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.

It does so by suggesting a *device composition* approach, that integrates smartphones and tabletops. Tabletops are a solution to this problem for several reasons. They provide the needed display space, and a touch-based interaction that comes naturally to the typical smartphone user. Moreover, they are designed for collocated collaboration, which brings an interesting new dimension to the scenarios discussed above. Examples of the possibilities include consulting a map to show a location to a friend, reading and

editing a document with a colleague, playing a multiplayer game, or viewing and sharing pictures between smartphones.

DEVICE COMPOSITION is an approach that can solve this.

Device composition started with SMART ROOMS, and composing various devices, which we want to do in this case. But smart rooms were designed as closed environments.

We would like AD-HOC, spontaneous device composition. That was addressed by projects like Obje, CompUTE, Platform Composition.

we are interested in mobile to TABLETOP - studies on collaboration (but closed environments and technical users) - studies on tangible interaction

tabletops emerge in public spaces ¿ non technical users ¿ we focus on standard hardware

we must solve 3 challenges: pairing, UI distributing, user interaction

To combine SP and TT, we need to pair them, which can be done with networked protocols or bluetooth (but that requires explicit input from user, or CPU/battery intensive continuous sniffing SO we use a trigger, that can be:) detecting synchronous events entering a key into display with RFID with computer vision to detect

we do it with computer vision detection and networked protocol (Bonjour)

we need a technology to do UI distribution: distr data distr code

best to distr graphics: (we choose VNC in spite of drawbacks ¿ why?)

we need a user interaction

FINISH with how we do differently!

1.2 Problem statement

This thesis addresses the problem of designing a composite device between a smartphone and a tabletop computer.

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 the Microsoft Surface and integrate different types of smartphones?

1.3 Research methods

A comprehensive literary review is made of the research work related to device composition and surface computing, and in particular, the following themes are investigated: device composition approaches, smart rooms, tabletop systems, tabletop applications, tangible integration, user identification, object tracking, UI distribution technologies. To complete this review, a study is made of the modern approaches and methodologies that are best suited for the design of interactive systems.

Following a user-centered approach, a solution design is completed, which requires to iterate between several design tasks. An analysis of the context of use is made with the support of scenarios and storyboards, which lead to the definition of solution requirements. A series of design options are produced 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 is produced and described.

An application prototype is implemented on a Microsoft Surface tabletop computer (first generation) running Windows Vista, an iPhone 4 running iOS and a HTC Legend running Android. Third-party applications are used on the smartphones, that requires rooting the devices. On the tabletop computer, the application is programmed within the .NET framework, using specifically WPF (Windows Presentation Foundations) and the Presentation layer of the Microsoft Surface API 1.0.

An evaluation of the design is achieved by way of a usability study, that involves the following steps. First, the aspects of the system that are to be evaluated are identified, and the evaluation method selected. Second, a user experiment is designed, based on co-operative evaluation, discovery, and a controlled test. Third, participants are recruited, and the experiment is conducted in the PITlab at the IT University of Copenhagen. Data is gathered via online forms as well as application logging. Lastly, the data is processed, analyzed, and the evaluation results derived.

1.4 Contributions

This work reports 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 (first generation) 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.

1.5 Thesis overview

Chapter 2 presents a literary review of the research that constitutes the background to this work, and the theoretical work on which the design approach is based.
Chapter 3 describes the process that lead to the design of the TIDE prototype.
The system itself is presented in Chapter 4, and its evaluation by way of a usability study in Chapter 5.
Chapter 6 is a discussion that addresses the results and lessons learned throughout this process, and brings suggestions for future work.
Chapter 7 concludes the report.

Chapter 2

Related Work

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.

Pairing

Pairing is an essential part of any device composition, and a challenge that can be solved in many ways. Some systems are designed for networked devices, and achieve pairing with TCP/IP based networking protocols such as Zeroconf [2012] for Obje and UPnP [2012] for CompUTE.

