A Surface Manager for Interactive Tabletops

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

by
Nur Al-huda Hamdan

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. James D. Hollan

Registration date: 02.10.2012
Submission date: 12.03.2013

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, March2013
Nur Al-huda Hamdan

Contents

List of Figures

List of Tables

Abstract

To date, most tabletop systems are designed with only a single application visible and accessible to the users at any time. This can lead to the underuse of the tabletop spacious surface, and is counter-intuitive to the normal working environment of a table. Desktop window managers provide facilities to launch and manage concurrent applications. However, these managers are designed for single-user systems and cannot be directly utilized in tabletops without sacrificing usability.

This thesis proposes *surface managers* as a new package of user-interface software, to enable interactive tabletops as multi-tasking, general-purpose workspaces. A surface manager is a user-interface that supports and facilitates users' concurrent work processes on large, horizontal, multi-touch surfaces. Like desktop window managers, surface managers provide interface elements and policies to lunch and manage concurrent applications. Surface managers additionally provide space management tools, and support simultaneous user interaction and workspace partitioning.

In this thesis, we begin by developing a descriptive model of the context of work in tabletop environments for the purpose of aiding surface manager design. We construct a conceptual framework for surface managers based on the theoretical perspectives of distributed cognition, and tabletop literature. We then apply the framework to design an exemplar surface manager, and describe the design and implementation of the interface. Finally, the findings of the conducted user studies are presented and discussed.

Acknowledgements

I am gratefully thankful to Simon Voelker, my advisor, for his contribution, insightful ideas, and valuable discussions. I would also like to extend my gratitude to Chatchavan Wacharamanotham for his continuous and thoughtful feedback, and to professor Jan Borchers, the head of the Media Computing Group, for his time and support.

Lastly, I would like to thank everyone who has supported me during this thesis, the participants, who voluntarily took part in the user studies, and my family and friends.

Nur Al-huda Hamdan.

Chapter 1

Introduction

"It's not enough that we build products that function, that are understandable and usable, we also need to build products that bring joy and excitement, pleasure and fun, and yes, beauty to people's lives."

—Don Norman

Interactive surfaces call for a paradigm shift in user-interface design

Until the beginnings of the 1990s, interactive computing research was essentially oriented towards personal computing. The yielded guidelines, models, and theories assisted designers in building efficient single-user interfaces. With the proliferation of a new range of shared interactive surfaces, including vertical whiteboards and horizontal tabletops, it has become important to understand the subsequent paradigm shift in user-interface (UI) design and to come up with new models and theories to guide designers.

Interactive tabletops are real-world, multi-touch surfaces augmented with computational capabilities. They form a unique class of computing devices by offering a horizontal surface that affords social interaction, provides unconstrained display orientation, and allows placing physical artefacts on it. In addition, their spacious surfaces can positively influence work dynamics by allowing more natural and direct content manipulation and extending the visibility of the workspace [Shen et al., 2006].

The research and industrial communities have taken great interest in designing applications for interactive tabletops. This has led to the development of novel interfaces and interaction techniques, as well as a body of design guidelines. Yet, most user-interaction on tabletop surfaces is limited to the single-application paradigm, i.e., only a single application is visible and accessible to the user at any time. This paradigm has several disadvantages in tabletop environments. Three of these disadvantages are:

- It constrains the user's interaction to a single application at a time, and requires her to remember the other running activities and how to switch between them.
- It limits the parallelism of co-located users.
- It can lead to the underuse of the surface's size.

To break away from the single-application paradigm, we propose the design of surface managers. A surface manager is a user-interface that supports multi-tasking and concurrent application interaction on interactive tabletops. Throughout this document, we refer to our version of surface manager design with Surface Manager.

Kirsh [2001] suggests that "there is a deep structure to well used workspaces. Any venue that has been adapted to the ongoing workflow needs of a user will support those task specific needs by providing an underlying structure, or context for that work" (p. 307).

In this thesis, we believe that to create a useful workspace in tabletop environments, it is important to first gain a deep understanding of people's physical interaction and work patterns around table surfaces. Therefor, we construct a conceptual framework that defines the tangible and conceptual structures users need during their work processes in tabletop environments. The framework will provide tabletop interface designers with coherence to empirical inquiry, and aid them in their design process. We illustrate the descriptive power of the framework by applying it to the design of Surface Manager.

1.1 Motivation

On classical desktop systems, window managers enable the user to launch multiple applications on the same viewing surface, and interact with them simultaneously without loss of context. However, existing window managers (e.g., in Microsoft Windows and Apple's OS X) are designed for single-user systems with limited screen size, standard mouse and keyboard, support of a single control point, and fixed vertical orientation. Attempts to migrate desktop window managers by merely scaling the interface and adapting the input modality can impose unnecessary limitations on tabletop interaction [Wang et al., 2008].

Desktop window managers provide facilitates to launch and manage concurrent applications

Mobile devices, such as tablets and smartphones, have the same direct-touch input modality as tabletops, but limited screen space. Window managers on these devices mostly allow users to switch between a full-screen application view and the "home" view, where the user can launch another application. On these devices the single-application design paradigm has proven successful, as it was utilized to account for the limited screen space and the mobile contexts-of-use. Currently, commercial tabletop systems, such as Microsoft Surface, implement window management systems which resemble those of mobile devices in form and function. With only one application visible and responsive on the surface at any time, users are forced to continually switch between contexts. This is unnatural and counter-intuitive to the normal working environment of a table, where the user is able to view and interact with multiple pieces of information in parallel.

Mobile window managers implement the single-application design paradigm successfully

In Table 1.1, we summarize the points in UI design that are affected by the properties of different interactive devices.

These observations drove us to the design of surface managers that provide similar window management facilities, such as user sessions, hosting concurrent applications, and layout policies, while being specifically designed for the form and context of interactive tabletops.

Surface managers design requires an understanding of how people work in interactive tabletop environments. How-

Property	Desktop PC	Mobile device	Tabletop	Effect on UI design
Concurrent users	1	1	n	Multi-tasking degree; collaboration support
Screen size	Medium	Small	Medium-large	UI layout
Context	Personal; stationary; private and semi-public contexts	Personal; mobile; private, semi-public, and public contexts	Shared; stationary; semi-public and public contexts	Privacy settings; filing system; features and task domain
Rendering orientation	Vertical	Vertical and horizontal	Arbitrary	UI layout; shared views
Contact points	1	10	10 <	Focus shift; access coordination policies

Table 1.1: Paradigm shift in UI design between different interactive devices.

ever, little is understood and has been studied of the kinds of interactions parallel and collaborative users perform when attempting to accomplish several different tasks in a single session. The lack of standard tabletop technologies and long-term users led us back to more fundamental theories of workplace and tabletop territoriality. In particular, we build on Kirsh's [2001] model of *context of work*. Kirsh studied office workplaces to define an invariant structure of work that abstracts from superficial physical attributes, and is shared between various office settings. We extend his work for interactive tabletop environments with observations and empirical results from tabletop literature.

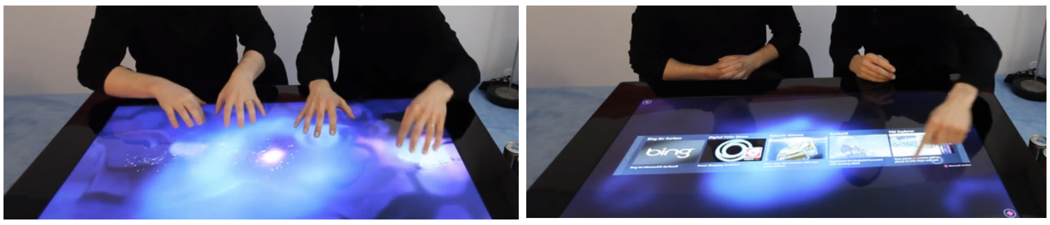

(a) (b)

Figure 1.1: Presentation of the single-application paradigm. (a) two users interacting with a single application simultaneously, (b) single user dominating the interaction.

1.2 Problem Statement

In a normal working environment of a traditional table, people are able to view and interact with multiple pieces of information in parallel. To date, most design and interaction on tabletop surfaces is limited by the single-application paradigm.

As an example, consider an interactive tabletop that implements some window management facilities, such as Microsoft Surface. In Fig.1.1 (a), two users are interacting with a single application simultaneously, when one participant interacts with the surface, her actions are visible to the other participant and supplementary to his interaction. At the moment when one participant decides to switch to another application, the other participant is forced to halt his or her interaction, and the table surface switches from displaying the full-screen application view to the manager's view, (see Fig. 1.1 (b)).

Current tabletop applications run in full-screen mode and assume co-located users share a single goal

Systems like this one, only allow a single application to be visible and accessible at any time. They assume that co-located users share a single goal that they want to achieve. In addition, they give control over the view of the entire surface to a single user. This could compel co-located users to stop what they are doing at any time, and interrupt their workflows. Unpredictable power structure issues can also rise from giving the control over the entire surface to a single user. Consequently, the table surface cannot support parallel users with different goals, which limits the types of supported interaction styles and can hinder users' productivity.

How can tabletop interface designers address the single-application paradigm limitations? One possible solution is to provide multiple access control points and enable concurrent applications to appear on the surface and be interactive simultaneously. As this may appear to address the described limitations, however, designers still need to determinant how people work on these large horizontal surfaces, how to distribute control, and what structures are needed to facilitate their tasks. The conceptual framework of surface managers is intended to abstract and define these structures to provide designers with a skeleton that can be then customized to the desired context.

1.3 Thesis Overview

In this section, we outline the structure of this thesis and give a brief overview of the content of each chapter:

Chapter 2–"Background and Related Work". In the second chapter, we provide context for this thesis by first reviewing tabletop research from HCI and CSCW perspectives, as well as tabletops in work environments and design. Next, we review desktop window systems, how they influence this current work, and describe the paradigm shift in UI design between desktops and tabletops. Last, a review of the related literature is conducted.

Chapter 3–"Conceptual Framework". In chapter three, we develop a descriptive model of the context of work on tabletops for the purpose of aiding surface manager design. We focus on the common structures needed to support the work processes of co-located users in tabletop environments. We describe the modeling approach of access point design and analyze the resultant design space.

Chapter 4–"System Design". In the fourth chapter, we outline the concepts, design, and implementation of our version of a surface manager, a.k.a Surface Manager. We apply the conceptual framework to the design process, describe our UI controller metaphor, and analyze our interface using the principles of direct manipulation

Chapter 5–"Users Studies and System Evaluation". In this chapter we describe the purpose, design, and findings of the two users studied we conducted during this thesis. These are: a preliminary, participatory study, and a qualitative, observational user study and system evaluation.

Chapter 6–"Summary and future work". We conclude this thesis with a summary of our findings and contributions, and potential future extensions of our work.

Chapter 2

Background and Related Work

"You see things vacationing on a motorcycle in a way that is completely different from any other. In a car you're always in a compartment, and because you're used to it you don't realize that through that car window everything you see is just more TV. You're a passive observer and it is all moving by you boringly in a frame."

—Robert M. Pirsig

In the 1990s, computer displays started to move beyond those dominated by the form factors of traditional desktop computers. Computer displays transformed in shape and size, and the pixels became input as well as output devices [Buxton, 2007].

Within the same time frame, Mark Weiser published his famous article "The computer for the 21st century", describing his vision of Ubiquitous Computing (UC). In his vision, a computing environment is a physical space augmented with perceptual information, not a virtual environment that exists to store and run software [Weiser, 1991]. His vision had two aspects: (a) a proliferation of devices at varying scales ranging in size from hand-held "inch-scale" personal devices to "yard-scale" shared devices; and (b) new applications that leverage off these devices and infrastructure with new paradigms of interaction. Consequently, Weiser noted, traditional window systems will not be able to cope with these substantial changes in application design and interaction.

Traditional window systems will not be able to cope with UC systems

Interactive tabletops are large, horizontal, multi-touch surfaces, which assume the shape of a traditional table. They are the embodiment of the yard-scale, shared devices in Weiser's vision.

Interactive tabletops are inherently ubiquitous

2.1 Interactive Tabletop Systems

Tabletop literature is mainly distributed between two research fields: Human-Computer Interaction (HCI) and Computer Supported Cooperative Work (CSCW).

In HCI, tabletop research have produced novel UIs (e.g., [Fitzmaurice et al., 1995]) and metaphors (e.g., [Besacier et al., 2007, Hinrichs et al., 2005]), interaction techniques (e.g., [Yoshikawa et al., 2012]), multi-touch frameworks and toolkits (e.g., [Shen et al., 2004, Hansen et al., 2009]), and resulted in multiple design guidelines for the development of tabletop interfaces [Apted et al., 2009, Scott et al., 2003].

In CSCW, detailed studies have been carried out describing how factors, such as tabletop ergonomics [Rogers et al., 2009], group size [Ryall et al., 2004], surface orientation [Rogers and Lindley, 2004], and indirect input techniques [Pinelle et al., 2009], can influence group processes. Other work focused on how horizontal multi-touch surfaces can increase workspace awareness of collaborators' actions [Pinelle et al., 2008].

2.1.1 Tabletops in Work Environments

Interactive tabletops have been studied in several contexts and physical contexts, such as public contexts or "in-the-wild" (e.g., restaurants and museums [Hornecker, 2008]), semi-public contexts (e.g., classrooms [Piper and Hollan, 2009] and work offices [Morris et al., 2008]), and private contexts (e.g., homes and domestic environments [Kirk et al., 2012, Wigdor et al., 2007]). However, when tabletops were first investigated, they were originally envisioned for work environments.

Tabletops were originally envisioned for work environments

The iRoom [Streitz et al., 1999], Roomware [Streitz et al., 2001], and DiamondTouch [Dietz and Leigh, 2001] are among the early systems that presented the vision of integrating interactive tabletops into collaborative offices for brainstorming, meetings, and presentations. For example, Roomware research project (Fig. 2.1), originated as the i-LAND project, integrated a table called the InteracTable, a plasma display that allowed multiple document views and touch based workspace rotations on the surface. DiamondTouch was the first tabletop system to be able to determine multiple user touch input simultaneously.

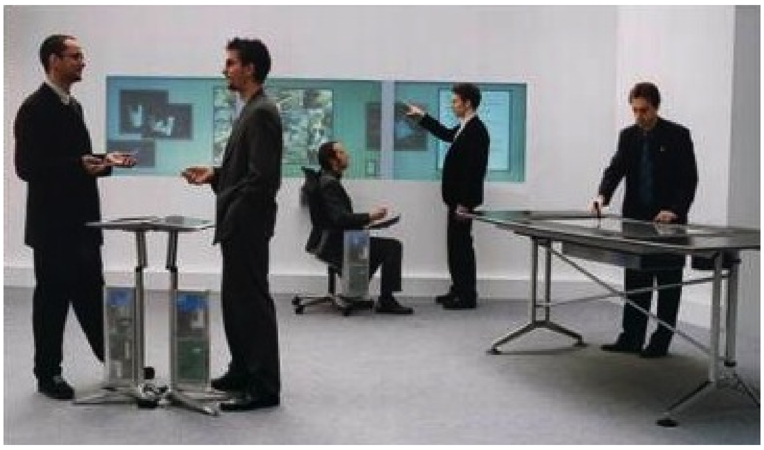


Figure 2.1: Roomware interactive work environment [Streitz et al., 2001].

Peltonen et al. [2008] reported that tabletop interaction in different physical contexts can be affected with a number of dimensions, such as the allowed number of co-located users, social relationships between users, and knowledge about the display and its use.

Isenberg et al. [2009] compared collaborative information exploration on tabletops in work environments and public spaces. In work environments, such as meeting rooms or research labs, users can be characterized by a vast amount of domain-specific knowledge, while in public spaces, people's level of knowledge is unpredictable. In work environments, people work within teams of colleagues, unlike public spaces, where strangers can approach and interact with the table at the same time. Moreover, work teams also often spend considerable time using the table, and are willing to put the time to learn the interface. In public contexts, people are pass byers, they approach the table with no predefined intention or knowledge, and only remain for a few moments. These core differences implicate the way a tabletop UI should be designed to accommodate for the needs and requirements of each setting.

UI requirements differ substantially between different physical contexts

In this thesis, we focus on interactive tabletops in work environments in two manners: (1) we attempt to define the common tangible and conceptual structures people need to perform their tasks in these environments, and (2) we investigate several design alternatives in Surface Manager to support these structures and the special needs of these settings.

2.1.2 Tabletop Metaphors

Metaphors have been an organic part of window managers since they first appeared in mid-1970s. Well-designed metaphors are significant components of the UI because they allow users to discover the purpose of interface elements and how to interact with them. This is usually achieved by building the metaphors on familiar, everyday concepts.

On traditional tables, people use pencils and papers for their activities. To facilitate users transition to tabletop systems, Besacier et al. [2007] proposed *peeling* and *slots* paper metaphors. Ståhl et al. [2002] designed Ponds, a tabletop interface that presents data elements in a three-dimensional virtual pond, providing a new interaction metaphor for searching and browsing digital media. Media items related to a recent search query float to the surface of the Pond.

Items which have not been interacted with for some time sink to the bottom of the Pond, gradually disappearing. This garbage collection also happens if the Pond becomes crowded (see Fig. 2.2).

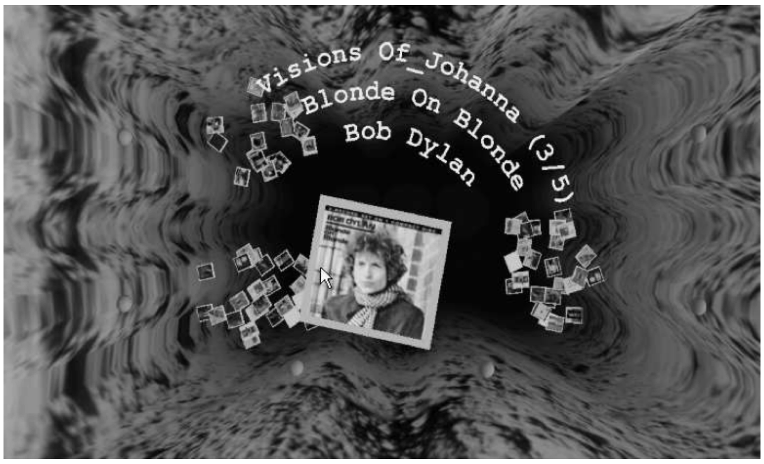


Figure 2.2: Ponds metaphor for information search and visualization on tabletops [Ståhl et al., 2002].

Metaphors designed for Natural User Interfaces (NUI) can also apply to interactive tabletops (e.g., Magnet, Sphere, and Unfold metaphors [Hofmeester and Wixon, 2010]).

In Surface Manager, we explore a new simple metaphor—*controller metaphor*—that provides users with flexible UIs and control over their work.

2.1.3 Tabletops Design Requirements and Constraints

Large, horizontal tabletop displays afford quite different interactions than common desktop displays. They introduce several issues for interface designers that vary considerably from those investigated in and inherited from the first official desktop window manager in Xerox Star and successive systems (see section 2.2.2).

Tabletops UI requirements

Hinrichs et al. [2005] define a set of requirements that should be considered when designing interactive tabletop

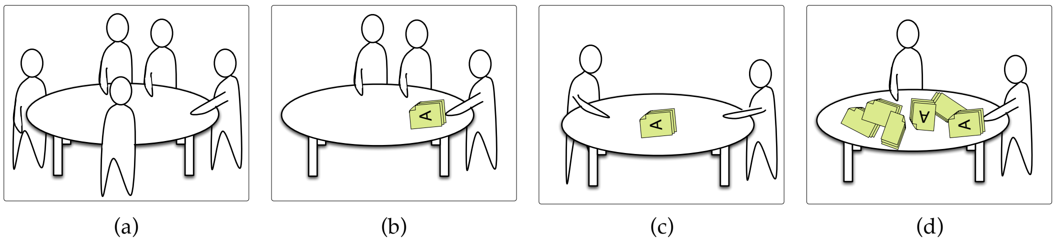


Figure 2.3: Tabletops design requirements and constraints. (a) seating arrangements, (b) orientation, (c) reachability, and (d) surface clutter.

UIs: seating arrangements, orientation, and reachability. In this section, we review these requirements, in addition to surface clutter and direct-touch (see Fig. 2.3).

Seating Arrangements and Ownership

When designing portable, scalable, or extensible tabletop interfaces the designer must consider all possible seating arrangements

Scott and Carpendale [2006] noticed that users during a collaborative task are mobile, some users leave, others join or change their seating positions. DiamondSpin [Shen et al., 2004] implements various important features to face tabletop challenges such as color coded frames to distinguish between different window owners, assumed the tabletop system provides user-identification information. It also supplies relocatable and non-modal menu-bars that adapt to several seating arrangements, as well as various tabletop sizes and shapes. We adopted similar techniques for ownership and flexible seating arrangements in Surface Manager.

Orientation

A unique feature of interactive tabletops is that they support face-to-face interaction. In a collaborative task, for example, this can lead to each user having a different view of the same content, and in some views text becomes in comprehensible [Kruger et al., 2003]. Orientation is relevant for visibility and comprehension, in addition to its social role of indicating willingness to share a document [Ringel et al., 2004, Tang, 1991].

The solutions proposed to address object orientation problem in tabletops include software-based approaches (e.g., allowing free manual object orientation, or having items on the table automatically orient themselves, to face the nearest edge [Shen et al., 2004], or the user who most recently touched them [Morris et al., 2004a]), and hardware-based approaches (e.g., a view-dependent tabletop display [Kakehi and Naemura, 2008]).

Reachability

Tools should be available to facilitate document sharing, passing and duplicating

Tabletop surfaces can be quite large. This means that some areas and workspace items may not be reached by every user around the table. [Scott et al., 2004] noticed that reaching across the table can cause territoriality issues and become socially awkward when reaching across other persons' work or into other persons' workspaces.

Ringel et al. [2004] proposed a series of four interaction techniques: release, relocate, reorient, and resize to support document sharing on multi-touch tabletop displays. Dynamic Portals [Voelker et al., 2011], is another lightweight interaction technique that supports passing a group of documents across large tabletop surfaces, while maintaining their relative spatial arrangement. Focusing only on reaching far items, [MacKenzie and Jusoh, 2001, Myers et al., 2002] proposed the use of indirect input devices, gyroscopic mice and laser pointer, respectively. However, when compared with the traditional mouse pointer, both devices performed poorly in terms of throughput and speed. In addition, users reported it to be cumbersome to switch between different input devices and shift input mode.

Surface Clutter

Tabletop surfaces tend to be large, nevertheless, due to the act of scaling digital objects to their life-size or because of low-resolution displays, and tabletop's affordance to accommodate more than a single user at a time, the table surface tends to clutter.

Leithinger and Haller [2007] investigated six different menu layouts for interactive tables under various cluttering conditions. In their study, they found out that menu types suffering from occlusion, pie menus, for example, showed significant disadvantages compared to their proposed user-drawn context menus. Occlusion can also occur from placing physical objects over virtual elements on the table surface. Furumi et al. [2012] presented a user interface widget, called SnapRail, that detects a physical object on the surface and the virtual elements under the object. It then snaps the virtual elements to a rail widget that appears around the object. The user can then manipulate the virtual elements along the rail widget.

