggested solution, a proof of concept was realized, in the form of an application prototype implemented on the Microsoft Surface tabletop computer, an iPhone 4 and a HTC Legend phone. The implementation was focused on the user interaction, and the aim of building a system that is easy to learn and use. The system is described in Chapter 4.

An evaluation of the design was conducted by way of a participant-based usability study, that involved the following steps. The focus of the evaluation was identified, in terms of specific aspects of the system, and the evaluation methods were selected. A user experiment was designed, that involved a discovery phase, a guided test and a questionnaire. Participants were recruited, and the experiment was carried out. Evaluation data was gathered via online forms as well as application logging. The experiment data was processed and analyzed, and the results were derived. The evaluation process and results are reported in Chapter 5.

1.3 Contributions

This report presents the design, implementation and evaluation of a composite device that integrates smartphones to the Microsoft Surface. There are three major contributions.

The user-centered design process, presented in chapter 3, shows that it is feasible to design a system that is immediately accessible to all types of users, by focusing on providing a *spontaneous user interaction*. In particular, to create an application whose learning curve is the lowest possible, a lot of focus is given to discovering which interaction techniques are familiar to the end users. To discover those features that will make the system both easy to learn and easy to use, two strategies are followed, that are both based on *design consistency*. The first strategy is to produce a design that is consistent

with the smartphone experience in general, in order to provide an familiar experience to smartphone users, and thus help introduce tabletops, that are novel devices, to non technical users. The second strategy is to realize a product whose design is consistent with the physicality of the devices involved. By getting inspiration for the tabletop application design from actual tables and the way people interact with them, the idea is to create an experience that will naturally appeal to end users.

The design process is based on a careful analysis of the application context, and focuses otherwise on how the user will interact with the system. To discover which interaction techniques are most familiar to smartphone users, several human-centered design techniques are used, all revolving around the setting up of cooperative design sessions based on low-fidelity prototypes. Figure 1.3 shows some of the paper prototypes that were used, together with an early prototype that was used as inspiration for this work.

Figure 1.3: Low-fidelity prototypes and an early digital prototype.

By involving end users early in the design process, it can be determined which actions users will naturally try the first time they will interact with the system, and which actions they would not think about. The result is the design of an application that allows the user to walk up to a tabletop, and easily transfer the content of his/her smartphone's screen to the interactive surface for an improvised experience. The replicated screen is surrounded by a frame that can be intuitively manipulated to control the size, position and rotation of the remote user interface. Multiple interaction interaction techniques are used to ensure that the application features are both easy to discover and efficient.

The implementation of the proof-of-concept application called *TIDE (Tabletop Interactive Display Extension)* is presented in chapter 4. It shows that it is feasible to implement

a system on the Microsoft Surface that can connect to a smartphone and extend its user interface. TIDE works by way of UI replication, it displays a copy of the smartphone's screen on the tabletop, and relays any touch input from the tabletop to the smartphone. Multiple simultaneous smartphones can be used, and the currently supported models are an iPhone 4 (iOS) and a HTC Legend (Google Android); but TIDE was designed to be extended to support other models as well.

TIDE uses shape recognition to detect smartphones that are in contact with the surface, and trigger a pairing procedure. The connection is quickly and securely established, and the smartphone can be easily disconnected and reconnected, enabling the application to support short successive interactions. Shape recognition is done by using computer vision algorithms to process and analyze the raw visual input provided by the Microsoft Surface camera-based system, and detect specific device features. This technique is also used to track the whereabouts of the smartphones during the application session, enabling the application to know where the devices are, and when they are removed. At the core of the system is the replication of the smartphone UI on the tabletop. This is done with the VNC protocol [Richardson et al. 1998], due to its stability and availability on many platforms. The client is integrated to the TIDE application, while the server relies on a third-party application installed on the smartphone. The mirrored UI is referred to as *replicated UI* in the rest of this report. It is contained by a visual frame that will be referred to as *surface UI*, and which implements all the UI elements that allow the user to manipulate the replicated UI on the tabletop.

The surface UI has the same visual aspect than the physical casing of the smartphone, to provide a natural user interaction. By using the Presentation layer of the Microsoft Surface SDK, it allows the touch-based manipulation of the replicated UI, with features such as moving, rotating, resizing or minimizing.

The final contribution is the evaluation of the TIDE prototype presented in chapter 5, which shows that the user-centered approach lead to the design of a composite device that provides a spontaneous user interaction. The system was evaluated by way of a participant-based usability study that confirmed that it was highly learnable, easy to use and useful in specific contexts. The usability study took the form of an experiment with thirteen volunteers, during which they used the application, as shown in figure 1.4.

The participants went through an exploratory test, a semi-controlled one, then answered a carefully designed questionnaire. The experiment generated qualified data, whose analysis helped determine which interaction techniques make TIDE easy to learn, which make it easy to use and in which contexts would TIDE be useful. The results showed that the touch-based techniques for shape manipulation came naturally to most users. For example, the possibility to enlarge the surface UI by touching both sides with two fingers and pulling apart. Other techniques that turned out to be familiar to most users were techniques that are part of a common IT knowledge due to the user's cultural background, an example of which is the double tap. However, the results did not confirm the idea that the form of the tabletop could inspire interaction techniques that would seem intuitive to the user, such as dragging a UI element to the edge of the table as one

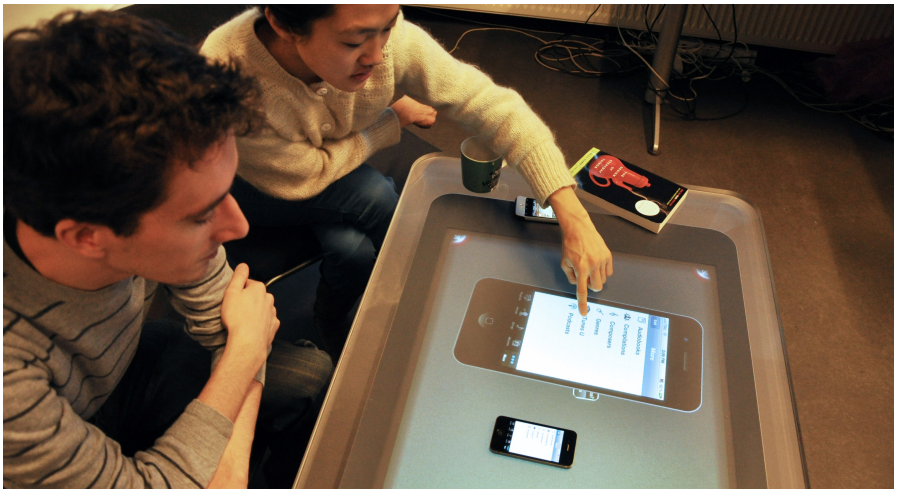

Figure 1.4: End users trying the TIDE prototype.

would with a real document.

Lastly, the participants' answers to the questionnaire showed that the system would be useful to perform specific activities, including browsing internet, looking at pictures and playing games. However, users do not feel comfortable using the system in a public environment, which is understandable due to the open aspect of the tabletop display, but a system limitation all the same.

Chapter 2

Related Work

This chapter describes the research context in which this work was conducted, by reviewing prior work belonging to five categories.

Device composition, presented in section 2.1, is the main background of this thesis. Combining smartphones and tabletops is the approach that was followed to solve the problem at hand, and is reviewed in section 2.2.

Section 2.3, 2.4 and 2.5 review the existing solutions to three fundamental challenges of device composition, i.e. respectively; (1) *Pairing*, the procedure through which devices associate with each other, (2) *UI distribution*, the technological approach to distributing user interfaces between devices and (3) *User interaction*, the type of metaphor that is used when combining devices for UI interoperation.

2.1 Device composition

Device composition is an important element of the ubiquitous computing research area, and the basis of this thesis. An essential aspect of ubiquitous computing is the idea that users can benefit freely from the resources that are available in the environment. Extensive research focused on realizing that idea by designing computer augmented spaces, or *smart rooms*, where heterogeneous devices can interoperate. Examples of such smart spaces are Augmented Surfaces [Rekimoto and Saitoh 1999], i-LAND [Streitz et al. 1999] and the Interactive Workspaces [Johanson et al. 2002]. Beside device interaction, these projects were aimed at supporting collaborative work between collocated users. Smart rooms are usually closed computing environments that rely on a centralized software infrastructure, such as BEACH developed for i-LAND [Tandler 2001], or Gaia OS for Active Spaces [Roman et al. 2002]. The advantage of working with a dedicated software platform is that the interaction between the enabled devices can be optimized, as they share a set of semantics. However, the drawback is that new devices cannot be integrated easily, as they require a specific software configuration.

To bring device composition out of the smart rooms, systems that support a more ad-hoc form of device interaction have been built. Obje [Edwards et al. 2009] is a middleware that relies on services transferring mobile code to achieve device compatibility at

runtime. It allows devices to interoperate upon their very first encounter, but relies on the user's semantic interpretation to do so. Pering et al. [2009] focused on understanding ad-hoc collaboration using pervasive technologies, and suggested Platform Composition as a technique to support collaborative work by using standard computing components. Bardram et al. [2010] developed CompUTE, a runtime architecture for device composition on Windows XP systems based on the extended desktop metaphor.

Tabletop displays are a recurring element in research projects that address device composition. They have two important characteristics. First, they are situated. Due to their size, they typically sit in a location and are not moved frequently. Second, they are shared among multiple users. Their horizontal interactive surfaces allow users to attend from all angles and to engage in face-to-face interaction. For these reasons, tabletops seem to perform the best in combination with the personal devices carried around by users.

In recent years, tabletops are being commercialized, and gradually appear in public spaces such as firm lobbies, shops, bars and restaurants. For example, the Microsoft Surface is part of the furniture in the new retail stores of the Royal Bank of Canada [Microsoft 2012b], offering service applications to the visiting customers. This tendency gives rise to a new type of tabletop interaction, more spontaneous, and targeting non technical users. It is within this context that the present work is set.

2.2 Integrating smartphones to tabletops

Due to the horizontal orientation of tabletop displays, they are an ideal platform for integration with the things that we naturally place on them. Much work has gone into the detection and tracking of material objects, with systems such as SurfaceFusion [Olwal and Wilson 2008], where the authors used RFID technology to sense the presence of objects, and computer vision to track their precise location.

Thus, computer vision can also be used to track other computing devices, such as smartphones. This approach has been followed by projects such as the BlueTable [Wilson and Sarin 2007], and the work by Echtler and Klinker [2008], where mobile phones are connected to a tabletop via Bluetooth, and the interaction is enhanced by having the tabletop track the location of the devices.

By using a personal device such as a mobile phone to interact with a tabletop, it is possible to provide user-identification to the interaction. An example of this is PhoneTouch [Schmidt et al. 2010], where the tabletop and a connected phone separately detect a touch event, and by comparing timestamps identify the phone. This technique can be used for personalization of the interface and user-authentication. It was also used by Berglund and Thomassen [2011], who implemented a framework to develop applications where smartphones can provide user-identification when interacting with a tabletop.

Another scenario for which device composition between smartphones and tabletops is promising, is the viewing and sharing of images. This has been shown by projects such as Throw Your Photos [Chehimi and Rukzio 2010] and Pour Images [Esbensen et al.

2010], where digital images can be transferred to a tabletop via a Bluetooth connection, and be viewed, edited and shared with other users.

2.3 Pairing

Pairing is the procedure through which two devices first discover each other, then connect to each other for interaction. It is an essential part of any device composition, and a challenge that can be solved in many ways.

Some systems are designed for networked devices, and achieve pairing with TCP/IP based networking protocols such as Zeroconf [2012] for Obje and UPnP [2012] for CompUTE.

Bluetooth [2012] supports a device discovery protocol that was used by the Personal Server [Want et al. 2002]. It allows the pairing of any bluetooth-enabled device, but the process has been shown to be slow, and can be inefficient if multiple devices are in the vicinity.

In systems that allow for spontaneous interaction between wireless mobile devices, additional techniques must be used to permit the identification of the correct device. A way to do that is by detecting synchronous events, as with the Smart-its [Holmquist et al. 2001], that connect when they are held and shaken together; and SyncTap [Rekimoto et al. 2003], that require the same key to be simultaneously pressed on both devices.

Another technique that has been used to connect mobile devices to a situated one, e.g. a large display, is having the display present a random key that has to be entered on the mobile device. The key can have the form of an alphanumeric string, a sequence of motions [Patel et al. 2004], or a visual pattern [Ballagas et al. 2005; Scott et al. 2005]. This approach allows authentication, but adds steps to the pairing process.

Established RFID technologies can be used for device association, but requires the mobile device to be equipped with the appropriate tag, as well as the situated device presenting an RFID reader. Upcoming NFC techniques present the great advantage that devices can function both as reader and transmitter, but few commercial devices are equipped as of yet.

Computer vision can be used to detect specific shapes, such as phone-like objects. It can make the pairing process easier, as shown with the BlueTable [Wilson and Sarin 2007], an interactive surface that uses vision techniques and Bluetooth to pair with a mobile phone.

2.4 UI distribution

Device composition has been extensively used for building systems that allow the sharing of user interfaces between devices. The typical setup is to be able to view and interact with the data and applications from a mobile device on a situated one that offers superior display resources. There are various technologies for UI distribution.

One way is to distribute only the data to the larger display, such as with the iRoom [Johanson et al. 2002], however this approach is only applicable when both devices have the same software installed.

Another possibility is to distribute code, sending whole applications to the situated device for execution. However, there is a series of issues with such an approach. The situated device might not be able to run the application, either because of compatibility problems (hardware, software) or because the device might choose not to trust unknown code. Software can be compiled into platform-independent code, such as with Flash [Adobe Systems 2012], Java [Gosling et al. 2000] and Silverlight [Microsoft 2012c]. However, issues that are mostly related to the excessive size of the data files and the security risks still remain.

Distributing graphics is generally a preferred approach. It allows for user data and credentials to be kept on the mobile device, and it has better potential for responsive interaction. Systems such as X-Windows (X11) [Scheifler and Gettys 1986], Remote Desktop Protocol (RDP) [Tritsch 2003] and Virtual Network Computing (VNC) [Richardson et al. 1998] utilize rendering-based protocols for UI distribution. Applications are executed on one device, and rendered on another. These protocols are stable, but they are based on the assumption that the machine executing the application has important resources. When combining mobile and situated devices, the situation is however reversed; the computer with the greater resources renders the graphics.

Another way of distributing graphics is by using web-based protocols, where Hypertext Transfer Protocol (HTTP) and Hypertext Mark-up Language (HTML) are used to send and present the application. Such an approach can either be built on a web server model, which is basically what we use when we log on to websites via a browser, or on a personal server model, which is what the Personal Server [Want et al. 2002] was based on. The Personal Server is a handheld device without a display, that transmits HTML pages to other available displays in the environment. Using a web-based protocol implies a series of drawbacks that are induced by the use of HTML; browsers behave differently, HTML 1 to 4 does not support generalized drawing, HTML does not handle varying display sizes and resolutions, and the use of Javascript introduces security issues related to programmable display.