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

UI distribution

Device composition has been extensively used for building systems that allow the distribution 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.

Arthur and Olsen [2011] address UI distribution with XICE (eXtending Interactive Computing Everywhere). XICE is 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 provide a flexible protocol that allows the annexation of different types of displays.

Gjerlufsen et al. [2011] have also taken the approach of building a framework to support the development of interactive multi-surface applications. Substance is a data-oriented programming model that was used to build a middleware that provides powerful sharing applications.

User interaction

Recent research efforts have been invested in combining mobile devices such as smartphones to different types of larger displays, such as tabletop computers. The aim is always similar: extending the user interface of the phone to improve the user experience. But different approaches are possible in terms of the interaction metaphor that the extension is 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 is added to the UI of the mobile device. It provides the possibility of transferring data or applications to the remote machine, while keeping the mobile UI independent and available.

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].

Chapter 3

Design

INCLUDE THIS:

there is a need for a system that allow for spontaneous and short interactions, and that is easy to use.

why wireless? use XICE

The thesis investigated here is that using tabletops as basic IO peripherals for other computing devices will help their adoption by the public. A user should be able to transfer the display of his/her device to a tabletop, and interact with it in a natural way. The success of such an interaction model depends on the usability of the system.

To avoid any confusion, a few concepts must be defined. The system involves a *person*, a *personal device* and a *tabletop*. Human computer interaction takes place on the screen of the tabletop. The application UI is divided in two main parts. The *display extension*, or *remote UI*, is a window that replicates the display of the personal device on the tabletop, relaying touch input and graphical output dynamically. The *surface UI* consists of UI elements on the tabletop that allows the manipulation and control of the remote UI on the interactive surface.

This chapter presents the design process of such a system. Section 1 focuses on understanding the context of use and section 2 is an analysis of the requirements. Together they present the conceptual design. Section 3, 4, and 5 describe the process of realizing a physical design for the surface UI element. They respectively present the generation of ideas, their evaluation through a participatory usability study, and the design decisions.

3.1 Enhancing mobile computing with tabletops

It is critical to understand fully the context in which the system will be used in order to achieve a good design. Most users own a computing device with personal data and applications that are tailored to their needs. Those personal devices are becoming smaller and more mobile, with devices such as tablet and handheld computers. In many cases, the display size of the personal device is a limitation in terms of graphical input and

output, and has a negative influence on the user experience. One of the main characteristics of tabletops, however, is that they have superior graphical I/O capabilities. This project focuses on situations where a tabletop can be used as a display extension to the personal device, thus enhancing the user experience.

Figure 3.1 describes the primary use case. The system should basically be a graphical peripheral unit for other computing devices. Its main functions are to forward user input to the device, and to display graphical output from the device.

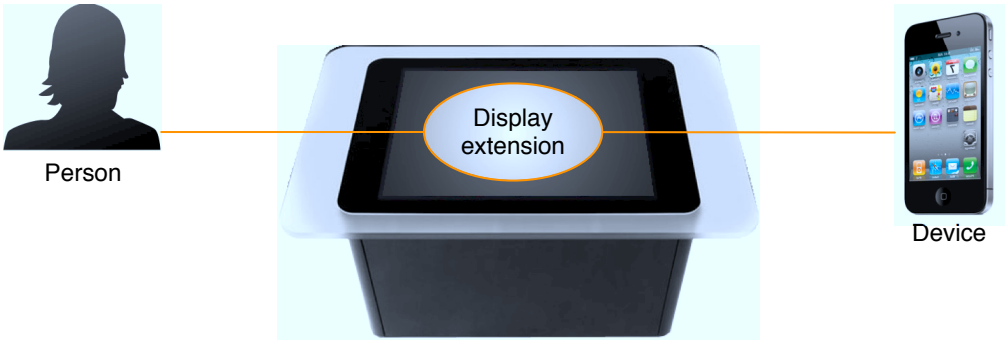

Figure 3.1: Main use case.