Direct-touch

Because people are accustomed to interacting directly with items on traditional tables, a direct-input device provides a familiar method of interaction

A natural collaboration is said to be afforded where the tabletop invites people to reach out and touch it with their fingers [Shen et al., 2004]. This form of direct-touch can increase visibility of actions, enabling implicit coordination and awareness among co-located users [Scott et al., 2003, Kruger et al., 2003, Forlines et al., 2007].

However, it has been found that touch surfaces are limited in the kinds of interactions they can effectively support. Human fingers come in different shapes and sizes, but even the smallest ones cannot achieve mouse-pixel precision [Seto, 2012]. This problem is referred to as fat-finger problem. Fingers are clumsier than pointing devices, and hence more error-prone for fine-grained and precise operations [Rogers and Lindley, 2004].

Other direct-touch disadvantages include occlusion from body parts such as hands and arms, [Scott et al., 2003, Forlines et al., 2007]. Brandl et al. [2009] designed occlusion aware menus that adapt to the user's handedness and position on the tabletop. The adaptive menu placement method was based on direct touch and pen tracking. Direct-touch also imposes limitation on text-input [Hinrichs et al., 2007].

2.2 Window Management Systems

In early 1970s, Alan Kay and his group at Xerox PARC introduced Smalltalk programming language and environment. Smalltalk demonstrated the power of graphical, bitmapped displays, mouse driven input, windows, and simultaneous applications. It became the common ancestor of all window systems [Gosling et al., 1989].

Smalltalk promoted the notion of personal computing and pioneered interactive programming environments

Basic window system tasks include, input and output handling, i.e., handling user and application requests, as well as window management, i.e., managing and providing the user controls for windows.

The window system operates between the hardware, i.e., input and output peripherals, and the application layer. In Fig. 2.4, we show the 4-layer window system architecture as described by Gosling et al. [1989].

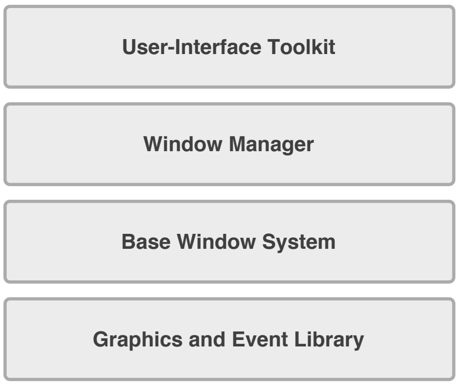


Figure 2.4: A 4-layer window system architecture [Gosling et al., 1989].

The graphics and event library, and the base window system layers handle low level operations, such as event routing and drawing on the screen. Together they are sometime referred to as the window system [Myers, 1988], reserving the term window manager to cover the UI and layout polices.

2.2.1 Window managers

A Window manager is associated with the user session

A window manager is a software package that helps the user monitor and control different contexts by separating them physically onto different parts of one or more display screens [Myers, 1988]. It enables a user-centered system view by providing a homogeneous look and feel across the interface.

A Window manager functions as screen manager and a session manager

Window managers assume three primary roles on the desktop computer:

- They allow separate activities to be visible and accessible concurrently by putting them in physically separate parts of the computer screen, a.k.a windows.
- They provide layout policies, and allow the user to bring up menus and dialog boxes associated with running applications, and manipulate the location and size of windows.
- They provide a higher level interface to input and output devices, and therefore can support: (a) much higher quality UIs, for example, the window managers on Star and Macintosh helped support the early Desktop metaphor; and (b) make applications portable from one machine to another, since the same window manager procedural interface can be provided on different machines [Myers, 1988].

Window Managers are usually accompanied with a UI toolkit

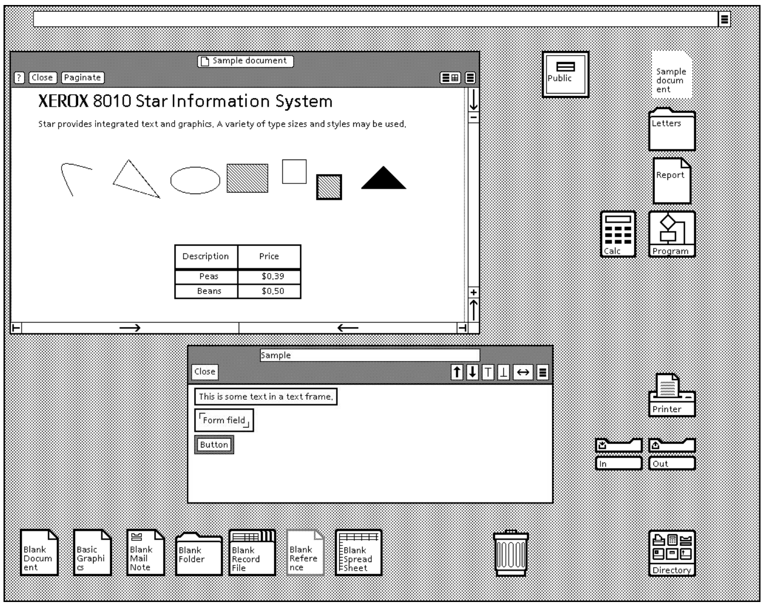
A toolkit is composed of several graphical elements and widgets, that are used to design a complete UI. Toolkits bring consistency across applications, and simplify the design process for developers by hiding the complexity of window management as well as handling input and output events. They also allow application developers to maintain the same look and feel of the underlying window manager. The widgets can be primitive such as buttons and switches, or complex such as windows and dialog boxes.

Surface Manager represents the window manager, and UI toolkit layers in Gosling et al.'s [1989] window system architecture.

2.2.2 The Evolution of Window Managers

From the mid-1970s to the mid-1980s, there was much research on window systems. With the introduction of Star Information System in 1981, Xerox changed notions of how interactive systems should be designed [Smith et al., 1982]. Star bitmapped-screen came with distinguishable interface features: windows, icons, pop-up menus and mouse driven interface (see Fig. 2.5).The WIMP interface paradigm, i.e., windows, icons, menus and pointers, is still adopted to the date on the predominant desktop operating systems (e.g., Microsoft Windows, and Apple's OS X).

Xerox changed the notion of interactive computing by introducing Graphical User Interfaces


XEROX 8010 Star Information System

Star provides integrated text and graphics. A variety of type sizes and styles may be used.

Description	Price
Peas	$0,39
Beans	$0,50

This is some text in a text frame.

Figure 2.5: Xerox Star UI.

Star was the first commercial system to provide the capabilities of a windowing system—allowing several programs to display information simultaneously in separate areas of the screen, rather than each consuming the entire display [Smith et al., 1982]. The system controlled the size and position of these windows when they were opened and closed. Star first allowed windows to overlap, but later found that users spent a lot of time adjusting windows. In Star's successor, ViewPoint, users were given the option to choose between an overlapped or tiled windows layout.

The Andrew window system was designed to be portable and hardware independent. It explored a space-filling tiled window layout, where windows are resized automatically, i.e., when one window grows others shrink. However, users found it confusing and the approach was abandoned [Morris et al., 1986] .

UI Metaphors

Tool metaphor vs. Desktop metaphor

In the 1980s, two interface design metaphors were prevalent: the Tools metaphor on Smalltalk-80, Cedar, and SunView; and the Desktop metaphor on Star. In the Tools metaphor, users dealt mainly with applications as tools. To accomplish a task, the user started an application, then specified the data files she wants to edit. In early systems, such as SunView, which application can open which files was a burden put on the user to remember. In contrast, in the early Desktop metaphor, users dealt with data files. To accomplish a task, the user opened a data file and the system automatically launched the relevant application.

Today, desktop window managers support both metaphors' file access capabilities. In mobile devices, the file management model is similar to that of the Tools metaphor. However, the model was improved by only showing the user the files that the selected application supports. In desktop and mobile systems the user can choose to open a file in the default supporting application, or switch to another application that supports that file type.

The Desktop metaphor employed analogies with the physical world to help make better memory associations. An advantage of the Desktop metaphor over the Tools' is that it oriented users toward their data and not the software.

In Surface Manager, we extend the file management model of the early Desktop metaphor to minimize users' interaction with the software, and provide users with advanced application switching mechanisms. We avoid the hierarchical filing and many analogies of the Desktop metaphor that are single-user oriented.

Icons first appeared in Tajo. The aim was to alleviate the problem of ”naming and remembering” [Johnson et al., 1989]. Icons followed the Desktop metaphor analogy and represented data files with pictures of office objects. This reduced search time, and users were allowed to organize icons, i.e., files and applications, spatially rather than by distinctive naming. In our interface, we abandon the Desktop metaphor analogy of icon presentation, and instead use thumbnails that were employed in X Window System [Scheifler and Gettys, 1986].

Direct Manipulation

Star designers based the system heavily on the principles of direct manipulation. They allowed users to directly manipulate objects on the screen by pointing at them or by using visible menus.

In multi-touch systems, direct manipulation is the prevalent interaction model. In section 4.5, we describe how direct manipulation principles, introduced by Shneiderman [1993], apply to the design of Surface Manager.

Commercial Systems

In 1984, the Macintosh from Apple Computer was released to the public. The Macintosh was a small machine with high-quality UI. Most of the ideas in Macintosh were browned from Smalltalk, Star, and Tajo. However, it was a landmark in UI design, providing several ideas in a coherent package that was easy to use, even though it was hard to program.

In 1985, X Window System [Scheifler and Gettys, 1986], a distributed window system, made a major innovation. X separated window managers from being intertwined with the complete window system. In X10, and later X11, window managers were treated as separate applications, and the user was able to choose, install, and exchange window managers over the window system.

Modern window systems are still constrained by the Desktop metaphor limitations

Most modern window systems (e.g., Apple's OS X, Microsoft Windows, and Ubuntu) are still based on the Desktop metaphor. The main improvements can be seen in the quality of graphics (e.g., Aqua theme on OS X and Aero Glass on Windows 7), and inter-application communication. For instance, animation is now heavily employed to give the user the sense of continuity and direct manipulation. Softer lines, shading, and textures are employed to reflect better affordance, feedback, and even feedforward [Vermeulen and Coninx, 2013]. Drag-and-Drop features have also improved across applications and online services.

The Modern UI Style is the new tile-based design in Microsoft Windows 8. Many of Windows UI changes are centered towards improving its experience on tablet computers and other touchscreen devices. The new UI replaced the Start menu with a new tile-based Start screen similar to that of Windows Phone. The window system opens applications in full-screen mode, and allows for limited multitasking. It also provides a vertical toolbar known as the charms bar to provide access to system and application-related functions, such as search, and settings. The traditional desktop environment is still accessible for running concurrent desktop applications.

2.2.3 A Paradigm Shift: from Personal Desktops to Shared Tabletops

Five aspects to consider when redesigning software from the desktop computer to interactive surfaces

In this section, we highlight five aspects that we believe are critical to investigate when attempting to redesign software from the desktop computer to interactive surfaces. These are: the WIMP paradigm, input focus shifting, single-user orientation, direct manipulation, and the Desktop metaphor.

WIMP paradigm. Most tools for building Graphical User interfaces (GUIs) are built around the WIMP paradigm. But WIMP interfaces do not work well for multi-touch input techniques because the paradigm is too closely tied to the single pointer model, and multi-touch interfaces lack the precision of indirect pointing devices.

Focus shift. In desktop environments, the term ″listener″ or ″input focus″ is used to refer to the window that receives the keyboard and mouse events. Window managers provide several policies to select the listener. In modern systems, the listener is no longer a singleton, for example, in Apple's OS X, the keyboard can be attached to one window and the mouse can be used to scroll another. On multi-touch surfaces, it is possible that one user has more than one control point at the same time, for example, two fingers. With multiple control points, multiple users can work on an interactive surfaces simultaneously. For example, if two users attempt to type-in text, each in a separate document, a mechanism should be provided to direct the input of a specific keyboard, hardware or software, to the correct document.

The notion of focus shifting is changed dramatically in multi-touch interfaces

Single-user orientation. The desktop computer, the Desktop metaphor, and consequently the desktop window manager are single-user oriented. Desktop window manager interfaces and policies were design for a limited, vertical screen where every part is assumed to be reachable by a single user. For example, the icons and task bar on the classical desktop, once projected on a large, horizontal surface become neither accessible from some locations around the table, nor readable from any orientation.

Direct manipulation. Current desktop interfaces do not totally comply with direct manipulation principles [Beaudouin-Lafon, 2000]. On interactive surfaces, directness is more sensible. For example, Hancock et al. [2006] designed a rotation and translation mechanism for tabletop interaction (RTN) that allows users to manipulate objects directly and incrementally.

Desktop Metaphor. Window management systems that are based on the Desktop metaphor have been an essential part of the personal desktop computer UI for the last 35 years. The Desktop metaphor served its role well by allowing users to view and switch between concurrent applications when the number of these applications was limited. Today, however, to accomplish a task, people engage with multiple applications and online services, and need to switch between them frequently and quickly [Henderson Jr and Card, 1986]. In the Desktop metaphor, switching between tasks can involve several operations (e.g., searching, rearranging, iconifying, opening, and resizing windows).

In Surface Manager, we avoid many of the Desktop metaphor analogies because they are mainly single-user orientated. For example, we attempt to narrow the gap between information access and information viewing, a known limitation of the Desktop metaphor [Kaptelinin and Czerwinski, 2007], by minimizing the number of operations required to open a file or an application. One way to achieve this is by designing a flatter system hierarchy.

2.3 Related Work

Current tabletop UIs are designed to support users' interaction with a single application. On the desktop, window managers are designed to support multiple simultaneously running applications, while desktop PCs have limited screen size compared to tabletops, and only single control point. As we lack and desire the application concurrency feature in tabletops, we are faced with two design alternatives: (1) we could adapt exiting window managers to tabletops' size and input modality, or (2) we could attempt to redesign these managers to be specifically customized for tabletops' design space and context.

Maintaining front end software from desktops to tabletops rises many issues

Wimmer and Hennecke [2010] suggested extending desktop window managers with multi-touch capabilities to enable researchers to transparently control user input and graphical output, simultaneously supporting both native multi–touch and single–pointer legacy applications. However, Andreychuk et al. [2010] and Wang et al. [2008] reported that it is not trivial to migrate front end systems that were originally built for vertical screens to be utilized on horizontal displays. For instance, while attempting to migrate the AgilePlanner, an application designed for the traditional PC, Wang et al. [2008] reported a number of issues such as the tabletop size, orientation, and the user group size. In addition, challenges like standard GUI components being unsuitable for the table input modality, and the support of multiple collaborators working at the same time.

Collins et al. [2011] investigate Thacts—Things that should be supported and Actions that the user should expect to be able do with those Things—to support core Operating System functions on tabletops. The authors suggested that, while it is important to acknowledge the mental models people bring from desktops to tabletops, tabletops are very different and the design of software support for Thacts on tabletops must be different from that on desktops.

Tabletops software should not be limited by the mental models people bring from desktops

Based on the presented reports and investigations and our previous knowledge of tabletops design constraints, we choose to adopt the second design alternative—attempt to design a window manager that is specifically tailored to the shape, size and display technology of large, horizontal surfaces, as well as to tabletop context requirements.

On tables, workspace partitioning is a natural human act, and one that should be supported in a tabletop system [Scott et al., 2004]. It has been observed that leaving this practice to social protocols can lead to several conflicts and social awkwardness [Morris et al., 2006a]. A software-based solution was proposed by Morris et al. [2004b] in the form of coordination policies that attempt to control a document's manipulation access rights. Klinkhammer et al. [2011] presented a hardware-based solution that employs tabletop-integrated multi-user tracking system to provide data on a user's location and movement. Using this data, the system assigns a visually separated display space to each user, the space serving them as a personal territory. In Surface Manager, we implement some of the software-based coordination policies suggested by [Morris et al., 2004b].

Workspace partitioning is a natural human act that should be supported by tabletop interfaces

Tabletops have been recognized to be well suited for collaborative interaction. Previous research has shown that collaborators tend to move back and forth between loosely and tightly coupled work [Tang et al., 2006]. Another study [Morris et al., 2004b] demonstrated that users preferred to work individually on some parts of a problem when the system used was capable of supporting such individual activities. Thus, for an interface to maintain and augment the social role of tabletops, it should support several collaboration styles, simultaneous user interaction, and workspace awareness.

Tabletop UIs should support several collaboration styles, simultaneous user interaction, and workspace awareness

In CoSpaces [Mahyar et al., 2012], color-coding has been used as a mechanism for providing awareness among users. Users partition their work using Worksheets (Fig. 2.6). Each Worksheet defines a work territory, either personal or shared, and contains color-coded tab-based views to enable access to other users' work. Similarly, we use color-coding to indicate ownership.

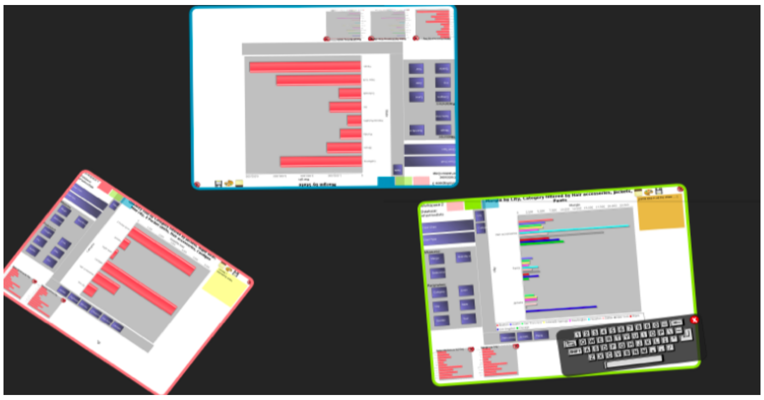


Figure 2.6: CoSpaces [Mahyar et al., 2012].

Tabletops should provide flexible space management and content sharing tools

As users interact over tabletops they will come to the need of space management and content sharing tools [Morris et al., 2010]. Storage bins [Scott et al., 2005] (Fig. 2.7) are mobile, adjustable container widgets. They offer lightweight interaction mechanisms to support information storage, organization, and sharing. Storage bins allow items to be added or removed as a group or individually. Users can use storage bins as personal, group or storage territories. Storage bins can be moved around the table, resized, and collapsed, allowing users to manage the available space more efficiently. In Surface Manager, we implement storage several units that provide similar facilities to organize and retrieve objects.

Tabletop UI widgets should account for reachability, orientation, and direct-touch issues

UI widgets on a large tabletop are constrained with reachability, orientation, and direct-touch issues. To address these problem, many gesture-based solutions have been proposed to invoke widgets at the location and orientation of the user, while maintaining a natural user interface experience [Jacob et al., 2008]. For example, [Strothoff et al., 2011] used the number of touches, while [Brandl et al., 2009,

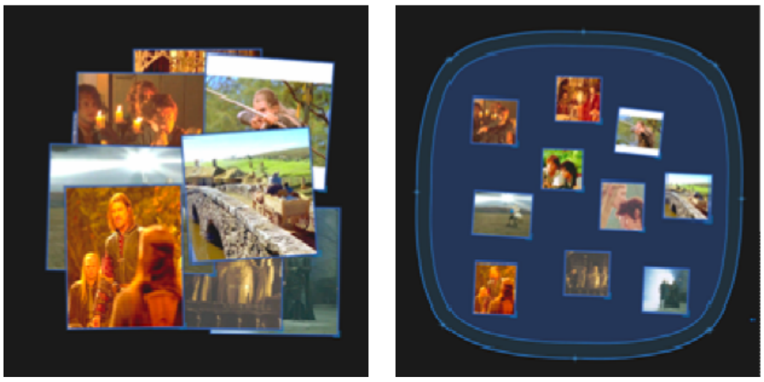


Figure 2.7: Storage bins [Scott et al., 2005].

Bartindale et al., 2011] utilized the contact shape of hands. HandyWidgets [Yoshikawa et al., 2012], are widgets localized around users' hands that are invoked by a bimanual multi-touch gesture called "pull-out" (see Fig. 2.8). HandyWidgets were designed to be robust against casual touching, and to be occlusion aware. Marking menus, proposed by [Lepinski et al., 2010], similarly support gestural activation and save screen real estate, by only popping-up when being used.

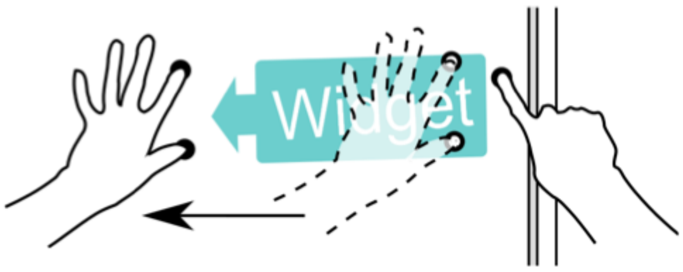


Figure 2.8: HandyWidgets [Yoshikawa et al., 2012].

Tabletop filing system need to account for limited text input, collaborative interaction, and context-of-use

To access shared and personal files on interactive tabletops, several approaches have been proposed. These approaches include: associative file access (e.g., [Collins et al., 2007]), faceted browsing and visualization approach (e.g., [Shen et al., 2006, Morris et al., 2006b]), and interactive file transferring techniques using personal carried devices, such as laptops and smart-phones, in conjunction with tabletop surfaces (e.g., [Shen et al., 2003, Rekimoto and Saitoh, 1999]). For example, in the Personal Digital Historian (PDH) application, Shen et al. [2006] allowed users to access archives of digital material like video clips, photographs, and documents, by visualizing these items on the rim of a circular tabletop. The application included other hierarchical layouts as well.

Several software toolkits (e.g., DiamondSpin Shen et al. [2004] and PyMT Hansen et al. [2009]) provide widgets and interaction techniques to accelerate the development of tabletop applications. However, these toolkits were designed for the single-application paradigm and do not include any space management or layout policies. Wu et al. [2011] developed uPlatform, a tool designed specifically for creating customizable multi-user windowing systems on interactive tabletops. Although compelling, uPlatform does not provide insights on user interaction and behavior in these environments, nor provide a coherent design for window managers on tabletops.

In a design effort to support application switching on tabletops, Ackad et al. [2010] proposed Switch. Switch provides four functionalities: change application, switch between the set of files available for an application, alter application settings, and activatedeactivate interface elements within an application.

2.3.1 Summary

Designing tabletop UIs allows the designer to exploit their unique design spaces

By merely adapting or migrating front end software from desktops to tabletops, we risk imposing limitations on the possible interaction afforded by tabletop surfaces. Interactive tabletop UIs should be designed to support workspace territories, various social interaction settings, and provide space management tools, UI widgets, and filing systems that account of tabletops design constraints.

Chapter 3

Conceptual Framework

"Ultimately, we are deluding ourselves if we think that the products that we design are the 'things' that we sell, rather than the individual, social and cultural experience that they engender, and the value and impact that they have. Design that ignores this is not worthy of the name."

