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

Master Thesis by Leo Sicard
lnsi@itu.dk

March 2012

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius. Claritas est etiam processus dynamicus, qui sequitur mutationem consuetudium lectorum. Mirum est notare quam littera gothica, quam nunc putamus parum claram, anteposuerit litterarum formas humanitatis per seacula quarta decima et quinta decima. Eodem modo typi, qui nunc nobis videntur parum clari, fiant sollemnes in futurum.

Acknowledgements

Juan David Hincapie Ramos, Aurlien Tabard, Jakob Bardram, Sebastian Bttrich, PitLab, experiment volunteers. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius. Claritas est etiam processus dynamicus, qui sequitur mutationem consuetudium lectorum. Mirum est notare quam littera gothica, quam nunc putamus parum claram, anteposuerit litterarum formas humanitatis per seacula quarta decima et quinta decima. Eodem modo typi, qui nunc nobis videntur parum clari, fiant sollemnes in futurum.

Contents

Chapter 1

Introduction

1.1 Background and motivation

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, on-the-go. They bring computing to situations where laptops don't fit, such as standing in a crowded train, or walking in the street. Furthermore, they 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 device can be a limitation to this form of improvised computer interaction, especially in situations with simultaneous users.

Figure 1.1: People viewing information on a smartphone.

Tabletop computers, on the other hand, are ideal in social contexts. They bring computing to a very common piece of furniture, the table. As such, they present a horizontal interactive surface around which multiple users can regroup, to share a common experience, or to 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 that they provide a tactile experience that reaches even the less computer literate. 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 a more natural one for the non-technical user, because it removes the mouse-and-keyboard abstraction layer and generates a more direct interaction.

Figure 1.2: People using an interactive tabletop exhibit at the Asia Society Museum.

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. But tabletops have a huge potential for collaboration. In following years, 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 an essential element 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], tabletops are gradually appearing in public environments such as museums, meeting rooms, public lobbies, bars, restaurants, etc. They are an ideal platform for spontaneous use and multi-user interactions, and they provide a touch-based experience that is similar than the one on most smartphones, making them easily accessible to the pubic.

State-of-the-art smartphones boast screen resolutions up to 1280 by 720 pixels, that exceed the naked eye's ability to distinguish separate pixels. However, a smartphone screen is too small to provide a satisfying 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 go only up to 5 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 one's eyesight, it usually takes 2 to 4 zooming gestures to enlarge the text to a size that is conveniently readable. At this point, the user generally tilts the phone to a landscape orientation, to give the paragraphs a more natural length. There is only space for 5 to 10 lines of text on the screen, which implies that the user uses successive pan gestures to update the content as s/he reads. Compared to the casual experience of reading a newspaper article or a book, this seems like a lot of trouble.

Maps are also graphically dense. They present a lot of different informations, such as topography, street names and sights, on limited space. To be able to read a street name on a smartphone maps 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, for example, his/her own location, the user must use a combination of pan and zoom out gestures that can be cumbersome. In certain cases, the area of interest is simply too vast to be viewed on a smartphone screen.

Modern smartphones play the role of point-and-shoot cameras, due to the fact that we carry them with us at all times, and that they take pictures of high quality. However, image viewing on a handheld device presents serious limitations. Beside the fact that the screen is too small to do pictures justice, it makes showing them to someone else a frustrating experience. Ideally, showing pictures to a friend implies both persons looking at the pictures together, to allow for commenting. This is hardly possible with a smartphone. A typical scenario 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 cumbersome, and another example of the need for smartphones to be able to integrate with larger displays such as tabletops.

In the above mentioned cases, the smartphone provides all the necessary resources, with the exception of a large enough display. This work focuses on these situations in which the smartphone experience would benefit from additional screen space.

A way to address this challenge is by using *device composition*, 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 aforementioned smart rooms, and a recurring element within device composition. They are an ideal platform for collocated collaboration, as was 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 casual form of interaction.

Within this context, designing for device composition between smartphones and tabletops implies addressing three fundamental challenges.

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, but improves this 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. There exists various stable technologies to choose from, but recent research works have chosen to build frameworks from scratch, to allow 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 destined to support a new form of nomadic experience, where users can annex various display servers with their mobile personal devices.

The last 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 be used to seemingly project its UI onto an available computer display.

1.2 Problem statement

This thesis addresses the problem of using *consistent* design to build a composite device between a smartphone and a tabletop computer, that allows for *spontaneous interaction*.

It does so by answering the following research questions:

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

1.3 Research methods

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

To place 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 investigated: main approaches, tabletop systems and applications, pairing, UI distribution, user interaction. The resulting study is presented in Chapter 2.

Following a user-centered approach, a solution design was completed. The process was divided into to the following tasks. The application's context of use was analyzed with the support of scenarios and storyboards, which lead to the definition of solution requirements. A series of design options were generated from the study of existing approaches to software design for surface computing. With the support of a study based on low-fidelity prototypes and the involvement of potential users, a final design was produced. The design process is reported 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, in the aim of showing that consistent design can lead to an easy user experience. The system is described in Chapter 4.

An evaluation of the design was conducted by way of a 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, cooperative evaluation, a semi-controlled 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 resulting data was processed, analyzed, and the evaluation results were derived. The evaluation process and results are presented in reported in Chapter 5.