3.1.1 The devices

Tabletops

A tabletop presents a range of characteristics that have concrete implications on the system design. A tabletop is ...

... a computer. This project investigates a strategy where a tabletop is used as a smart IO peripheral. The system should allow a tabletop to function as a relay between a personal device and users, forwarding both input and output as required.

... a table. Its working surface is horizontal by nature, with the implication that it cannot support any prolonged interaction, because of the bad ergonomics of the "hunched over" working position. Furthermore, a table gives naturally rise to various activities, such as eating, and it supports all sorts of objects, often in a collocated collaborative atmosphere. The system should therefore support simultaneous users, and handle limited space availability. Finally, a table offers no specific orientation, implying that the orientation of the system UI will have to adapt to the user's position around the tabletop.

... a situated device. A tabletop is not mobile. It usually sits at a specific location, and users come to it in order to use it. The main implication is that tabletops seem to naturally fit in public spaces, where they are shared among multiple users. This tendency is accentuated by the price factor, that makes a private person not likely to buy such a device for private use. The system should handle public

use, characterized by short anonymous sessions and an often interrupted interaction flow. Other scenarios should be considered as well, such as a tabletop in an individual office, or in a family home. In any case, it can be expected to have dedicated power supply and network connection, removing such concerns from the developer's mind.

. . . a shared device. In many cases the tabletop will be shared by multiple users, raising concerns of user identification and data protection.

. . . an interactive surface. As such, it typically offers large graphical output, as well as a range of input techniques to allow for user interaction. Most tabletops support multitouch-based input. This introduces a new kind of interaction model, more intuitive, that the system should be based upon. In some cases, such as with camera-based devices, it is possible to add tangible objects to the tabletop experience. This project reports on the possibilities to use a personal device as a tangible control integrated to the tabletop.

Mobile computing devices

A mobile computing device is . . .

. . . a computer. It offers enough computing power, storage space and connectivity to support most users' daily tasks. The system should allow the user to access his/her applications and personal data.

. . . a small device. Hence the mobility. Such a device is carried around by users. However, the small form factor implies a suboptimal graphical user experience. The system should try to improve this, by offering superior IO resources.

. . . a mobile device. With the implications that its power supply is limited by battery life, and its connectivity is unstable. The system should be developed with those concerns in mind.

. . . an interactive device. Most mobile devices are now equipped with touch-based displays, making them naturally suitable for display extension on a tabletop. However, all devices present physical buttons to the user as well. The physical buttons implement strategic functions that the system should support.

3.1.2 The situations

There exists different types of everyday situations in which the system can be put to use. They vary depending on whether they involve one or more users, and whether the tabletop is a public or private device. A tabletop is considered a public device as soon as there are more than one user that have access to it. As a consequence, the scenario of multiple users on a private tabletop is not considered. Each situation implies a slightly different set of application features. The original scenarios are included in appendix A.

Single user on a public tabletop

It should be possible for the user to wirelessly pair his/her personal device with a public tabletop computer. This implies that the devices are both connected to the local wireless networks, that they are able to detect each other and discover each other's identity on the network. It would not be safe to establish this connection automatically in a public space. Therefore, dialogs should be used both on the mobile device and on the tabletop to gather user input. The UI of the mobile device should be transferred to the interactive surface as graphical output, and this transferred display should be able to accept touch input to be forwarded back to the device. The transferred display should be contained in an application window, and this window should be manipulable (drag, resize, rotate, minimize, hide, ..).

The application window should react to the state of the interactive surface. An example of this is that the application window should turn inactive if it is obstructed by an object on the table.

Finally, the mobile device could be a participant in the interaction model, i.e. listing it off the table should interrupt the connection and exit the application.