Recent research efforts have been invested in the development of whole frameworks that can support flexible UI distribution. This is the case with XICE (eXtending Interactive Computing Everywhere) [Arthur and Olsen 2011], a programming framework that uses wireless networks to connect portable devices to display servers. It takes the limited CPU and battery capacities of mobile devices into account, and provides a flexible protocol that allows for annexation of different types of displays.

Substance [Gjerlufsen et al. 2011] was designed to support the development of interactive multi-surface applications. It is a data-oriented programming model that was used to build Shared Substance; a middleware that provides powerful sharing applications.

2.5 User interaction

Combining mobile devices with larger displays is not a new idea. It is already widely available for laptop computers, that can be connected to external monitors or projectors via a cable such as VGA or DVI. There are serious limitations to this type of setup. Some are technical; said cables are limited in terms of graphical output, the large size of the connectors is prohibitive on mobile devices. Others are related to the user experience; the user must have the required cable, the user must find the connector on the larger display, the process of connecting/disconnecting is susceptible to take time. This thesis focuses therefore on providing a wireless user experience.

Recent research efforts show different approaches, to improving the experience of the smartphone user, by allowing the extension of the phone UI to an available display surface. The main approaches are listed here in terms of the interaction metaphor that they are based on.

Streaming is a straightforward approach where only the visual output of the mobile device is forwarded to a remote display. It is what happens when we connect a laptop to a projector, e.g. to give a presentation. This is a satisfactory solution in specific situations, and presents the advantage of being secure, since the input is kept to the mobile device. However, it presents serious limitations in terms of the interaction potential.

Replication goes one step further, by allowing the user to provide input on the replicated UI. This type of integration is already available for personal computers, but requires a cable, or a docking station.

Expansion is when the larger display provides additional space for the existing UI of the mobile device. It is also available for laptops via a cable or docking station. It provides the possibility of transferring data or applications to the remote machine, while preserving the integrality of the mobile UI.

Projection is a metaphor similar to replication, with the difference that the mobile device itself is used for projecting its UI onto an available display surface. Winkler et al. [2011] have built a device composed of a smartphone and a personal projector, that provide powerful in-call collaboration features, by projecting a shared interface on any available surface. Virtual Projection (VP) [Baur et al. 2012] is a system where a smartphone can be used to project its UI onto an available display.

Adaptation refers to an improved UI transfer where the UI is modified to make full use of the available resources on the remote display. This approach was followed by the authors of XICE [Arthur and Olsen 2011].

In comparison to prior work, this thesis focuses on learnability and ease of use. The goal is to design a system that can support spontaneous interaction with non technical users successfully, by implementing interaction techniques that are consistent with the smartphone user experience. Additionally, this work examines the use of standard hardware and protocols to build a system that can be adapted to multiple platforms, and is compatible with various smartphone types.

Chapter 3

Design

This chapter reports the design process for building a composite device that integrates smartphones to tabletops and provides a spontaneous user interaction.

A user-centered design approach was followed, in the purpose of understanding the system's context of use, and identifying the interaction techniques that would make for an intuitive application. This process was supported by methodological tools described in Designing Interactive Systems [Benyon 2010]. Brainstorming, scenarios, storyboards and low-fidelity prototypes were used, and a study involving end users was carried out, the results of which informed the final design.

Figure 3.1: The TIDE prototype.

The focus of the design was on learnability and ease of use. The goal was to build an intuitive system, i.e. a system that users understand without conscious reasoning.

To reach this goal, the design principle of conceptual consistency, as defined by Benyon [2010], was used. In this particular case, consistency was used in relation to two important aspects. First; given that all the users of the system are smartphone users, the final system should provide a user experience that is as consistent possible with the smartphone experience. Second; the user interaction should be consistent with the implications that the devices' form have on the user's expectations. In particular, the interaction should be consistent with the form of the table, e.g. various objects could

be placed on the tabletop during an application session.

It was decided to limit the focus of this work to one interaction metaphor: *UI replication*. This metaphor was chosen among the different possibilities enumerated in Chapter 2, for the following reasons. Firstly, UI replication allows for a strong consistency between the experience on the smartphone and the experience on the tabletop. Secondly, UI replication implies that the smartphone's application logic stays unchanged, and allows for the development effort to be focused on the tabletop. Thirdly, for the same reason as mentioned above, the implementation can be made to support different types of smartphones.

Thus, the system is designed to present the user with a mirrored image of the smartphone UI on the tabletop. This is referred to as the *replicated UI*. It is interactive, relaying touch input and graphical output between the smartphone and the tabletop. The replicated UI is controlled by the *surface UI*. The surface UI consists of elements that allow for manipulation of the replicated UI on the tabletop.

An analysis of the context of use is presented in section 3.1, from which solution requirements are derived in section 3.2. Section 3.3 describes the generation of design options for the surface UI and their evaluation. The final design of the TIDE prototype is presented in section 3.4.

3.1 Understanding the application context

Some initial design constraints are already known; the system integrates smartphones to tabletops, and the system uses UI replication to extend the smartphone UI to the tabletop. In this section, additional design constraints are derived from an analysis of the devices, and the context of use is investigated through the use of scenarios.

The devices

The basic characteristics of the devices have concrete implications on the system design.

A tabletop is...

...a computer. The tabletop provides computing resources such as processing power, memory and connectivity; as well as an operating system that supports software applications.

...a table. Its physical form comprises a horizontal surface that is commonly used to support various material objects. A table is used for various activities, such as studying, eating, playing games, holding meetings, etc. It can be approached from all angles, encouraging face-to-face interaction between multiple users.

...a situated device. It usually sits at a location, and is not moved often. It can be expected to have dedicated power supply and network connection. Users approach

it to use it, and leave when they are finished. Thus, the interaction flow is likely to be interrupted, taking the form of short successive sessions. The nature of the location has an impact on the user interaction, and should be considered. Examples are a conference room, a public lobby, a mall, an individual office, a living room in a family house, etc.

. . . a shared device. When located in a public space, the tabletop is shared among multiple users. Depending on the context and the relationship between the users (friends, colleagues, strangers, etc), the level of trust varies.

. . . an interactive surface. It typically offers a large graphical output, and a range of input techniques that allow for user interaction. Most tabletops support multitouch-based input. Some tabletop models support computer-vision techniques that allow for the integration of tangible objects to the interaction.

A smartphone is. . .

. . . a computer. It provides the computing resources necessary to allow a user to connect, store personal data and install/use applications. It runs on a mobile operating system that supports software applications.

. . . a small device. It is designed to fit in a pocket. Smartphones are typically less than 5" high, 3" wide, and 0,6" deep. The small form factor implies some hardware limitations: resources such as battery life and processing power can not be taken for granted.

. . . a mobile device. It is carried around by users at all times. It is mostly used as a wireless device, across networks, implying a level of instability in the connectivity.

. . . an interactive device. Modern smartphones typically include touch-based screens, as well as physical buttons, to allow user interaction.

The situations

Understanding the situations in which the system will be used is a step towards the elicitation of solution requirements. The scenarios that were used to help define those situations are included in appendix B.

As shown in figure 3.2, the scenario depends on the level of privacy of the location (public/private) and the amount of involved users (single/multiple).

There are four combinations, however the situation with multiple users in a private location is not addressed here, for the reason that the tabletop cannot be considered private if multiple users are present.

Single user with a public tabletop

This situation was derived from a scenario involving a person sitting alone at an interactive table in a coffee shop.

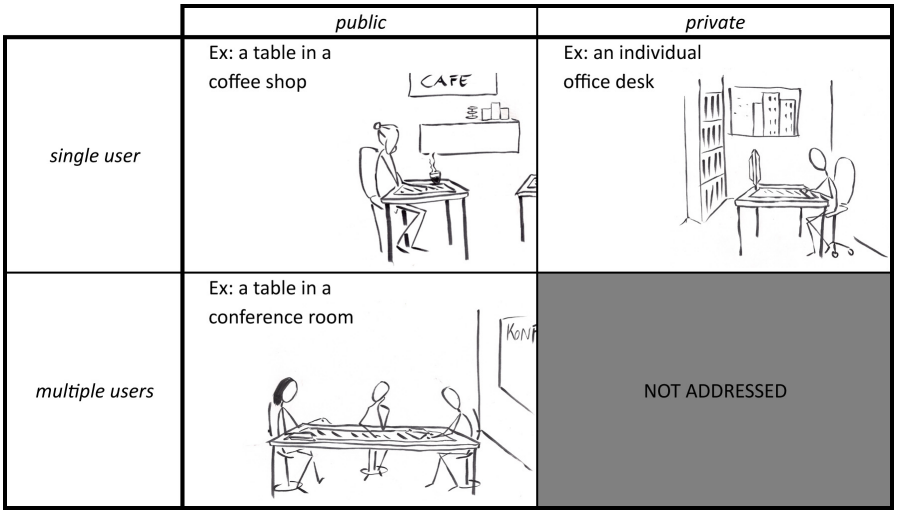

Figure 3.2: The different situations in which the system can be used.

In this environment, the motivation for using the system is an activity that involves graphically intense content, such as internet browsing, reading, consulting a map, etc. To start using the system, the user pairs the smartphone with the tabletop. The process is quick and easy, but not completely automatic. The user must explicitly allow the connection to be established. The user does not usually need cables to use the smartphone. Ideally, the connection with the tabletop is handled wirelessly. The replicated UI appears on the tabletop, and the user starts interacting with the application. The surface UI provides the user with features to manipulate the replicated UI, in particular: dragging, rotating and resizing. Via the surface UI, the user can minimize and hide the replicated UI, as well as exit the application.

The smartphone itself is involved in the interaction. Placing it on the table triggers the pairing process. During the interaction, the replicated UI is collocated with the device, and moves together with it. Lifting the smartphone off the table ends the connection.

Multiple users with a public tabletop

This situation was derived from a scenario involving some work colleagues having a meeting in a conference room, equipped with an interactive tabletop. It can be argumented that such an environment is only partly public, being only accessible to the people working in the company. However, in an effort to produce a design as generic as possible, and for the sake of simplicity, the tabletop is here considered a public device.

The motivation to use the system in this context is to view, comment and possibly edit digital data in a collaborative meeting.

Multiple users might want to connect their smartphones to the system simultaneously. Due to the lack of space on the table, the users remove their devices, but keep

the replicated UI active on the tabletop.

Single user with a private tabletop

This situation was derived from a scenario involving a user whose office desk is a tabletop.

The motivation for using the system is that it is a tool that the user interacts with daily, to support various activities related to the work routine.

The tabletop being private, can be configured to provide extended functionalities. Examples of extended functionalities are: the pairing procedure can be made automatic and data can be shared between the devices.

3.2 Solution requirements

Based on the analysis of the application context, the following requirements were derived. Only the requirements that were relevant to the problem at hand were included. They are divided into four categories, that each correspond to a different aspect of the system: the pairing procedure, the UI replication, the surface UI and the use of the smartphone as tangible UI.

However, the following fundamental requirement is not attached to any category, as it relates to several system aspects:

R-1 *The system should support multiple simultaneous devices.*

Pairing

The pairing procedure is responsible for device discovery and connection. It should fulfill the following requirements:

RA-1 Pairing the devices should be easy and quick.

RA-2 Discovery between the devices should be automatic.

RA-3 Establishing the connection should require explicit confirmation from the user.

RA-4 Closing the application should be easy and quick.

Tracking

Tracking the location of the smartphone on the tabletop allows to link specific features to the physical device. It should fulfill the following requirements:

RB-1 Placing the smartphone on the tabletop should trigger the pairing procedure.

RB-2 The replicated UI should be collocated with the smartphone, i.e. the replicated UI should be seemingly linked to the device, and should move when the device is moved.

RB-3 Lifting the smartphone off the table should trigger the application exit.

RB-4 It should be possible to remove the smartphone, but keep the replicated UI active on the tabletop.

Replicated UI

The replicated UI is the interactive mirror image of the smartphone displayed on the tabletop. It should fulfill the following requirements:

RC-1 The UI of the smartphone should be replicated on the tabletop.

RC-2 The graphical output from the smartphone should be dynamically relayed to the replicated UI.

RC-3 The touch-based input from the replicated UI should be dynamically relayed to the smartphone.

RC-4 The responsiveness of the replicated UI should be under 1 second, for the user experience to be satisfactory.

Surface UI

The surface UI is the interface on the tabletop that controls the replicated UI. It should fulfill the following requirements:

RD-1 The surface UI should be easy to use.

RD-2 The surface UI should allow the user to manipulate the replicated UI, i.e. the user should be able to move, rotate and resize the replicated UI across the tabletop display.

RD-3 The surface UI should allow the user to minimize and restore the replicated UI.

RD-4 The surface UI should allow the user to hide the replicated UI.

RD-5 The surface UI should allow the user to exit the application.

RD-6 The surface UI should include controls that implement the functionalities provided by the physical buttons on the smartphone.

3.3 Designing the user interaction

This section focuses on the design of the surface UI, which comprises the UI elements that provide control over the replicated UI on the tabletop. The other system aspects did not require a detailed design process, for the following reasons.

First, it was decided that the pairing procedure should rely on already available solutions.

Second, given that UI replication had already been decided upon, no design effort would be required for this system component.

Lastly, tracking the location of the smartphone is a technical requirement that does not require interaction design.

The input technique considered in the design of the surface UI is touch-based interaction. The reason for this is that it is strongly consistent with the smartphone and tabletop experiences. Both rely on touch-based interaction. Moreover, the presence

of other input peripherals cannot be expected, thus relying on touch-based input will ensure the system is usable in any setup.

The steps followed to design the surface UI were; (1) The generation of ideas, (2) The definition of *interaction strategies* and (3) A study involving end-users to identify the interaction techniques that they find the most familiar.

Generating ideas

The generation of ideas is an important part of the design process. Benyon refers to it as envisionment [Benyon 2010], which he defines as the process of externalizing design thoughts. The techniques that were used for this process are brainstorming, sketching, storyboarding and prototyping.

Storyboards

The scenarios mentioned in section 3.1 are used as a base for the making of storyboards. Storyboarding helps finding the general flow of the system interaction. It also gives a visual dimension to the definition of the different system features, and raises new design issues.

As an example, several system features were described in the initial scenarios as being the result of a tap on a button. When storyboarding, it became obvious that having too many UI buttons would crowd the user interface. This caused consideration of other interaction techniques, such as touch gestures. Similarly, the issue of the location and size of the replicated UI on application launch first became apparent on a storyboard.

Low fidelity prototypes

Paper prototypes were used to aid the process of generating and evaluating as many design solutions as possible. Screenshots of the iPhone UI [Apple 2012b] were printed in various sizes, and used on a normal table to simulate interaction with the replicated UI. Figure 3.3 shows some prototypes and a working session.

Defining interaction strategies

The terms of *actions* and *commands* were used to refer to the different aspects of the user interaction. They are inspired by the work done by Wobbrock et al. on defining hand gestures for interactive surfaces, [Wobbrock et al. 2009].

A command is an *effect* that the user wants to obtain, such as rotating a picture on a tabletop display. An action is the *cause* of an effect. For example, using two fingers and performing a rotating gesture. On a tabletop display, actions are typically hand gestures.