1.4 Contributions

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

1. The user-centered design process shows that it is feasible to design a system that is immediately accessible to all types of users, and that the user interaction plays an essential role in the design of an intuitive device. To support the process of selecting the most intuitive interaction techniques, this work introduces two design principles: *consistency* and *physicality*. The consistency of the design refers to the selection of interaction techniques that users already know from any type of prior experience. The physicality of the design implies taking into consideration the

form factor of the devices, to identify interaction techniques that are natural to the user.

2. The implementation of the proof-of-concept application called *TIDE (Tabletop Interactive Display Extension)* shows that it is feasible to implement a system on the Microsoft Surface that allows users to integrate any type of smartphone to the tabletop by way of UI replication.

3. The final contribution is the evaluation of the application design. It shows that consistency and physicality allow the design of a device that is easy to use for all types of users. Furthermore, it shows that it is a good idea to implement several interaction techniques for a same feature, in order to reach users with different background. Finally, the evaluation confirms that there is a need for a system that allows users to extend their smartphone screen space to a larger display.

Chapter 2

Related Work

2.1 Device composition and tabletops

Device composition is an important element of the ubiquitous computing research area, and the main background 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 such an approach is that the interaction between the enabled devices can be hugely optimized, as they shared a set of semantics. However, the drawback is that new devices cannot be integrated easily, requiring at least a software installation or configuration.

To bring device composition outside of the smart rooms, systems have been built, that support a more ad-hoc form of device interaction. 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 a combination of 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 an recurring element in research projects that address device composition. They present two important characteristics. First, they are *situated*, i.e. due to their size, they typically sit in a location and are not moved frequently. Second, they

are *shared* among multiple users, mostly due to their horizontal interactive surfaces that allow users to attend from all angles and engage in face-to-face interaction. For these reasons, tabletops seem to be best used in combination with the mobile devices carried around by users, especially when they sit in a public space.

Even though early systems such as the DigitalDesk [Wellner 1993] were designed for the individual office desk, most of the research on tabletops has been focused on their potential as support for collocated collaboration. Thanks to technology advances such as DiamondTouch [Dietz and Leigh 2001], tabletops support multiple simultaneous touch input, and thus simultaneous users. They are an essential element of smart spaces, with systems such as the InteracTable for the i-LAND [Streitz et al. 1999] and the iTable for the iRoom [Johanson et al. 2002]. Other projects have shown that they are an ideal tool to support collaboration, with systems such as the MemTable [Hunter et al. 2011], that permits the capture and recall of meetings, or SketchTop [Clifton et al. 2011], a multitouch sketching application for collocated design collaboration.

Another basic characteristic of tabletops is the horizontal orientation of their interactive surface. The obvious implication is that physical objects can be placed on them. Much work has gone into the detection and tracking of such objects, with systems such as SurfaceFusion [Olwal and Wilson 2008], where the authors used a combination of RFID technology to sense the presence of objects and computer vision to track their precise location. Tracking techniques can be used to detect other computing devices, and can therefore help to solve certain challenges of device composition.

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.

The task of the present work is to design a composite device made up of a smartphone and a tabletop for spontaneous user interaction. This task poses three fundamental challenges: device pairing, distributing the user interface and designing the user interaction.

2.2 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 different 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 there are multiple devices 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, like with the Smart-its [Holmquist et al. 2001], that connect when they are held and shaken together; and SyncTap [Rekimoto et al. 2003], that requires 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 to have the display present a random key to be entered on the mobile device. The key can take 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 that the mobile device is equipped with the appropriate tag, and that the situated device presents 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.

2.3 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 was done for the iRoom [Johanson et al. 2002], but 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 are 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 is the case for Flash [Adobe Systems 2012], Java [Gosling et al. 2000] and Silverlight [Microsoft 2012c]. However, issues remain, that are mostly related to the excessive size of the data files and the security risks.

Distributing graphics is generally a preferred approach, because it allows to keep user data and credentials 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 a 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, however, the situation is 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 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 into account the limited CPU and battery capacities of mobile devices, and provides a flexible protocol that allows the 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.

2.4 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 via a cable—such as VGA or DVI—to external monitors, projectors, etc. 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; and 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 work focuses therefore on providing a wireless user experience.

Recent research efforts have shown that there are 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, for example to give a presentation. This is a satisfying solution in specific

situations, and present the advantage of being secure, because input is kept to the mobile device. However, it presents serious limitations in terms of 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 using a cable, or a docking station.

Expansion is when the larger display becomes 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 additional resources available 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 building a system that provides an easy interface and a spontaneous user experience. Additionally, the system is built on standard hardware, and is compatible with most smartphone types.

Chapter 3

Design

This chapter presents the design process that was conducted to build a composite device that integrates smartphones to tabletops and provide 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 process was focused on producing a *consistent* design. This concept was used in an effort to build an intuitive system, i.e. a system that users can understand without the need for conscious reasoning. Consistency can apply to various aspects of a system design. In the present case, the following were found to be the most relevant. The design should aim at being consistent with (1) the user background and (2) the physicality of the devices. The user background is a broad notion that includes any prior knowledge or experience that is relevant to the current system, and that the designer can derive from the design analysis. In the present case, for example, the system is aimed at smartphone users, so it can be expected of the end-user some prior experience with a smartphone.