Multi users on a public tabletop

In a collocated collaborative context, it should be possible for more than one personal device to simultaneously have their display extended to the interactive surface. This implies that the implementation should support parallel connections and simultaneous use.

Mobile computing devices come in many forms, and ideally the system should support all of them. Devices vary in terms of software and hardware specifications. Some parameters that are especially important here are the programming platform, as well as the display resolution.

When a tabletop is used simultaneously by multiple users, there is a very concrete risk of lack of space on the surface. This fact introduces a new need for the system, to allow a user to remove his/her personal device from the table, while keeping the display extension active.

Single user on a private tabletop

If the tabletop is private, such as a home computer or a working desk, the system should offer extended functionalities to the user. He/she should have the option to configure the tabletop in order to allow/initiate those extended functionalities. Some suggested functionalities are:

- automatic launch of the display extension application
- push application widgets from the extended display to the tabletop
- share data between the personal device and the tabletop

3.2 Solution requirements

The general focus is on a seamless user experience.

3.2.1 Pairing

Connecting a personal device to a tabletop should be a quick and easy process. The system should include a detection mechanism that would allow the devices to become aware of each other, as well as a discovery protocol to gather the information necessary to the pairing, such as a Network IP. The connection should be wireless to guarantee a smooth experience. However, a public tabletop should not be allowed to gather data from, let alone connect to, a personal device without the explicit consent of its owner. In the case of a trusted setup, it should be possible to bypass any explicit user input. Exiting the application and closing the connection should also be easy, allowing the system to handle short successive sessions.

3.2.2 Remote UI

At the core of the system is the display extension. The UI of the personal device should be transferred to the tabletop, allowing the user to interact with it in a natural way. Graphical output from the personal device should be forwarded to the tabletop, and touch-based input from the tabletop should be forwarded to the personal device.

3.2.3 Surface UI

The user experience should be improved by using the display extension on the surface. Therefore, it is important to provide for a rich interaction. The transferred UI should be contained within a manipulable window on the tabletop. Specifically, the user should be able to move, rotate and resize the window; as well as minimize, hide and restore it. The system should include UI elements that implement the functions supported by the physical controls present on the personal device. Those elements and their function should be obvious to the user.

Modern smartphones include sensors that allow to switch the display orientation by tilting the device. This feature is strategic to certain applications, and should be implemented by the system. Obviously, tilting the tabletop is unfeasible, so another solution is necessary.

As mentioned earlier, a tabletop's screen space would typically be shared among different applications and/or objects. Therefore, the system should handle limited screen space, and obstruction of the display extension.

In a trusted setting, the user should have the possibility to push application widgets from the personal device to the tabletop, outside of the display extension, thus saving space on the latter.

3.2.4 Tangible UI

It is a natural thing to place an object on a table, and tabletops are designed to allow for the integration of physical artifacts. Therefore, it should be possible to use the personal device as a tangible UI. For example, it would seem obvious that placing the device on the tabletop would launch the pairing process, or that lifting it off would interrupt the connection. Furthermore, it should be possible to control the position of the display extension by sliding the personal device on the surface.

3.3 Interaction design: the surface UI

The solution requirements are divided into 4 elements: pairing procedure, remote UI, surface UI and tangible UI. The rest of this chapter investigates the physical design of the surface UI only, for the following reasons.

The pairing procedure does not introduce anything new to the field. There are known solutions to this challenge, which are described in section **??**. The remote UI replicates the UI of the personal device on the tabletop, and does not require any supplementary design. Using the personal device as a tangible UI for the display extension raises a series of design and implementation challenges. It is the opinion of the author that this research angle is promising, but it was decided to leave it out of the project for reasons of time constraints.
The surface UI is the part of the system that allows the user to manipulate the remote UI on the tabletop using touch-based input. Touch-based interaction removes the necessity of having input peripherals such as keyboard and mouse, and provides the user with a more direct and sensual experience. The aim is to achieve a design that allow the user interaction to be intuitive.