—Bill Buxton

In this chapter, we develop a descriptive model of the *context of work* in tabletop environments for the purpose of aiding surface manager design. We focus on the common structures needed to support the work processes of colocated users. In particular, structures that facilitate users' access to system resources, support flexible workspace construction, and coordinate users' actions on a horizontal interactive surface. We are building a conceptual framework that extends Kirsh's [2001] model of context of work in office environments that is based on the theoretical perspectives of situated and distributed cognition.

Three concepts to define tabletop work structures

Our framework utilizes three concepts to define the structures people need to perform basic tasks in tabletop environments. These concepts are: *workspace access*, *surface partitioning*, and *coordination policies* (see Fig. 3.1). Workspace access is the set of structures users need to initiate and move through a task flow. Surface partitioning is the act of dividing a work area into spaces, each with different characteristics and purpose. A coordinating policy is an underlying mesh of mechanisms and rules that coordinate users' actions or govern the layout of work items during a workflow. Guided by these abstract concepts, designers should be able to design a variety of surface managers that support users' workflows in different technological, physical, and social interaction settings.

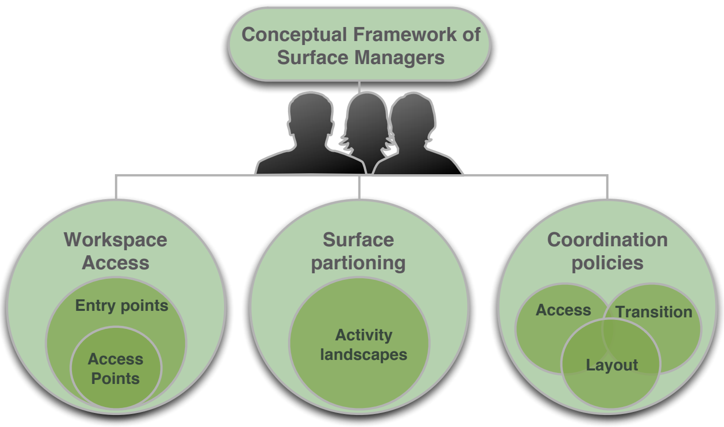


Figure 3.1: Conceptual framework of surface managers.

3.1 Context Of Work

Context of work is the set of underlying structures that support work activities in an environment

Kirsh defines the context of work as the structure of informational, conceptual and physical resources that goes beyond the superficial attributes of a work environment. It is used to describe how people come to obtain resources from an environment or a task, construct an activity representation, and coordinate their access to the resources and their interaction with the environment and people.

Kirsh analyzed office environments from a cognitive science perspective, in order to understand the ecology and key components that define the deep structures of these environments and make them portable. In his analysis, he used three key concepts that he believes to be shared among many work settings: entry points, activity land-

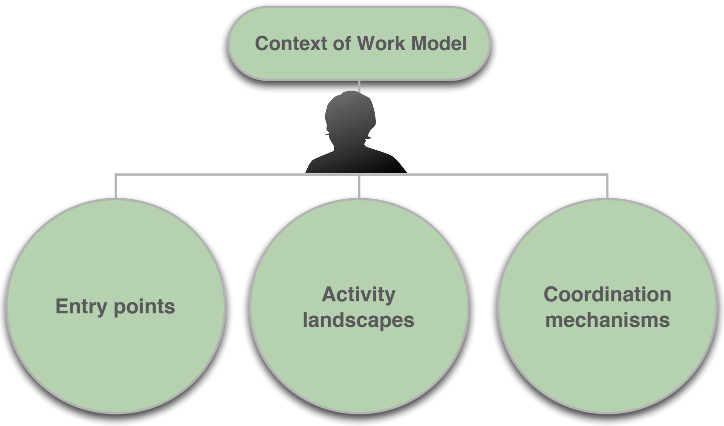


Figure 3.2: Kirsh's model of context of work.

scapes, and coordination mechanisms (see Fig. 3.2). An entry point is a structure or cue that represents an invitation to enter an information space. An activity landscape is the space users interactively construct out of the resources they find when trying to accomplish a task. A coordinating mechanism is an artefact or structure which helps a user manage the complexity of his task.

Distributed cognition theory proposes cognition and knowledge are not confined to an individual, rather, they are distributed across objects, individuals, artefacts, and tools in the environment [Hutchins and Lintern, 1995]. Kirsh proposes that distributed cognition can help achieve the "ultimate goal" of ubiquitous computing:

> "The ultimate goal of ubiquitous and context aware computing will not be achieved until we have a theory of the interaction of these elements [that make up the context of work], and more particularly, an account of how we humans are dynamically embedded in this contextual nexus. The theory of distributed cognition has a special role to play in understanding this relationship" (p. 306).

Distributed cognition theory to bridge research in CSCW and HCI

In HCI research, Norman [1986] advocated a cognitive engineering approach, "knowledge in the head, knowledge in the world", that is premised upon this view of cognition as distributed between user and artefact. Nardi [1995] discussed distributed cognition as one component of a theory to bridge research in CSCW and HCI. Wright et al. [2000] suggest that this theory serves to soften the boundary between the user and system, and brings into focus the design question of the information requirements for interaction. *What information is required in order to carry out some task and where should it be located?* A question that we are keen on answering in this framework.

Our conceptual framework builds on Kirsh's model in three ways:

1. It attempts to frame the ideas from distributed cognition research in a way that is more usable by HCI designers.
2. It adapts the abstract concepts of the model from traditional office desks and environments to interactive tabletop environments.
3. It expands the previous model, which is single-user oriented, to account for multiple co-located user settings.

3.2 Workspace Access

Entry and access points are the first levels of interaction in tabletop environments

Many attributes of an environment contribute in shaping the intention of a user and consequently his actions towards that environment [Norman, 2002]. The tabletop designer's first concern should be to construct the tabletop environment in a way that mediates the correct sequence of actions. The concept of workspace access provides designers with the basic information and structures needed to initiate an interaction with users. In this section, we describe these structures at two levels: an entry level to invite users, formulate intention, afford interaction, and mediate meta data in an information space; and an access level to enable interaction and execute a sequence of actions. We investigate the properties and roles of these levels through the design of *entry points* and *access points*, respectively.

3.2.1 Entry Points

Entry points are related to Gibson's notion of affordance

Entry points are cues and mechanisms that provide visibility, relevance, and overview of a space, and advance information about it [Hornecker et al., 2007, Kirsh, 2001, Lidwell et al., 2010]. For example, a highlighted headline in a newspaper is an entry point to the corresponding article. In shared interfaces, entry points can contribute to the work context in terms of providing the users with a continuous perception of the state of digital and physical resources [Rogers et al., 2009].

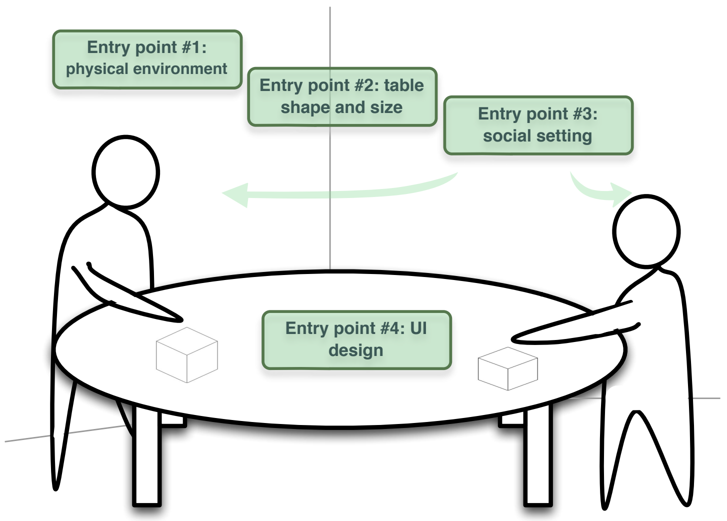

Entry point #1: physical environment
Entry point #2: table shape and size
Entry point #3: social setting
Entry point #4: UI design

Figure 3.3: Types of entry points in tabletop environments.

In interactive tabletop environments, entry points can be environmental, physical, social, or digital structures (see Fig. 3.3). The physical environment in the vicinity of a tabletop is the first entry point the user encounters. The second point the user faces is the horizontal, relatively large and familiar surface of the table. Rogers et al. [2009] found that the ergonomics of a table, i.e., size and shape, can play an important role in luring people to approach that table. Other users who are already at the table can either have what [Hornecker et al., 2007] describe as the *honeypot effect* or discourage further approach. As the user finds space around the table or by merely observing others interact and experience the table's interface, whatever design decisions the designer had made shape the primary entry points to the tabletop experience.

In public contexts the interaction principles of the display should be communicated implicitly to bystanders

Brignull and Rogers [2003] found two personal thresholds that users have to overcome before they can start interacting with a display in a public setting. First, they have to drive away from any other activity they are engaged with. Second, they have to wait for their turn and be willing to use the display in the presence of others. As an implication, the authors suggested positioning the display along the thoroughfares of traffic and improving the ways in which the interaction principles of the display are communicated implicitly to bystanders.

Lidwell et al. [2010] describe entry points as one of the universal principles of design, and list three key features for them: minimal barriers, points of prospect, and progressive lures. In the rest of this section we present these features and demonstrate how entry points can be designed in tabletop environments to: (a) encourage the user to enter a workspace, (b) guide his behavior, and (c) inhibit undesired actions.

Minimal barriers means allowing the user to get to and move between entry points with minimum interference. Placing a tabletop in an obscured location introduces a barrier of getting to the table. An entry point that vaguely communicates its purpose can prevent the user from moving to subsequent points. Other forms of barriers can be explicitly designed to prevent harmful actions. For example, in a classroom setting, the tabletop designer can make some entry points harder to reach for the young students and easier for the teacher by placing them away from the edges of the surface. Visibility, accessibility, and aesthetics are some of the properties a designer can manipulate to include or exclude barriers.

Points of prospect means entry points must provide the user with enough time and space to review his options and understand the context. Visible and meaningful layouts of entry points are one way to bring context to the user. For example, the flow, typeface and size of newspaper's headlines provide the observer with "information scent" necessary to obtain a high-level conception of the content, and a rough plan to navigate through this information landscape [Kirsh, 2001]. Entry points should give users enough time to capture a meaning. While animation can draw people's attention, using flashing animations, for instance, can require additional time to capture the meaning of a point—time that people might not be willing to give.

Progressive lures means entry points should be designed incrementally to guide the user to enter and move through the space. In social settings, [Brignull et al., 2004] suggest that a UI should start with simple and low commitment activities in order to encourage users to interact with the interface. The response of an interface to the user's first interaction should be simple, clean and to an extent familiar, in order to seduce the user to continue trying the system. Given the limited text input on direct-touch surfaces, a repulsive interface could be one that forces the user to enter his personal information in order to initiate the system. On interactive tabletops, the designer can offer a diverse set of incremental entry points to enable users to engage at different levels of interaction, gradually allowing mechanics of the system to disappear, leaving the user with a sense of familiar and natural interaction with the content. Entry points can also be designed to facilitate learning complex interfaces.

3.2.2 Access Points

An access point enables the user to access a resource or interact with an activity

Access points are structures that users perceive and interact with to gain access to an interface or a resource. For example, a desktop icon is an access structure that the user interacts with to launch an application or open a file. A variety of access points can be distributed over an interface, each point can lead to an information space that contains yet another set of access points. A UI cannot be considered interactive if it does not provide access structures with which users can convey their intentions to the system.

Hornecker et al. [2007] distinguish between entry points and access points in that the former denote design characteristics that invite people to an activity, while the latter enable the user to actually interact and join that activity.

A main attribute of large multi-touch surfaces is the appropriation of multiple inputs that can support multiple users interacting simultaneously. One way of increasing access on these surfaces is by providing multiple interaction objects that distribute control, or through spatial distribution of access points and areas [Hornecker et al., 2007].

Modeling the design of access points to investigate design alternatives and effectiveness

In order to investigate possible access point design alternatives, and to understand the effect each design decision can have on tabletop interaction, we follow Card et al.'s [1990] approach and describe how to model the space of access point designs.

To model the design of access points, we use two key ideas:

- The presentation of access structures, and
- The distribution of access points.

The presentation of an access structure describes the properties users perceive and interact with to gain access to an interface. The distribution of access points models the relation between the location and number of access structures that lead to a common information space.

Presentation

Presentation refers to the perceived look and feel of these points

Going back to the desktop icon example, if the user has the goal of launching an application from the desktop, a typical scenario: the user looks for the corresponding desktop icon, perceive its click to open affordance, and use an input device, such as a mouse, to activate it and launch the application.

In tabletop systems, we use three attributes to describe the presentation of an access point: visibility, activation modality, and spatial properties (see Fig. 3.4). Visibility indicates the initial visual state of an access structure, i.e., visible or invisible. Activation modality is the mediating tool or mechanism that the user uses to activate an access structure. Spatial properties are the perceived cues of a structure (e.g., size, shape, affordance).

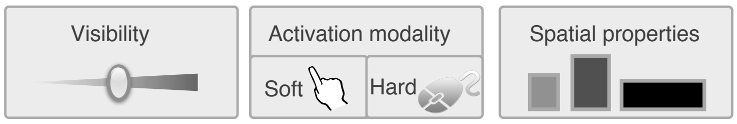


Figure 3.4: Access points presentation attributes: visibility, activation modality, and spatial properties.

In most tabletop projects, two classes of visible structures have been used to access system resources: graphical elements (e.g., menus and buttons) and tangibles (e.g., custom-designed objects [Lepreux et al., 2012], or ad hoc devices such as laptops [Rekimoto and Saitoh, 1999]). Invisible structures such as spatial locations have been utilized in systems such as Microsoft Surface. Access points can be designed to transform from one initial visibility state to another via incentives, such as user actions (e.g., the user taps on the surface or issues a voice command to receive a visible access structure), timeouts (e.g., animated structures), context-aware mechanisms (e.g., an access menu appears on the surface when it detects users in the vicinity), or mounting a head device in order to see access structures in virtual reality.

Input modalities can themselves serve as access structures

Once recognized by the user, an access point can be activated or accessed with a number of input devices and modalities. These include soft modalities (e.g., direct-touch, complex gestures, voice, and eye-gaze [Holman, 2007]), and hard modalities (e.g., tangibles, styli, gyroscopic mice [MacKenzie and Jusoh, 2001], and laser pointers [Myers et al., 2002]). The input modalities can themselves serve as access structures instead of only being mediators or instruments. For example, [Lepreux et al., 2012] used tangible objects to store the user's current state on a table, afterwards the user was able to access the stored data by merely placing the tangible on any table surface.

An access structure can have several static and dynamic spatial properties to mediate its current state, usage affordance (e.g., multi-user interaction), and relevance.

Distribution

Distribution affects the number of possible co-located users, seating arrangements, and users' mobility

We describe the distribution model of a system of access points using three measures (see Fig. 3.5): the number of access structure sets within the system, i.e., single set, fixed number of sets, or unlimited number; and their locations relative to each other and to the table, i.e., whether the access structures are initially distributed at fixed or arbitrary locations, and whether they are positioned on- or beyond-the surface.

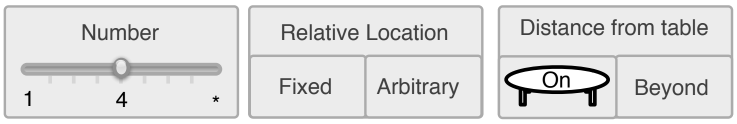


Figure 3.5: Access points distribution dimensions: number of access points, relative location, distance from table.

Morris et al. [2006a] cite two access distribution models on tabletops: centralized, a single set of access points is fixed and shared by all users, and replicated, a set of access points is replicated around the borders of the shared surface (see Fig. 3.6). We refer to the control pattern in both these models as singleton—while a user uses an access structure in this control patten, he obtains a system resource that other co-located users cannot obtain simultaneously, and the system response to the user's action can hinder or interfere with other users' work. For example, Microsoft Surface provides a replicated distribution model with singleton control pattern in the form of four access structures located at the corners of the table.

On the other end of the distribution spectrum is the distributed model, an arbitrary number of access point sets

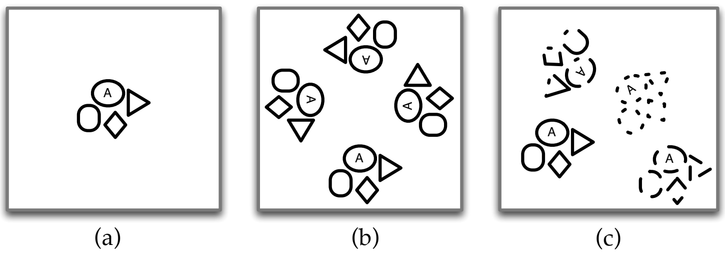


Figure 3.6: Access points distribution models.

located at arbitrary spatial locations. The control pattern in this model can be singleton, or independent—each user can obtain any system resource regardless of other users. In an independent control pattern, it is important to design the response of the system to be "local" to the user's workspace, rather than affecting the entire table surface.

Access structures can be distributed and accessed on-surface, such as graphical elements, spatial locations, and tangibles, or beyond-surface such as in-air gestures and ad hoc devices (e.g., smart-phones and laptops).

Access points need not be mounted on the table

The Design Space for Access Points

In Fig. 3.7, we graph a visualization of the design space of tabletop access points to show the possible presentation and distribution design combinations. For example, Surface Manager appears as a vertical line in the design space. In our system, we tired to combine the benefits of visible structures and invisible structures by allowing the access structures to be hid and shown on user request. Another example from the space is the MemTable [Hunter, 2009]. In this project, the author designed a single, animated "Start menu" graphical presentation that can move around the table and closer to the active users, thus, minimizing the object reachability problem of a typical centralized distribution model.

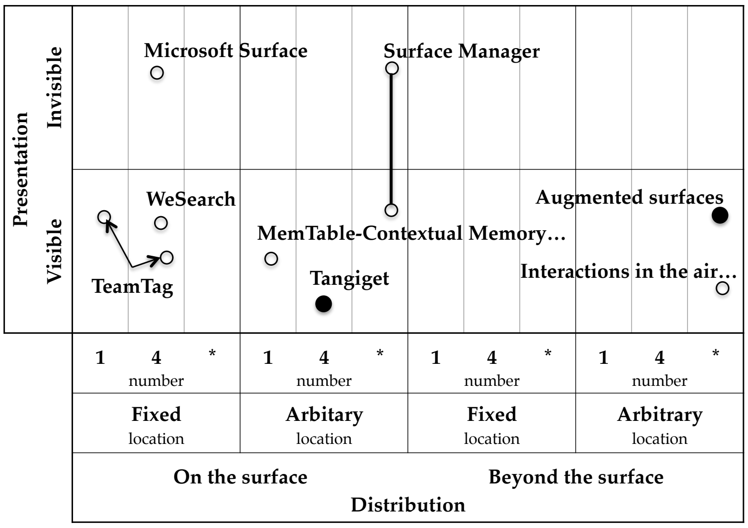


Figure 3.7: The design space of tabletop access points. A circler indicates a tabletop system. Hollow circles indicate that the corresponding system uses soft activation modalities, and filled circles indicate systems with hard modalities. A line represents a single tabletop system with access point structures that have varying dimensions. The numbers from 1 - 4 - * measure the number of access point sets within a tabletop system, single - fixed - unlimited, respectively.

Design Space Analysis

Up to this point, we have described how to model the design of access points, and presented a visualization of this design space. We now attempt to analyze the effectiveness of several design alternatives in different tabletop contexts, in order to help designers make informed design decisions.

An access system should be designed to minimize or avoid the effects of these constraints

In our analysis, we use two sets of figures of merit:

- From tabletop design constraints: reachability, orientation, and direct-touch.
- From Card et al.'s [1990] input device effectiveness: desk footprint, time to grasp a device, and cost.

Visible structures are discoverable but expensive to obtain and maintain

Visible structures have the advantage of being easily discoverable. However, on the table surface, visible structures can: (a) obscure or be obscured by other objects, (b) be oriented against the user's position, and (c) contribute to the surface clutter [Kruger et al., 2003]. Tangible visible structures add the cost risk of getting lost or stolen. In contrast, invisible structures have zero footprint on the table surface but are harder to discover.

Hard modalities and more reliable than soft modalities but less flexible

An access system that is designed with soft activation modalities can be less expensive to scale than with hard modalities. Soft activation modalities have three benefits: (1) they integrate seamlessly and maintain the desired invisibility in a ubiquitous computing system, (2) they reduce the time to grasp an input device and avoid the overhead of switching between different input devices, for example, switching to use a laser pointer to reach distant objects [Myers et al., 2002], (3) soft modalities can provide a large set of interaction vocabulary (e.g., via complex gestures or voice commands). However, compared to hard modalities, soft modalities can be harder to discover (e.g., complex gestures), less reliable (e.g., voice commands), and imprecise (e.g., direct-touch).

Many of the direct-touch issues can be addressed by manipulating a structure's spatial properties (e.g., by increasing the size of the structure or surrounding it with an iceberg area).

Different structure presentations can be utilized depending on the context-of-use

Isenberg et al. [2009] discussed contextual, technological, perceptual and collaborative challenges arising when designing tabletop systems for information exploration in two different contexts: workplace settings where domain experts gather to explore and analyze often large and complex datasets, and public spaces where the design has to support a much more diverse set of people, tasks, and goals. They found that several issues are common in both settings, other challenges are unique to workplace environments or public spaces and need to be addressed accordingly. For example, in public contexts people mostly would not have any experience of tabletops and may not be willing to put the time to explore them, thus, visible structures activated via simple direct-touch or tangibles are preferred in these settings. However, with tangibles, the number of allowed simultaneous participants is predetermined. In semi-public contexts such as work environments, [Isenberg et al., 2009] cite that work teams are often prepared to invest time in learning the tabletop interface, thus, employing invisible or ad hock structures in these environments can be more advantageous.

In a comparison between replicated model and centralized model Morris et al. [2006a] found that users preferred the replicated model for two main reasons: (1) the desire to use the center of the table for other semantically important tasks, and (2) to avoid accidentally touching a teammate's hand when using the shared controls.

A centralized model has the advantage of providing better workspace awareness among users where each can see the others interacting with the same access point set [Zanella and Greenberg, 2000]. However, this model can cause accessibility and single-orientation issues. In addition, [Scott et al., 2003, Stanton and Neale, 2003] report that this model can allow the stronger users to dominate, and can raise coordinating concerns, such as invading other users' personal spaces by reaching out to a centralized location. The replicated model can overcome the reach and orientation problems until the table becomes very large and the number of access point sets limited. Both models, centralized and replicated, share the disadvantage of limiting the overall flexibility of the system by providing a singleton control pattern. In this sense, the distributed model is more flexible, but its evaluation also depends on the reachability of the spatial locations used and the number of access point sets provided.

Access systems that are designed to operate beyond-surface, can be more flexible in supporting users' mobility, can be utilized as a form of privacy, and are largely scalable. In addition, the structures of such systems, whether visible or invisible, have zero footprint on the table surface. Access systems that operate directly on-surface can offer a more engaging user experience and facilitate better human-to-human interaction.