Figure 3.3: Working with low fidelity prototypes.

Commands

Based on the requirements formulated in section 3.2 for the surface UI, the following six commands were identified. They are the *interaction primitives* for the surface UI, i.e. the commands that must be implemented by the surface UI, to allow the user full control of the replicated UI.

1. *Dragging* the replicated UI across the interactive surface.
2. *Rotating* the replicated UI across the interactive surface.
3. *Resizing* the replicated UI across the interactive surface.
4. *Minimizing* the replicated UI, and restoring it.
5. *Hiding* the content of the replicated UI.
6. *Closing* the replicated UI.

Additionally, the surface UI should include controls that implement the functionalities provided by the physical buttons on the smartphone.

Actions

To invoke the commands that are defined above, a user needs to perform actions. Various interaction techniques can be used to define these actions. Figure 3.4 shows five *interaction strategies* that were identified while working with paper prototypes. Each strategy can be consistently implemented for all six interaction primitives.

1. *Action Tabs* are traditional buttons/tabs that implement functionalities.
2. The *Action Bar* can be compared to a virtual touchpad, it includes a manipulation area and buttons.
3. *Window Toggle* refers to using a switch to toggle the window between inactive and active states. In its inactive state, the window is made manipulable as a common digital picture.

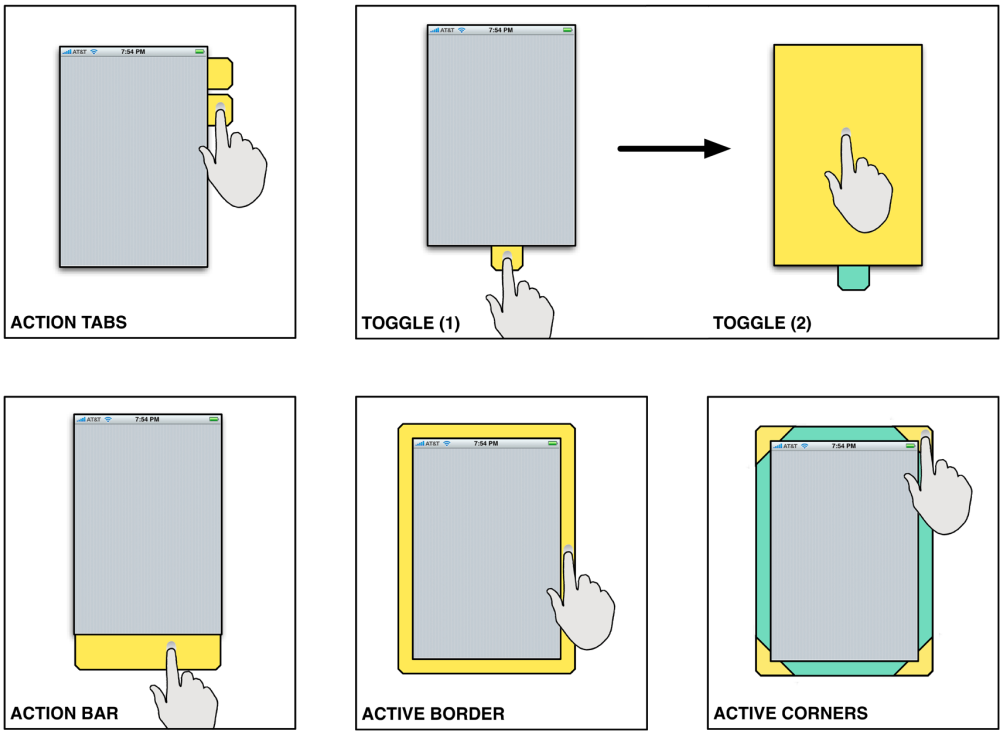

Figure 3.4: Interaction strategies.

4. The *Active Border* is a digital frame around the application window used for manipulation.
5. *Active Corners* is a strategy similar to Active Border, with the difference that the border's corners implement specific functionalities.

Beside these five interaction strategies, an additional category was used, that regrouped interaction techniques that did not correspond to any of the strategies. This category is called *Other*, and is discussed on page 28.

Involving users

To discover which of the defined interaction strategies were most intuitive, an experiment was carried out, that involved end users. The experiment took the form of cooperative design sessions, where users engaged with low-fidelity prototypes of the system, in the aim of exploring and evaluating ideas.

The parameters of the experiment are the commands, also referred to as *interaction primitives*, and the actions, also referred to as *interaction strategies*, defined in section 3.3.

Experiment

Figure 3.5: Designing the surface UI with paper prototypes.

Twelve participants were recruited on a voluntary basis. A session involved one participant and the designer, as shown in figure 3.5. The participant sat next to the Microsoft Surface tabletop [Microsoft 2012a], and was presented with an iPhone [Apple 2012b], but both devices were turned off. On the tabletop were paper prototypes, that were to be used as representations of UI elements throughout the session. The designer lead the experiment by reading instructions from a script (included in appendix C) and answering the participant's questions.

As an introduction, the following things were explained to the participant:

- The purpose of the experiment
- The purpose of the application
- The tasks that the participant will perform
- The principles of working with paper prototypes

The session revolved around a task that the participant was asked to perform using the prototyped application. The task was to write an email, and it required the user to go through six phases. Each phase was dedicated to an interaction primitive, and they had the same structure, which is as follows:

1. The primitive was explained to the participant in terms of a command to the application.

2. The user was asked to express an open-ended suggestion, i.e. suggest an action that s/he would perform to obtain the desired effect, and to demonstrate the action using the prototypes,
3. The designer described action suggestions, that the user was asked to try out and rank by order of preference.

Data processing

Participant answers were gathered in a form such as the one included in appendix D. The form is a matrix where an entry corresponds to a pair (primitive, strategy). The entries contain the position from first (highest) to third (lowest), that was given by the user for using the suggested strategy for implementing the primitive. After processing all answers, each entry contained six positions. In order to obtain a numeric score for each entry, a weighted average was calculated. A weight of 3 was given to a first position, a weight of 1 to a second position, and a weight of 0 to a third position. Finally, the results were normalized to a [0-1] interval, where a 1 meant that the entry was awarded a first position by all participants, and a 0 meant that all participants ranked the entry third. Figure 3.6 summarizes the normalized scores, with colored cells containing values above 0.6. This is considered a superior score, because it can only be obtained if half of the participants awarded the first position.

The experiment data also included the user suggestions, gathered separately on paper. User suggestions were not limited to one per primitive. Thus, the processed data was a disparate list of suggestions that each had a counter variable indicating how many users independently expressed said suggestion.

	Action Tabs	Window Toggle	Action Bar	Active Border	Active Corners	Other
Dragging	0.50	0.22	0.61	0.89	0.44	0.00
Rotating	0.06	0.17	0.56	0.61	0.78	0.50
Resizing	0.56	0.17	0.22	0.50	0.67	0.56
Minimizing	0.44	0.00	0.72	0.67	0.00	0.67
Hiding	0.33	0.39	0.17	0.78	0.44	0.56
Closing	0.28	0.50	0.06	0.50	0.33	1.00
Avg	0.36	0.24	0.39	0.66	0.44	0.55

Figure 3.6: Normalized weighted average of the ranks given to each pair (primitive, strategy).

Result Analysis

It is possible to divide the primitives into two groups. The first half—dragging, rotating, resizing—have a concrete visual signification. The second half—minimizing, hiding, closing—are more abstract.

For the first three primitives, there is a strong coherence in the choice of the participants. The favored strategies are the active border, the action bar and active corners.

All three require the user to interact with an area directly around the window in order to manipulate it and modify its position, orientation or size. This interaction technique is similar to the current standard for manipulating pictures on interactive screens. However in the present case, the replicated UI is logically avoided because of its role as IO relay between the tabletop and the smartphone.

For the last three primitives, the action bar and active border continue to score high, even though there is no apparent relation between the visual aspect of the strategy and the effect implied by the command. To understand this, it is necessary to look at the actual suggestions. The favored strategies for minimizing were double tapping on the action bar and double tapping on the active border. For hiding, the favored strategy was double tapping on the active border. It is thus obvious that it is the double tap that participants have a preference for, possibly because it is a common technique in many other application contexts, as well as a quick and easy one.

It is not possible to analyze the scores of the sixth category as a whole, as it does not represent a consistent strategy, but more a patchwork of various suggestions. They are:

1. Drag by holding a finger on a specific tab, and using another finger to tap a destination target to move the window.
2. Rotate by performing a one finger dragging gesture on a corner of the window.
3. Resize by pulling the window apart with both whole hands.
4. Minimize by dragging the window to the bottom of the surface.
5. Hide by placing and holding a full hand on the window.
6. Close by dragging the window to a specific location on the surface.

Suggestions 4 and 6 are similar, and they are the only ones that scored above 0.6. This suggests that moving the window off screen; is a natural way to remove focus from the application. Interestingly, this correlates with the analysis of the user suggestions.

User suggestions

The user suggestions were multiple and heterogeneous, but data processing revealed a clear tendency in three situations.

In the case of dragging, half of the participants suggested using one or more fingers to touch within the replicated UI, and perform a drag gesture. This is interesting, because it is an obvious conflict for the developer, i.e. any touch inside the replicated UI is forwarded to the smartphone, and logically can not be interpreted by the surface UI. It must be noted that dragging was the first primitive, and it can therefore be assumed that the participants did not yet have a full understanding of the application concept. However, this result shows what the ideal system should be able to do: interpret the intention of the user.

In the case of resizing, 8 out of 12 participants suggested grabbing the sides of the window with 2 fingers, and pulling the window apart to enlarge it. This shows that the

gesture is very intuitive, and that implementing it would definitely add to the usability of the system.

The third consensus was for minimizing, where 7 out of 12 participants suggested dragging the window offscreen (or to a specific location along the surface edge). The same suggestion reoccurred for the primitives hiding and closing, though with less decisiveness. Once again, it shows that the gesture is an intuitive one. Moreover, it is consistent with the action of removing a real piece of paper from the center of a table.

3.4 Final Design of TIDE

Based on the design analysis and in particular the feedback from the user experiment, the following design decisions were taken. They relate to the requirements defined in section 3.2.

The requirement *R-1 The system should support multiple simultaneous devices* will be fulfilled by the design decision:

D-1 The system will be implemented with focus on loose coupling between the main application session classes and the objects handling individual devices, in order to support simultaneous devices, as shown in figure 3.7.

Figure 3.7: (A) Pairing second device (B) Simultaneous devices

Pairing

The following design decisions address the requirements RA-1 to RA-4 (see p. 21).

DA-1

- DA-1a Pairing is triggered by placing the smartphone on the tabletop, as shown in figure 3.8, for the reason that it is a simple gesture that is consistent with the form of the tabletop.
- DA-1b Pairing is based on wireless connectivity, in order to provide spontaneity to the interaction.

DA-2 Device discovery relies on a standard networking protocol - in order to make it automatic, and provide spontaneity to the interaction.

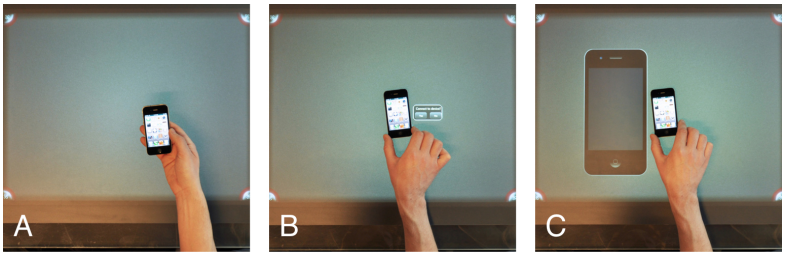

Figure 3.8: (A) Device in hand (B) Device detected (C) Device connected

DA-3 The user must confirm the connection, both on the tabletop and on the smartphone, for reasons of security. The tabletop should not be allowed to connect to an unknown device without explicit authorization, but the user should also be able to confirm that the smartphone is the correct one.

DA-4 See DD-5.

Tracking

The following design decisions address the requirements RB-1 to RB-4 (see p. 21).

DB-1 Smartphones that come in contact with the tabletop are detected, using computer vision and shape recognition.

DB-2 The location of the smartphones on the tabletop is tracked, using computer vision.

DB-3 Removal of smartphones are detected using computer vision.

DB-4 The replicated UI can be made active on the tabletop independently of the smartphone.

Replicated UI

The following design decisions address the requirements RC-1 to RC-4 (see p. 22).

DC-1 The smartphone screen is replicated to the tabletop, using a standard desktop sharing protocol, for the reason that such protocols are platform independent and stable.

DC-2 See DC-1.

DC-3 See DC-1.

DC-4 See DC-1.

Surface UI

The following design decisions address the requirements RD-1 to RD-6 (see p. 22). It was decided to use multiple techniques to activate certain features. The reason for this is to benefit from the advantages of all interaction techniques. Some are easy to discover, and

provide the learnability and intuitiveness that is looked for, while others are quicker and easier to use. This is standard practice within software design, a good example being keyboard shortcuts, that are not easily discovered by a novice user, but are preferred by the expert user for their efficiency.

DD-1 The surface UI takes the form of an active border that contains the replicated UI, based on the user feedback. It is a virtual replication of the physical body of the smartphone, to provide consistency in the user experience.

DD-2 The replicated UI can be manipulated by using touch-based gestures on the surface UI, for reasons of consistency with both the smartphone and tabletop experiences.

DD-2a Dragging is done by performing a dragging gesture with one or more fingers.

DD-2b Rotating is done by

1. Dragging the corner of the surface UI, as shown in figure 3.9

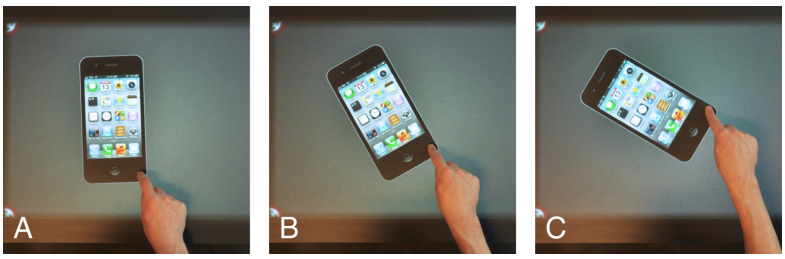

Figure 3.9: (A,B,C) Rotating using 1 finger.

2. Performing a rotating gesture with two fingers of the same hand, as shown in figure 3.10

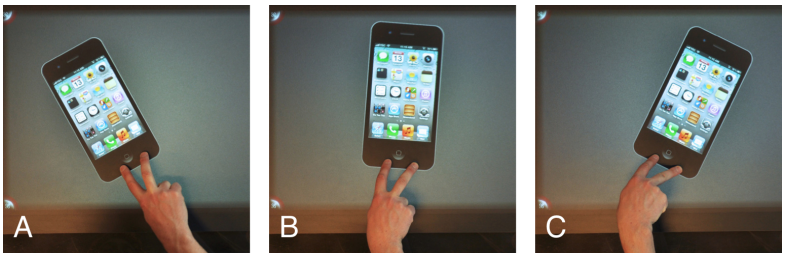

Figure 3.10: (A,B,C) Rotating using 2 fingers of the same hand.