3.3.1 Generating ideas

The generation of ideas is an important part of the design process. David Benyon refers to it as envisionment [Benyon 2010], and defines it 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. At the same time, it gives a visual dimension to the definition of the different system features, and raises new design issues.

Several system features were described in the scenario 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.

Similarly, the challenge of the location and size of the display extension on application launch became apparent with the first storyboard.

Low fidelity prototypes

Paper prototypes were used to aid the process of generating and evaluating as many possible design solutions as possible. Screenshots of the iPhone UI [Apple 2012] were printed out in various sizes, and used on a normal table to simulate interaction with the surface UI. Figure 3.2 shows some prototypes and a working session.

Figure 3.2: Working with low fidelity prototypes.

3.3.2 Defining interaction stragegies

To support the interaction design process, the concepts of *actions* and *commands* are used. They are inspired by the work done by Wobbrock et al. on 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, he/she 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.

1. *Dragging* the application window across the interactive surface.
2. *Rotating* the application window across the interactive surface.
3. *Resizing* the application window across the interactive surface.

4. *Minimizing* the application window, making it possible to restore it easily.
5. *Hiding* the content of the application window.
6. *Exiting* the application, thus closing the application window.

It is also agreed that the surface UI should offer an implementation for any supplementary command that is controlled by a physical button on the personal device.

Actions

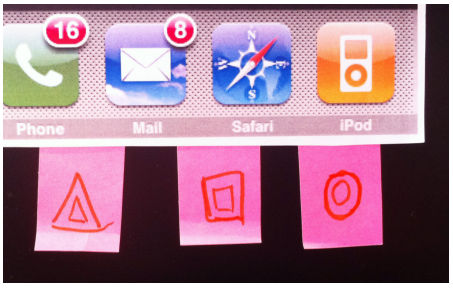

Figure 3.3: action tabs prototype

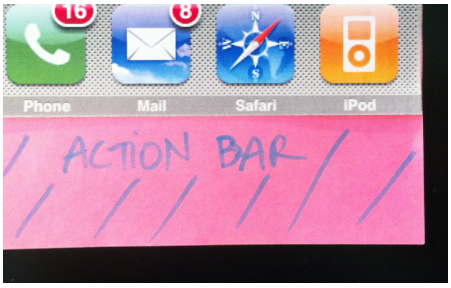

Figure 3.4: action bar prototype

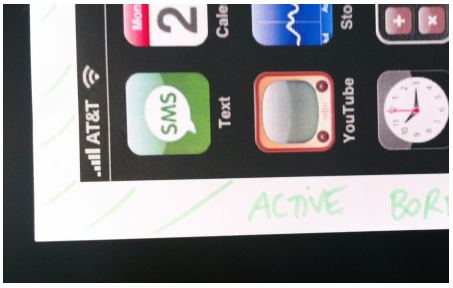

Figure 3.5: active border prototype

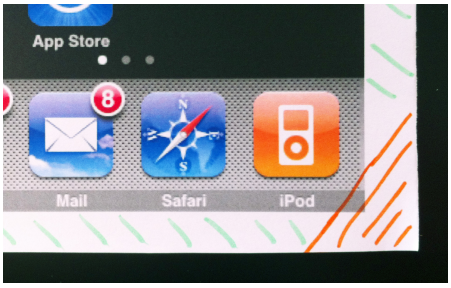

Figure 3.6: active corners prototype

Various interaction techniques can be used to invoke application level commands, as shown in figures 3.3 to 3.6. Working with paper prototypes to generate ideas lead to the definition of a set of five interaction strategies. Each strategy can be consistently implemented for each previously defined command. There is a sixth category, that regroups design suggestions that are not part of a 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.

3.4 Preliminary usability study

In order to design an interactive system that is both usable and engaging, it is important to keep the process human-centred. This can be done by taking a participatory approach, where users explore and evaluate ideas. This study was run with this goal in mind. It is based on an experiment where users engage with low fidelity prototypes of the system, and are asked to evaluate the interaction strategies that are defined in section 3.3.2.