In summary, a tabletop interface designer can combine several access point presentations and distributions to control the number of concurrent users on the surface at any time, and the amount and type of accessible resources. Access points can be designed to impose access privileges to affect the power structure of an interaction (e.g. in a classroom two levels of privilege can be provided for the teacher and the students). In addition, access points help increase workspace awareness and can be used to define different roles and ownership [Hornecker, 2005]. Access points are mediators and not content, and so they should have a limited footprint on the surface. For example, by extending the number of manipulative access points by facilitating the sharing of digital content across ad hoc devices [Rekimoto and Saitoh, 1999]. A designer should also be aware of the effect various access point designs can have in different social interaction settings. For instance, in a parallel interaction setting, a centralized distribution model may raise intolerable privacy issues as opposed to its usage in a collaborative setting.

The choice of access point design is determined by the task and context requirements

3.3 Surface Partitioning

Surface partitioning concept helps the designer understand how people divide a shared surface and construct workspaces, in order to facilitate and maintain these spaces in simple and straightforward ways. Once users gain access to system resources they will start performing sequences of actions on these resources to achieve their goals. Several investigations of tabletop work practices have observed that users partition the surface into three different territories when performing activities to acquire resources and mediate group interactions: personal, group, and storage territories [Scott et al., 2004]. Within these territories, users construct what Kirsh describes as activity landscapes. These landscapes are composed of collections of concepts, the layout of artefacts, users actions and consequences, and constraints imposed by a task or environment.

Each tabletop territory can be perceived as a space that needs workspace access, space partitioning, and coordination policies structures

The establishment of territories depends on the user's acquired space and is part of the manager's partitioning and coordination process. On a tabletop surface, each territory can be perceived as a workspace with a set of entry and access points, activity landscapes, and coordination policies. Territories can be defined implicitly by relying on social practices. For instance, associating a space with a user's work habits, i.e., period of usage and frequency of access [Edney, 1976], or by merely depending on the orientation of tabletop artefacts to distinguish separate regions on the surface [Tang, 1991]. However, while implicit territories seem natural and intuitive, researchers observed that users attach different characteristics to different types of territories [Scott and Carpendale, 2006]. To support a variety of tasks and interaction styles, the designer should understand the fundamental concept of territories and design the UI to augment their roles in tabletop environments.

From tabletop territoriality research, we synthesized four aspects a designer should investigate when partitioning territories, the territory's definition, properties, functionality, and policies. In Table 3.1, we present these aspects, the related questions a designer should attempt to answer, and a suggested set of investigation elements.

On personal desks the surface is divided to a personal territory and several private territories

While defining a territory, a designer can depend on the ergonomics of the table and use the edges, for example, as references to personal territories. However, fixed and predefined territories have been found to be limiting to the user's mobility [Scott et al., 2004], and can impose constraints on the number and arrangement of co-located users. Taking a user-oriented approach instead, requires the tabletop system to have user-identification mechanisms or behavioral models to identify the user and adapt her personal territory to her location and orientation. Alternatively, personal territories can reside on ad hoc devices such as laptops (e.g., [Rekimoto and Saitoh, 1999, Shen et al., 2003]).

Group territories usually start from the center of the surface and expand to the areas unoccupied by personal or storage spaces [Scott and Carpendale, 2006]. When deciding on the properties of each territory, the designer should consider if the interface should be scalable or portable. Scott and Carpendale [2006] noticed that users casually pile documents in their storage territories and suggested providing these spaces with facilitates to organize and search the stored documents.

Territory Aspect	Designer questions	Investigation elements
Definition	What kinds of territories should be supported?	Personal, storage, group.
	Which approach should be adopted in dividing and maintaining territories?	User-oriented or table-oriented.
Properties	What spatial properties does each territory have?	Size, shape, visual presentation, etc.
	Should the properties be dynamic or fixed?	Yes/No
	What affects the designer's choice of properties?	Table ergonomics, task activities, group size, seating arrangements, etc.
Functionality	What functionality should each territory provide?	Organization, searching, personalization, etc.
Policies	What policies should each territory implement?	Access, transition, and layout policies.

Table 3.1: Aspects of surface partitioning.

Each territory type can require different coordination policies. The selection of these policies is influenced by the tabletop hardware, and desired interaction styles among other factors. For example, the DiamondTouch table [Dietz and Leigh, 2001] is capable of assigning touch points to specific users, thereby enabling simultaneous access and avoiding conflicts. When interacting on the table, the users shouldn't need to change their work practices between different types of territories. Thus, the designer should be cautious in maintaining a consistent look and feel among the territories while augmenting and supporting the purpose each territory serves during an interaction.

3.4 Coordination Policies

Coordination policies are agents that facilitate manipulating objects and coordinate the interaction of single and multiple users. In offices, Kirsh describes the clock as one mechanism to facilitate time coordination between people. On traditional tables, social policies, i.e., standards of polite behavior, are used to coordinate people's interaction. However, in interactive tabletops, research shows that additional coordination policies for direct manipulation should be provided to coordinate access and solve conflicts [Morris et al., 2004b, Yuill and Rogers, 2012].

Coordination policies should be flexible

Morris et al. [2004b] suggest that coordination policies should be implemented to provide a more structured and predictable interaction while still being more flexible than rigid access permissions. They proposed several global coordination mechanisms, among these are: (a) making all elements accessible to everyone and relying on social protocols, (b) allowing users to lock objects so that others cannot access them, (c) giving each user a ranking, and allowing higher ranking users to take objects from lower ranking users, and d) allowing physical measurements, such as speed and force applied by the user, to determine who is able to retrieve a contested object. Other techniques were proposed to improve handoffs, where one user transfers an object to another [Jun et al., 2008], as well as to improve awareness of other users' actions and locations [Pinelle et al., 2008].

Based on this, we identify three kinds of policies that should be supported on interactive surfaces (see Fig. 3.8):

- Layout policies to help increase workspace visibility, organize activity landscapes, exploit spacial cognition, and facilitate artefacts organizing.
- Access policies to enable flexible access to table resources and workspaces, as well as to implement privacy settings.
- Transition policies to achieve fluid transitions between individual work and active collaboration, and

to facilitate content sharing.

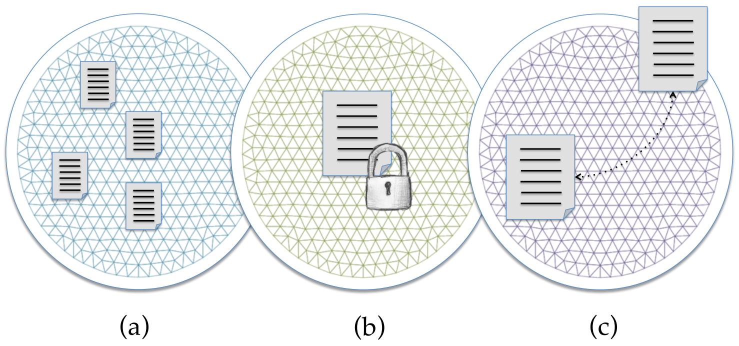


Figure 3.8: Coordination policies on interactive tabletops. (a) layout policies, (b) access policies, and (c) transition policies.

Coordination policies can be designed on different layers and activated or configured by the users

These polices can vary between different user roles and territories. For example, access policies in personal territories should be more strict than in a group territory. Kirsh noticed two patterns of structured office desks, the neat—tidy desks where limited number of work items are visible and located in a standardized place, and the scruff—messy office that can hold large amounts of information. In tabletops, both patterns should be supported, for example, by relaxing the layout policies inside the territories. The designer's choice of policies depends largely on the type of social interaction she wants to support.

3.5 Summary

The framework helps designers assess design alternatives for tabletop structures

This chapter presented three concepts to help define the structures people need to perform basic tasks in tabletop environments, namely, workspace access, surface partitioning, and coordination policies. The framework builds on observed human behavior around tables as captured in tabletop investigations and described in workplace theories. To help designer assess different design alternatives, we demonstrated, in detail, the assessment of tabletops' access points. We described a modeling technique for access point design based on the presentation and distribution of these structures, and produced the corresponding design space. We then used the space to evaluate design alternatives in the light of empirical tabletop literature, as well as effectiveness figures of merit from Card et al. [1990].

Before applying the framework to the design process of tabletop UI, designers should define the system's context-of-use. The context includes the physical, technological, and social settings of the digital table, as well as target users, supported interaction styles, and task domain.

The described framework recognizes the role of tables in physical contexts and does not attempt to design holistic work environments for interactive surfaces. Instead, it focuses on observed tabletop practices and provides structural guidance to support these practices and enhance the overall user experience. The proposed framework still requires long-term and observational studies to understand the deep effect the table's form factor and contexts have on the workflows of co-located users. With the use of surface manager software and considering the current state of commercial tabletops, we believe that this data will become available in the near future.

Chapter 4

System Design

"Any intelligent fool can make things bigger, more complex, and more violent. It takes a touch of genius, and a lot of courage, to move in the opposite direction."

—Albert Einstein

This chapter outlines the concepts, design, and implementation of our version of a surface manager, a.k.a Surface Manager. An essential requirement for software development on interactive tabletops is to take advantage of tabletops' unique features and design space, and attempt to at least one of the following goals:

- Augment the utility of the traditional table with computational power in order to digitize, access, and visualize information in new forms and contexts.
- Enhance the user experience with new styles of engaging and enjoyable interaction.
- Provide new products and services that are not possible on other computing devices.

The design of Surface Manager attempts to achieve the first two goals, which we believe will consequently enable the third.

4.1 Design Goals

A nonintrusive UI makes the mix between digital and physical objects more natural

The overall goal of Surface Manager is to support and facilitate users' concurrent work processes on interactive surfaces, in various social interaction settings. Our aim is to augment the current role of traditional tables in semi-public contexts with a general purpose, nonintrusive, "calm" UI that enables flexible resource access and manipulation. We envision Surface Manager to enable a similar scenario to the one described by Streitz et al. [1999] in the i-LAND project:

> "[Imagine] meeting a colleague by chance in the hallway and starting a discussion that might result in the intention to explain something by drawing a sketch on the wall and annotate it by drawing. Besides the fact that this is usually not accepted in office buildings, traditional walls do not support storing and later modifying the elements of the discussion. It is also not possible to search for related information in a background information base and to link this information to the sketch and the scribbles on the wall. In the future, we like to be able to turn to the wall and do just this. Think of the wall as an "interactive wall" or as one being "covered" by a high resolution electronic wallpaper providing the functionality needed" (p. 121).

To achieve our overall goal, Surface Manager is designed with the following design goals:

1. Equal Access Privileges. The size, shape, and horizontal tilt of tabletop surfaces afford approaching them from any angle, and by multiple users. Surface Manager UI should be designed to provide users with equal access privileges to the tabletop system from any location on the surface. This is a vital feature for large tabletops in particular, and for interactive systems that do not provide static user interfaces in general.

2. Local Control Distribution. Surface Manager should provide each tabletop user with full control over his or her work items. The scope of one user's control should be localized to the user's workspace and must not affect or interrupt the work of others.

3. Window Management Facilities. As a general purpose UI, Surface Manager is required to provide: (a) facilities to launch and interact with concurrent applications, (b) policies to coordinate the layout of, and access to interface and work items, and (c) techniques to interact with and manipulate user content.

4. Handle Design Constraints. The focus of Surface Manager is mainly on the constraints that can interrupt the workflow of users, such as surface clutter and object reachability.

5. Calm Computing Environment. Surface Manager interface should inherit from tabletops' ubiquity. The interface should integrate with the traditional table role without being intrusive, loud, or demanding.

We argue that the minimum requirement for any surface manager, regardless of the context-of-use and target users, is to handle tabletop design constraints.

4.2 Applying the Conceptual Framework

The conceptual framework in chapter 3 describes the tangible and conceptual structures needed to support users' work in tabletop environments. In this section, we look at ways designers can apply the knowledge of the framework to the design of surface manager UIs. We demonstrate the descriptive power of the framework by showing how we applied the framework, and give examples of how a designer can use the framework: to think about the suitable presentation and distribution of structures and widgets to provide workspace access in a specific context, to partition the surface to work areas of different purposes, and to decide on the types of policies that serve the desired social interaction settings.

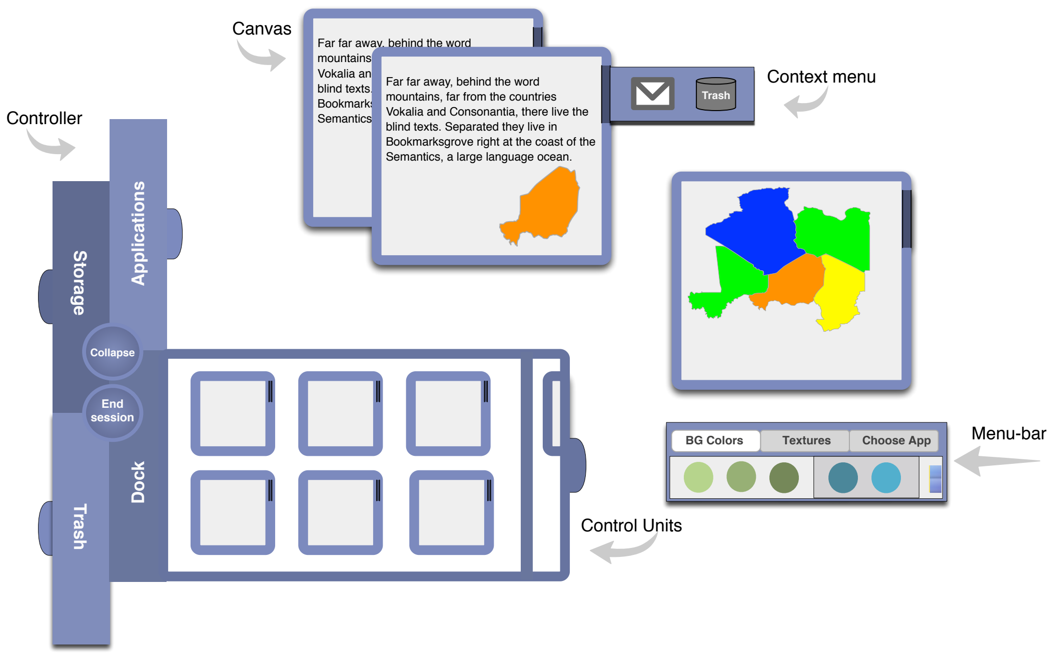

Canvas
Controller
Context menu
Trash
Applications
Storage
Collapse
End session
Dock
Trash
BG Colors
Textures
Choose App
Menu-bar
Control Units
Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts. Separated they live in Bookmarksgrove right at the coast of the Semantics, a large language ocean.

Figure 4.1: Surface Manager UI overview.

Fig. 4.1 shows an overview of Surface Manager's UI elements to provide a context for subsequent explanations of software features in this section.

Workspace access

Our requirements for the interface in Surface Manager are: (a) to be scalable to any table size, shape, and number of users, (b) to be portable across several tabletop technologies, (c) enable flexible seating arrangements , and (d) to be discoverable. A discoverable interface in this context refers to how easily users recognize how to access and interact with the interface by robust exploration or by relying on the affordance and layout of the design.

Informed by the design space of access points (section 3.2.2), the presentation of Surface Manager's access points is: an on-surface, hidden access structure that is activated via a soft modality. The access points are arranged in a distributed access model with independent control pattern.

The hidden presentation is a graphical widget, a controller, which is initially invisible. Using a soft activation modality, a simple access gesture, on any location on the table, a controller can be invoked to become visible and provide the user access to system resources (e.g., applications and files). This workspace access design allows us to avoid static UI layouts, thus, adhering to our first design goal, equal access privileges, and minimizing object reachability problem.

Additionally, the system applies the design principle of entry points by making use of familiar visualizations, avoiding flash animations, time-outs, and moded interaction, in order to provide the user with a continuous and consistent system state.

Surface Partitioning

To support multiple social interaction settings, Surface Manager supports the three main types of territories: personal, storage, and group territories. We followed a user-oriented approach to partition these territories—when the user performs the access gesture at any location on the surface, a virtual personal territory is created for her at the location and orientation of the gesture. A storage territory is available for each user from the controller widget. The group territory coves the entire unoccupied space on the surface.

The spatial properties for the personal territory are not visible, i.e., no visible boundaries to show the size or shape of the territory. We did not want visible boundaries to divide the table or add chrome to the interface in away that can distract users or over ”technify” the table, which contradicts with our fifth design goal, calm computing environment. The size and location of a personal territory are dynamic and adapt to the distribution of the user's work items. A virtual personal territory is thus, independent of the table ergonomics. Storage territories, called control units, are also mobile but have a fixed shape and size. Control units provide facilities to store, organize, and navigate stored items.

Coordination Policies

We chose a relaxed set of access and transition policies, namely, making all elements accessible to everyone and relying on social protocols [Morris et al., 2004b]. The layout policies are implemented system-wide to manage the tree of work item widgets on the surface. The user can drag, rotate and scale items freely. When an item is dragged by the user it is located at the point of release, and on top of the surface tree, thus, covering all items lying under it. Item cans be manipulated while obscured. To rearrange an occluded item to the top of the tree, the user must attempt to drag it.

Other policies are implemented to facilitate accessing and controlling distant work items (e.g., *collapse&expand*, and *remote control* interaction techniques), and to enable flexible layouts (e.g., *in-place* interaction technique). To direct input streams from input devices, such as keyboards, to the correct output devices, we designed a simultaneous peripheral associate mechanism.

Many of our design decisions were constrained by the lack of any user-identification mechanism on the tabletops in our possession.

4.3 Controller Metaphor

Large, shared tabletops, unlike desktop computers and smartphones, can have co-located users working simultaneously, with new users coming and others leaving, at any time. Thus, the notions of accessing the system and exiting the system are complemented and replaced, respectively, with the notions of staring and ending a table session on tabletops.

Surface Manager implements, what we refer to as the controller metaphor (see Fig. 4.2). In this simple metaphor, the UI of the system is compacted in a single widget, the controller, and duplicated and distributed among table-

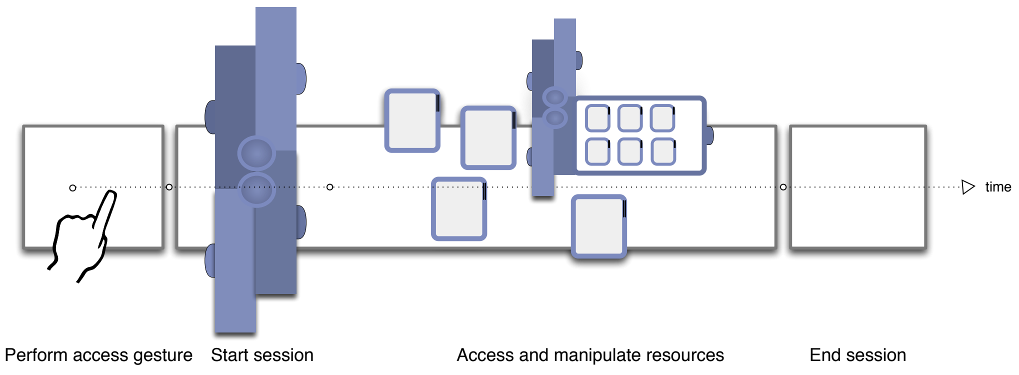


Figure 4.2: Controller metaphor.

top users. Staring a table session means obtaining a controller. Ending a session means destroying the controller and clearing any related work items. An essential part of the metaphor is the UI mobility—a user can move her workspace, controller and work items, around the table while maintaining their relative spacial arrangement. The metaphor is composed of a set of core UI elements and interaction techniques, which we discuss further in the next section.

4.4 User Interface

Following a series of prototypes, interaction metaphors, and design iterations, we concluded to the general look & feel of Surface Manager UI. Fig. 4.3 summarizer the main building blocks of the interface.

Our design guidelines for the interface were as follows:

- Design the interface to be discoverable, predictable and consistent.
- Keep the interface simple and familiar. Minimize chrome and allow users to interact with content directly.
- Avoid deep hierarchies. Minimize the number of steps and transitions required to perform an action.

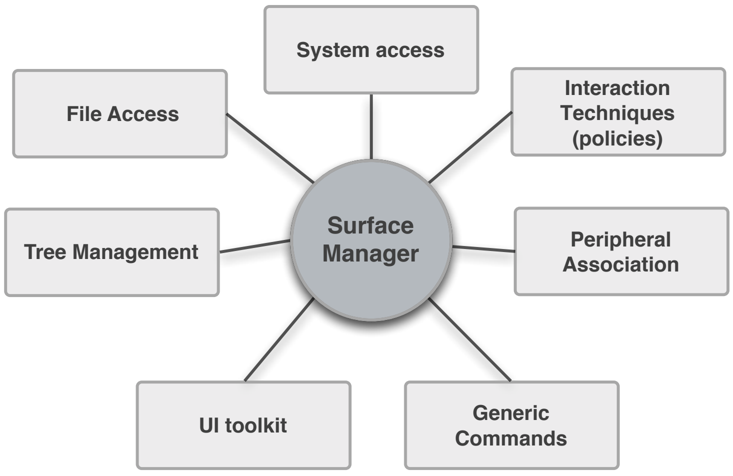


Figure 4.3: Building blocks of Surface Manager UI.

- Provide a flexible and expressive set of UI components for variant social interaction settings.
- Provide a continuous visual state of the system and resources.

4.4.1 User-Interface Toolkit

We designed a lightweight UI toolkit for Surface Manager in order to maintain a consistent interface, and to embed design solutions for some of the direct-touch issues. For example, we adjusted the size of UI elements, and surrounded them with an additional iceberg to increase the touch responsive area, and account for different contact-point's angels and sizes. In addition, we designed menu and file navigation to require horizontal scrolling rather than vertical to prevent items from being obscured by the hand of the user.

The toolkit is composed of three groups of UI elements: (a) primitive elements, such as buttons and text-fields, (b) layout containers, and (c) core elements: controllers, canvases, menu-bars, and soft-keyboards. This section describes the design and properties of the core UI elements.

Controller

The controller is technically the access point structure form which the user can access and manage all Surface Manager resources and functionality (see Fig. 4.4). Conceptually, a controller presents a table session. Each controller is bound to a single territory, and controls the orientation and layout of items lying in this territory. The controller can also function as a system notification center by directing system messages to available users.

When a controller is first generated it is assigned a specific color-code to indicate ownership. All items generated using the controller are similarly color-coded. The controller is composed of four control units that provide users with access to system resources. It also provides two territory control buttons: an end session button to move all work items of a territory off the table surface and into storage, and terminate the table session; and a collapse&expand button (details in section 4.4.6).

The controller has three main characteristics:

- Mobility. The controller can be moved on the surface to be reachable to the user from any working area. This features enables the user to move around the table and still have full access to the system and his control units.
- Orientation. The controller can be freely reoriented. Items generated from a controller are oriented to the same angle. This allows collaborators to share a single controller and orient it towards a group's view angle. Other scenarios where this feature might come useful is when an existing user needs to adjust his workspace to accommodate for a newcomer.
- On request. The controller can be hid, when it is not in use, to avoid cluttering or occluding the surface. Performing the access gestures anywhere *near* the previous controller's location will make it appear in the new location.