The physicality of the devices refers to the implications that the form of the devices have on the user's expectations. For example, the table form implies that a user will expect to be able to place an object on it. If one of the devices had wheels, the user will expect it to be able to roll.

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

Thus, the system is designed so as 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 smartphone and tabletop. The replicated UI is controlled by the *surface UI*. The surface UI consists of elements that allow the 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

The initial design constraints are provided by the parameters that are already known. First, the system integrates smartphones to tabletops. Second, 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.

3.1.1 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.

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

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

A smartphone is. . .

. . . a computer. It provides the computing resources necessary to allow a user to store personal data and install/use applications, as well as connectivity. 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.

3.1.2 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 A.

There are four general cases, depending on the location (public/private) and the users (single/multiple). However, it is unclear if a location can be private when multiple users are present. Without taking into account the nature of the relationship between the users, this configuration is an impossibility, and thus not addressed in this analysis.

Single user on a public tabletop

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

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. 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 closes the connection.

Multiple users on 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 company's employee. However, in an effort to produce a design as generic as possible, and for the sake of simplicity, the tabletop is considered as 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 on 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 daily routine.

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

3.2 Solution requirements

There are four aspects to the system, that each relate to a specific challenge within device composition. For each of those aspects, requirements were derived from the analysis of the application context. Some requirements were left out, that were considered out of scope for the present work. The requirements that were deemed essential to solving the thesis problem are formulated in this section.

3.2.1 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 Exiting the application should be easy and quick.

3.2.2 Replicated UI

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

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

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

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

RB-4 The responsiveness of the replicated UI should be under 1 second, to provide a satisfying user interaction.

3.2.3 Surface UI

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

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

RC-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.

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

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

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

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

3.2.4 Tangible UI

The tangible UI is the set of features that are used by physically interacting with the smartphone. It should fulfill the following requirements:

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

RD-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.

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

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

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.

It was decided to focus exclusively on touch-based interaction in the design of the surface UI, for the following reasons. First, it is consistent with the physicality of tabletops, who rely on touch-based input to provide an appealing experience to all types of users. Second, it is consistent with the smartphone experience, which also relies on touch-based input. Lastly, the presence of other input peripherals cannot be expected, so relying on touch-based input will ensure that the system is compatible with 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 most intuitive interaction techniques.

3.3.1 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 to permit this process are brainstorming, sketching, storyboarding and prototyping.

Storyboards

The scenarios mentioned in section 3.1.2 are used as a base for the making of storyboards. Storyboarding helps getting a feeling for the general flow of the interaction with the system. It also gives a visual dimension to the definition of the different system features, and raises new design issues. Putting orally expressed ideas on the paper is often a first test of their validity.

For example, several system features were described in the initial scenarios as being the effect of a tap on a button. When storyboarding, it became obvious that having too many UI buttons would be cumbersome, which lead to the consideration of other interaction techniques, such as touch gestures. Similarly, the issue of the location and size of the replicated UI on application launch became first 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 out in various sizes, and used on a normal table to simulate interaction with the replicated UI. Figure 3.2 shows some prototypes and a working session.

Figure 3.2: Working with low fidelity prototypes.

3.3.2 Defining interaction strategies

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

Human computer interaction can be modeled as a simple cause-effect relationship. The user wishes the computer to execute a command. To achieve that, s/he performs an action to provide input. In the case of a touch-based interactive surface, the action is typically a hand gesture. The action is the cause, the command is the effect, and together they form a single interaction between user and machine.

Commands

The following six basic commands are identified as interaction primitives for the surface UI. They are directly derived from the requirements formulated in section 3.2.

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. *Exiting* the application.

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

Actions

Various interaction techniques can be used to invoke application level commands. While working with paper prototypes to generate ideas, it became apparent that some interaction techniques could be regrouped into categories that could be consistently implemented for each of the previously defined commands. This realization lead to the definition of five *interaction strategies*, presented in figure 3.3. There is a sixth category, that regroups design suggestions that are not part of any consistent strategy.

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.
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.
6. *Other* regroups suggestions that do not fit with any specific strategy.