3.4.1 Method

Parameters

The parameters of the experiment are the commands, referred to as *interaction primitives*, and the actions, referred to as *interaction strategies*, defined in section 3.3.2. The six interaction primitives represent each a system feature that is deemed critical to the overall design. For each primitive, there are six interaction strategies. The aim of the study is to have real users compare and evaluate those strategies, and to determine which ones are best fitted for each primitive.

Figure 3.7: Participant and designer during an experiment session.

Experiment

Twelve participants were recruited on a voluntary basis. An experiment 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 2012], but both devices are 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.

During the introduction phase, the following things are explained to the participant:

- the purpose of the study
- the purpose of the TIDE application
- the tasks that the participant will perform
- the principles of working with paper prototypes

During the main experiment, the user is asked to perform a task 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 suggest an action that he/she would perform to obtain the desired effect, and to demonstrate the action using the prototypes
3. the designer gives three action suggestions, the user is asked to demonstrate the actions, then rank them by order of preference.

There is a seventh phase focusing on the pairing procedure. This phase occurs first and is meant as an example to the participant, describing the common structure. At the end of this phase, a slide animation is used to describe all six primitives to the user.

For each command, there are six possible actions. However, it was decided that asking a user to choose between six choices would be overwhelming. Therefore the volunteers are split into 2 groups, each evaluating a subset of the interaction strategies. The repartition is shown in table 3.8.

3.4.2 Results

Participant answers were gathered by the designer 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 contains the position from 1 (highest) to 3 (lowest), that

	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 3.8: 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.

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 3.9 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 same results are presented in the form of charts in figure 3.10.

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

Analysis

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

For the first three primitives, there is a strong coherence in the choice of the participants. The favored strategies are the active border, the action bar and active corners. All three require the user to interact with an area directly around the window in order