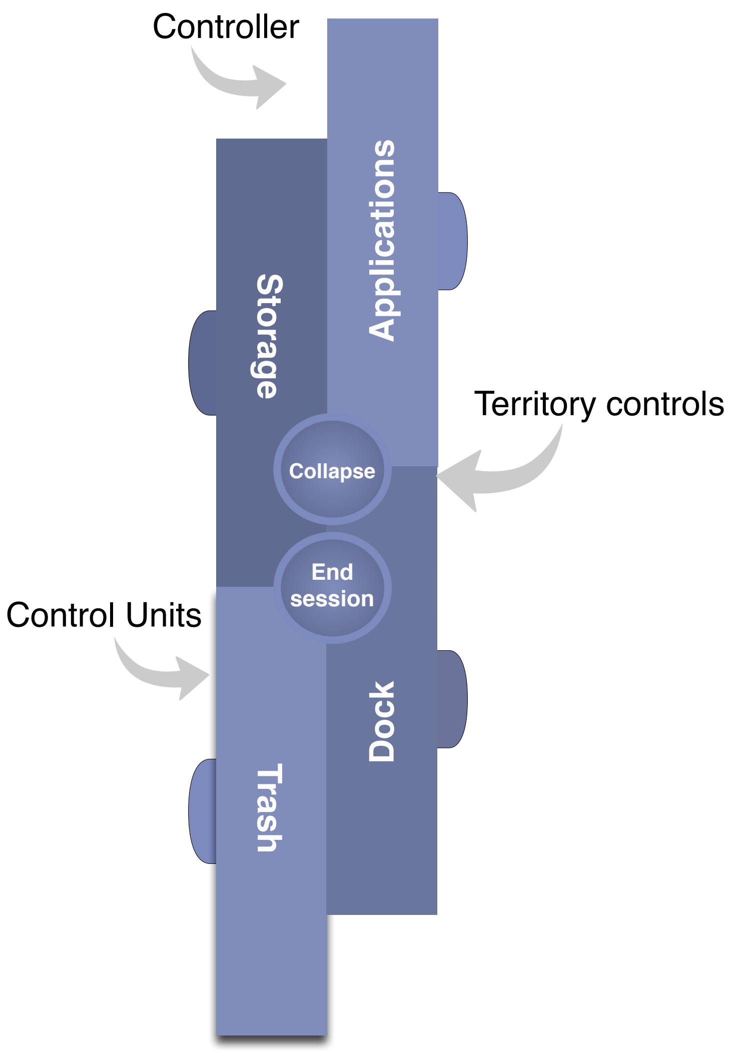


Figure 4.4: Surface Manager controller.

Control Units

A control unit is a rectangular shaped widget that provides users access to system applications and user files. The controller is composed of four control units: application, dock, trash, and storage units. Application unit displays available applications that the user can launch concurrently. The dock, storage, and trash units allow the user to add and remove digital items in them. The dock unit, like the dock in

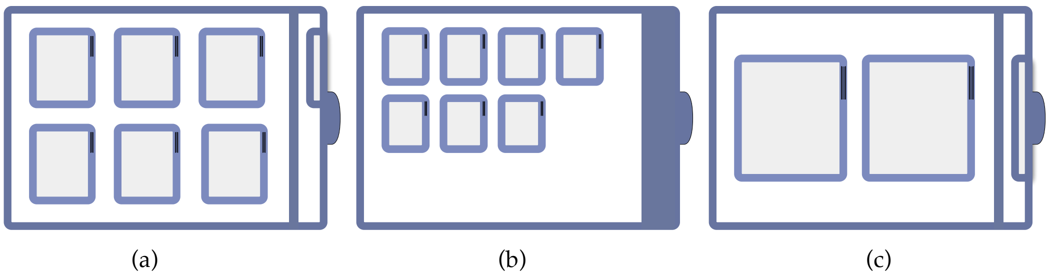


Figure 4.5: Control units. Three navigation views: (a) normal view ,(b) scan view , (c) fine view.

Apple's OS X and the task bar in Microsoft Windows, can be used to minimize the footprint of work items on the surface. Other than storage capabilities, control units provide facilities to organize, and navigate files.

Items in a control unit are arrange in a grid. When a user adds an item to a control unit, the item is scaled-down in size, and snapped to a grid position. This organization mechanism allows for maximum visibility as the content of a scaled-down item remains visible, and the spacing between gridded items prevents potential occlusion.

The units provide users with a lightweight interaction mechanism to navigate files (see Fig. 4.5). To quickly scan the files, the user performs a zoom-out gesture, the gridded items scale down and are rearranged to show more items per view. To fine-tune the search the user performs a zoom-in gesture, the items scale up in size for the user to have a better visual.

To distinguish between dragging an item over a control unit and into one, we implemented a scaling feedback. When an item is gridded, it is scaled down to a size that maintains its recognizability. When an item is dragged out a unit, it is scaled up to its original size. While dragging an item over a unit, if the user wants to grid that item, she must drag it slower around the edges of the unit. Once the unit recognizes that this item must be gridded, it notifies the user by scaling down the size of the item. If the user releases a scaled-down item, the item snaps to the unit's grid, otherwise the item stays at the release location on the table.

Canvas

A canvas is a rectangular shaped widget that acts as a place holder for user content. It is surrounded by a color-coded border that has a rectangular hot spot to open the context menu-bar (Fig. 4.6). The canvas can be moved, scaled and rotated. Once the user drags an item from a control unit and onto the table, a new canvas is created and attached to that item. Attaching a canvas to each item on the table provides interactive and visual consistency.

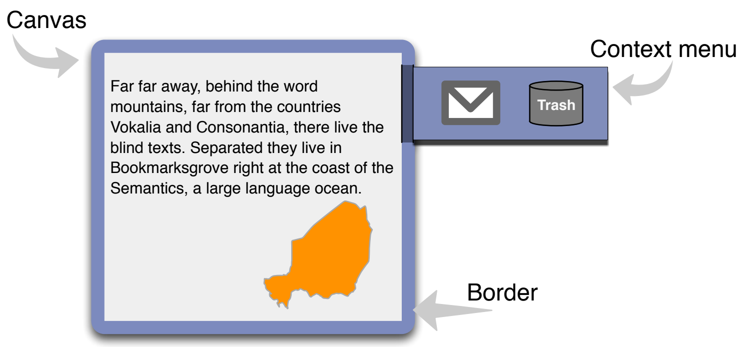

Canvas
Context menu
Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts. Separated they live in Bookmarksgrove right at the coast of the Semantics, a large language ocean.
Trash
Border

Figure 4.6: Surface Manager canvas.

Logically a canvas is made of two parts: user content (e.g., a file that contains images and text) and a set of application references (e.g., an image editor and a text editor). When a canvas is created and attached to a file, it creates a reference to the application that supports that file, and launches that application. A single canvas can have references to several applications, we discuss this feature in section 4.4.5.

The toolkit contains two floating objects: a menu-bar and a soft-keyboard. A floating object is generated and invoked by a canvas, and responds to the canvas dragging and griding. Floating objects resides on root objects' level, rather than within the canvas. The rational behind this design was to account for three problems: (1) if the canvas is too narrow, the menu-bar, for example, is either wrapped or truncated, and that can destroy the spatial memory of menu item locations or limit their accessibility; (2) if the canvas was far or slightly oriented off the user's position, the user will need to change his seating position, for example, to access the virtual keyboard; (3) rendering floating objects within the canvas limits how simulations users can access that canvas.

To manipulate the content of a canvas, the user can either perform direct transformation gestures or perform a menu invoking gesture over any element to invoke an editing menu-bar. In response, the canvas contacts the supporting application and generates a floating menu-bar, in-place.

Menu-bar

A menu-bar is composed of two parts: menu tabs, and menu items (Fig. 4.7 (a)). The menu-bar arranges the menus consecutively in a single row. The user can scroll horizontally, or use the tabs to view different menus and menu options.

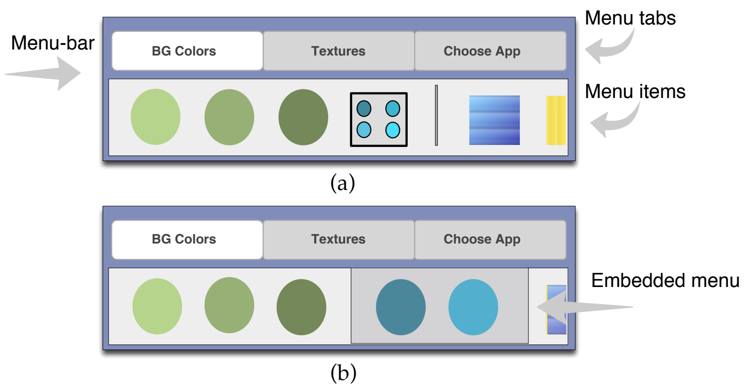


Figure 4.7: Surface Manager menu-bar. (a) single menu-bar of three menus and one embedded menu, (b) expanded embedded menu.

An embedded menu is represented as a single menu item in the parent menu. The item provides a miniature preview of the first four menu items in the embedded menu. The user can tap that item to expand the embedded menu horizontally and in-place. The embedded menu items appear at the level of the parent menu items but with a darker background shade (see Fig. 4.7 (b)). Like other floating objects, menu bars can be flicked-away and invoked when needed.

Soft-keyboard

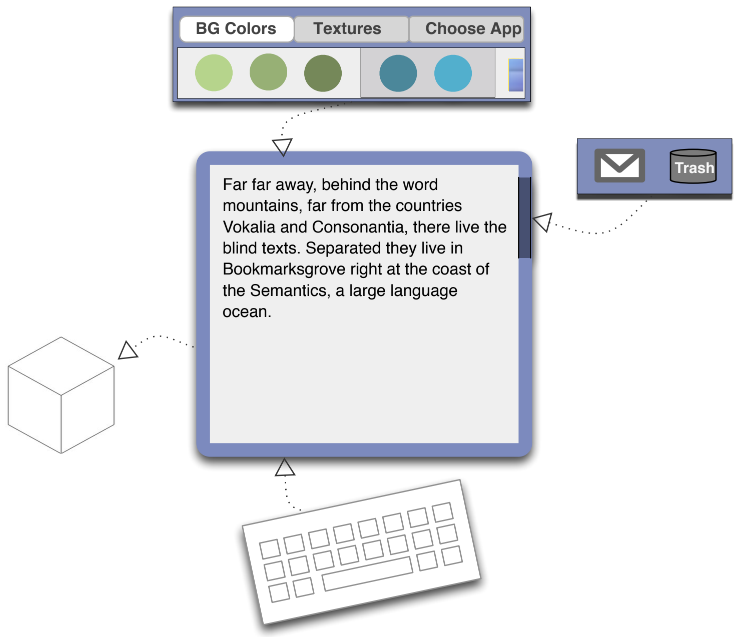
When the user taps a text field a software keyboard appears floating under that field. When finished, the user can flick-away the keyboard to hide it.

4.4.2 Generic Commands

To maintain gestural consistency as well as visual, we designed a minimum set of generic commands that apply similarly to all items in the system. These commands include: transformation gestures (e.g., move, rotate, and scale), menu invoking gesture, floating objects hiding gesture, and access gestures. Most of these gestures are from standard bi-manual gestures adopted by most contemporary multi-touch platforms. We kept the gestural commands to a minimum to reduce interference with application specific gestures. The system also allows application designers to disable generic gestures. Unlike keyboard shortcuts, gestures are not perceived as unique command units. For example, the exerted force, the handedness of a user, and the number of fingers can be used to distinguish between gestures on the system level, but not by users [Wobbrock et al., 2009]. Consequently, some gestures are further supported with visual options in the context menu.

4.4.3 Peripheral Association

Given that there could be many canvases on the surface, each of which can be controlled by a different user, there must be a mechanism for directing input to the right canvas. In our system, we implement a simultaneous peripheral association mechanism. Association is achieved between a peripheral, input or output device to pass events



Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts. Separated they live in Bookmarksgrove right at the coast of the Semantics, a large language ocean.

Figure 4.8: Simultaneous peripheral association mechanism.

between them (see Fig. 4.8). For example, an image can be a target item (output device) that is associated and manipulated with a menu-bar. Simultaneous association means that several items can be associated with peripherals at the same time. This mechanism functions based on three rules:

1. All items on the surface are in-focus, whether the item is obscured, embedded, or on the top of the surface. This means that any item in the system can be manipulated directly, without first selecting it and then issuing an action for it, even if partially obscured.
2. Each item is associated with the touches directly on its surface. Competing touches are resolved by the gesture detector (section 4.6.4).
3. Each item within a canvas can be associated to a single soft-keyboard and or menu-bar via a one-to-one relation. This means that each device, menu-bar or soft-keyboard, can only send events to a single target.

4.4.4 System Hierarchy and Tree Management

The system is built over a relatively flat hierarchy of three levels (Fig. 4.9). The *surface level* is the visible background covering the entire table area that receives users' access gestures. The *work level* is the level above the surface level on which all users' work items exist. The *content level* is a level embedded in the work level items and where system resources and user's content elements are arranged and manipulated. A flat hierarchical system provides users with visibility and transparency, potentially increasing workspace awareness. It can also minimize the number of transitions needed to access and manipulate system resources.

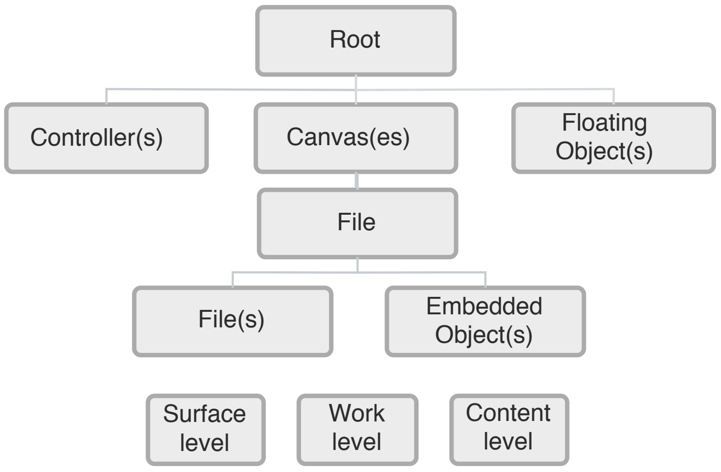


Figure 4.9: System hierarchy and work items tree.

Surface Manager manages two types of trees: surface tree and content trees. The *surface tree* is singleton and dynamic. The root of this tree is the table surface, which is the only object on the surface level. Controllers, canvases, and floating objects, which reside on the work level, are the branches of this tree, the children of the root. *Content trees*, start and branch at the content level. Content trees can be built of static UI elements arrangements, for example, the interface of an application, or dynamic user content elements, for example, by dragging and dropping files in a canvas. Every item on the table is part of a dynamic or static tree and is within a child-parent relation with another item. Layout policies control these trees.

Files and canvases can be dragged and dropped inside other canvases. We distinguish between a canvas dragged over or into another canvas by changing the border color of the dragged canvas, momentarily.

4.4.5 File Management

There are two primary types of content elements in the system. These are embedded objects and files. An embedded object is one that an application generates and has no extension outside that application (e.g., shapes in a paint application). A file is an information object with an extension or file-type (e.g., an image can have the extensions jpeg or png). The extension is important to identify the applications that support the file.

The system provides users with three different ways to access files: *speed access*, *system access*, and *application access*. All methods aim to narrow the gap between information access and information viewing by minimizing the number of transitions required to open a file or an application:

- Speed access. A user can approach a table from any location, perform a canvas invoking gesture, and a canvas is created, and appears in-place. The aim of providing speed access is to allow hasty users to skip the system access process and start generating content. This method was designed to mimic the situation of traditional tables which have papers and pencils laid out and allow any person to grab a pencil and start writing or drawing. The user can create an unlimited number of canvases. When the user is done, she can perform the corresponding gesture to save or delete her work. This method augments the utility of traditional tables by allowing users to attach applications and services to a canvas. For example, a user can create a canvas and attach a paint application to it in order to draw, or a calculator utility to calculate some numbers.

- System access. The user invokes a controller, opens the storage unit, and drags the desired files to the table for editing. This method is designed for users who want to navigate and edit exiting files.

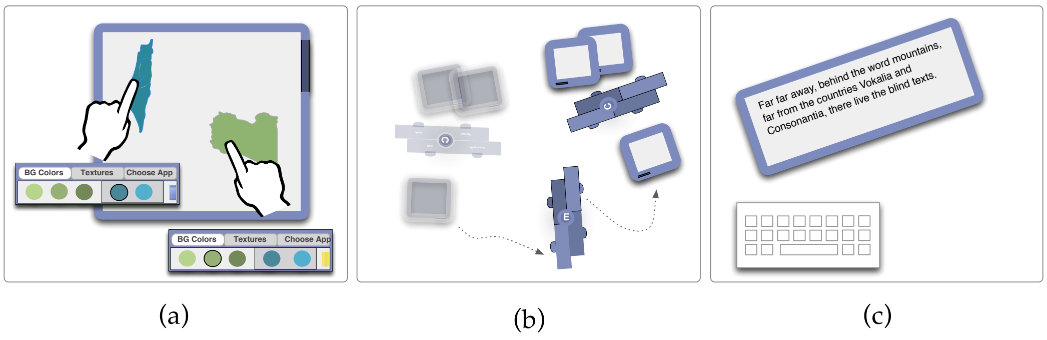
- Application access. To create new content, or interact with the interface of the application (e.g., in game applications), the user can access the application unit, drag a new instance of an application and start generating content or interacting with the interface.

Application switching. In Surface Manager, each file is associated with a single application. When the file is dragged to open on the table, a canvas is attached to it, and the canvas invokes the file's associated application to launch. This file becomes the root of the content tree in the canvas, and its application becomes the "current application". To switch from the current application, the user invokes a menu-bar on the root file. The menu-bar contains the menus of the current application, and an additional menu, labeled "Choose App", of other applications that support that file type. After the user selects an application, the canvas aggregates the menus of the new application in the file's menu-bar. The new application becomes the file's associated application.

Invoking a menu-bar on an embedded file, child of the root file, can result in one of two results: if the embedded file type is supported by the current application, the menu-bar is filled with the current application's menus, if the file type is not supported, the menu-bar shows the menus of this file's associated application.

4.4.6 Interaction Techniques

To handle many of the horizontal surface constraints and support collaborative interaction, we designed three interaction techniques: in-place, collapse&expand, and remote control interaction (see Fig. 4.10).



Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts.

Figure 4.10: Interaction techniques. (a) in-place, (b) collapse&expand, and (c) remote control.

In-place Interaction

In-place interaction means providing the desired information at the time and location of request (e.g, menu-bars and soft-keyboards). On interactive surfaces in-place interaction can have the following benefits:

- Enhance system's visibility and predictability. This becomes more important in cluttered workspaces, and when triggered items are very large that depending on their boarders to show a requested menu, for example, becomes inefficient.
- Grantee accessibility of requested information regardless how large the table surface is. It can also reduce homing time [STUART et al., 1978].
- Support collaboration. This is specially true when collaborators attempt to manipulate two different but close items. An in-place menu-bar, for example, will provide a clear associate of which menu can affect which item (Fig. 4.10 (a)).

Collapse & Expand Interaction

Collapse&expand technique is mainly used to handle object reachability problem. When the user wants to access a far item, she can tap the "Collapse" button on the controller, to force all her work items to the dock control unit. She can then drag the item in request, and tap the now "Expand" button for the system to return all the other collapsed items to their previous locations. This interaction technique also supports user's mobility and flexible seating arrangements by maintaining the relative spatial arrangement of the collapsed items. For example, collapse&expand allows the user to collapse all his items, move the controller to another location on the table, then expand the items to their original relative-to-controller spacial arrangement (Fig. 4.10 (b)).

Remote Control Interaction

Remote control is an interaction technique that allows users to control and manipulate distant or oriented items. We illustrate the benefits of this technique with two scenarios. The first, when editing a text document, instead of the user reaching out to use the keyboard in an uncomfortable posture, or dragging the document away from a desired group's view, the user can simply drag or rotate the soft-keyboard towards him and start typing (Fig. 4.10 (c)). The second, if the user is working with overlapping canvases, she can drag the menu-bar of a partially obscured canvas and control it without having to change the canvas arrangement on the table.

4.5 Analyzing the Interface against Direct Manipulation Principles

In this section, we use Schneiderman's principles of direct manipulation [Shneiderman, 1993] to analyze and compare Surface Manager UI with general desktop window managers.

Continuous representation of objects of interest
An object of interest refers to digital artifacts that users want in order to perform a task (e.g., text, images, URLs, etc.). In this thesis, we refer to these objects as content elements. This principle asserts that objects of interest should

be always present to the user. Similar to desktop window managers, Surface Manager provides scrolling and zooming techniques to control the presence of these objects. Zoomable interfaces [Bederson and Hollan, 1994] break away from the windowing metaphor. They implement the metaphor of all objects exit on one surface, and the user can zoom in or out to reach different objects.

Within traditional GUI interfaces, objects of interest and secondary objects (e.g, menu-bars and pallets) are overlapping and occupy the same display area. Window systems handle this problem by providing a full-screen option, i.e., scaling the content to the entire screen and hiding menu-bars, providing "auto hide" mechanisms, or "stay on top" option. In our UI, we allow users to flick-away secondary objects, i.e., menu-bars, controllers and soft-keyboards, when they are not in use, to minimize surface clutter and occlusion.

Physical actions on objects vs. complex syntax
Single control point interfaces such as desktop computers are limited to mouse pointers, indirect touch pad gestures, and keyboard shortcuts, to trigger actions. Consequently, mediators such as menus, are needed to issue complex commands. In multi-touch interfaces, tangibles and gestures can provide an unlimited command vocabulary. However, it was found that people generally do not agree on which gesture is suitable for an action, nor do they agree on how to perform a certain gesture (e.g., with how many or which fingers) [Wobbrock et al., 2009]. Thus, currently, we still need mediators and tools on multi-touch surfaces. However, typical action sequences such as scaling and translating can be reduced with gesture manipulation. In our UI, we support a generic set of commands and gestures, as well as provide menu-bars to control and manipulate the interface. However, to enhance the user experience, we implemented the in-place technique to reduce the typical homing time on the desktop computer.

Fast, incremental and reversible operations with an immediately-apparent effect on the objects of interest
With advances in display and computing technologies, systems are able to react to a user's command in time and with minimum lag. On typical window managers, to change

the color of a text, the user must select the text, go to the menu bar and enter the desired color and commit. This interaction in neither fast nor incremental. On the other hand, scaling a window only requires selecting the window and dragging an edge or corner to see the window scale in time. In Surface Manager, gestures achieve a similar incremental effect. However, to change text color, for example, the user must go through a similar action cycle of typical window managers, but with reduced homing time.

Layered or spiral approach to learning
In desktop window managers, the transition from a novice user to a power user usually means the user knows most of the keyboard shortcuts. Windows systems provide users with *ctrl* and *alt* buttons to make this transition incremental [Wigdor and Wixon, 2011]. In Surface Manager, the user can incrementally scale, rotate, and translate an object with one gesture. Additional complex gestures can be used to provide incremental effect, and supported with visual options in the context menu.