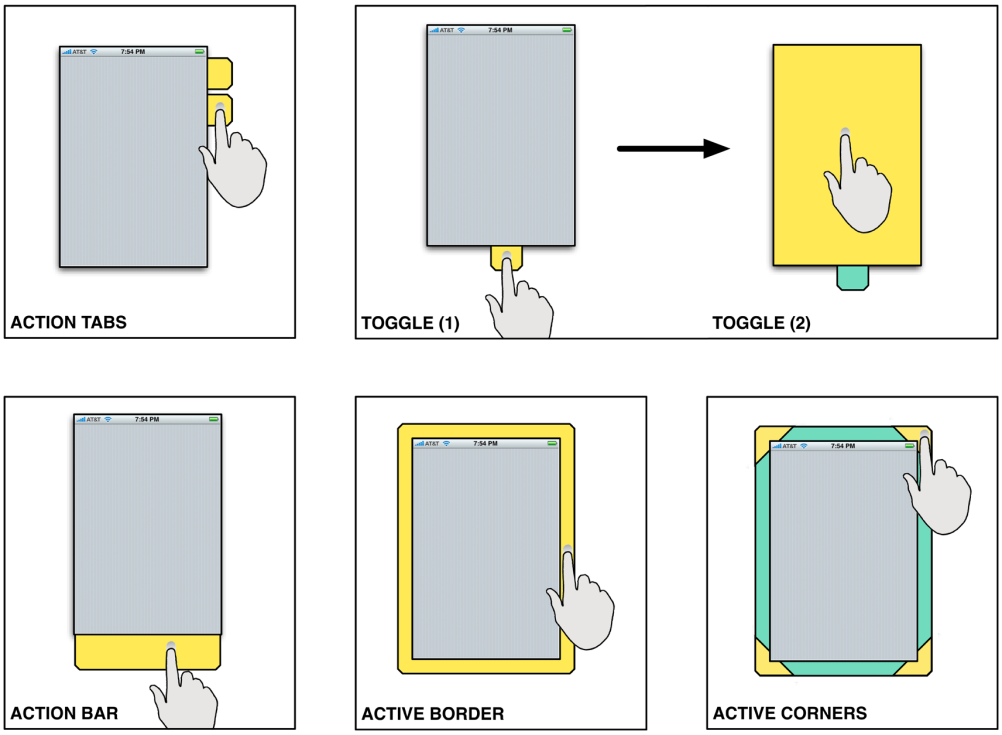

Figure 3.3: Interaction strategies.

3.3.3 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 3.3.2.

The interaction primitives are central system features. For each primitive, the participants were asked to express an open-ended *user suggestion*, then to *rank* the interaction strategies.

Experiment

Twelve participants were recruited on a voluntary basis. A session involves one participant and one designer. The participant sits next to the Microsoft Surface tabletop [Microsoft 2012a], and is presented with an iPhone [Apple 2012b], but both devices are

Figure 3.4: Experimenting with prototypes.

off. On the tabletop are paper prototypes, that are to be used as representations of UI elements throughout the session. The designer leads the experiment by reading instructions from a script (included in appendix B) and answering the participant's questions.

As an introduction, the following things are 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 revolves around a task that the participant is asked to perform using the prototyped application. The task is to write an email, and it requires the user to go through six phases. Each phase is dedicated to an interaction primitive, and they have the same structure, which is as follows:

1. the primitive is explained to the participant in terms of a command to the application,
2. the user is 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 describes action suggestions, that the user is asked to try out and rank by order of preference.

For each command, there are six possible actions. However, it was decided that presenting a user with six options to rank would be overwhelming. The volunteers were therefore split into two groups, each evaluating a subset of the interaction strategies. The repartition is shown in table 3.5.

	Group 1	Group 2
Dragging	1,2,3	4,5,6
Rotating	2,3,4	5,6,1
Resizing	3,4,5	6,1,2
Minimizing	4,5,6	1,2,3
Hiding	5,6,1	2,3,4
Exiting	6,1,2	3,4,5

Figure 3.5: The repartition of the evaluated interaction strategies between the two groups of participants. The strategies are 1) Action Tabs 2) Action Bar 3) Window Toggle 4) Active Border 5) Active Corners 6) Other.

Data processing

Participant answers were gathered in a form such as the one included in appendix C. The form is a matrix where an entry corresponds to a pair (primitive,strategy). Those 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 contains 6 positions. In order to obtain a numeric score for each entry, a weighted average was calculated, giving a weight of 3 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 signifies that the entry was awarded a first position by all participants, and a 0 signifies that all participants ranked this 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, so the processed data was a disparate list of suggestions that each had a counter variable indicating how many users independently expressed said suggestion.

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, exiting; are more abstract.

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

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

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 strategy is similar to the current standard for manipulating pictures on interactive screens, with the difference that 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 to.
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. Exit 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.

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 can logically 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 exiting, 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

The design decisions reported in this section were derived from the user-centered analysis, with focus on design consistency, and relate to the requirements defined in section 3.2.

3.4.1 Pairing

DA-1 Pairing is triggered by placing the smartphone on the tabletop, and based on wireless connectivity.

DA-2 Device discovery relies on a standard networking protocol such as UPnP [UPnP Forum 2012] or Bonjour [Apple 2012a].

DA-3 The user must confirm the connection, both on the tabletop and on the smartphone.

DA-4 See DC-5.

3.4.2 Replicated UI

DB-1 The smartphone screen is replicated to the tabletop, using a standard rendering-based protocol such as VNC [Richardson et al. 1998].

DB-2 See DB-1.

DB-3 See DB-1.

DB-4 See DB-1.

3.4.3 Surface UI

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 intuitiveness that is looked for, while others are quicker, and provide an efficiency that is beneficial for the application. This is standard practice within software design, a good example being keyboard shortcuts: they can not be easily discovered by a novice user, but are preferred by the expert user for their efficiency.

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

DC-2 The replicated UI can be manipulated by using touch-based gestures on the surface UI.

- DC-2a Dragging is done by performing a dragging gesture with one or more fingers.
- DC-2b Rotating is done by: (1) dragging the corner of the surface UI, (2) performing a rotating gesture with two fingers of the same hand or (3) performing a rotating gesture with two hands.
- DC-2c Resizing is done by: (1) performing a pinching gesture with two fingers of the same hand or (2) performing a resizing gesture with two hands.