3. Performing a rotating gesture with two hands

DD-2c Resizing is done by

1. Performing a pinching gesture with two fingers of the same hand
2. Performing a resizing gesture with two hands, as shown in figure 3.11

DD-3 Minimizing the replicated UI can be done by

Figure 3.11: (A,B,C) Resizing down using both hands.

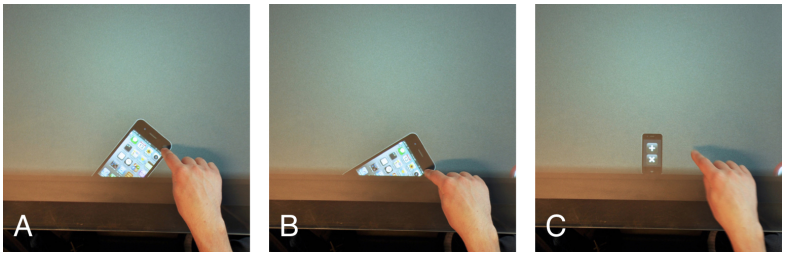

Figure 3.12: (A,B,C) Minimizing by dragging off the edge.

1. Resizing the surface UI down
2. Dragging the surface UI off an edge of the table, as shown in figure 3.12
3. Double tapping the surface UI, as shown in figure 3.13
4. Placing a full hand on the surface UI, as shown in figure 3.13

Figure 3.13: (A) Minimized state (B) Double tap (C) Whole hand tap

When minimized, the surface UI appears as an icon on the tabletop, and can be restored by tapping a button.

DD-4 Hiding the replicated UI is done by minimizing the surface UI, based on user feedback, see DC-3.

DD-5 Closing the replicated UI is shown in figure 3.14, it can be done by

1. Closing the minimized surface UI
2. Dragging the surface UI to a corner of the tabletop
3. Pressing and holding a finger on the active border

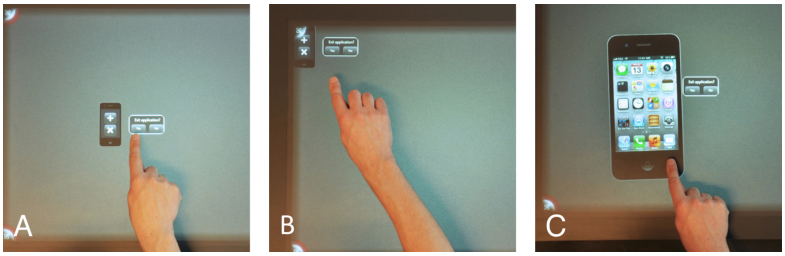

Figure 3.14: (A) Minimize then close (B) Close by dragging to a corner (C) Press and hold to close

DD-6 The functionalities provided by the smartphone's physical buttons can be accessed by tapping virtual replications of the buttons on the surface UI.

Chapter 4

The TIDE prototype

TIDE (Tabletop Interactive Display Extension) is a composite device between smartphones and a tabletop computer. It replicates the smartphone's user interface on the tabletop, where it can be manipulated and interacted with. The application can handle simultaneous devices. It runs on the Microsoft Surface tabletop computer, referred to as *Surface* in this chapter. It currently supports two smartphone models: the iPhone 4 (iOS 5) and the HTC Legend (Google Android 2.1), and can be extended to support other models.

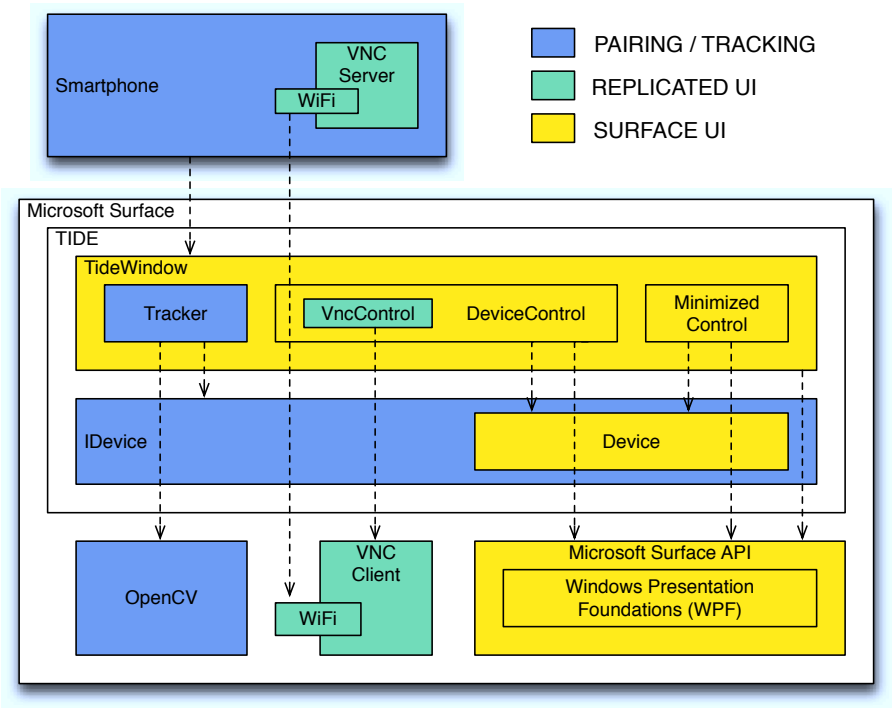

Figure 4.1: TIDE component diagram.

Figure 4.1 is a component diagram that presents an overview of the system. TIDE can be divided into three parts, which together implement the design aspects formulated in section 3.4, i.e. pairing, tracking, replicated UI and surface UI.

The pairing and tracking elements (blue) are presented in sections 4.1 and 4.2. They are both based on shape recognition to detect the presence and location of the smartphone on the tabletop. To do that, TIDE uses the visual input provided by the Surface camera, and relies otherwise on OpenCV (Open Source Computer Vision) [2012], a library that provides the tools necessary to process images and perform object detection.

Replicating the smartphone's UI (green) is described in section 4.3. It is done with VNC [Richardson et al. 1998], a platform independent system for standard desktop sharing. TIDE includes a VNC client that is based on the VncSharp library [VncSharp 2012], and that connects over a WiFi connection to a VNC server running on the smartphone.

Section 4.4 presents the surface UI (yellow). It was implemented within the .NET framework, using the Surface SDK and WPF (Windows Presentation Foundations). It links the other components together in a Surface application, and provides the overall user interface. It is the system aspect that has the most relevance for the user interaction, and has therefore received most focus in this implementation.

System interaction

Using the system involve a user, a smartphone and a tabletop computer interacting with each other. During a typical session, the interaction revolves around the *surface UI*, which is the interface between all system parts, and is contained by the main application window on the tabletop, the `TideWindow`. Figure 4.2 is a sequence diagram that describes this interaction.

The *user* starts the session by placing the smartphone on the table (①), where it is detected by the *tracker* component (②). Optionally, the tracker can track the device during the whole application session. Upon device detection, a virtual device is created and added to the surface UI, then forwarded to the *VNC client* who is responsible for establishing the connection (③). The VNC client attempts to connect to the *VNC server* running on the smartphone (④), at which point an explicit authorization from the user is required. If the connection is authorized, The server sends the smartphone UI to the client for replication.

At this point, the session enters a *loop* where the user interacts with the surface UI for an undefined amount of time. The user provides input to the surface UI, a part of which is destined to the replicated UI. The VNC client sends the said input to the server, who responds with an new copy of the smartphone screen, that is used by the surface UI to update the replicated UI.

On the initiative of the user, the session is terminated, at which point the virtual device is removed from the `TideWindow`, and the connection is interrupted by the VNC client.

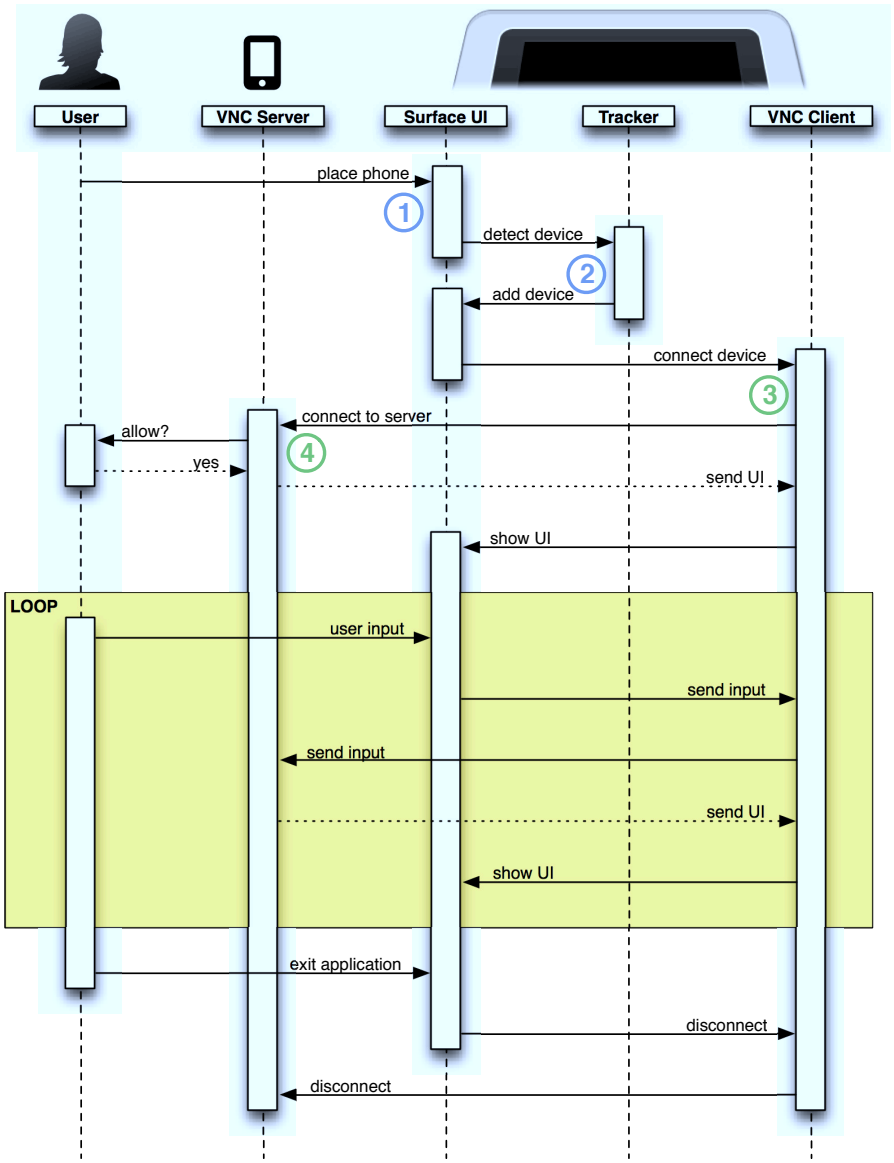

Figure 4.2: TIDE: main interaction diagram.

4.1 Pairing

This section describes and explains the implementation steps that were taken based on the design decisions from section 3.4 relating to the pairing procedure.

The pairing procedure consists of three steps: (1) a *trigger* starts the procedure, (2) *discovery* allows the smartphone's local IP address to be made available to the tabletop and (3) the *connection* is established, given that the user explicitly authorizes it.

Trigger

To realize the design decision: *DA-1a Pairing is triggered by placing the smartphone on the tabletop*, TIDE uses the Surface computer vision to detect when a smartphone is placed on the tabletop.

The implementation details of the smartphone detection and tracking are presented in section 4.2.

Discovery

For the tabletop to discover the local IP address of the smartphone, a straightforward solution is to present a text field to the user in which s/he can enter it. However, this approach has several limitations, including an obvious lack of spontaneity. Not all smartphone users know where to look in the device's settings to find the IP address, and in any case, it is a process of several steps that would remove the spontaneity that is the overall aim with this prototype.

Therefore, to realize the design decision: *DA-2 Device discovery relies on a standard networking protocol*, TIDE should use an automatized discovery process based on a standard networking protocol such as the ones provided by UPnP [UPnP Forum 2012] and Bonjour [Apple 2012a].

However, given that the focus of this thesis lies on the user interaction problem, it was decided not to implement the discovery protocol. The system currently functions with IP addresses hardcoded into the TIDE implementation on the Surface, to serve the purpose of the proof-of-concept.

Connection

The design decision: *DA-1b Pairing is based on wireless connectivity*, does not require any implementation effort, but represents an assumption about device and environment. There must be a local area network for the system to function, and both devices must be connected to it. The tabletop being in a fixed location, it can use either WiFi or an ethernet cable. On the smartphone, WiFi must be used.

To realize the design decision: *DA-3 The user must confirm the connection, both on the tabletop and on the smartphone*, TIDE uses dialogs, that appear successively on the tabletop, then on the smartphone. They are described further in section 4.4.

4.2 Tracking

This section describes and explains the implementation steps that were taken based on the design decisions from section 3.4 relating to tracking the location of smartphones on the tabletop. To realize the design decisions DB-1 to DB-3 (see p. 30), TIDE uses the computer vision provided by the Surface, as well as the OpenCV library.

The Surface uses camera-based vision to detect touch input. Infrared light is projected towards the surface, and reflected by the objects that are in contact with the

table. This reflection is captured by infrared cameras, allowing the Surface to "see" all types of shapes. When such a shape is small enough, the Surface interprets it as a finger, and reacts accordingly.

Beside fingers, the Surface is designed to natively recognize certain types of 2D visual tags (Byte and Identity tags). Such tags can be used to integrate objects to Surface applications.

However, using visual tags in the present case would imply that the system only be usable by users that had applied such a tag to the smartphone. To remove as many constraints as possible, and to make the system usable in as many situations as possible, it was decided to use shape recognition.

OpenCV provides algorithms to perform shape analysis and recognition on the raw images that are captured by the Surface system.

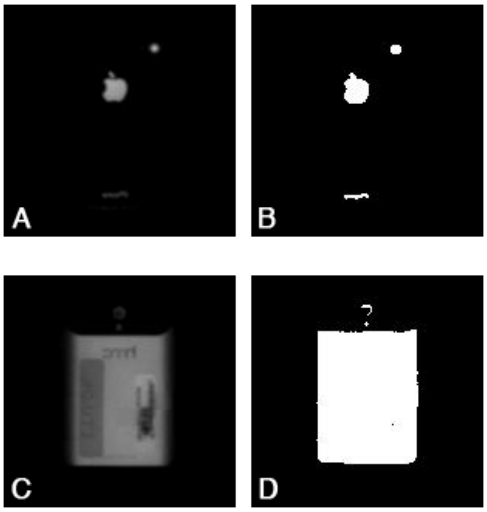

Figure 4.3: Microsoft Surface raw visual input: (A) iPhone 4 (B) iPhone 4 after threshold processing (C) HTC Legend (D) HTC Legend after threshold processing.

Figure 4.3 shows the raw Surface input for both tested smartphones. The features that allow TIDE to detect the iPhone 4 are the Apple logo and the camera point in the back of the casing. For the HTC Legend, the system looks for the larger silver rectangle formed by the aluminum casing.