4.6 Implementation

This section covers the technical implementation and structural design of Surface Manager's UI and gesture detector. We begin by describing the general software development process, and the overall class diagram of the system. Next, we describe in more detail the class hierarchy and software architecture for the graphic front end of the table system. Finally, the process of gesture detection is illustrated.

4.6.1 Software Development Process

Surface Manager was built over Apple's OS X 10.8 operating system. The software was developed using Objective-C and C programming languages in the Xcode IDE, and can be launched using a single .app file. Surface Manager was built over the foundational framework Table Engine, which was developed at our chair (Simon Voelker, 2012).

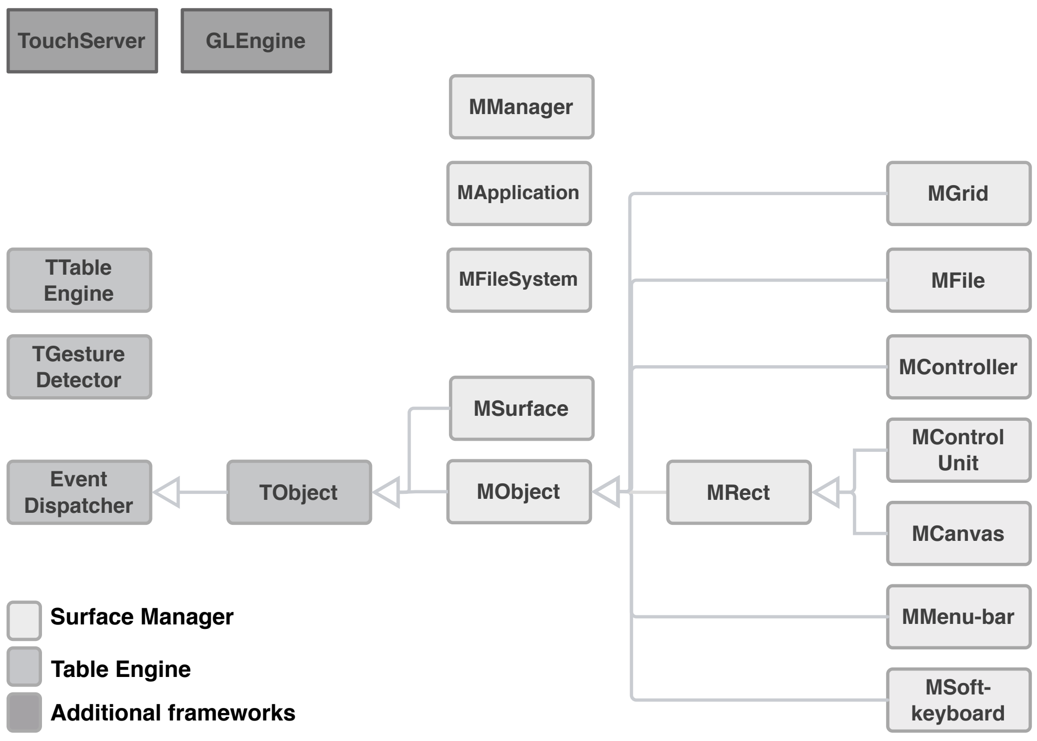


Figure 4.11: Surface Manager class diagram.

Table Engine includes: (a) a Touch Server to deliver touch events to the framework, (b) a GLEngine to provide convenient functions to draw and load texture files with OpenGL 3.2 framework, and (c) a set of base classes to start the rendering pipeline, dispatch events, and implement standard affine transformations.

The software was designed to run independent from any tabletop's technical settings and ergonomics. Our development process has been evolving for approximately six months, including the initial paper prototypes and the design of the system architecture.

4.6.2 System Class Diagram

In Fig. 4.11, we preset a simplified class diagram of the overall system.

In Surface Manager, the general look and feel of the UI are implemented in the base class MObject. MObject implements shared methods and properties in the UI toolkit, such as the generic commands, the widget iceberg, the context menu, and peripheral association mechanism. The base object grantees uniform visual and behavioral characteristics over the interface.

4.6.3 Surface Manager Software Architecture

Surface Manager implements the equivalent of the UI toolkit and window manager layers in desktop window system architecture (see section 2.2). In Fig. 4.12, we show the system architecture of Surface Manager. The system was designed to be simple and modular by using an interface of delegates and protocols between communicating classes. MSurface, MManager and MFileSystem are singleton classes, i.e., they are shared among all table sessions. MController and MControlUnit(s) are created once per session. The id of MController instance is used to access the singleton classes and identify the corresponding session. Several MCanvas, MMenu-bar and MSoft-keyboard instances can exit per session.

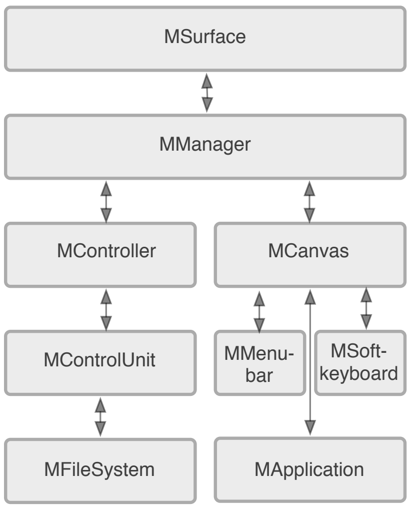


Figure 4.12: Surface Manager software architecture.

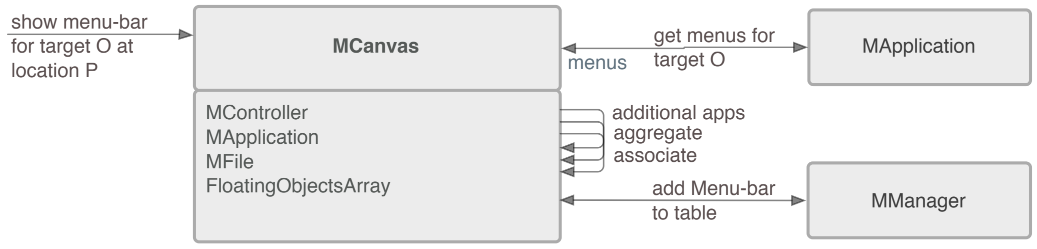


Figure 4.13: Peripheral association process.

MSurface is a non-transformable object that represents the table surface. It controls the background of the table, receives users' access gestures, and maintains the surface tree. When MSurface receives a user event it translates the request to: request a controller, or request a canvas, and calls MManager to handle that request.

MManager receives the user request from MSurface with two pieces of information: the user's location and orientation. MManager uses the location information to find or create a new MController. The decision is made by consulting a simple nearest neighbor algorithm. The algorithm was implemented to account for the lack of any user-identification mechanism. Once an instance of MController is allocated, MManager initiates a table session and adds MController to the top of the surface tree and at the location and orientation of the request. MManager implements surface layout policies to coordinate the location and orientation of root objects, and their arrangement in the surface tree.

MController initiates the storage, application, dock and trash MControlUnit(s). The units communicate with MFileSystem to fill up their grids with the available files and applications. Each MControlUnit maintains a record of its items and can work independently. Thus, MController only acts as a container widget and can be replaced with any other presentation.

MCanvas aligns its items in the center of its bounds if the size of items is less than the canvas's, and to the top-left corner otherwise.

When a show menu or keyboard invoking gesture is performed on MCanvas, MCanvas receives the event and the target object from the TEventDispatcher and continues with the peripheral associating process.

Fig. 4.13 illustrates the peripheral association process: (1) request the menus that apply to the target object from the current application, (2) create a menu of the other system applications that support the target object, (3) aggregate the menus in a single MMenu-bar, (4) associate that MMenu-bar with the target, and (5) sends MMenu-bar to MManager to be added to the table.

To end a table session, the user taps end session button on his or her MController, the MController adds all its canvases to the storage unit, and is then removed by MManager from the surface tree.

4.6.4 Gesture Detection Process

The purpose of designing a gesture detector was to enable multi-user interaction. In particular, we wanted a gesture server that can concurrently detect and respond to two or more users performing the same gesture (e.g., an access gesture) on the same object (e.g., the table surface). We achieved this by using a nearest neighbor algorithm to roughly classify concurrent touches into sets of potential gestures, before sending them to the gesture server.

In Fig. 4.14 we show the gesture detection process. TGestureDetector is a singleton class that receives user touch events from Table Engine. When TGestureDetector receives a new touch on a target object, it assembles an array of the TGestureRecognizers registered on the target object and its parent object, down to the root parent object. TGestureDetector gives registered TGestureRecognizers on the target object higher priority than TGestureRecognizers on parent object chain.

TGestureDetector starts by sending the touch object to the last registered TGestureRecognizer with highest priority. If a TGestureRecognizer accepts the touch, TGestureDetector

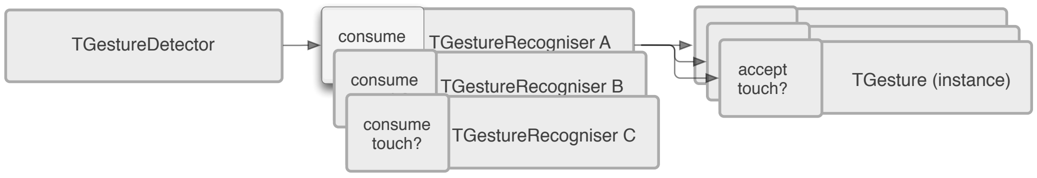


Figure 4.14: Gesture detection process.

binds the touch object with that TGestureRecognizer. Otherwise, TGestureDetector sends the touch to the next in line TGestureRecognizer.

Each TGestureRecognizer is associated in a one-to-one relation to a TGesture. TGestureRecognizer takes the touch, and finds or creates an instance of the associated TGesture to process that touch. TGestureRecognizer uses the location of the touch to find a gesture instance in its vicinity. Once an instance is found, it is asked if it can process this touch. TGesture designer can override a method in the TGesture class to constrain which touches can be accepted. If the touch was accepted, TGestureRecognizer binds the touch object with that gesture instance. Otherwise, TGestureRecognizer uses the touch and target object info to decide if it should create a new gesture instance to process that touch. For example, if the object of TGestureRecognizer only allows one user to interact with it at a time, TGestureRecognizer does not create a new TGesture instance if one already exists. In this case, the touch is not accepted in that TGestureRecognizer, and is returned to TGestureDetector.

Chapter 5

User Studies and System Evaluation

"The only way of discovering the limited of the possible is to venture a little way past them into the impossible."

—Author C. Clarke

During this thesis, we conducted two user studies: a primarily study, before we started the design process, and a qualitative user study and system evaluation, at the end of the process. This chapter describes the purpose, design, and findings of each study. Our overall goal is to enhance the UI of Surface Manager, test the underlying conceptual design, and begin the process of developing well-grounded theories of the use of tabletops as workspaces to inform our framework with empirical results.

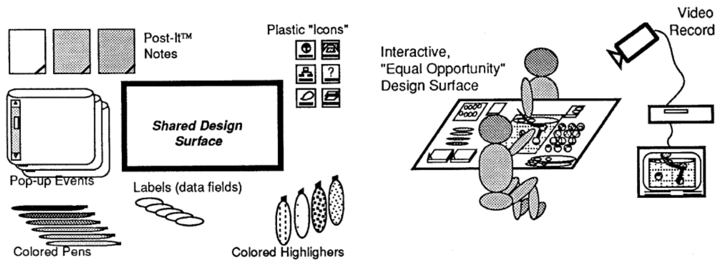
5.1 Preliminary Study: Conceptual Model Elicitation

The purpose of the preliminary study was to understand how people, experienced with desktop and smartphone UIs, perceive a table surface as a digital workspace.

5.1.1 Study Protocol

A PICTIVE prototype gives the user an early sense of how the finished product could look and feel

This study was conducted using the participatory research method PICTIVE (Plastic Interface for Collaborative Technology Initiatives through Video Exploration) [Muller, 1991]. PICTIVE is a paper mock-up technique that allows users to take part in the early design process in collaboration with the research or design team (see Fig. 5.1).



(a) (b)

Figure 5.1: PICTIVE research method [Muller, 1991]. (a) design material, (b) PICTIVE setting.

Tabletops are relatively new interactive devices considering their current availability and population of frequent users. Limitations and problems in tabletop design are mainly available from short-term and laboratory studies. Consequently, we had to find another source of information to help us frame our research question—how to enable tabletops as useful workspaces.

PICTIVE research method was selected to allow us to explore early UI and conceptual design of tabletop interfaces with potential users. The method provides participants with a high degree of freedom to express their thoughts and generate design ideas. The participatory nature of the method enables the researcher to engage with the user and understand the rational of each step of the user's actions instantly, thus, minimizing the effect of false memory, and providing a chance to explore design alternatives.

The study was conducted on a traditional physical table with one user and one researcher at a time (see Fig. 5.2). At the beginning of the study, users were provided with simple office supplies, such as Post-it notes, pens, paper, scissors, and paper clips. They were asked to use these material to symbolize components of the interface, such as icons, windows, or any custom widgets .

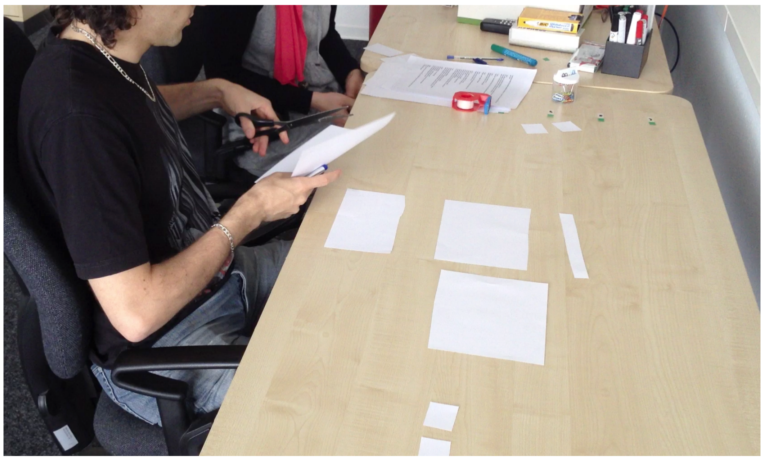


Figure 5.2: Preliminary study setup. Participant (wearing blank) engage in a participatory design session with the researcher.

The study session started by the researcher describing the desire to design a digital UI on a tabletop surface to be later integrated in the university campus. The interface should allow users to browse the internet, design presentations, and play games. The session proceeded as a collaborative brainstorming session. Users were asked to perform a series of simple tasks, such as start the table, open an application, copy an image from one application to another. In each task the user was asked to visualize the interface using the provided material, and to avoid lifting the objects off the surface during virtual interaction. The sessions were video recorded, and followed by a semi-structured feedback session.

Five users took part in the study. All users were males, their ages ranged between 22-27 years old. All participants were Computer Science students, and owned a multi-touch smartphone or tablet.

5.1.2 Results

Data sources, video recordings and interviews, were processed by annotating, coding, and finally, categorizing the observations.

Users had limited imagination for new UI widgets and interaction techniques

User-designed widgets were subsets of the desktop and smatphone interfaces: icons, pull-down menu-bars, windows, and virtual keyboards (Fig. 5.3). In [Wigdor and Wixon, 2011], the authors expressed concerns about the ability of users conducting a study to come up with new and concrete design ideas. In this study, the results confirmed that users had limited imagination for new UI widgets that go beyond the desktop or smartphone metaphors.

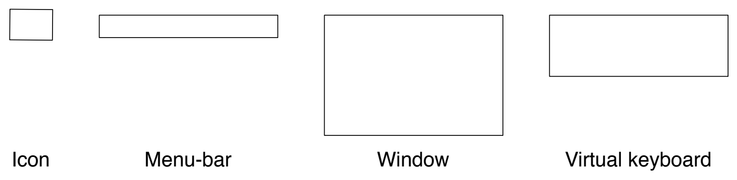


Figure 5.3: User-generated widgets.

Worspace layouts and metaphors

The interface design process started from the table's startup view—the first view the user sees when she approaches the table. All users used icon widgets to represent this view, but arranged them in different distributions. Fig. 5.4 shows the two layouts of the suggested distribution of icons on a table surface. The users were asked to design the interface to work for them and the researcher at the same time. In the first layout, U1 and U2 treated the table as one workspace that can offer widget duplicates for each co-located user according to the user's location and orientation. In the second layout, U3-U5 split the table surface to two independent workspaces, each running a separate instance of the UI.

Users had different ideas of the dynamics and metaphors of their workspaces. In Fig. 5.5 (a), U1, U2, and U5 worked only within a single, fixed area, located in front of them, on the table surface. In this work area, users had only the application they were currently interacting with. Other applications were spread out of the work area, on the table surface. To interact with another application, the users

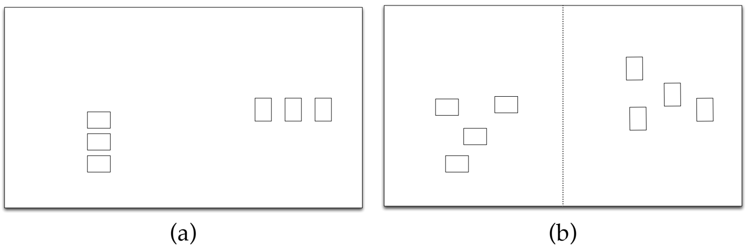


Figure 5.4: Users' perception of the table as a workspace for two. (a) table as single, shared workspace, (b) table as set of independent workspaces.

dragged the previous application out of their work area, and dragged the new application in. Users had fixed menubars on the bottom or top edges of the virtual work area.

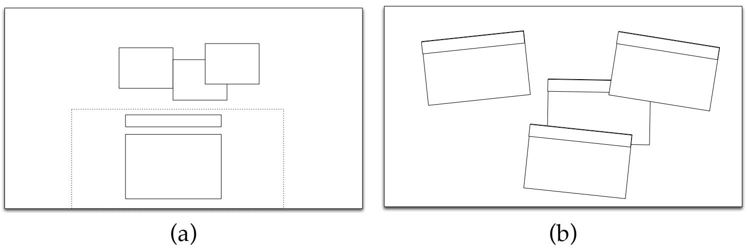


Figure 5.5: Users' workspace metaphors. (a) single work area on table surface, (b) table surface as a work area.

In Fig. 5.5 (b), U3 and U4 worked on the entire table surface. They spread their working items, and interacted with them in different locations and orientations.

Users used physical-like gestures during interaction

When we asked users how they would react if another person comes and sits on the opposite edge of the table, they used different grouping gestures to pull their items closer to them. Users shared items with the researcher by dragging them towards her. Users reached for far items by stretching themselves or standing up. When told to do so without standing up, two users made the gesture of pulling a virtual table cloth towards them, while another user suggested the use of a laser beamer that can be pointed to the target object.

In Table 5.1, we synthesized users' basic requirements for a digital workspace on a table surface.

User	Requirement
U1-U5	Provide a startup view that informs the user of what he can do on the table
U2	Make visible which window is receiving keyboard events, for example, by highlighting window border, or drawing a connecting line
U3	Provide context menu to manage root windows, and undo surface-level actions
U4	Provide dock area to reduce number of windows on surface
U4	Provide storage area
U1	Provide window grouping containers
U2, U4	Provide user credentials to access previous window arrangements

Table 5.1: Users' requirements for a digital workspace on a table surface.

Collaborating with potential users allowed us to focus on their basic requirements in the beginning of the design process

In summary, this study was intended to explore early UI and conceptual design of tabletop interfaces with potential users. The observations provided us with insights on how users perceive the table as a workspace, and enabled us to focus on basic user requirements in the context of design. However, in such exploratory interface, we were not able to extract concrete design ideas from users. In the future, other research methods could be explored.

5.2 Observational Study and System Evaluation

The purpose of the second user study was to: (1) evaluate Surface Manager UI, and (2) be the first of multiple rounds to develop well-grounded theories of tabletop usage from empirical observations. Surface Manager's flexibility allow us to investigate several aspects of task performing on interactive surfaces. The study was conducted with the following objectives:

- Assess and gather feedback on the overall usability, discoverability and learnability of the system's UI components, functionality, and interaction techniques.
- Understand how users perceive the controller metaphor.
- Detect usage and behavioral patterns during a task flow.

5.2.1 Task Design

Each participant worked individually on the tabletop, and performed three tasks in the same order. For the purpose of this study, we designed two simple applications, Text Editor and Poster Creator. The tasks required the user to design a poster in different themes. Below, we provide the description of the first task. The description of the rest of the tasks is available in Appendix A.

Dennis was working on the table before you, however, he didn't finish his task yet, and needs your help. He left the cover of a technical poster in the "storage unit" of the "controller" with a number of images and notes that he we would like you to arrange for printing later today. Feel free to choose the arrangement, rotations, sizes, background color, and text format that work for you. But please, use only the items Dennis provided and stick to the size of the given poster. After you finish, store your work and end the table session. Dennis will come later to check it.

The design and purpose of each task were as follows:

- The first task was designed to mimic, to an extent, the desktop interface. When the user starts this task he sees a controller mounted at a static location on the table. The controller in this task was not movable, and other than the control units, it only had an end session button. The task was designed to require minimal creativity—user were asked to rearrange a predefined set of images and text labels inside an existing poster. In the task description, we attempted to familiarize the user with the interface vocabulary, (e.g., controller, control unit, and table session).

  The purpose of this design was to asses the discoverability of the UI, familiarize users with the general interface, and see which UI features the users used or missed.

- The second task was designed to observe users from the point they encounter the table in a standby mode, to the point they leave it. The user was required to first start a table session by invoking the invisible controller to appear, in order to complete the design task. The controller in this task was mobile and provided additional buttons, i.e., hide, shrink and collapse, to manage it and the surface items. We provided the user with a set of predefined images and asked him to arrange them in a new poster and create suitable text labels. At the end of the task, users were asked to describe the functionality provided by the controller, and then end the table session.

  The aim of this task was to observe user's behavior towards the tabletop, particularly, at the entry and access stages, and to understand how users use the controller and perceive the controller metaphor.

- The third task was designed to observe and detect users' work patterns. The user was required to complete a full poster design. The theme we chose for this task was a movie ranking poster, under the assumption that users will find some of the available movie images familiar and focus more on the content rather than on the interface.

At this point of the study, we assumed the users to be familiar with the system, thus, allowing us to conclude on the learnability and usability of the interface.

5.2.2 Method and Procedure

We conducted a qualitative, observational, think aloud user study. The study sessions were video recorded, and each lasted about 30 minutes.

Before starting the sessions, participants were asked to fill out a consent form and provide demographic background information. Then, they were provided with a brief description of the tabletop and the overall system, as well as the goal of the study. The users were asked to explore the system freely during the tasks, and describe what they are doing, and why, at every step. Each user was handed a sheet of paper that described the tasks. After finishing the tasks, the users filled out a questionnaire (see Appendix B), and took part in a semi-structured interview session.

5.2.3 Participants

Nine participants took part in the study, 2 female users and 7 male users. They ranged in age between 19-34 years old, with the median of 25. All participants, but U6, were Computer Science students. U6 was the only left-handed participant.