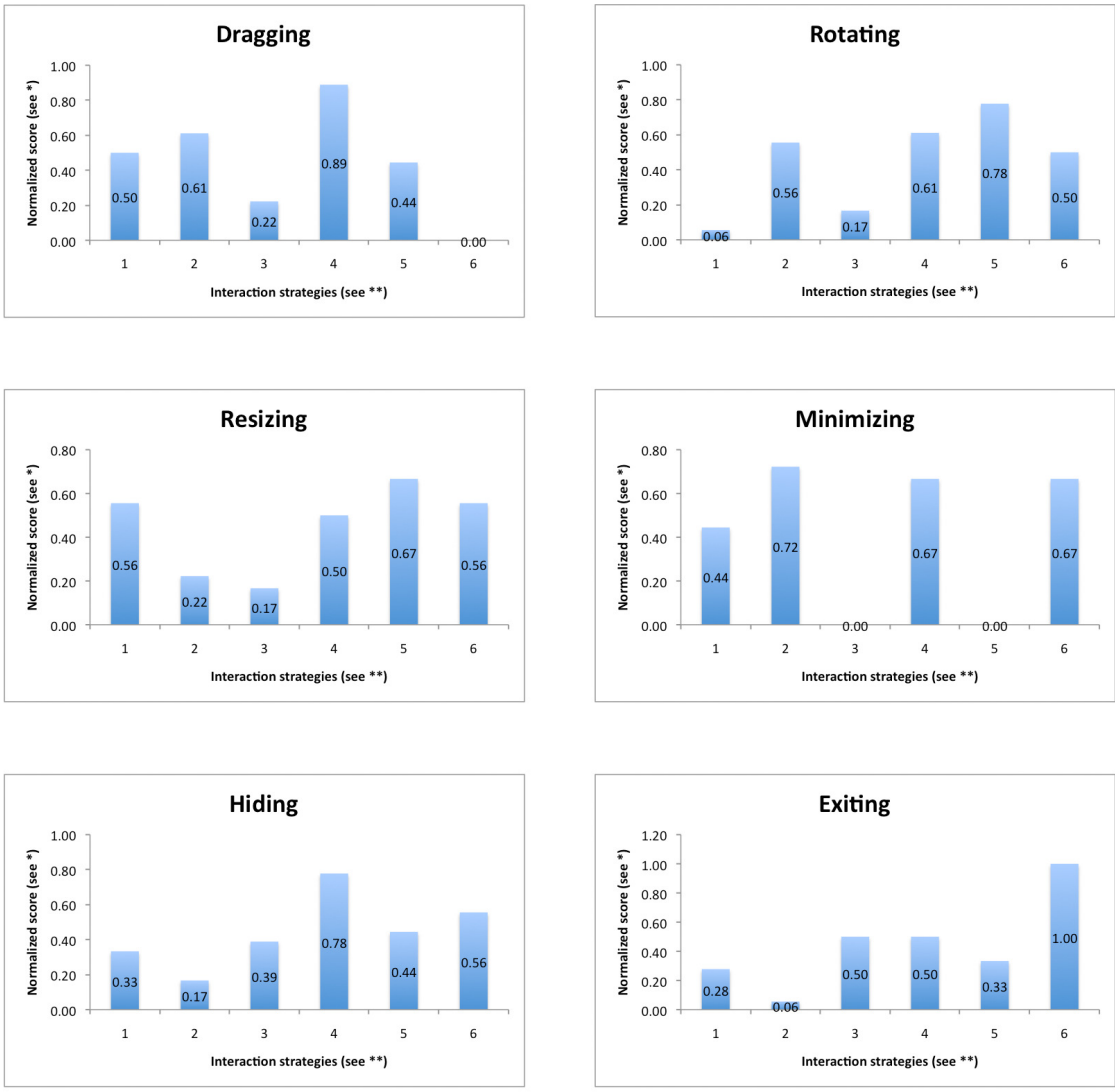

Figure 3.10: The score of the different interaction strategies for each primitive.

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 case of the display extension, the window containing the remote UI is logically avoided because of its role as IO bridge between the tabletop and the personal device.

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 relevant entries in more detail. 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 interaction strategy as a whole, as it does not represent a consistent strategy, but more a patchwork of various implementation suggestions. Those suggestions were chosen because of their originality and interest, and are listed here:

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. It is interesting to see how this correlates with the open suggestion analysis below.

Open suggestion analysis

The open suggestions were multiple and heterogeneous, but further analysis showed a definite tendency in three situations.

In the case of dragging, half of the participants suggested using one or more fingers to touch inside the window containing the remote UI, and perform a drag gesture. This is interesting, because it is an obvious conflict for the developer, i.e. any touch inside the window is forwarded to the remote UI, and can therefore not be interpreted as a surface UI manipulation. 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 remote UI concept. However, this result shows well that the ideal goal would be to design an interactive system whose UI could 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 a good usability would require implementing this interaction technique.

The third consensus was for minimizing, where 7 out of 12 participants suggested dragging the window offscreen (or to a specific location on 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, there is a direct parallel between removing a real piece of paper from the center of a table, and the act of minimizing, hiding or exiting the display extension application. Moving the application out of focus seems like a good solution, and using a simple dragging gesture is a natural way of doing it.

3.5 Design Decisions

The usability study provided precious input from potential users of the system, and this input supported the designer's views that in order to build a successful system, the focus should be on usability and intuition. In this context, intuitive interaction means that the user should be able to learn the system through free exploration, all the time guessing and discovering functionalities. To allow this phenomenon, the implementation should focus on three things. First, it should be coherent. If the features are consistent, the user will be able to derive one from the other. Second, common interaction techniques should be used, such as the picture manipulation gestures (drag, pinch, rotate) that are already successful on touch-based interactive screens. Third, the implementation should refer to the table metaphor when possible, using the user's familiarity with normal table objects to hint at specific features.