Two things can be remarked on. First, black casing areas do not reflect infrared light, which is a basic, yet serious limitation, given that certain models of smartphones could be invisible to the system. Second, detecting a phone depends on the specific features that the phone presents on its casing. This implies that the system must take into considerations that the phone could be placed on the table face down, or wear a protective sleeve.

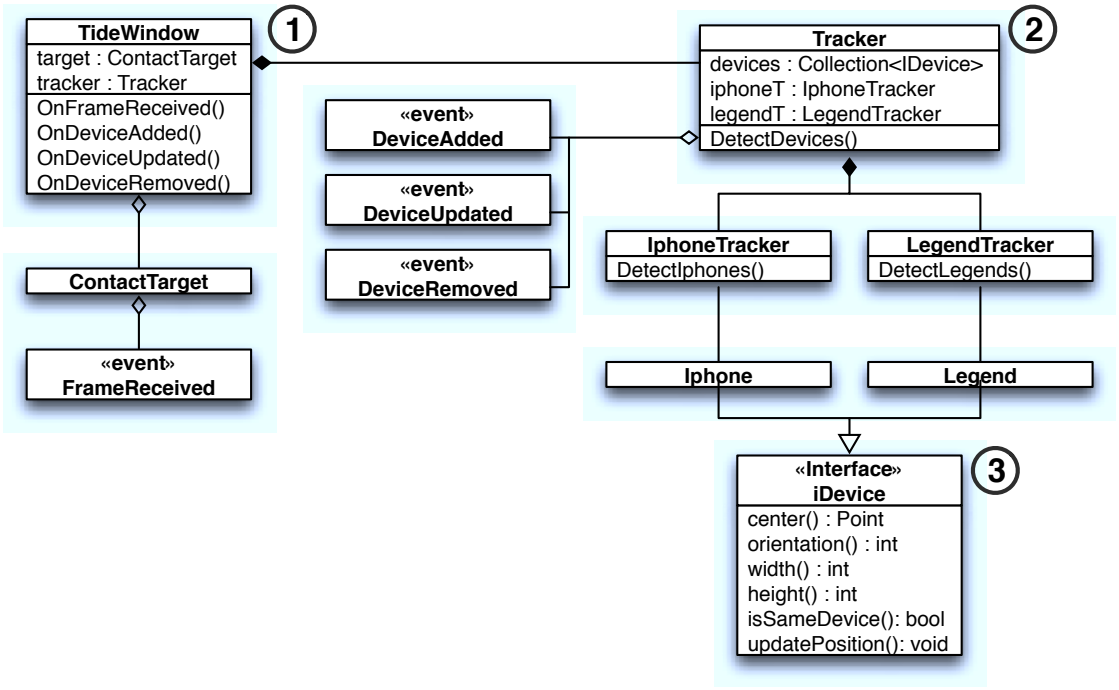

Figure 4.4: Class diagram of the tracking component.

Figure 4.4 shows the implementation that allows TIDE to detect specific smartphones, and track their whereabouts on the surface during application sessions.

The `TIDEWindow` (①) is an application-specific instance of a `SurfaceWindow` control provided by the Surface SDK Presentation layer. It represents the main application window, and comprises most of the application logic. The core layer of the Surface SDK provides the `ContactTarget` class, which raises the `FrameReceived` event for each available frame of visual input.

The `TideWindow` sends screen captures to the `Tracker` class (②), that is responsible for processing the images, detecting and tracking the smartphones. The `Tracker` keeps track of the devices by using additional tracker classes that are specific to the supported device types. In the current implementation, the `IphoneTracker` detects `Iphone` objects, and the `LegendTracker` detects `Legend` objects. All device types implement the `IDevice` interface (③), which provides properties that are used by the main `Tracker` to keep track of an updated list of current devices. Upon device apparition, movement or removal, the `Tracker` raises relevant events, i.e. `DeviceAdded`, `DeviceUpdated`, `DeviceRemoved`. The `TIDEWindow` listens for such events, and handles them accordingly.

The tracking component provides the application with the means to know when new devices are placed on the table, but also when and where they are moved, and when they are removed entirely. The initial device detection was used in the evaluated prototype, as a trigger for pairing.

However, the tracking of smartphone devices during the application session was de-

activated for the test sessions. This decision was based on feedback gathered during the prototyping sessions, that pointed towards a lack of usefulness. Users expressed that they would be reticent to leave their smartphones on the table for fear of being robbed, or to drag the device across the table for fear of damaging it.

4.3 Replicated UI

This section describes and explains the implementation steps that were taken based on the design decisions from section 3.4 relating to the replicated UI.

To realize the design decision: *DC-1 The smartphone screen is replicated to the tabletop using a standard desktop sharing protocol*, it was decided to use VNC [Richardson et al. 1998], a standard system that uses a rendering-based protocol to allow desktop sharing between remote computers. It works with a network connection between a client and a server, where the client sends input events to the server, who responds by sending a pixel-based copy of its updated screen. It was chosen for its stability and library availability on many platforms.

On the tabletop, the VNC client is integrated to the Surface application using a `VncControl` object. The `VncControl` is a surface user control that is contained within a `DeviceControl` object that provides other UI elements, as shown in figure 4.5. The `DeviceControl` object monitors all contacts, in order to detect the user inputs that are destined to the replicated UI. This is done by checking if the coordinates of the contact are within the image displayed by the `VncControl`. When such a contact is detected, it is forwarded to the `VncControl` object, that relays it further to the smartphone.

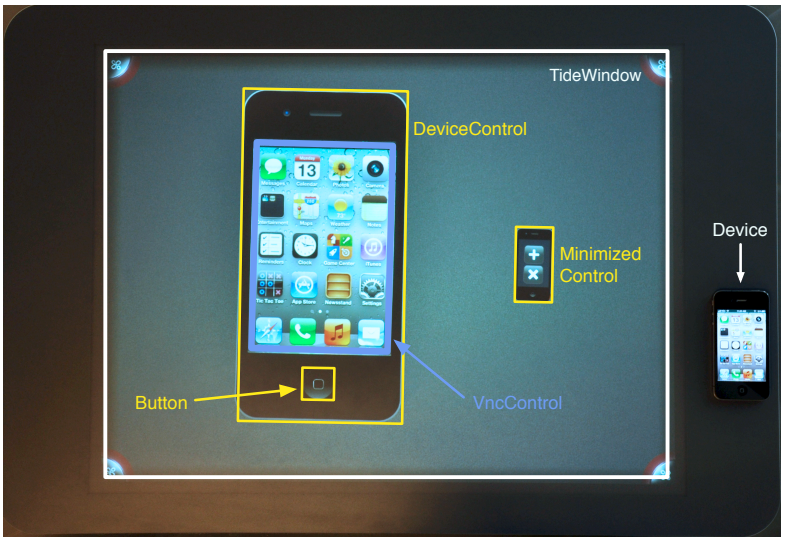

Figure 4.5: TIDE main UI elements.

The `VncControl` uses the VncSharp library to communicate with the VNC server on the smartphone. VncSharp provides methods to transmit mouse and keyboard

events. The server responds with an updated image of the smartphone's screen, that the `VncControl` displays on the tabletop.

On the smartphones, the VNC server relies currently on third-party applications: Veency on iOS, and Droid VNC Server on Android. The reason for this was to allow the easy integration of devices running on different platforms.

4.4 Surface UI

This section describes and explains the implementation steps that were taken based on the design decisions from section 3.4 relating to the surface UI. To realize the design decisions DD-1 to DD-6 (see p. 31), TIDE is implemented using the Surface SDK, which is itself based on WPF (Windows Presentation Foundations).

The elements from figure 4.5 are described below in figure 4.6, which is a class diagram of the TIDE surface UI implementation.

As mentioned for the tracking component, the `TideWindow` (①) is the main application window. It contains all other UI elements and is responsible for most of the application logic. The `DeviceControl` (②) contains the replicated UI of a connected smartphone, and provides the main surface UI elements to the user. The `MinimizedControl` (③) is the minimized version of the `DeviceControl`.

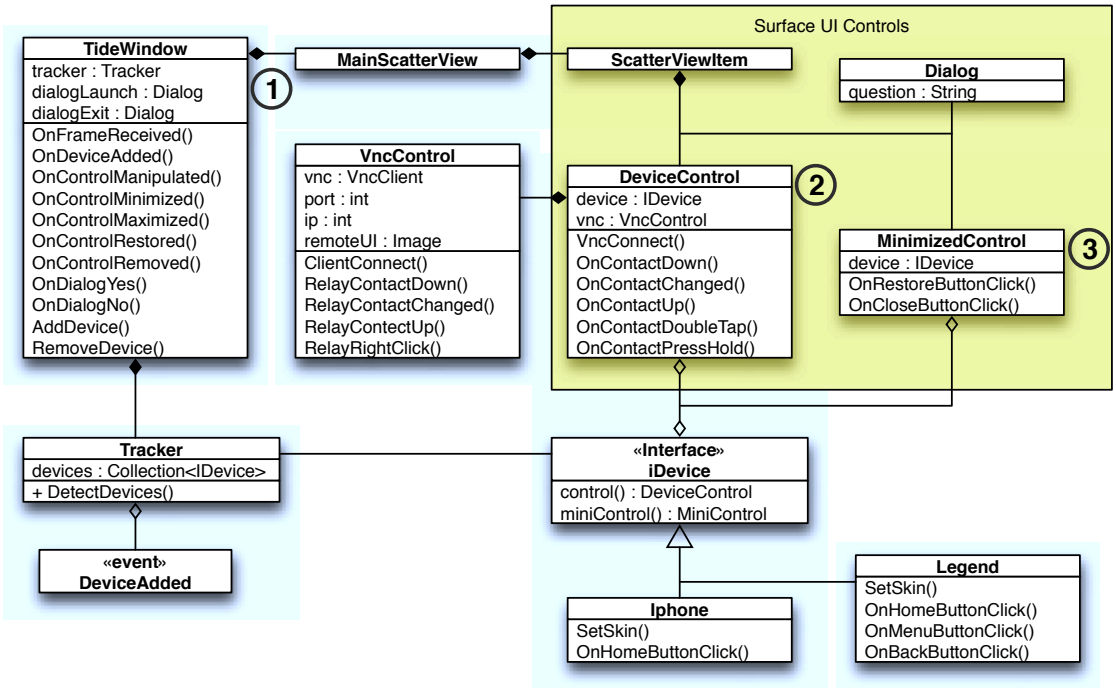

Figure 4.6: Class diagram of the surface UI.

The implementation of the surface UI is based on the Presentation layer of the Surface SDK. To realize the design decision: *DD-2 The replicated UI can be manipulated by*

using touch-based gestures on the surface UI, TIDE uses a `ScatterView` object called `MainScatterView`. The `ScatterView` control displays UI elements that can be manipulated, by wrapping them inside `ScatterViewItem` objects. The `ScatterViewItem` class provides manipulation capabilities to the surface controls, that include move, rotate and resize with one or several fingers. However, some of these capabilities are deactivated for specific controls, such as the `MinimizedControl`, which cannot be resized.

Each virtual device consists of three objects: a `Device` implementing the `IDevice` interface, and two surface controls: the `DeviceControl` and `MinimizedControl` objects.

Upon creation, the `DeviceControl` object gets specific visualization features from its `Device`. For example, an `iPhone` object provides a `SetSkin()` method to show the appropriate visualization, and a `OnHomebuttonClick()` that implements the effect of the home button, specific to the iPhone.

The `DeviceControl` monitors all its touch inputs, and reacts differently depending on their location. If the input is within the `VncControl` image, it is forwarded to the replicated UI. Otherwise, the input is interpreted as a manipulation on the containing `ScatterViewItem` object. The features that are activated by a double tap, press and hold and whole hand tap are also implemented within the `DeviceControl` class.

When a `DeviceControl` is minimized, it raises an event that is handled by the `TideWindow`, causing the `DeviceControl` to be hidden and the `MinimizedControl` to be shown. Similar events are used to handle restoring and closing controls.

An event called `ControlManipulated` is used by the `TideWindow` to monitor the position of the `DeviceControl` and activate the features that depend on position, size or rotation; such as the minimization when the control is dragged to an edge.

Lastly, `Dialog` objects are used to obtain explicit user confirmation when establishing the connection and when disconnecting the device. They are surface controls that are wrapped in `ScatterViewItem` objects, but whose manipulation capabilities are deactivated.

Chapter 5

Evaluation

This chapter presents an evaluation of the solution designed in chapter 3, by way of a usability study of the TIDE prototype. The evaluation focuses on three system aspects: learnability, ease of use and usefulness.

The methods used are based on Designing Interactive Systems by Benyon [2010], and described in section 5.2. The evaluation results are presented in section 5.4.

5.1 Parameters

To demonstrate that the solution proposed is valid, this evaluation shows that TIDE allows spontaneous user interaction, and that it is a useful application. The evaluated parameters are the system's learnability and ease of use, that are major aspects of usability; and the system's usefulness.

Learnability is the capability of a software product to enable the user to learn how to use it.

Ease of use is the self-perceived level of effort that the use of a software product requires of the user.

Usefulness is the quality of being useful, and a requirement for user satisfaction.

Variables

The six *interaction primitives* that were defined in section 3.3 are independent variables. They are the application features that are implemented by TIDE, and the minimum set of features to allow a successful user interaction. However, only five primitives were evaluated: *dragging, rotating, resizing, minimizing and closing.* The sixth primitive, *hiding*, was removed, due to user feedback from the design process that showed it was similar, if not interchangeable, with minimizing.

The interaction techniques that were implemented as a result of the design decisions from section 3.4 are dependent variables. The goal of this evaluation is to discover

which of these interaction techniques are intuitive to the user and which ones the user prefers.

5.2 Methods

Given the strong focus of the present work on user interaction, it was decided to use participant-based methods to evaluate the system. A usability study involving end-users was designed, that regrouped the selected methods in one experiment session.

Discovery To evaluate learnability, users are engaged in discovering the system on their own. With no prior knowledge of the application, participants are given three minutes to *learn by doing*, i.e. explore the system without exterior help. Qualified data is gathered to evaluate how much of the system the users were able to learn.

Guided test To evaluate ease of use, a semi-controlled experiment is designed. Participants are asked to perform specific tasks using the system. Detailed instructions are given to the user during the test. These instructions mention only the independent variables, and the behavior of the user provides data to evaluate the dependent variables. In the present case, the instructions only address the interaction primitives, and it is up to the user to choose between the available interaction techniques. Qualified data is gathered to evaluate the dependent variables.

Questionnaire A questionnaire is used to gather data that supports the evaluation of all three parameters. Participants are asked to assess the learnability, ease of use and usefulness of the system by answering carefully designed questions.

5.3 Experiment

In order to generate qualified data, a user experiment was designed and conducted. Figure 5.1 shows a participant during an experiment session.

Participants

Thirteen participants aged between 25 and 40 were recruited on a voluntary basis. Five out of the thirteen had no technical background, and three out of those five were female participants. All were regular smartphone users.