DC-3 Minimizing the replicated UI can be done by: (1) resizing the surface UI down, (2) dragging the surface UI off an edge of the table, (3) double tapping the surface UI or (4) placing a full hand on the surface UI. When minimized, the surface UI appears as an icon on the tabletop, and can be restored by tapping a button on said icon.

DC-4 Hiding the replicated UI is done by minimizing the surface UI, see DC-3.

DC-5 Exiting the application can be done by: (1) closing the minimized surface UI, (2) dragging the surface UI to a corner of the tabletop or (3) pressing and holding a finger on the active border.

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

3.4.4 Tangible UI

Based on a careful analysis of the context of use, supported by oral feedback gathered from the prototyping sessions, a decision was taken to limit the use of the smartphone as a tangible UI. The main reason supporting this decision is the realization that users might 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. The physical device is therefore only involved in the pairing procedure.

DD-1 The pairing procedure is triggered by placing the smartphone on the table.

DD-2 Requirement cancelled.

DD-3 Requirement cancelled.

DD-4 The replicated UI is active on the tabletop independently of the smartphone, as default.

Chapter 4

The TIDE prototype

TIDE (Tabletop Interactive Display Extension) is a prototype that combines smartphones and tabletops by way of UI replication. It was implemented in .NET for the Microsoft Surface (Windows Vista), referred to as MS in this chapter. It was tested with an iPhone 4 running iOS 5 and a HTC Legend phone running Google Android 2.1, and it can be extended to support other smartphone models.

As shown in figure 4.1, TIDE consists of three components, that are responsible for pairing, UI replication, and the surface UI. Pairing is done through camera-based object detection. TIDE relies on OpenCV (Open Source Computer Vision) [OpenCV 2012] to detect phone-like objects to connect to, and to track the devices during the application session. The UI replication is based on the VNC protocol [Richardson et al. 1998]. TIDE includes a VNC client that is based on the VncSharp library [VncSharp 2012], and that connects over the network to a VNC server running on the smartphone (third-party application).

The surface UI is implemented using the Microsoft Surface SDK and WPF (Windows Presentation Foundations).

Figure 4.2 shows the basic interaction flow that TIDE supports. First, the user places the smartphone the MS, which triggers the pairing process, described in section 4.1. Section 4.2 describes how smartphones are detected and tracked during TIDE application sessions. Second, the smartphone's UI is replicated to the tabletop via the VNC protocol, as presented in section 4.3. Lastly, the user interacts with the application via the surface UI, described in section 4.4.

4.1 Pairing

The pairing procedure requires that both devices are connected to the local network. In the case of the smartphone, this connection is very likely wireless. Moreover, a VNC server instance should be up and running on the smartphone, for the system to function.

To achieve successful pairing, the following steps must be completed: (1) a *trigger* starts the procedure, (2) *discovery* allows the smartphone's local IP address to be made

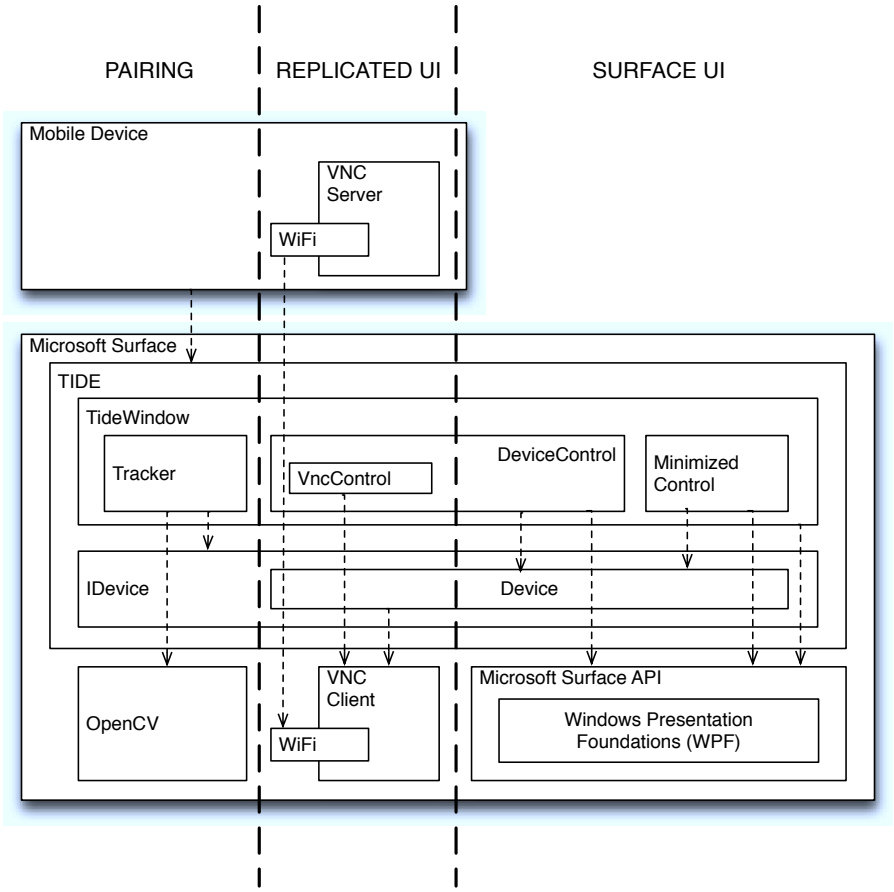

Figure 4.1: TIDE overview.

available to the tabletop and (3) the *connection* is established, given that the user explicitly authorizes it.