The implementation choices are as follows:

- Dragging, rotating and resizing are done by manipulating an active border that frames the window. The active border has the visual appearance of the body of the physical device.
- When the window is dragged off an edge of the table, the application is minimized. Minimizing can also be done by double tapping the active border.
- In its minimized state, the application appears as an icon on the tabletop. This icon can be moved around, restored, or closed.
- On closing the application, the user will be prompted to confirm exiting the application.
- Exiting can also be done by dragging the window to a corner of the tabletop, or by pressing and holding one finger on the active border.
- Hiding is achieved by minimizing the application. Minimizing is therefore also implemented as a result of holding a whole hand over the window.
- When reduced under a certain threshold, the application is minimized. 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.

- the buttons present on the physical device are a logical part of the active border, and their functions are implemented accordingly.

In most software applications, there exists two ways to activate the same command. The advantage is that one implementation can be made to be easy to discover, while the other is more obscure, but a faster or easier interaction. The best example of this are keyboard shortcuts. They can not be guessed by the novice user, but they are used by the expert user for their efficiency. Similarly, the display extension can be minimized easily and quickly by double tapping the active border, but the novice user can easily discover the feature by simply dragging the window off screen.

Chapter 4

The TIDE application: Tabletop Interactive Display Extension for mobile devices

refer to problem formulation:

Following is an open list of problems that we will address in order to achieve device composition by means of implicit interaction.

1. *Setup*: How is a device enabled for integrating with a tabletop? The setup should be simple, to be performed only once by non-technical users. An initial survey of possible solutions points towards the use of tagging mechanisms and/or camera-based object recognition.

2. *Discovery*: How do the tabletop and the device discover and communicate with each other? How do we solve the issues of discovery, handshake, network connectivity, and encryption mechanisms to ensure privacy?

3. *UI transfer*: Given the computational constraints of mobile devices, how can the UI transfer be efficiently implemented so as to support native applications and guarantee a seamless user experience?

4. *Input*: How can the users interact with their applications on the tabletop (touch and other peripherals)?

5. *Interaction Design*: What means of interaction are best-fitted for the tabletop-based systems that we propose to develop? How can we best adapt to public/private uses and single/multiple users? How can we take advantage of the larger interaction surface?

4.1 Device Composition

4.1.1 Device setup

4.1.2 Detection and discovery

–¿ how it is not new, what are the existing options, what would I recommend in this context. Discussion. How did I solve it and why.

Bluetooth isa used by personal server

4.1.3 Vision-based device tracking

vision-based device tracking detection options: iPhone App, Tag, camera based

4.1.4 UI transfer

(I/O approach) - technology issues (slow Veency)

4.2 Surface application UI

4.2.1

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. iphone. http://www.apple.com/iphone/, 2012. (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.

Paul H. Dietz and Benjamin D. Eidelson. Surfaceware: dynamic tagging for microsoft surface. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, TEI '09, pages 249–254, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-493-5. doi: http://doi.acm.org/10.1145/1517664.1517717. URL http://doi.acm.org/10.1145/1517664.1517717.

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.

Juan David Hincapié-Ramos, Aurélien Tabard, and Jakob E. Bardram. Mediated tabletop interaction in the biology lab - exploring the design space of the rabbit. *Ubicomp*, 2011.

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.

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.

Brygg Ullmer and Hiroshi Ishii. The metadesk: models and prototypes for tangible user interfaces. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, UIST '97, pages 223–232, New York, NY, USA, 1997. ACM. ISBN 0-89791-881-9. doi: http://doi.acm.org/10.1145/263407.263551. URL http://doi.acm.org/10.1145/263407.263551.

UPnP Forum. Upnp. http://upnp.org/sdcps-and-certification/resources/whitepapers/, 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.