Apparatus

The experiment sessions were carried out at the PITLab (Pervasive Interaction Technology) at the IT University of Copenhagen. The TIDE prototype was installed on the Microsoft Surface tabletop computer, and used in combination with an iPhone 4 and a HTC Legend smartphone, both equipped with third-party VNC applications. A additional desktop computer was available for filling out online evaluation forms.

Figure 5.1: A volunteer using the TIDE prototype.

Procedure

A session lasts 30 minutes, and involves a participant and the designer. The session is structured by an experiment script, included in appendix E, that is read by the designer to the user.

First, the participant is introduced to the following things:

- the structure of the session (a discovery phase, a guided test and a questionnaire),
- the devices (Microsoft Surface, iPhone and HTC Legend);
- the TIDE prototype:
  - the pairing procedure,
  - the lack of responsiveness of the replicated UI,
  - the difference between the replicated UI and the surface UI;
- the fact that only the system is under test, not the participant.

The participant is then taken through the following phases: discovery, guided test and questionnaire.

Discovering TIDE

The participant has been shown how to pair the phone to the Microsoft Surface. S/he is given three minutes to freely explore the application features of the application. At the end of this discovery phase, the designer fills out the evaluation form *Tide-EF1*, included in appendix F, where he records for each of the interaction primitives, which of the interaction techniques the user discovered.

Testing TIDE

In order for the participant to be able to use TIDE to its full potential, s/he is taught all the available interaction techniques before the beginning of the guided test.

The test consists of two small scenarios, that require of the participant that he performs specific tasks with the application. The first scenario involves the user and the designer playing a game of tic-tac-toe, and the second one involves the user looking up a location on Google Maps and showing it to the designer. The tasks are explained, and detailed instructions are given to the user during the test, but for each instruction, it is up to the user to decide which interaction technique to use. For example, the user is told to minimize the window, but not *how* to activate the command.

This test allows the user to engage several times with all the interaction primitives, each time requiring that s/he chooses a technique of choice. For example, a given user will prefer to drag a window using two hands, while another one will rather use a single finger.

The evaluation data is gathered via the questionnaire filled out during the next phase.

Answering questions about TIDE

The participant is asked to fill out a questionnaire, the evaluation form *Tide-EF2*, included in appendix F. The user is asked to express an opinion about the following things:

- which interaction techniques are his/her favorites?
- is the application easy to learn, easy to use, useful?
- what is best/worst/missing in the application?
- in which situations would the application be useful?

Data

The evaluation data was gathered via the aforementioned forms, then converted to excel files to be processed. The data was processed manually, analyzed, and results were derived. Additional data was generated through runtime logging during the application sessions. The logging data was not used to derive conclusive results, but to confirm the findings presented hereunder.

5.4 Results

Analysis of the results revealed that the system shows good learnability, ease of use and usefulness. These three system aspects were assessed by end users at the end of the experiment sessions. A questionnaire was presented to them, based on the Likert scale, a common approach to scaling responses in survey research. The answers are presented in figure 5.2. They confirm that the majority of the participants found the application easy to learn, easy to use and useful in context.

For each of the evaluated parameters, further results were derived from the experiment data.

Values are expressed as percentage of the participants.	Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree
It is easy to learn how to use the application.	69	23	0	0	8
The application is intuitive.	38	46	0	0	8
It is easy to use the application.	54	38	0	8	0
There are situations in which the application is useful.	54	38	0	0	8

Figure 5.2: End users' assessment of the system's learnability, ease of use and usefulness.

Learnability

Analysis of the logged data from the discovery phases showed that all participants were able to discover at least one technique to perform each of the necessary primitives to a successful interaction with the system. In other words, all participants could find out on their own and within three minutes how to drag, rotate, resize, minimize and close the main application window on the tabletop.

It can thus be concluded that the system implements interaction techniques that comes naturally to most users. Figure 5.3 shows which specific techniques were discovered by the participants. The colored cells contain values above 50%.

Primitive	Technique	Percentage of participants that discovered the technique
Dragging	finger gesture	100
Rotating	1 finger gesture	85
	2 fingers gesture, same hand	38
	2 fingers gesture, both hands	69
Resizing	2 fingers gesture, same hand	69
	2 fingers gesture, both hands	85
Minimizing	resize down	85
	drag to edge	46
	double tap	69
	whole hand tap	0
Closing	minimize then close	100
	drag to corner	0
	press and hold	0

Figure 5.3: Interaction techniques that were discovered by the end users.

The first thing that can be remarked upon is that, although all the basic shape manipulation techniques were easily discovered, the two-handed rotating and resizing gestures scored higher than their single-handed counterpart. This points towards the fact that it is more intuitive to perform these gestures with both hands.

Given that all participants could somehow resize the window, 85% logically found out that they could use it to minimize it as well. In most cases, the user stumbled upon

this feature by accident, while exploring the resizing feature. This is an example of how design can enable the user to *learn by doing*.

Only 46% of the users found out that the window could be minimized by being dragged to the edge, and none discovered the possibility of closing the window by bringing it to a corner. These techniques were implemented because of their success among end users during the design sessions based on low-fidelity prototypes. They were expected to be easy to discover because of their consistency with how people would interact with a document on a normal table. However, it seems that this success was prompted by the fact that the prototypes were actual pieces of paper, which could only be minimized/hidden/closed by being physically put away.

Similarly, none discovered that the window could be minimized by tapping it with the whole hand, even though the gesture was thought to be natural due to its resemblance to the action of hiding a document on a normal table. It can be remarked that this gesture is not used on common devices such as smartphones or personal computers. It is a gesture that requires a lot of space and is therefore specific to large interactive surfaces, and thus not part of the common IT knowledge of most users.

69% of the participants found out about the double tap, which is a large amount, considering the fact that the feature is hidden, i.e. it can not be discovered by accident while exploring the manipulation possibilities, but has to be explicitly attempted. It shows that, as opposed to the whole hand tap, the double tap is part of the common IT knowledge.

Pressing and holding on the window to close it is also a hidden feature, and it was discovered by none. However, the results concerning the primitive *closing* are not considered conclusive, due to the fact that users did not attempt to discover different ways to close the application during the exploratory phase, but only performed it once to terminate the session.

Ease of use

To determine which interaction techniques were the most efficient, participants were asked to select their favorites. The answers are presented in figure 5.4, where colored cells contain values above 50%. This study only addresses the primitives for which multiple techniques were implemented.[1] Participants were not limited to select one technique per primitive.

It is interesting to see that users preferred to resize and rotate by using a two-handed gesture, even though it could be done single-handedly. An explanation might be that the two-handed gesture seems to provide more control to the user, especially when s/he has little experience with touch shape manipulation on large surfaces.

To minimize, the favored techniques were double tap and the whole hand tap. Resizing the window down to minimize it was only favored by 8% of the users, even though it was the most easily discovered technique, whereas using a whole hand tap could not

[1] Two techniques were not included in the study due to a human mistake: rotating with 1 finger and closing with press and hold.

Primitive	Technique	Percentage of participants that favor the technique
Rotating	2 fingers gesture, same hand	38
	2 fingers gesture, both hands	54
Resizing	2 fingers gesture, same hand	54
	2 fingers gesture, both hands	62
Minimizing	resize down	8
	drag to edge	15
	double tap	69
	whole hand tap	54
Closing	minimize then close	62
	drag to corner	38

Figure 5.4: End users' favorite interaction techniques.

be discovered, but ended up among the most appreciated ones. This shows that familiar and efficient features are not necessarily the same, and that both are needed. The familiar gestures make the system easy to learn, while the efficient ones, such as the double tap or the whole hand tap, make it easy to use.

The gestures that involved dragging the window to an edge or corner of the tabletop did not score as high as expected among the participants. The hypothesis that such gestures would be appreciated because of their consistency with normal table interaction, i.e. their similarity with the action of putting a document away, was not confirmed. The first explanation that comes to mind is that a dragging gesture requires more time and space to be performed than, say, a double tap, which is quicker and can be done on the spot.

Usefulness

The participants expressed the opinion that the application would be useful in specific situations. As part of the questionnaire, they were asked which activities they would consider using the system for, and in which context. The answers are presented in figure 5.5, where colored cells contain values above 50%.

The participants were presented with a list of 7 activities that could be performed on a smartphone. For each of those activities, the users were asked first if they would use their smartphone to do them, then in which context they would consider using the application instead. The contexts suggested were: alone in a private space, alone in a public space, together with friends or colleagues.

The results show that whether a user would choose the application or not depends mostly on privacy and trust.

The first realization is that the system is not a proper platform for activities that involve private data. There are two activities that most users do on smartphones, but very few would do on a tabletop, whatever the context: reading and writing emails or sms messages. Both activities involve viewing text that would potentially contain

Values are expressed as percentage of the participants.	Which activities do you use your smartphone for?	For which activities would you consider using the application instead of your smartphone,		
		if alone in a private space?	if alone in a public space?	If together with friends / colleagues?
Browse the internet	77	69	46	77
Read / write emails	69	46	23	23
Look at pictures	77	77	23	100
Read / write sms messages	92	15	8	8
Play games	62	69	46	69
Look up a location on a map	92	54	62	92
Read documents	31	62	38	62

Figure 5.5: The contexts in which the application is useful.

personal conversations. It is therefore natural, that users would not want to view this data on the large open display of a tabletop.

The second realization is that there are indeed a series of activities for which the system is suitable. They are: browsing the internet, looking at pictures, playing games and looking at a map. However, most users would only use the application for these activities in a *trusted* context, i.e. in a private space, or together with trusted acquaintances. In a public context, the only activity that a majority of users would still consider doing with the system is looking up a location on a map. Again, the explanation to this can probably be found in the fact that tabletops are very public devices, in the sense that they can be seen by any person in range. Unlike smartphones, laptops, or even desktops, whose screen can be rotated more or less freely, tabletop displays are there for the world to see. Thus, users are reticent to use them in public, even for semi-private occupations such as browsing the internet or playing games.

Last but not least is the realization that the system has most potential for the activity of reading documents. It is the only activity that only a minority of users (31%) would do on a smartphone, which confirms the hypothesis that smartphone screens are too small for this type of graphically intense data. 62% of the participants would consider using the application for reading documents, though only in a trusted context. Thus, the system seems to address a real need.

Chapter 6

Discussion

This thesis addressed the problem of building a composite device between a smartphone and a tabletop, by focusing on designing a system that would enable the user to spontaneously interact with it. This approach implied addressing two types of challenges. The first one relates to the design of the user interaction, and is presented in section 6.1. The second consists of the technical system aspects, and is presented in section 6.2.

6.1 User interaction in context

To achieve that, one of the design strategy that was followed was to design a system that was consistent with the opportunities and constraints that are implied by the characteristics of tabletops.

This lead to the realization that the social nature of tabletops have precedence over many other user concerns. Due to the form of their interactive surface, i.e. horizontal and approachable from all angles, tabletops are social artifacts, but also very public. The social aspect implies that they are ideal for situations where multiple users that know each other perform an activity together. Examples of such social situations include a work meeting, a groups of friends in a bar or a family in their home. The public aspect, however, addresses the fact that their display is open for all to see, and thus a constraint to any form of private use. The implications include that a person alone would be reticent to use a tabletop for anything more private than looking at a map, and that two strangers would be reticent to share a tabletop to perform parallel activities. Tabletops seem thus better fitted for prolonged interactions in trusted or semi-trusted environments, such as in a home, a workplace, or among friends. The most relevant implication for a system such as TIDE is a limitation of the contexts in which the application would prove useful.

On a more interactional level, it proved interesting to witness that end users were able to quickly learn the touch-based shape manipulation techniques that are specific to tabletops. Examples of those are rotating and resizing UI elements. Even though those manipulations are not conventionally supported on smartphones, whose small screen size makes them irrelevant, the directness of the touch interaction is such that users were able to discover them almost instinctively. This confirms the idea that the tabletop

experience is one that appeals even to the less computer literate. This work investigated the idea that interaction techniques could be discovered, that would be familiar to the user, due to their consistency with normal table interaction. The main example is the action of minimizing or closing the application by moving the window to the edge (or corner), similar to the way one would remove a real document from a table. However, the evaluation did not confirm this idea, and showed that the users were generally very conscious of not interacting with a normal table. When having to improvise, they would use knowledge related to computers in general, and to common touch-based devices such as smartphones in particular, before relying on the obvious table-like aspect of the device. A good example of this behavior is that users would ask before putting a cup or a book down on the table.

Due to the early involvement of users in the design process, it was possible to determine which interaction techniques were most familiar to them. Users with different backgrounds would naturally prefer different techniques. The decision to implement several techniques to activate the same action feature proved to be a simple, but efficient way of targeting several types of users at once, thus improving the system's learnability and ease of use.

6.2 Technical challenges

VNC [Richardson et al. 1998] was chosen to enable UI replication between the devices. This decision allowed the quick development of a functioning prototype that supports various smartphone models. However, VNC comes with a set of limitations, the most relevant being its lack of responsiveness. VNC was designed to remotely access the UI of a desktop computer. Its architecture includes a server, that is located on the remote computer, and implements most of the application logic. The server is thus accessed by a lightweight client that is only responsible for transmitting inputs, and receiving the updated UI copies. This architecture is not suited to the configuration investigated by this work. VNC requires the server to run on the smartphone whose resources are limited, while the tabletop only runs a small client. The result was a system whose replicated UI was only poorly responsive. The benefit of a well-known system like VNC is that it is available on most platforms, and could be for example installed both on the iPhone and on the HTC Legend. However, this work confirms that the rendering-based protocol it uses is too heavy for a smartphone to handle. For a real-world deployment, another technical solution would have to be used, to implement a responsive UI replication. This challenge is being addressed by current research, with projects such as XICE [Arthur and Olsen 2011], which is a programming framework that allows a more flexible form of UI distribution for the annexation of displays by nomadic users.

Another shortcoming of VNC is that it constrains the configuration to UI replication, and its pixel-based protocol generates a replicated UI that can difficultly be adapted to various display sizes and resolutions. This work has revealed a range of use cases that could be addressed by this type of composite device, but for which UI replication is not

sufficient. An example is the possibility of transferring whole applications or processes to the tabletop, while keeping the smartphone independent.

This project demonstrated that computer vision could be used to track the position of smartphones on a tabletop display. However, it also showed that this approach has some limitations. In short, smartphones can only be detected if the system "sees" them—e.g. in the case of an infrared based system, black phones are invisible—and if the application "knows" them, i.e. the features specific to the phone must be encoded in the application. Moreover, visual tracking does not allow the identification of a specific device. It seems that the upcoming NFC technologies (Near Field Communication) would allow a more efficient way of detecting and identifying mobile devices, though they would not solve the problem of precisely tracking their location on the display.

Chapter 7

Conclusion

This thesis addressed the limitations that are induced by the small screen size of smartphones, when used to display graphically intense content and in social situations. The approach followed was to use *device composition* to build a system that would integrate a smartphone to a tabletop, and provide a *spontaneous user interaction* to novice users. This approach was inspired by related research efforts that addressed the issue of enabling nomadic users to annex the large displays that are available in the environment [Want et al. 2002], [Arthur and Olsen 2011], [Baur et al. 2012].