5.2.4 Apparatus

Participants sat at a custom-made desk embedding a capacitive touch-sensing 27″ Perceptive Pixel display (see Fig. 5.6). The display had the area and resolution of 597×336 mm, 2560×1440 pixels. The display was connected to a Mac Pro running the Surface Manager software.

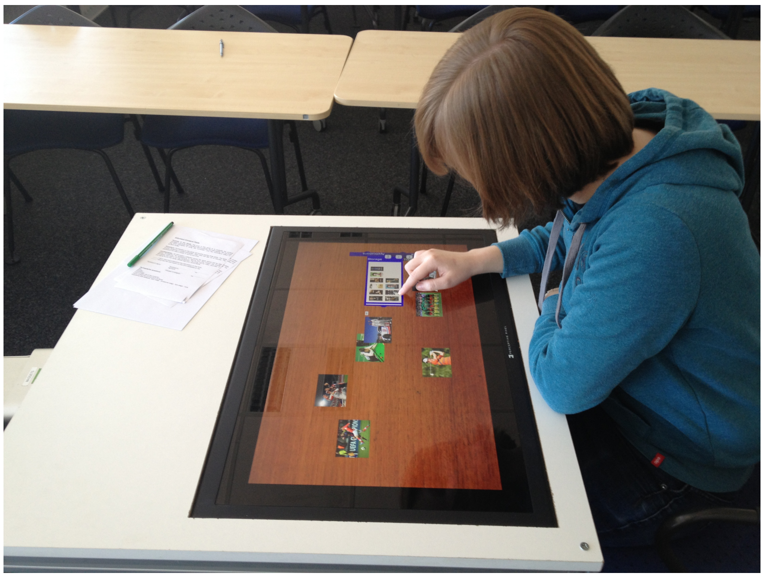


Figure 5.6: Observational study setup. Participant interacting directly with on the horizontal surface.

5.2.5 Data Collection and Analysis

Data was collected from three different sources: the video recordings, questionnaire, and feedback interviews, as a form of data source triangulation. We used grounded theory as a qualitative research method to develop theory from empirical results [Glaser and Strauss, 1967]. The data analysis procedure was as follows:

1. An open coding round was performed by one researcher. The researcher watched the video recordings and annotated actions, relevant time stamps, and vocabulary, such as terms users used to describe the interface, or verbs to describe actions.

2. Another round of coding was performed but this time the intention was to distinguish form from function. Users' behavior and comments were coded under two main categories: *visual and gestural enhancement*, and *concepts and functionality*. The former category includes users' comments on the used gestures and spatial properties of the interface elements. The latter is

composed of our observations of user behavior, and how users described and interpreted the system.

3. The data under concepts and functionality category was further categorized under 5 sub-categorizes : *system entry*, *system access*, *controller metaphor*, *activity landscape construction*, and *content manipulation*.

4. Questionnaire and feedback interview results were also coded in a similar fashion.

5.2.6 Results

The convention used to present the results of the study is as follows: under each category we describe our *observations* of users behavior, the *implication* of these observations and users' answers to questionnaire and interview questions, and, when available, *recommendations* for tabletop designers.

System Entry

Observation. In the first task, users saw the controller widget and started interacting with the tabletop immediately. In the second task, where the controller was not visible at startup, we noticed that the users paused for a few minutes before they started touching the tabletop surface. At different points in time, users began performing exploratory gestures at different locations on the table until they invoked the controller. In the third task, users engaged directly with the table, except for U7 who could not remember the gesture he used previously to invoke the controller.

The lack of a digital entry point confused users

Implication. In the conceptual framework (section 3.2.1) we described four types of entry points in interactive tabletop environments: environmental, physical, social and digital. In the first task, physical and digital entry points—the shape and size of the table, and the controller widget—were present. In the second task, despite the users having already interacted with the system, the lack of a digital entry point confused the users and made some of them hesitant to touch the table's surface.

Recommendation. Tabletop designers who want to depend on gestures in their UI design should seek to understand users' gesture patterns, and provide continuous feedback to users' touches in order to enhance the chance of users continuing their interaction with the surface. In our interface, we rendered a white circle of 7 mm (30 px) radius under each detected finger as a feedback to indicate system responsiveness, however, this was not enough, for example, for U3 who gave up trying to find the gesture after only a few failed trials.

System Access

The gestural access structure was hard to discover

Observation. To invoke the controller, users performed a sequence of repeated gestures at different locations on the table until the controller appeared. When asked to describe the gesture that invoked the controller, they all gave nonmatching descriptions. In the interview, users agreed that, unless known to them previously, the access gesture—a two-finger drag—was not intuitive, and is hard to discover.

Implication. The detected behavior, of gesture exploration, arises three questions:

1. Was this continuous quest to find the correct gesture the effect of the study environment? Would the users give up earlier in other situations?

2. Was this behavior the result of having seen the interface in the first task already? How could this behavior change in a new, undiscovered interface?

3. What is the effect of the nonmatching description users gave to their gestures on system acceptance? In future encounters with the system, will the user remember to repeat her complex sequence of gestures? Will the user be willing to try and discover the gesture again?

In the case of Surface Manager, the interface is intended for semi-public contexts—where users are assumed to be frequent users who are willing to put the time and effort to discover and learn the system. Thus, users could be tolerable to discover the access gesture and invoke the controller, but a false conceptual model of the access gesture can have a sever effect. In contrast, in public contexts, the system's immediate response is more important to attract the user, while the correct conceptual model of the access gesture can be sacrificed, since most users in these contexts are usually one-time users [Isenberg et al., 2009].

When we compared the access gesture each user used to invoke the controller, and the user's description of that gesture, we noticed that users rarely differentiate between a four-finger gesture and a five-finger gesture if the gestures were structurally the same (e.g., a four-finger and a five-finger tap).

Patterns in gesture discovery behavior

We traced users' gesture discovery behavior at the beginning of the second task in order to detect any useful patterns. Fig. 5.7 illustrates the locations where users tested gestures on the table, and the frequency of returning to each location. Fig. 5.8 shows the direction of users' movement and order of table locations each user tested.

All users started at, and if still hadn't found the correct gesture, returned to the area in front of them. Out of 9 users, 7 users tested tabletop locations in clockwise direction, while U9, and U6, the only left-handed user, moved counter-clockwise. Among participants, 55% of the users started their gesture discovery sequence with a single-finger tap, while the other 45% started with a single-finger drag.

A limitation of the collected data is that it was traced on a relatively small table surface, where each edge and corner was reachable by the sitting user, in addition to the relatively small group of users.

Recommendation. If the designer is to take advantage of a gesture as an access point, he should consider the illustrated patterns. The patterns show that users start looking for an access gesture in the center area in front of them then move to explore the accessible edges and corners of the ta-

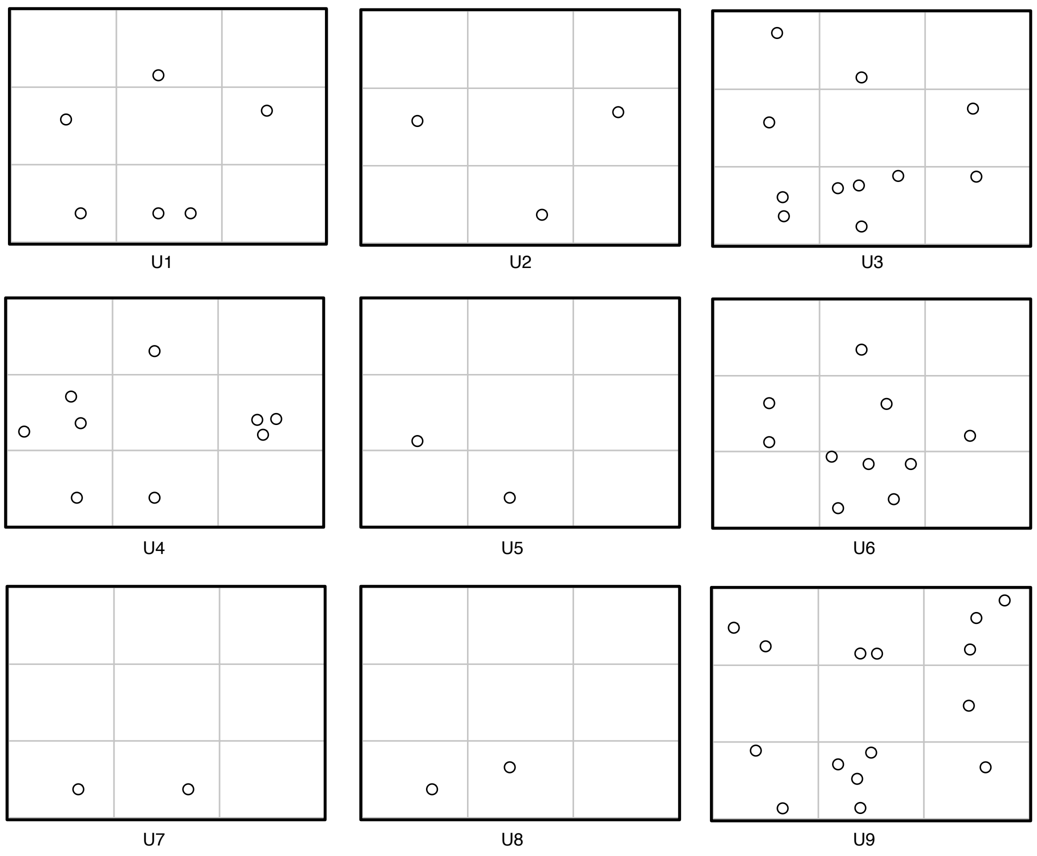


Figure 5.7: Gesture discovery behavior: test locations, and visiting frequency. Each rectangle represents one user's activity on the surface. The grid view is used to divide the table into 9 locations. Starting form the lower right corner of each rectangle, and moving clockwise to the area in front of the user on the table, the lower left corner of the table, the left edge, upper left corner, upper edge, etc. Each circle symbolizes a set of successive gestures performed at a location. Number of circles in each location reflects the frequency of revisiting the location after leaving it.

ble. Users begin their gesture sequences with single-finger gestures, tap or drag. The designed gesture should be tolerable to how users differentiate the number of involving fingers. Additionally, the designer should consider that users who once knew the gesture can forget it after a period of time.

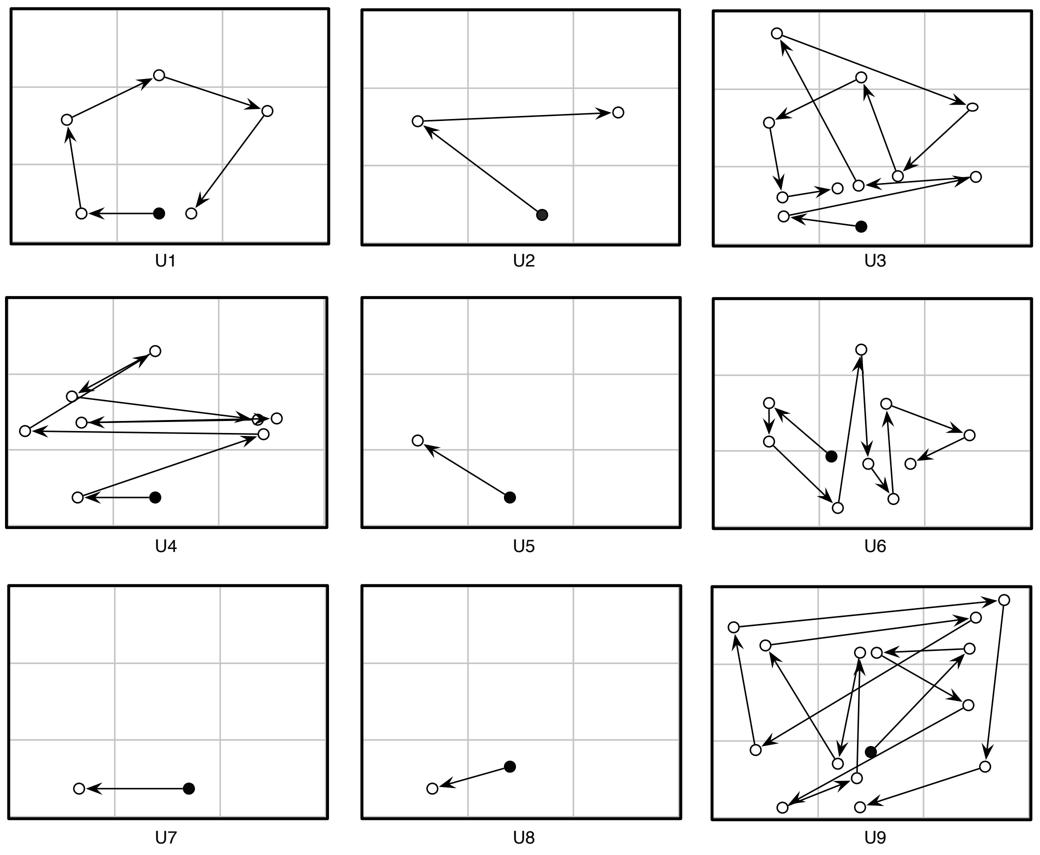


Figure 5.8: Gesture discovery behavior: direction of movement and order. The blacked circles represent the starting point of each user's path. The arrows show how the user moved from one location to another.

Controller Metaphor

Observation. In all tasks, once the users saw the controller, they started interacting with it, and were able to figure out how it works.

Users felt familiar with the controller metaphor

Implication. We asked the users during the interview "What did the controller represent in the system?". We received similar answers, such as the task-bar in Microsoft Windows, the dock in Apple's OS X, or simply the "main menu". While all answers are valid, when we designed the controller we intended for it to represent a complete user session, including the file system, application host, and the dock. To try and understand more how the users viewed the system, we asked them the following: ”If two users used the same table, will each user have his or her own controller or will they share one?”. Two users said they will share one, while the others felt that each user should have his own controller. U8 commented that what is inside the controller, files and applications, should be shared, but the widget itself shouldn’t, so the user who has one control unit open at a specific location on the table will always find it in the same way.

In the questionnaire and interviews, users reported the controller metaphor to be simple. When asked if they felt the controller was in their way while working, U6 said that it was in the first task, while 3/9 users said they had to move the controller from their focus area more than once.

Users favored controller mobility on any other feature

In Fig. 5.9, users ranked the features of the controller according to their usefulness. The features were: controller’s mobility, the ability to hide the controller using the hide button, the ability to close all opened control units using the shrink button, and the collapse&expand feature. Users mostly appreciated the mobility of the controller, and had varied opinions about other features. Four users reported they used the collapse&expand feature to clear and arrange their workspace, while the rest of users didn’t use the feature.

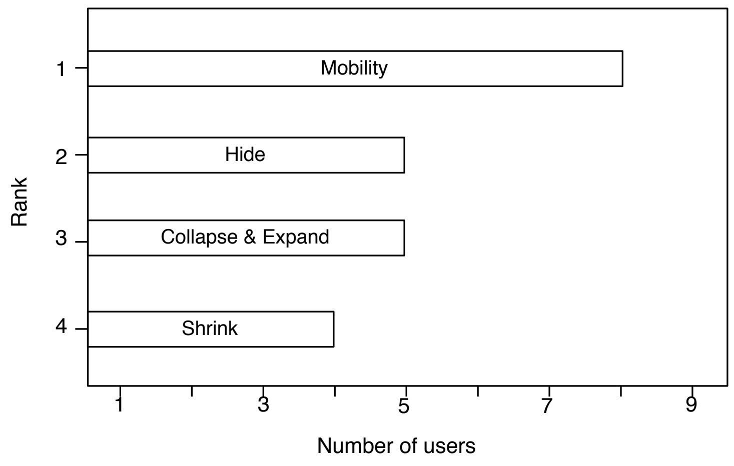


Figure 5.9: Users’ ranking of controller features: 1 most useful, 4 least useful.

In general, the controller was used and identified the way it was designed for. However, the type of tasks, and the setting of this study were not sufficient to test all aspects of the controller.

Activity Landscape Construction

Patterns in activity landscapes construction

Observation. We observed two reoccurring patterns of landscape construction in all three tasks (see Fig. 5.10). In the first pattern, 7 users spread all images, whether exiting on the poster like in the first task, or in the storage unit like in the third task, on the surface. The users then moved the items one by one to the poster, where they positioned them in the desired location before going to look for the second image. This pattern is beyond the typical desktop metaphor, and more closely related to traditional tables' affordance. In the second pattern, 2 users accumulated all images on the poster itself, and selected and arranged the images by repeatedly shuffling between the images.

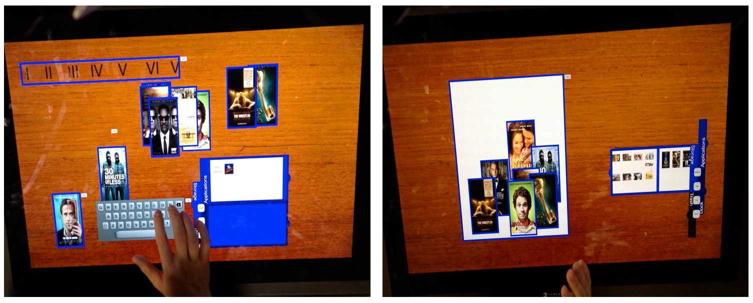

(a) (b)

Figure 5.10: Activity landscape construction patterns. (a) working on the surface (b) working in the application.

Implication. In the conceptual framework we mentioned how Kirsh [2001] distinguishes between two structures of office desks, the neat—tidy desks where limited number of work items are visible and located in a standardized place, and the scruff—messy office that can hold large amounts of information. In the context of this study, the first pattern resembles that of the messy office structure.

Recommendation. Finding a connection between Kirsh's office desk structures and the observed patterns could inform the design of surface managers, particularly, surface partitioning and layout policies.

Content Manipulation

Observation. In Surface Manager, interface and content manipulation is done using direct manipulation techniques, gestures, and menus. Users discovered how to invoke menu-bars quickly. They were observed to request the menu-bars when needed, use them intuitively, and then hide them when they were done. We also noticed that users would have several menu-bars appear simultaneous, edit the associated items, and then go back to hide the menu-bars. Only two users made use of menu-bars' mobility.

Users utilized the scaling feedback in the controller to facilitate file searching

In the third task, where the users needed to find images in the storage unit, we noticed the users had hard time distinguishing the images. Users reported that despite the zooming feature in control units, images were still small. Alternatively, two users made use of the scaling feedback of control units, and dragged each image to the edge of the unit, where it grows to its original size. While still holding the image, the user examined it, then returned it or dragged it out for usage. The other users dragged the images out of the storage unit completely and returned them if not wanted.

Menu-bars were found to be simple to use, and require only few action steps

Implication. We were keen to find how users use the menu-bars, in particular, how the in-place interaction technique improved or hindered their task flow. Diagram 5.11 shows how the users responded to the questionnaire about the usability of menu-bars. About 50% of the users reported that they were confused by the menu-to-target item association. In our interface, we relied merely on proximity. We wanted to avoid any visible connectors, for example, between the peripherals. However, we observed that proximity was not sufficient. Users found menu-bars appearing under the target item to be useful. In the interview, three users commented that using the menu-bars required only few steps to achieve the desired effect on the target item compared to the desktop.

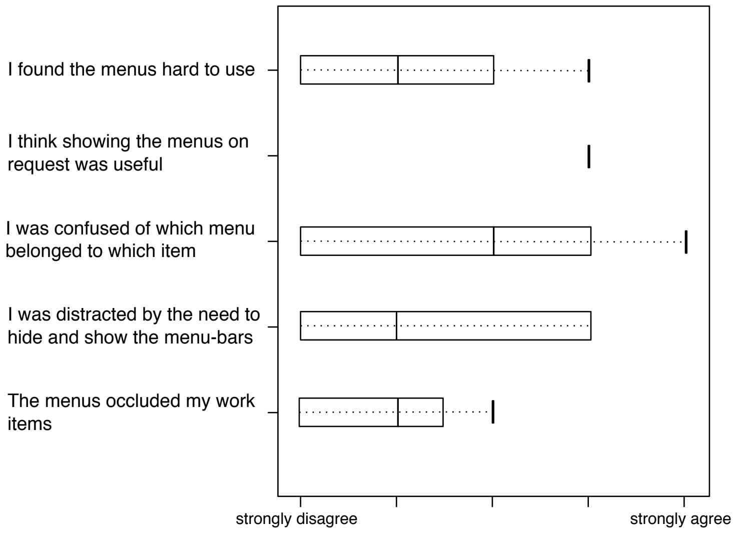


Figure 5.11: Box plot for menu-bar related points in the questionnaire.

5.3 Discussion

We achieved simplicity and usability in a new computing environment and interface-design

During both conducted studies, we noticed that users enjoyed the familiarity, of UI and gesture design, but were open to trade familiarity for simplicity and speed. The controller metaphor, the core of Surface Manager, was approved by users for its simplicity, reachability, and mobility. We also noticed that users felt comfortable placing physical sheets of paper on the surface while interacting with it. Users suggested that the system can work significantly better if we enhanced the visual appearance and feedback with animations. In the questionnaire, we asked the users if they imagine that most people will be able to use this interface without prior training and they all confirmed.

We collected rich sets of data during the first and second studies, however, we defer to provide any conclusive manifestations, and bound to only reflect our observations and analysis, as we were limited in the size of table and group of users. Nevertheless, our findings, as we have mentioned, are the result of the first round of a series of user studies that aim to develop well-grounded tabletop usage theories. Designers can make use of our initial findings, for example, the illustrated gesture discovery behaviors and landscape construction patterns, as guidelines in the early stages of the design process.

Chapter 6

Summary and Future Work

"Successful design does not necessarily perpetuate users' current ways of working, but it is built on a deep understanding of those ways and of how a new design will change them."

—Catherine Courage

6.1 Summary and Contributions

Research question: how to enable tabletops as multi-task, general-purpose workspaces

This thesis started with the broad research question—how to enable tabletops as multi-task, general-purpose workspaces. On desktops, window managers have been responsible for supporting and enabling these features. However, desktop window managers are designed for single-user systems and cannot be directly utilized in tabletops without sacrificing usability. Consequently, we proposed surface managers as a new package of user-interface software on interactive tabletops.

Surface managers provide equivalent facilities of window managers tailored for tabletops

A surface manager is a user-interface that supports and facilitates users' concurrent work processes on large, horizontal, multi-touch surfaces. Like desktop window managers, surface managers provide interface elements and policies to launch and manage concurrent applications. Surface managers additionally provide space management tools, and support simultaneous user interaction and workspace partitioning.

In order to design a surface manager, we started by analyzing the design space, requirements and constraints of tabletops, and the paradigm shift in UI design between desktops and tabletops.

We highlighted five aspects that we believe are critical to investigate when attempting to redesign software from the desktop computer to interactive surfaces. These are: the WIMP paradigm, input focus shifting, single-user orientation, direct manipulation, and the Desktop metaphor.