4.1.1 Trigger

To start the pairing procedure, a trigger must be used. In TIDE, this trigger is provided by the detection of a smartphone object lying on the tabletop.

Interactive displays that are based on computer-vision technologies function by seeing the shape of the objects that come in contact with the surface. That implies that not only are fingers seen, but also other objects such as books, cups, or cell phones. The MS is an example of a tabletop that was designed for integrating with physical objects. This strategy is based on the use of visual markers [EXPLAIN MS VISUAL TAGS STRTATEGY HERE]

However, the MS camera-based system is able to see all contact shapes, as long as they reflect infra-red light. The display's visual input can be captured and analyzed with computer vision algorithms such as the ones provided by OpenCV, and specific shapes

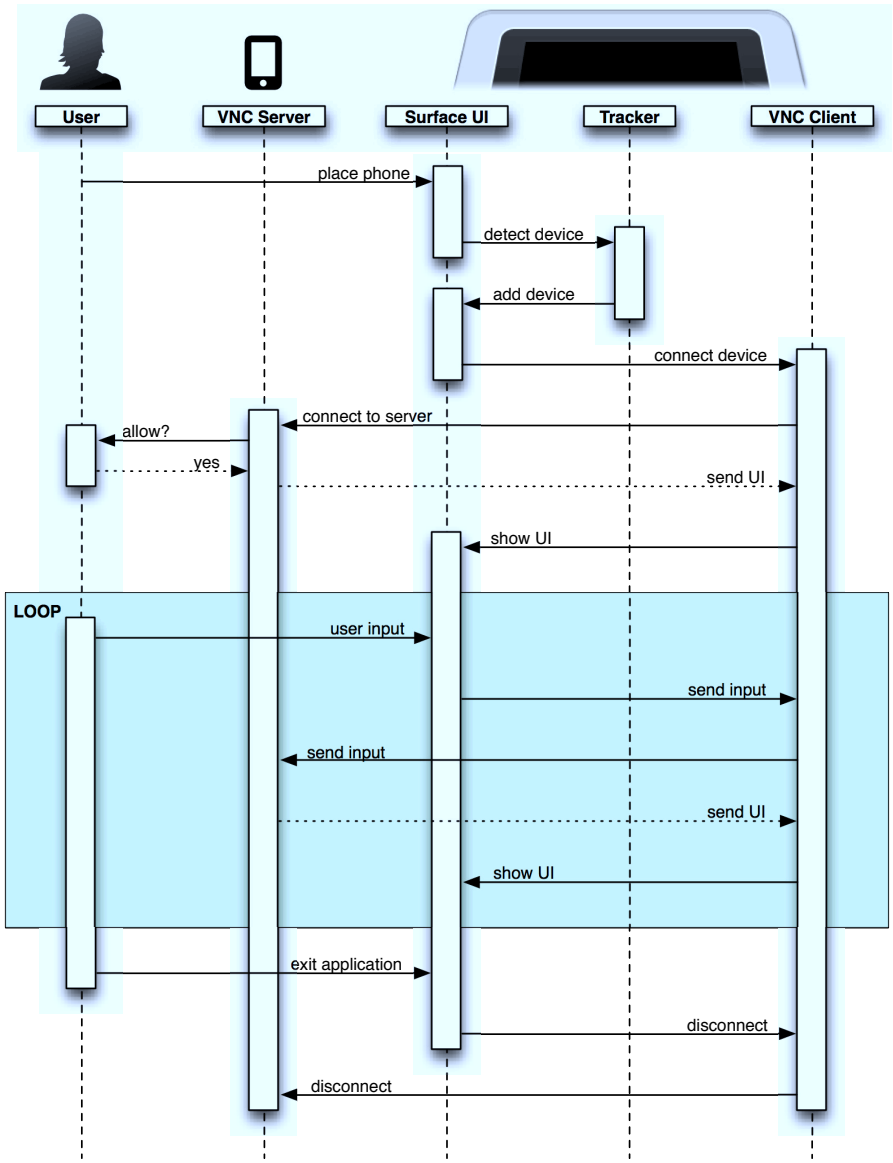

Figure 4.2: TIDE overview.

can be recognized. This strategy was used in the TIDE implementation, to detect specific smartphones, and trigger the procedure. The details of the smartphone detection steps are explained in section 4.2.

The reason for using shape recognition instead of the MS tag recognition system is to remove as many constraints as possible, to make the system accessible to all, whatever the context. Using visual tags would imply that the system be usable only by users that had applied a visual tag to the smartphone.

4.1.2 Discovery

The smartphone's local IP address must somehow be made available to the TIDE application, in order to establish the connection.

There is one easy solution to this problem, which is having the user enter the IP address in a dialog window on the tabletop, but this approach has serious limitations. Finding out what the IP address is not obvious for most users, it requires digging into the networking settings of the phone. In any case, it is a process of several steps. Moreover, entering the address on the tabletop takes time and can lead to mistakes. IP addresses are numeral strings of typically more then 10 digits, which makes them cumbersome to remember and type.