The solution presented is a composite device called *TIDE (Tabletop Interactive Display Extension)*, that replicates the UI of a smartphone to a tabletop display. TIDE was designed based on a user-centered approach that focused on discovering the interaction techniques that are familiar to most end users, in order to create a touch-based user interface that seems intuitive to the user.

The implementation of the prototype runs on the Microsoft Surface tabletop computer and supports multiple simultaneous smartphones. TIDE includes a visual tracking mechanism that is used to detect smartphone devices and track their location on the display during the application session. The UI replication is based on VNC [Richardson et al. 1998], which allows the system to support most smartphone models, but otherwise proved to be unable to provide sufficient responsiveness for the replicated UI.

TIDE was evaluated by way of a participant-based usability study whose results showed that the system is highly learnable and easy to use. The evaluation also revealed that the system is useful to perform specific activities such as reading documents and looking at pictures, though only in trusted and semi-trusted environments.

Bibliography

Adobe Systems. Flash. http://get.adobe.com/flashplayer/, 2012. (accessed 2/12).

Apple. Bonjour. https://developer.apple.com/opensource/, 2012a. (accessed 2/12).

Apple. iphone. http://www.apple.com/iphone/, 2012b. (accessed 2/12).

Richard Arthur and Dan R. Olsen, Jr. Xice windowing toolkit: Seamless display annexation. *ACM Trans. Comput.-Hum. Interact.*, 18:14:1–14:46, August 2011. ISSN 1073-0516. doi: http://doi.acm.org/10.1145/1993060.1993064. URL http://doi.acm.org/10.1145/1993060.1993064.

Rafael Ballagas, Michael Rohs, and Jennifer G. Sheridan. Sweep and point and shoot: phonecam-based interactions for large public displays. In *CHI '05 extended abstracts on Human factors in computing systems*, CHI EA '05, pages 1200–1203, New York, NY, USA, 2005. ACM. ISBN 1-59593-002-7. doi: http://doi.acm.org/10.1145/1056808.1056876. URL http://doi.acm.org/10.1145/1056808.1056876.

Jakob E. Bardram, Christina Fuglsang, and Simon C. Pedersen. Compute: a runtime infrastructure for device composition. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI '10, pages 111–118, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0076-6. doi: http://doi.acm.org/10.1145/1842993.1843014. URL http://doi.acm.org/10.1145/1842993.1843014.

Dominikus Baur, Sebastian Boring, and Steven Feiner. Virtual projection: Exploring optical projection as a metaphor for multi-device interaction. CHI '12. ACM, 2012.

David Benyon. *Designing interactive systems: a comprehensive guide to HCI and interaction design*. Addison Wesley, 2010. URL http://books.google.com/books?id=P923PwAACAAJ.

Thomas Berglund and Michael Thomassen. Non-anonymous user interaction on tabletop displays. Master's thesis, IT University of Copenhagen, Denmark, 2011.

Fadi Chehimi and Enrico Rukzio. Throw your photos: an intuitive approach for sharing between mobile phones and interactive tables. In *Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing*, Ubicomp '10,

pages 443–444, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0283-8. doi: 10.1145/1864431.1864479. URL http://doi.acm.org/10.1145/1864431.1864479.

Paul Clifton, Ali Mazalek, Jon Sanford, Claudia Rébola, Seunghyun Lee, and Natasha Powell. Sketchtop: design collaboration on a multi-touch tabletop. In *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*, TEI '11, pages 333–336, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0478-8. doi: http://doi.acm.org/10.1145/1935701.1935778. URL http://doi.acm.org/10.1145/1935701.1935778.

Florian Echtler and Gudrun Klinker. Tracking mobile phones on interactive tabletops. In *2nd Workshop on Mobile and Embedded Interactive Systems*, MEIS'08, September 2008.

W. Keith Edwards, Mark W. Newman, Jana Z. Sedivy, and Trevor F. Smith. Experiences with recombinant computing: Exploring ad hoc interoperability in evolving digital networks. *ACM Trans. Comput.-Hum. Interact.*, 16:3:1–3:44, April 2009. ISSN 1073-0516. doi: http://doi.acm.org/10.1145/1502800.1502803. URL http://doi.acm.org/10.1145/1502800.1502803.

Morten Esbensen, Stina Matthiesen, and Lise M. Petersen. Pour images - an interactive system for sharing images. 2010.

T. Geller. Interactive tabletop exhibits in museums and galleries. *Computer Graphics and Applications, IEEE*, 26(5):6 – 11, sept.-oct. 2006. ISSN 0272-1716. doi: 10.1109/MCG.2006.111.

Tony Gjerlufsen, Clemens Nylandsted Klokmose, James Eagan, Clément Pillias, and Michel Beaudouin-Lafon. Shared substance: developing flexible multi-surface applications. In *PART 5 ——– Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 3383–3392, New York, NY, USA, 2011. ACM. doi: http://doi.acm.org/10.1145/1979442.1979446. URL http://doi.acm.org/10.1145/1979442.1979446.

James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. *Java Language Specification*. The Java Series. Addison-Wesley Longman Publishing Co., Inc., 2nd edition edition, 2000.

Lars Holmquist, Friedemann Mattern, Bernt Schiele, Petteri Alahuhta, Michael Beigl5, and Hans-W. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In Gregory Abowd, Barry Brumitt, and Steven Shafer, editors, *Ubicomp 2001: Ubiquitous Computing*, volume 2201 of *Lecture Notes in Computer Science*, pages 116–122. Springer Berlin / Heidelberg, 2001. ISBN 978-3-540-42614-1. URL http://dx.doi.org/10.1007/3-540-45427-6_10. 10.1007/3-540-45427-6_10.

Seth Hunter, Pattie Maes, Stacey Scott, and Henry Kaufman. Memtable: an integrated system for capture and recall of shared histories in group workspaces. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 3305–3314, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9. doi: http://doi.acm.org/10.1145/1978942.1979432. URL `http://doi.acm.org/10.1145/1978942.1979432`.

IETF. Zeroconf. `http://www.zeroconf.org/`, 2012. (accessed 2/12).

B. Johanson, A. Fox, and T. Winograd. The interactive workspaces project: experiences with ubiquitous computing rooms. *Pervasive Computing, IEEE*, 1(2):67 – 74, apr-jun 2002. ISSN 1536-1268. doi: 10.1109/MPRV.2002.1012339.

Microsoft. Microsoft surface. `http://www.microsoft.com/surface/`, 2012a. (accessed 2/12).

Microsoft. Microsoft surface at royal bank of canada. `http://www.microsoft.com/casestudies/Microsoft-Surface/Royal-Bank-of-Canada/Royal-Bank-of-Canada-delights-customers-with-innovative-Microsoft-Surface-experience/4000011029`, 2012b. (accessed 2/12).

Microsoft. Silverlight. `http://www.microsoft.com/silverlight/`, 2012c. (accessed 2/12).

Alex Olwal and Andrew D. Wilson. Surfacefusion: unobtrusive tracking of everyday objects in tangible user interfaces. In *Proceedings of graphics interface 2008*, GI '08, pages 235–242, Toronto, Ont., Canada, Canada, 2008. Canadian Information Processing Society. ISBN 978-1-56881-423-0. URL `http://dl.acm.org/citation.cfm?id=1375714.1375754`.

OpenCV. Opencv. `http://opencv.willowgarage.com/wiki/`, 2012. (accessed 2/12).

Shwetak N. Patel, Jeffrey S. Pierce, and Gregory D. Abowd. A gesture-based authentication scheme for untrusted public terminals. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, UIST '04, pages 157–160, New York, NY, USA, 2004. ACM. ISBN 1-58113-957-8. doi: http://doi.acm.org/10.1145/1029632.1029658. URL `http://doi.acm.org/10.1145/1029632.1029658`.

Trevor Pering, Roy Want, Barbara Rosario, Shivani Sud, and Kent Lyons. Enabling pervasive collaboration with platform composition. In *Pervasive Computing*, volume 5538 of *Lecture Notes in Computer Science*, pages 184–201, 2009. doi: 10.1007/978-3-642-01516-8_14.

Jun Rekimoto and Masanori Saitoh. Augmented surfaces: a spatially continuous work space for hybrid computing environments. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, CHI '99, pages 378–385, New York, NY, USA, 1999. ACM. ISBN 0-201-48559-1. doi: http://doi.acm.org/10.1145/302979.303113. URL `http://doi.acm.org/10.1145/302979.303113`.

Jun Rekimoto, Yuji Ayatsuka, and Michimune Kohno. Synctap: An interaction technique for mobile networking. In Luca Chittaro, editor, *Human-Computer Interaction with Mobile Devices and Services*, volume 2795 of *Lecture Notes in Computer Science*, pages 104–115. Springer Berlin / Heidelberg, 2003. ISBN 978-3-540-40821-5. URL `http://dx.doi.org/10.1007/978-3-540-45233-1_9`. 10.1007/978-3-540-45233-1_9.

Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, and Andy Hopper. Virtual network computing. *Internet Computing, IEEE*, 2(1):33–38, 1998.

M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, and K. Nahrstedt. A middleware infrastructure for active spaces. *Pervasive Computing, IEEE*, 1(4):74 – 83, oct-dec 2002. ISSN 1536-1268. doi: 10.1109/MPRV.2002.1158281.

Robert W. Scheifler and Jim Gettys. The x window system. *ACM Trans. Graph.*, 5: 79–109, April 1986. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/22949.24053. URL `http://doi.acm.org/10.1145/22949.24053`.

Dominik Schmidt, Fadi Chehimi, Enrico Rukzio, and Hans Gellersen. Phonetouch: a technique for direct phone interaction on surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 13–16, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0271-5. doi: http://doi.acm.org/10.1145/1866029.1866034. URL `http://doi.acm.org/10.1145/1866029.1866034`.

David Scott, Richard Sharp, Anil Madhavapeddy, and Eben Upton. Using visual tags to bypass bluetooth device discovery. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9:41–53, January 2005. ISSN 1559-1662. doi: http://doi.acm.org/10.1145/1055959.1055965. URL `http://doi.acm.org/10.1145/1055959.1055965`.

Bluetooth SIG. Bluetooth. `http://www.bluetooth.org/apps/content/`, 2012. (accessed 2/12).

Norbert A. Streitz, Jörg Geissler, Torsten Holmer, Shin'ichi Konomi, Christian Müller-Tomfelde, Wolfgang Reischl, Petra Rexroth, Peter Seitz, and Ralf Steinmetz. i-land: an interactive landscape for creativity and innovation. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, CHI '99, pages 120–127, New York, NY, USA, 1999. ACM. ISBN 0-201-48559-1. doi: http://doi.acm.org/10.1145/302979.303010. URL `http://doi.acm.org/10.1145/302979.303010`.

Peter Tandler. Software infrastructure for ubiquitous computing environments: Supporting synchronous collaboration with heterogeneous devices. In *Ubicomp 2001: Ubiquitous Computing*, volume 2201 of *Lecture Notes in Computer Science*, pages 96–115, 2001. doi: 10.1007/3-540-45427-6_9.

Bernhard Tritsch. *Microsoft Windows Server 2003 Terminal Services*. Microsoft Press, Redmond, WA, USA, 2003. ISBN 0735619042.

UPnP Forum. Upnp. `http://upnp.org/sdcps-and-certification/resources/whitepapers/`, 2012. (accessed 2/12).

VncSharp. Vncsharp. `http://cdot.senecac.on.ca/projects/vncsharp/`, 2012. (accessed 2/12).

Roy Want, Trevor Pering, Gunner Danneels, Muthu Kumar, Murali Sundar, and John Light. The personal server: Changing the way we think about ubiquitous computing. In Gaetano Borriello and Lars Holmquist, editors, *UbiComp 2002: Ubiquitous Computing*, volume 2498 of *Lecture Notes in Computer Science*, pages 223–230. Springer Berlin / Heidelberg, 2002. ISBN 978-3-540-44267-7. URL `http://dx.doi.org/10.1007/3-540-45809-3_15`. 10.1007/3-540-45809-3_15.

Pierre Wellner. Interacting with paper on the digitaldesk. *Commun. ACM*, 36:87–96, July 1993. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/159544.159630. URL `http://doi.acm.org/10.1145/159544.159630`.

Andrew D. Wilson and Raman Sarin. Bluetable: connecting wireless mobile devices on interactive surfaces using vision-based handshaking. In *Proceedings of Graphics Interface 2007*, GI '07, pages 119–125, New York, NY, USA, 2007. ACM. ISBN 978-1-56881-337-0. doi: http://doi.acm.org/10.1145/1268517.1268539. URL `http://doi.acm.org/10.1145/1268517.1268539`.

Christian Winkler, Christian Reinartz, Diana Nowacka, and Enrico Rukzio. Interactive phone call: synchronous remote collaboration and projected interactive surfaces. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '11, pages 61–70, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0871-7. doi: http://doi.acm.org/10.1145/2076354.2076367. URL `http://doi.acm.org/10.1145/2076354.2076367`.

Jacob O. Wobbrock, Meredith Ringel Morris, and 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, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-246-7. doi: http://doi.acm.org/10.1145/1518701.1518866. URL `http://doi.acm.org/10.1145/1518701.1518866`.

Appendix A

TIDE video and code

A short video describing the system and the TIDE source code are available online at:

`http://www.itu.dk/pit/?n=Projects.Devicecomposition`

Appendix B

Design - scenarios

The coffee shop: single user on a public tabletop

Alice is sitting in a Coffee Shop, waiting for her friend Bob. The table is an interactive surface, which allows her to order her drink via the digital interface.

She takes her phone and notebook out of her purse and places them on the table. A dialog pops up on her smartphone, asking her if she wants to establish the connection between the two devices. Alice confirms this by a simple tap. A menu appears on the table beside her phone. Alice taps the 'Connect' button, and her phone's display appears on the table beside her phone.

She resizes the window to her convenience, and moves it closer to her by sliding her phone on the surface. Alice accesses her phone's applications, and starts typing an email.

When Bob arrives, Alice minimizes the surface application, keeping her phone in place. Bob orders a drink and they start catching up. After a while, Bob leaves. Alice restores the application, finishes up her email, and exits the application by lifting the phone off the table.

The meeting: multiple users on a public tabletop

Jim, Jack and Jill are having a meeting about the development of a product. They are sitting around an interactive table, with different artifacts, including paper, pens, computing devices and coffee cups. Jill is responsible for the meeting's agenda, which is stored on her smartphone.

Jill places her phone on the top right corner of the table and establishes a UI transfer. She uses the 'Grab' button to drag the display window to the left of the phone where there is space. She opens the document containing the agenda, and uses the 'Resize' corner of the window to enlarge the display, so as to allow convenient visual reference for all present.

It is now time for Jack to present a diagram of the development process. He switches on his tablet computer, opens said diagram, and places the tablet on the table for the others to see. The screen is however too small, so Jack decides instead to use the UI transfer application. With a single tap on the 'Grab' button, Jack pins the UI to the

table, allowing him to remove the physical tablet while keeping the transferred display active. By using the 'Resize' corner, he enlarges and flips the orientation of the window to a landscape view. By the use of a double touch on the 'Grab' tab and the active window, Jack rotates the display to a convenient position, and presents the diagram to his colleagues. When done, Jack minimizes the window by tapping a button. The window takes the shape of an active icon, ready to be restored or closed as convenient. When the meeting is over, Jack taps the 'Close' tab on the icon to exit the application.

The office: single user on a private tabletop

It is monday morning and Bill arrives at his office. His working desk is made up of an interactive table, extended with a vertical screen, mouse and keyboard. On it are various physical objects, including a stack of papers, some books, pens, an empty cup and a lamp.

Bill wakes the tabletop up from its standby state by simply placing his smartphone on it. The devices know each other, so a UI transfer is automatically launched. Bill uses a widget on his phone to push application widgets to the table space. Bill places his calendar up in one corner, together with his Skype widget.

After reading through his mail on the vertical screen, Bill starts typing an answer using the keyboard. He needs to refer to a document that is stored on his phone. Bill uses the 'Grab' button beside his phone to attach the display beside the device. He enlarges the window and moves the display to a convenient location by sliding the phone. Bill types on..

Suddenly the phone rings. Bill taps the 'Grab' button to effectively pin all applications and UI display to the tabletop, allowing him to pick up the phone without interrupting the UI transfer.

Appendix C

Design - experiment script

C.1 Introduction

"You are here to participate in a short experiment whose purpose is to assist in the design of an application called Displex. The experiment will last about 30 minutes and is being recorded, both as audio and video. I will give you instructions by reading from this script. After an introduction you will be able to ask questions before we start the experiment.

First, let me introduce Displex. Displex is an application that connects a smartphone (iPhone) with a tabletop computer (Microsoft Surface). It allows you to interact with your phone on a larger screen, by transferring the display of your phone to a window on the screen of the interactive surface. Do you understand the basic concept of the application?

During this experiment, I will ask you to perform a task using the application. This will lead you to perform a number of actions that use the basic features of Displex. For each action, there will be 3 steps:

- First, I will explain the action, and show you its effect on this screen
- Second, I will ask you to describe to me how you would suggest performing this action with the user interface of Displex.
- Third, I will give you 3 suggestions of how to perform the action, and ask you to order them according to your preference.

This experiment is based on prototypes, which means that we will use the available paper representations in order to describe the user interface of Displex. There are iPhone screenshots and different types of buttons and controls. Paper, pen and scissors are available for building your own prototypes if necessary. You are welcome to draw on the prototypes if you want. We will also use the iPhone, the MS Surface, and of course verbal communication.
This iPhone screenshot printout is a representation of the main window of the Displex application. The idea is that you can interact with this window in exactly the same way as you would interact with your phone's screen. For example, by tapping the Photos

icon, you would launch the Photos application, and if you are viewing a picture, by performing a two finger pinching gesture, you could zoom on the picture. At the same time, we need a way to manipulate the window, and that is what this experiment is going to focus on.
Do you have any questions concerning the general course of the experiment?
Let us begin. Your general task is to write an email to a friend using your iPhone and the Displex application on the Microsoft Surface. We will talk about 7 basic actions."

Example: Pairing

This first action is only an example, meaning that I will go through all the steps myself. The action is called pairing.
Scenario: In order to use Displex, I have to pair my iPhone with the Surface and launch the Displex application. Here is a visual representation of the effect of this application. (SHOW PPF)
Suggestion:
I asked my advisor Juan, and his suggestion was to launch Displex on the iPhone, then search for available surface computers within the application, and connect to the Surface.
Selection:

A The application launches automatically when the smartphone is placed on the surface, and a dialog window appears on the smartphone, offering the user to establish the connection.

B The application launches automatically when the smartphone is placed on the surface, and 2 dialog windows appear, first on the surface, then on the smartphone, offering the user to establish the connection.

C The application launches automatically when the smartphone is close enough to the surface, and a dialog window appears on the surface, offering the user to establish the connection.

Juans order of preference was B, A, C. What about you?
(SHOW ALL PPF) There are 6 actions left, and I will now show you visual representations for each of those actions.

C.2 Experiment

Dragging

Scenario: Your iPhone screen is now active on the surface, and you need to move it closer to yourself. Therefore, you drag the window across the surface.
Suggestion
Selection (group 1)

A By performing a one finger dragging gesture on a specific action tab. (1)

B By performing a one finger dragging gesture on the action bar. (2)

C By tapping a tab to render the window inactive, then performing a one finger dragging gesture anywhere on the window. (3)

Selection (group 2)

A By performing a one finger dragging gesture on the active border of the window. (4)

B By performing a one finger dragging gesture on the active border (excl. corners) of the window. (5)

C By holding a finger on a specific tab, and using another finger to tap a destination target to move the window to. (6)

Rotating

Scenario: the application window is not oriented correctly, so you need to rotate it to the correct orientation.
Suggestion
Selection (group 1)

A By performing a two finger touch rotating gesture on the action bar. (2)

B By tapping a tab to render the window inactive, then performing a two finger touch rotating gesture anywhere on the window. (3)

C By performing a two finger touch rotating gesture on the active border. (4)

Selection (group 2)

A By performing a two finger touch rotating gesture on the active border. (5)

B By performing a one finger dragging gesture on a corner of the window. (6)

C By performing a two finger touch rotating gesture with one finger placed on a specific tab, and the other anywhere on the window. (1)

Resizing

Scenario: Now you open the Safari App by taping on the correct icon, but the window is too small for you to type an email, so you resize it to make it bigger.
Suggestion
Selection (group 1)

A By tapping a tab to render the window inactive, then performing a two finger pinching gesture anywhere on the window. (3)

B By performing a two finger pinching gesture on the active border. (4)

C By performing a one finger dragging gesture on an active corner. (5)

Selection (group 2)

A By pulling the window apart with both whole hands. (6)

B By performing a one finger dragging gesture on a specific tab. (1)

C By performing a two finger pinching gesture on the action bar. (2)

Minimizing

Scenario: Before you start writing your email, you want to verify some facts in a document. You decide to minimize the Displex application to make room for the paper, and be able to restore the window to its previous state soon after.
Suggestion
Selection (group 1)

A By double tapping the active border. (4)

B By using Resizing on an active corner to reduce the window until it snaps to icon shape. (5)

C By dragging the window to the bottom of the surface. (6)

Selection (group 2)

A By tapping a specific tab. (1)

B By double tapping the action bar. (2)

C By tapping a tab to render the window inactive, then performing a specific gesture anywhere on the window. (3)

Hiding

Scenario: Later, you are writing another email of a personal nature, and one of colleagues is approaching. You wish to quickly and temporarily hide what you are doing.
Suggestion
Selection (group 1)

A By double tapping an active corner. (5)

B By placing and holding a full hand on the window. (6)

C By tapping a specific tab. (1)

Selection (group 2)

A By performing a specific gesture on the action bar. (2)

B By tapping a tab to render the window inactive. (3)

C By double tapping the active border. (4)

Exiting

Scenario: Finally, you are finished and want to leave. You exit the Displex application.
Suggestion
Selection (group 1)

A By dragging the window to a specific location on the surface. (6)

B By tapping a specific tab. (1)

C By performing a specific gesture on the action bar. (2)

Selection (group 2)

A By tapping a tab to render the window inactive, then performing a specific gesture on the window. (3)

B By using Resizing on the active border to Minimize, then tapping a specific tab. (4)

C By using Resizing on an active corner to Minimize, then tapping a specific tab. (5)

Appendix D

Design - experiment form

TIDE Early Usability Study (1)

	Action Tabs	Action Bar	Active /Inactive Window	Active Border	Active Border + Corners	Other	Open Suggestion
Dragging							
Rotating							
Resizing							
Minimizing							
Hiding							
Exiting							

Figure D.1: Form used to gather participant answers during the preliminary usability study.

Appendix E

Evaluation - experiment script

Introduction speech to the user:

"The experiment is divided in 3 parts. The first part will allow you to discover the application and learn to use it. In the second part, I will guide you through two scenarios where you will play a role, and have to do specific things with the application. The third part is an online questionnaire that you will fill out on the computer.

But first, let me present what we will be working with. This is the Microsoft Surface, a tabletop computer with a multitouch screen. It works like a big smartphone, please try.

We will use an application called TIDE. This application allows you to connect a smartphone to the tabletop.

TIDE is a only a prototype, meaning that there are things that are not finished, and some that are not implemented, so please be patient in case of unexpected behavior. Especially, please be aware of the following things:

- to pair the phone with the table, you place it, and the table detects it. what happens is that sometimes the table does not see the phone, so you have to move it around until it works. Sometimes the table will think it sees a new phone, so you will have to click 'no'.
- when you touch inside the window, you interact with the phone. But the application is very slow. So the golden rule is to only give one input at a time, so you atom then you wait until the screen refreshes, then you continue.
- as you can see, there are two elements to the application. Inside the window is for using the phone, but you can also manipulate the window. This is what this experiment focuses on.
- Finally, one important thing, remember that it is the system that is under test here, not you. Nothing you do will be a mistake, only the application will make mistakes.

Let us start."

E.1 Discovering Tide (10 min)

Free exploration

“ You have a few minutes to discover the application on your own. Here is a list of what you can try to do with the application. Please, comment aloud on what you are doing, and what is happening. ”

The list of suggestions should be apparent, for example hanging on the wall.

Features (designer fills out EF1)

“ Together, we will go through all the application features, and I will teach you the features that you haven't discovered on your own. I fill out this form, where I register which features you discovered yourself. ”

E.2 Using Tide (10 min)

“ Now we will do a little bit of role playing. We will go through two scenarios, where I will direct you to do specific things with the application. I will tell you exactly what to do, step by step, but it is up to you to decide how to do it. ”

Gaming

“ The first scenario is about two friends in a waiting room. They are bored, and decide to play a game of tic-tac-toe to pass the time. ”

1. connect the iPhone to the tabletop
2. launch the game called 'tic tac toe'
3. start a 2 players game
4. enlarge the window and position it so we can play
5. now we play
6. exit the game
7. close the application

Browsing

“ The second scenario is about two friends that are planning a surprise dinner for the birthday of a third friend named Bob. You will look up a place called cafe alma on the internet, and you will show me where it is and how to go there from here. When Bob approaches, you should hide what we are doing. ”

1. connect the HTC Legend to the tabletop
2. launch the app called Internet
3. adjust the position and size of the window

4. type on the google search bar
5. type 'cafe alma' and hit search
6. tap 'contact'
7. Bob is passing by, minimize the window
8. restore the window
9. use your phone to zoom on the address
10. go back to home screen
11. launch the app called Maps
12. hit search
13. type 'isafjordsgade 5' and hit enter
14. zoom in on the address
15. find ITU on the map
16. adjust the position and size of the window to show it to me
17. Bob is here, minimize the window
18. restore the window
19. show me how to walk from ITU to cafe alma
20. return to phone home screen
21. exit application

E.3 Questionnaire (10 min)

The user fills in the evaluation form Tide-EF2.

Appendix F

Evaluation - experiment forms

Tide-EF1

Filled by the designer.
The goal is to record which interaction techniques does the user know after 5 minutes of free exploration of the application.

Enter session number

Dragging

- [ ] finger gesture
- [ ] Other:

Rotating

- [ ] gesture with 1 finger
- [ ] gesture with 2 fingers, 1 hand
- [ ] gesture with 2 hands
- [ ] maximize then rotate
- [ ] Other:

Resizing

- [ ] gesture with 2 fingers, 1 hand
- [ ] gesture with two hands
- [ ] Other:

Minimizing

- [ ] resize reduce
- [ ] drag off edge
- [ ] double tap
- [ ] blob touch
- [ ] Other:

Maximizing

- [ ] resize enlarge
- [ ] Other:

Exiting

- [ ] Minimize, then close
- [ ] drag off corner
- [ ] press and hold
- [ ] Other:

Submit

Tide-EF2

1/5

Questionnaire to be filled by the participant after performing the scenarios.
* Required

please enter a session number *

What is your favorite way of using the application?

You do not have to answer based on the gestures that you used during the experiment session.

To resize the window, which action do you prefer?

- [ ] perform a pinch gesture with 1 hand
- [ ] perform a resizing gesture with 2 hands

To rotate the window, which action do you prefer?

- [ ] perform a rotating gesture with 1 hand
- [ ] perform a rotating gesture with 2 hands

To minimize the window, which action do you prefer?

- [ ] drag the window to the edge of the table
- [ ] double tap the border of the window
- [ ] touch the window with the whole hand
- [ ] reduce the window until it minimizes

To close the window, which action do you prefer?

- [ ] drag the window to a corner of the table
- [ ] minimize the window, then close it

Do you agree with the following statements?

It is easy to use the application.

- ( ) strongly disagree
- ( ) disagree
- ( ) neither agree nor disagree
- ( ) agree
- ( ) strongly agree
- ( ) no opinion

It is easy to learn how to use the application. 2/5

- ○ strongly disagree
- ○ disagree
- ○ neither agree nor disagree
- ○ agree
- ○ strongly agree
- ○ no opinion

The application is intuitive.

- ○ strongly disagree
- ○ disagree
- ○ neither agree nor disagree
- ○ agree
- ○ strongly agree
- ○ no opinion

The look and feel of the application is engaging.

- ○ strongly disagree
- ○ disagree
- ○ neither agree nor disagree
- ○ agree
- ○ strongly agree
- ○ no opinion

There are situations in which the application is useful

- ○ strongly disagree
- ○ disagree
- ○ neither agree nor disagree
- ○ agree
- ○ strongly agree
- ○ no opinion

What do you think of the application?

You can write in english or danish.

What was the best thing about the application?

What was the worst thing about the application? 3/5

Can you think of any suggestion to improve the application?

Are you a smartphone user?

Do you have a smartphone?

- ○ yes
- ○ no

Do you use a smartphone on a daily basis?

- ○ yes
- ○ no

Which of the following activities do you use your smartphone for?

- ☐ browse the internet
- ☐ read / write emails
- ☐ look at pictures
- ☐ read / write sms messages
- ☐ make a video phone call (Skype)
- ☐ play games
- ☐ look up a location on google maps
- ☐ read documents

4/5

In which situations would you use the application rather than using your smartphone alone?

You can check as many boxes as you like.

When you are on your own in a PRIVATE space?

- [ ] to browse the internet
- [ ] to read / write emails
- [ ] to look at pictures
- [ ] to read / write sms messages
- [ ] to make a video phone call (Skype)
- [ ] to play games
- [ ] to look up a location on google maps
- [ ] to read documents

When you are on your own in a PUBLIC space?

- [ ] to browse the internet
- [ ] to read / write emails
- [ ] to look at pictures
- [ ] to read / write sms messages
- [ ] to make a video phone call (Skype)
- [ ] to play games
- [ ] to look up a location on google maps
- [ ] to read documents

When you are together with friends or colleagues?

- [ ] to browse the internet
- [ ] to read / write emails
- [ ] to look at / show pictures
- [ ] to read / write sms messages
- [ ] to make a video phone call (Skype)
- [ ] to play games
- [ ] to look up a location on google maps
- [ ] to read documents

5/5

Last details

How old are you?

What is your gender?

- male
- female

Any last comments? (english or danish)

Submit