An easier solution is to automatize the discovery process, using networking protocols such as UPnP and Bonjour, based on Zeroconf. [HOW MUCH SHOULD I EXPLAIN HERE??]

Seeing that the focus of this work lies elsewhere, and for reasons of time constraints, it was decided not to implement the discovery protocol. The system currently functions with IP addresses hardcoded into the TIDE implementation on the MS, to serve the purpose of the present proof-of-concept.

4.1.3 Connection

To protect the user's privacy, the connection needs explicit authorization. This is handled by a dialog that appears on the tabletop, described further in section 4.4.

The connection is otherwise handled by the VNC components on tabletop and smartphone, whose implementation details are presented in section 4.3.

4.2 Tracking

The MS computer vision is based on infrared light that is projected towards the interactive display, and reflected by the objects that are in contact with it. The reflection is captured by multiple cameras, and processed for system use. By using OpenCV, it is possible to analyze the captured images to detect specific shapes and interpret the nature of the object on the table.

Figure 4.3 shows the implementation of a mechanism that allows TIDE to detect specific smartphones, and track their whereabouts on the surface during application sessions. The core layer of the Surface SDK provides the `ContactTarget` class, which raises the `FrameReceived` event for each available frame of visual input. Figure 4.4 shows the raw MS 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

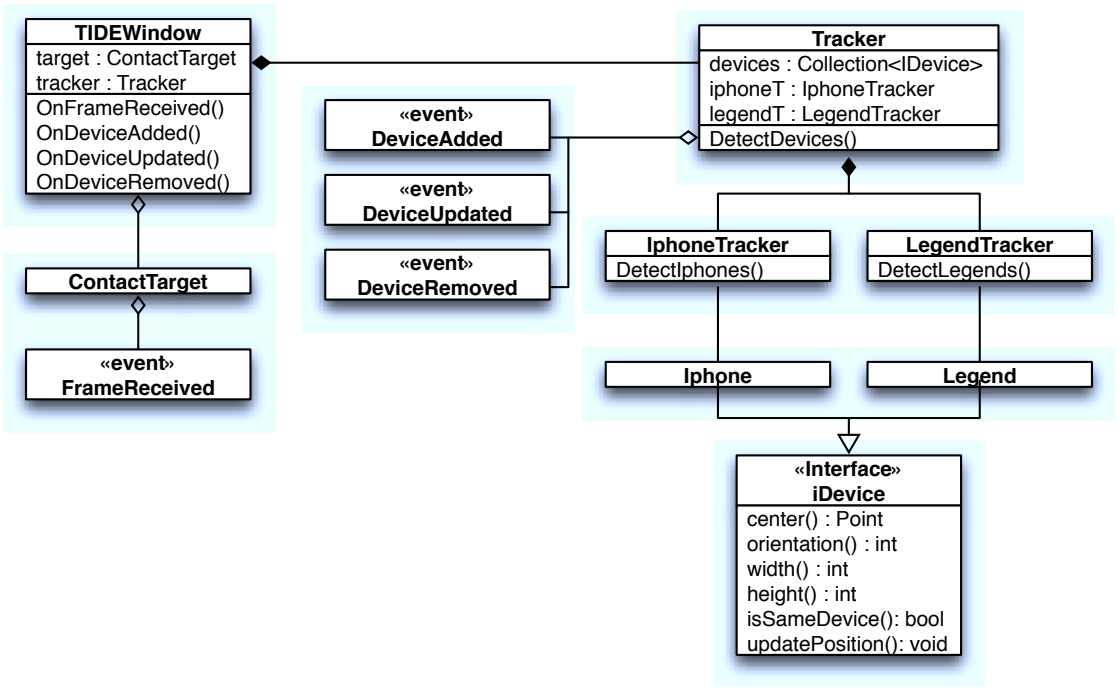

Figure 4.3: Tracking overview.

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.

The `TIDEWindow` 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. It sends screen captures to the `Tracker` class, 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, 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-

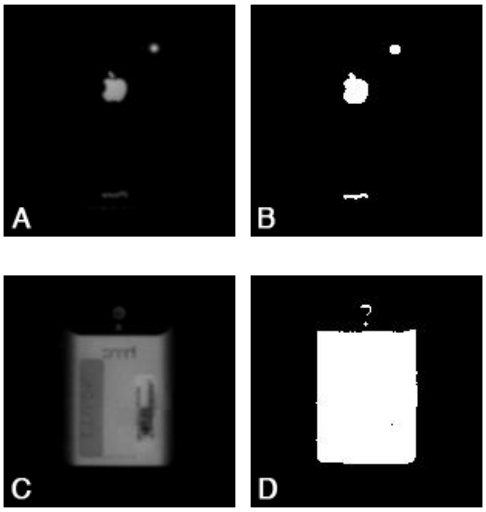

Figure 4.4: Microsoft Surface raw visual input: (A) iPhone 4 (B) iPhone 4 after threshold processing (C) HTC Legend (D) HTC Legend after threshold processing.

activated for the test sessions, for the following reason. The purpose of tracking devices was to allow the use of the smartphone as a tangible user interface, a remote control for the replicated UI. This feature was part of the initial requirements, as presented in section 3.2, but were left out of the final design, the reasons of which were explained in section 3.4.

4.3 Replicated UI

The UI replication is based on the VNC protocol [Richardson et al. 1998]. The pairing procedure completes with the VNC client on the tabletop connecting to the VNC server running on the smartphone, and receiving the first image of the remote UI.

The VNC client is implemented by the `VncControl` class, as shown in figure **??**, that relies on the VncSharp library for sending input and receiving the remote UI. The `VncControl` is responsible for displaying the replicated UI, and is contained within a `DeviceControl` object that provides other UI elements. When the `DeviceControl` object detects a contact on the replicated UI, it forwards it to the `VncControl` object, that relays it further to the smartphone. The server responds with an updated image of the smartphone's screen, that the `VncControl` displays on the tabletop.

The VNC server on the smartphones relies currently on third-party applications. The reason for this was to avoid developing software that would be tied to a certain platform. It permitted to develop an prototype that can support all types of smartphones, given that they run a VNC server.

However, due to limitations implemented by the manufacturers, installing the VNC

applications required rooting the devices. Rooting a smartphone is a procedure that allows the user to obtain full access rights on the system, though with the implication that the device warranty becomes void. Thus, it cannot be expected of users that they will perform this procedure, and this approach would be problematic in a real-world scenario. However, this setup was satisfying for the purpose of the proof-of-concept.

[VNC IS NOT ADAPTED TO THIS SITUATION; IT IS TOO SLOW]

4.4 Surface UI

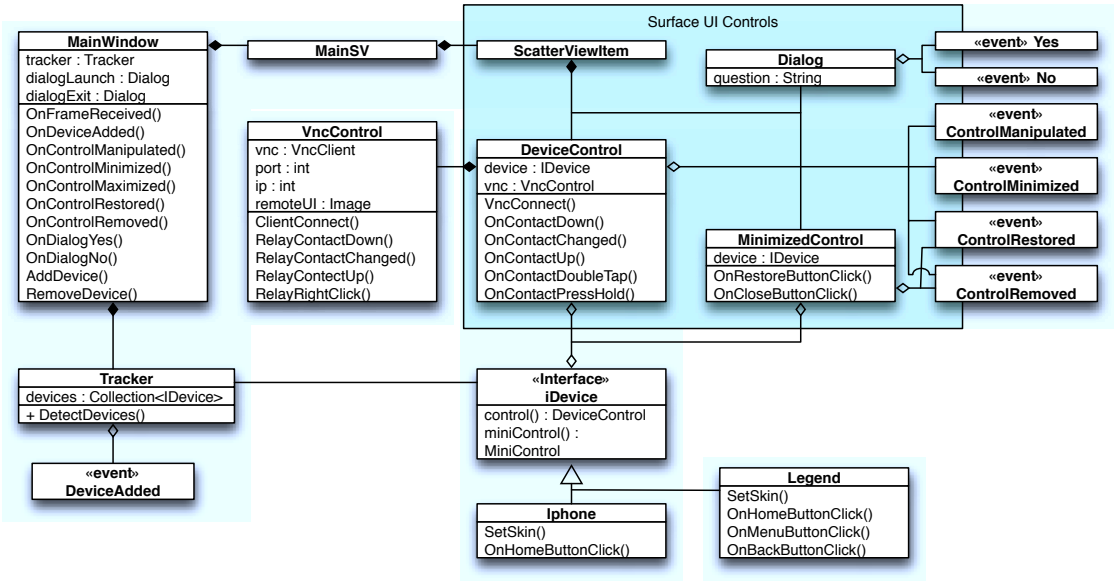

Figure 4.5: Surface UI overview.

4.4.1

we added MAXIMIZE: Similarly when enlarged above a certain size, it is maximized to a fullscreen mode. To escape the fullscreen mode, buttons are implemented in the corners of the tabletop.

Chapter 5

Evaluation

in some cases, compare different interaction strategies for same command.

Compare same implementation with no visual aids, discrete visual aids, and overkill visual aids.

5.1 Experiment

Compare and discuss

5.2 Results

Chapter 6

Discussion

Implementing same functionality with several parallel interaction techniques is a good choice because it will augment the number of situations in which a user will obtain the desired effect intuitively on his first try, without referring to any manual.

Chapter 7

Conclusion

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.

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.

Paul Dietz and Darren Leigh. Diamondtouch: a multi-user touch technology. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01, pages 219–226, New York, NY, USA, 2001. ACM. ISBN 1-58113-438-X. doi: http://doi.acm.org/10.1145/502348.502389. URL `http://doi.acm.org/10.1145/502348.502389`.

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`.

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.

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

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 id 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. The screen goes gray to notify Alice that an object (the notebook) is in the way. Alice removes the notebook, and the window becomes active again. She 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 B

Preliminary usability study - experiment script

B.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."

B.1.1 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.

B.2 Experiment

B.2.1 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)

B.2.2 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)

B.2.3 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)

B.2.4 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)

B.2.5 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)

B.2.6 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 C

Preliminary usability study - result 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 C.1: Form used to gather participant answers during the preliminary usability study.