A vital part of the design process was to understand users' physical and behavioral interaction around tables. We conducted a preliminary, participatory user study in order to explore early UI and conceptual design of tabletop interfaces with potential users. The observations provided us with insights on how users perceive the table as a workspace, and enabled us to focus on basic user requirements in the context of design.

Distributed cognition theories to understand the structure of tabletop environments

Next, we reviewed fundamental theories of workplace and tabletop territoriality. We extended Kirsh's [2001] model of the context of work in office environments to define the structures users need while working on tabletops. This has resulted in the conceptual framework of surface manages.

The conceptual framework builds on Kirsh's model in three ways: (1) it attempts to frame the ideas from distributed cognition research in a way that is more usable by HCI designers, (2) it adapts the abstract concepts of the model from traditional office desks and environments to interactive tabletop environments, and (3) it expands the previous model, which is single-user oriented, to account for multiple co-located user settings.

The framework provides a descriptive model of the tangible and conceptual structures users need to perform tasks in tabletop environments. It sets out three concepts to define and design these structures: workspace access, surface partitioning, and coordination policies.

The conceptual framework elaborates on the design of access points to help designers assess different design alternatives. We described a modeling technique for access point design, produced the corresponding design space, and evaluated design alternatives in the light of empirical tabletop literature, as well as effectiveness figures of merit from [Card et al., 1990].

The described framework recognizes the role of tables in physical contexts and does not attempt to design holistic work environments for interactive surfaces. Instead, it focuses on observed tabletop practices and provides structural guidance to support these practices and enhance the overall user experience.

The proposed framework still requires long-term and observational studies to understand the deep effect the table's form factor and contexts have on the workflows of co-located users. With the use of surface manager software and considering the current state of commercial tabletops, we believe that this data will become available in the near future.

The aim of Surface Manager design was to augment the utility of traditional tables in semi-public contexts

We applied the framework to the interface design of Surface Manager. Our overall goal was to support and facilitate users' concurrent work processes on interactive surfaces, in various social interaction settings. We aimed to augment the current role of traditional tables in semi-public contexts with a general purpose, non-intrusive, "calm" UI that enables flexible resource access and manipulation. Consequently, we designed the controller metaphor. This metaphor allows single users as well as collaborators to approach and interact with the tabletop, while maintaining flexible seating arrangements.

Surface Manager design goals

Surface Manager was designed to achieve five design goals: (1) equal access privileges: users should be able to access the system from any location or orientation on the table; (2) local control distribution: the scope of one user's control should be localized to the user's workspace and must not affect or interrupt the work of others; (3) window management facilities, such as application hosting and layout policies; (4) handling tabletops design constraints, mainly the constraints that can interrupt the workflow of users, such as surface clutter and object reachability; and (5) designing a calm computing environment: Surface Manager should integrate with the traditional table role without being intrusive, loud, or demanding.

The main building blocks of the Surface Manager interface are: system access model, UI toolkit, file system, generic commands, tree manager, peripheral association mechanism, and a set of interaction techniques (policies). The interaction techniques provided by Surface Manager aim to handle tabletops design constraints. The techniques include: (a) in-place, (b) collapse&expand, and (c) remote control interaction techniques. For example, collapse&expand allows the user to reach any distant object by tapping the collapse button on the controller. Once the user does, she can press the now expand button to return her objects to their original spatial arrangement.

Surface Manager UI toolkit

The UI toolkit in Surface Manager was designed to maintain a consistent interface, and to embed design solutions for some of the direct-touch issues. The toolkit is composed of three groups of UI elements: (a) primitive elements, such as buttons and text-fields, (b) layout containers, such as scroll and grid views, and (c) core elements: controllers, canvases, menu-bars, and soft-keyboards.

Gesture detector

In addition, we designed a gesture detector server for Surface Manager that can concurrently detect and respond to two or more users performing the same gesture (e.g., an access gesture) on the same object (e.g., the table surface).

A qualitative, observational user study to detect tabletop usage patterns and evaluate Surface Manager UI

Surface Manager was evaluated in a qualitative, observational user study. The study results highlighted tabletop usage patterns, such as gesture discovery behavior and activity landscape construction. Users feedback on the overall design, learnability, and discoverability of the system was positive and encouraging.

6.2 Future Work

Developing well-grounded theories of tabletop usage

The qualitative user study we conducted during this thesis is only the first of multiple rounds to develop well-grounded theories of tabletop usage from empirical observations. Our next steps include iterating on Surface Manager UI design, and designing a user study to test the controller metaphor and overall UI with co-located users. This shall be followed with a series of studies in different technological, physical, and social interaction settings.

Further investigations and quantitative studies can be performed to follow up on the tabletop usage patterns that emerged during our user study. For example, the gesture discovery behavior can be studied further and processed with Artificial Intelligent techniques to build heuristic models of users' gestural behavior. This can be used as input to enhance our gesture detector.

Our work in this thesis can be extended as two complementary paths: refining the conceptual framework with more theoretical analysis and empirical results, and iterating and extending Surface Manager UI design.

Exploring new metaphors

The controller metaphor was received well by participants in the user study. Nevertheless, we intend to test several other metaphors in Surface Manager, for example, the metaphors designed for NUI (e.g., Magnet, Sphere, and Unfold metaphors [Hofmeester and Wixon, 2010]).

Improving file navigation and search

In our interface, we designed a simple file system that allows users to access files in three ways: speed access, system access, and application access. For example, speed access allows the user to start creating content on tabletops after a single gesture. However, we only used basic file navigation and visualization techniques. A future work could be to investigate with more visualization tools. Additionally, due to text input limitations, we only offered file search via simple navigation, scaling, and scrolling mechanisms. Alternatively, a visual tagging system could be designed. This system would allow users to tag a file by cropping part of its view and then utilizing grouping and associative searching tools to replace hierarchical file storing, naming, and extensive text input requirements.

A new filing system

In addition, in the design of the file system we explored two new filing ideas. In the first idea, we tried to substitute the notion of files and applications, with content and services. Content represents users' objects of interest that are needed to accomplish a task (e.g., images, text, URL). A service is a set of options provided by a service provider that can be requested and received from Surface Manager in a uniform format. This led to the second idea: the user can request two services for one content object and the system aggregates the options of both services in one format. Accordingly, the user can, for example, edit the images of a single pdf file with an image editor, and the text in a text edit, simultaneously. We designed a prototype for this file system, but the design still requires several iterations and user evaluations.

Surface Manager as a general-purpose UI

For Surface Manager to function as a general-purpose UI we need to achieve the following: (a) simple and clean visual design of UI elements, (b) expressive and continuous feedback, (c) comprehensive gestural language, (d) user-identification mechanisms, (e) improved filing system, and (f) additional interaction techniques to improve the overall user experience.

Surface Manager as a complete window system

A longitudinal goal is to extend Surface Manager to design an equivalent of a complete window system. Such a system on tabletops will provide abstractions from tabletop technologies and hardware, provide a reliable multi-touch and gesture recognition system, plug-ins that allow seamless integration with other user devices, tracking and scanning mechanisms, as well as support for tangible objects. It could also implement several interaction models to handle different social settings. It would also be essential for the system to provide privacy settings, and be connected to the internet and cloud storage.

Appendix A

Task Description for Observational User Study

First task Dennis was working on the table before you, however, he didn't finish his task yet, and needs your help. He left the cover of a technical poster in the *storage unit* of the *controller* with a number of images and notes that he we would like you to arrange for printing later today. Feel free to choose the arrangement, rotations, sizes, background color, and text format that work for you. But please, use only the items Dennis provided and stick to the size of the given poster.

After you finish, store your work and end the table session. Dennis will come later to check it.

Second task Start a new session on the table. You will see several images randomly spread out on the surface. Your task is to make use of these images, and create suitable text labels to design a sports poster. You can use the Poster Creator and Text Editor applications to accomplish this task.

Can you describe the controller features?

After you finish please store your work and end the session.

Third task This is your final task. Sahra was given the task of creating a poster that shows the best and worse movies from the past 5 years. Her boss already decided on the movies she could rank and stored them in the storage unit. But Sahra does not watch movies! She needs your help. Please, design a poster that shows a few movies and their rankings (according to your preference). You can use any resource provided by the system. After you finish please store your work and end the session.

Appendix B

Questionnaire Form for Observational User Study

	strongly disagree	disagree	neutral	agree	strongly agree
I used the controller to:					
Access applications					
Access stored items					
Store items					
Delete items					
Dock items					
Exit session					
I found the controller to be in my way while working					

I found the controller's mobility useful					
I found the controller's hide button useful					
I think I will use the shrink button frequently					
I found the menus hard to use					
I was distracted by the need to hide and show the menubars					
I think showing the menus on request was useful					
I was confused of which menu belonged to which item					
The menus occluded my work items					
The interface distracted me from the content					
I imagine that most people will be able to use this interface without prior training					
I used the collapse button to:	reach far items	store all items	clear the surface	arrange my space	didn't use it
I rank controller features from 1-4 as follows (1 is most useful, 4 is least):	hide button	collapse button	shrink button	mobility	
I used the zooming functionality of the control units to:	fast search for items	fine search for items	estimate storage size	avoid scrolling	didn't use it

Table B.1: Questionnaire form for observational user study.

Bibliography

C.J. Ackad, A. Collins, and J. Kay. Switch: exploring the design of application and configuration switching at tabletops. In *Proc. ITS '10*, pages 95–104. ACM, 2010.

D. Andreychuk, Y. Ghanam, and F. Maurer. Adapting existing applications to support new interaction technologies: technical and usability issues. In *Proc. SIGCHI '10*, EICS '10, pages 199–204. ACM, 2010.

T. Apted, A. Collins, and J. Kay. Heuristics to support design of new software for interaction at tabletops. In *CHI'09 Workshop on Multitouch and Surface Computing*, 2009.

T. Bartindale, C. Harrison, P. Olivier, and S.E. Hudson. Surfacemouse: supplementing multi-touch interaction with a virtual mouse. In *Proc. TEI '11*, pages 293–296. ACM, 2011.

M. Beaudouin-Lafon. Instrumental interaction: an interaction model for designing post-wimp user interfaces. In *Proc. SIGCHI '00*, pages 446–453. ACM, 2000.

Benjamin B. Bederson and James D. Hollan. Pad++: a zooming graphical interface for exploring alternate interface physics. In *Proc. UIST '94*, UIST '94, pages 17–26. ACM, 1994.

G. Besacier, G. Rey, M. Najm, S. Buisine, and F. Vernier. Paper metaphor for tabletop interaction design. *Human-Computer Interaction. Interaction Platforms and Techniques*, pages 758–767, 2007.

P. Brandl, J. Leitner, T. Seifried, M. Haller, B. Doray, and P. To. Occlusion-aware menu design for digital tabletops. In *CHI EA '09*, CHI EA '09, pages 3223–3228. ACM, 2009.

H. Brignull and Y. Rogers. Enticing people to interact with large public displays in public spaces. In *Proc. INTERACT '03*, volume 3, pages 17–24, 2003.

H. Brignull, S. Izadi, G. Fitzpatrick, Y. Rogers, and T. Rodden. The introduction of a shared interactive surface into a communal space. In *Proc. CSCW '04*, pages 49–58. ACM, 2004.

B. Buxton. Multi-touch systems that i have known and loved. *Microsoft Research*, 2007.

S.K. Card, J.D. Mackinlay, and G.G. Robertson. The design space of input devices. In *Proc. SIGCHI '90*, pages 117–124. ACM, 1990.

A. Collins, T. Apted, and J. Kay. Tabletop file system access: Associative and hierarchical approaches. In *TABLETOP'07*, pages 113–120. IEEE, 2007.

A. Collins, C.J. Ackad, T. Apted, P. Sztajer, P. Ward, H. Weng, and J. Kay. Core functionality and new applications for tabletops and interactive surfaces. In *Proc. UbiComp '11*, pages 607–608. ACM, 2011.

P. Dietz and D. Leigh. Diamondtouch: a multi-user touch technology. In *Proc. UIST '01*, pages 219–226. ACM, 2001.

J.J. Edney. Human territories: Comment on functional properties. *Environment and Behavior*, 1976.

G.W. Fitzmaurice, H. Ishii, and W.A.S. Buxton. Bricks: laying the foundations for graspable user interfaces. In *Proc. SIGCHI '95*, pages 442–449. ACM Press/Addison-Wesley Publishing Co., 1995.

C. Forlines, D. Wigdor, C. Shen, and R. Balakrishnan. Direct-touch vs. mouse input for tabletop displays. In *Proc. SIGCHI '07*, pages 647–656. ACM, 2007.

G. Furumi, D. Sakamoto, and T. Igarashi. Snaprail: a tabletop user interface widget for addressing occlusion by physical objects. In *Proc. ITS '12*, ITS '12, pages 193–196. ACM, 2012.

Barney G Glaser and Anselm L Strauss. *The discovery of grounded theory: Strategies for qualitative research*. Aldine de Gruyter, 1967.

J. Gosling, D.S.H. Rosenthal, and M.J. Arden. *The NeWS book: an introduction to the network/extensible window system*. Springer, 1989.

M. S Hancock, F. D Vernier, D. Wigdor, S. Carpendale, and C. Shen. Rotation and translation mechanisms for tabletop interaction. In *TableTop '06. First IEEE International Workshop on*, pages 8–pp. IEEE, 2006.

T.E. Hansen, J.P. Hourcade, M. Virbel, S. Patali, and T. Serra. Pymt: a post-wimp multi-touch user interface toolkit. In *Proc. ITS '09*, pages 17–24. ACM, 2009.

D.A. Henderson Jr and S. Card. Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *TOG '86*, 5(3): 211–243, 1986.

U. Hinrichs, S. Carpendale, S. Scott, and E. Pattison. Interface currents: Supporting fluent collaboration on tabletop displays. In *Smart Graphics*, pages 924–924. Springer, 2005.

U. Hinrichs, M. Hancock, C. Collins, and S. Carpendale. Examination of text-entry methods for tabletop displays. In *TABLETOP'07*, pages 105–112. IEEE, 2007.

K. Hofmeester and D. Wixon. Using metaphors to create a natural user interface for microsoft surface. In *CHI EA '10*, pages 4629–4644. ACM, 2010.

D. Holman. Gazetop: interaction techniques for gaze-aware tabletops. In *CHI EA '07*, pages 1657–1660. ACM, 2007.

E. Hornecker. A design theme for tangible interaction: embodied facilitation. In *ECSCW 2005*, pages 23–43. Springer, 2005.

E. Hornecker. "i don't understand it either, but it is cool"-visitor interactions with a multi-touch table in a museum. In *TABLETOP '08*, pages 113–120. IEEE, 2008.

E. Hornecker, P. Marshall, and Y. Rogers. From entry to access: how shareability comes about. In *Proc. DDPI '07*, pages 328–342. ACM, 2007.

S. Hunter. *MemTable-Contextual Memory in Group Workspaces*. PhD thesis, Massachusetts Institute of Technology, 2009.

E. Hutchins and G. Lintern. *Cognition in the Wild*, volume 262082314. MIT press Cambridge, MA, 1995.

P. Isenberg, U. Hinrichs, M. Hancock, M. Tobiasz, and S. Carpendale. Information visualization on interactive tabletops in work vs. public settings. *CoVIS '09*, page 28, 2009.

R.J.K. Jacob, A. Girouard, L.M. Hirshfield, M.S. Horn, O. Shaer, E.T. Solovey, and J. Zigelbaum. Reality-based interaction: a framework for post-wimp interfaces. In *CHI '08*, volume 1, page 201. ACM, 2008.

J. Johnson, T.L. Roberts, W. Verplank, D.C. Smith, C.H. Irby, M. Beard, and K. Mackey. The xerox star: A retrospective. *Computer*, 22(9):11–26, 1989.

L. Jun, D. Pinelle, C. Gutwin, and S. Subramanian. Improving digital handoff in shared tabletop workspaces. In *TABLETOP '08.*, pages 9–16. IEEE, 2008.

Y. Kakehi and T. Naemura. Ulteriorscape: Interactive optical superimposition on a view-dependent tabletop display. In *TABLETOP '08*, pages 189–192. IEEE, 2008.

V. Kaptelinin and M. Czerwinski. *Beyond the desktop metaphor: designing integrated digital work environments*, volume 1. The MIT Press, 2007.

D. Kirk, S. Izadi, O. Hilliges, R. Banks, S. Taylor, and A. Sellen. At home with surface computing? In *Proc. CHI '12*, pages 159–168. ACM, 2012.

D. Kirsh. The context of work. *Human–Computer Interaction*, 16(2-4):305–322, 2001.

D. Klinkhammer, M. Nitsche, M. Specht, and H. Reiterer. Adaptive personal territories for co-located tabletop interaction in a museum setting. In *Proc. ITS '11*, pages 107–110. ACM, 2011.

R. Kruger, S. Carpendale, S.D. Scott, and S. Greenberg. How people use orientation on tables: comprehension, coordination and communication. In *Proc. SIGGROUP '03*, pages 369–378. ACM, 2003.

D. Leithinger and M. Haller. Improving menu interaction for cluttered tabletop setups with user-drawn path menus. In *TABLETOP '07*, pages 121–128. IEEE, 2007.

G. J. Lepinski, T. Grossman, and G. Fitzmaurice. The design and evaluation of multitouch marking menus. In *Proc. CHI '10*, CHI '10, pages 2233–2242. ACM, 2010.

S. Lepreux, S. Kubicki, C. Kolski, and J. Caelen. From centralized interactive tabletops to distributed surfaces: the tangiget concept. *International Journal of Human-Computer Interaction*, 28(11):709–721, 2012.

W. Lidwell, K. Holden, and J. Butler. *Universal Principles of Design, Revised and Updated: 125 Ways to Enhance Usability, Influence Perception, Increase Appeal, Make Better Design Decisions, and Teach through Design*. Rockport publishers, 2010.

I. MacKenzie and S. Jusoh. An evaluation of two input devices for remote pointing. *Engineering for Human-Computer Interaction*, pages 235–250, 2001.

N. Mahyar, A. Sarvghad, T. Weeres, and M. Tory. Cospaces: Workspaces to support co-located collaborative visual analytics. In *DEXIS '11*, page 36, 2012.

J.H. Morris, M. Satyanarayanan, M.H. Conner, J.H. Howard, D.S. Rosenthal, and F.D. Smith. Andrew: A distributed personal computing environment. *Communications of the ACM*, 29(3):184–201, 1986.

Meredith Ringel Morris, Andreas Paepcke, Terry Winograd, and Jeannie Stamberger. Teamtag: exploring centralized versus replicated controls for co-located tabletop groupware. In *Proc. CHI '06*, CHI '06, pages 1273–1282. ACM, 2006a.

M.R. Morris, D. Morris, and T. Winograd. Individual audio channels with single display groupware: effects on communication and task strategy. In *Proc. CSCW '04*, pages 242–251. ACM, 2004a.

M.R. Morris, K. Ryall, C. Shen, C. Forlines, and F. Vernier. Beyond social protocols: Multi-user coordination policies for co-located groupware. In *Proc. CSCW '04*, pages 262–265. ACM, 2004b.

M.R. Morris, A. Paepcke, and T. Winograd. Teamsearch: Comparing techniques for co-present collaborative search of digital media. In *TableTop '06*, pages 8–pp. IEEE, 2006b.

M.R. Morris, A.J.B. Brush, and B.R. Meyers. A field study of knowledge workers' use of interactive horizontal displays. In *TABLETOP '08*, pages 105–112. IEEE, 2008.

M.R. Morris, D. Fisher, and D. Wigdor. Search on surfaces: Exploring the potential of interactive tabletops for collaborative search tasks. *Information processing & management*, 46(6):703–717, 2010.

Michael J. Muller. Pictive: an exploration in participatory design. In *Proc. CHI '91*, CHI '91, pages 225–231. ACM, 1991.

B.A. Myers. A taxonomy of window manager user interfaces. *CG '88*, 8(5):65–84, 1988.

B.A. Myers, R. Bhatnagar, J. Nichols, C.H. Peck, D. Kong, R. Miller, and A.C. Long. Interacting at a distance: measuring the performance of laser pointers and other devices. In *Proc. CHI '02*, pages 33–40. ACM, 2002.

B.A. Nardi. *Context and consciousness: activity theory and human-computer interaction*. mit Press, 1995.

D. Norman. *The design of everyday things*. Basic books, 2002.

D.A. Norman and S.W. Draper. *User centered system design; new perspectives on human-computer interaction*. L. Erlbaum Associates Inc., 1986.

P. Peltonen, E. Kurvinen, A. Salovaara, G. Jacucci, T. Ilmonen, J. Evans, A. Oulasvirta, and P. Saarikko. It's mine, don't touch!: interactions at a large multi-touch display in a city centre. In *Proc. CHI '08*, pages 1285–1294. ACM, 2008.

D. Pinelle, M. Nacenta, C. Gutwin, and T. Stach. The effects of co-present embodiments on awareness and collaboration in tabletop groupware. In *Proc. GI '08*, pages 1–8. Canadian Information Processing Society, 2008.

D. Pinelle, M. Barjawi, M. Nacenta, and R. Mandryk. An evaluation of coordination techniques for protecting objects and territories in tabletop groupware. In *Proc. CHI '09*, pages 2129–2138. ACM, 2009.

A.M. Piper and J.D. Hollan. Tabletop displays for small group study: affordances of paper and digital materials. In *Proc. CHI '09*, pages 1227–1236. ACM, 2009.

J. Rekimoto and M. Saitoh. Augmented surfaces: a spatially continuous work space for hybrid computing environments. In *Proc. CHI '99*, pages 378–385. ACM, 1999.

M. Ringel, K. Ryall, C. Shen, C. Forlines, and F. Vernier. Release, relocate, reorient, resize: fluid techniques for document sharing on multi-user interactive tables. In *CHI EA '04*, CHI EA '04, pages 1441–1444. ACM, 2004.

Y. Rogers and S. Lindley. Collaborating around vertical and horizontal large interactive displays: which way is best? *Interacting with Computers*, 16(6):1133–1152, 2004.

Yvonne Rogers, Youn-kyung Lim, William R. Hazlewood, and Paul Marshall. Equal opportunities: Do shareable interfaces promote more group participation than single user displays? *Human–Computer Interaction*, 24(1-2):79–116, 2009.

K. Ryall, C. Forlines, C. Shen, and M.R. Morris. Exploring the effects of group size and table size on interactions with tabletop shared-display groupware. In *CSCW '04*, volume 6, pages 284–293. Citeseer, 2004.

R.W. Scheifler and J. Gettys. The x window system. *ACM Transactions on Graphics (TOG)*, 5(2):79–109, 1986.

S. D. Scott, M. Sheelagh, T. Carpendale, and K. M. Inkpen. Territoriality in collaborative tabletop workspaces. In *Proc. CSCW '04*, pages 294–303. ACM, 2004.

S.D. Scott and S. Carpendale. Investigating tabletop territoriality in digital tabletop workspaces. Technical report, Technical Report 2006-836-29, Department of Computer Science, University of Calgary., Calgary, AB, Canada, 2006.

S.D. Scott, K.D. Grant, and R.L. Mandryk. System guidelines for co-located, collaborative work on a tabletop display. In *Proc. CSCW '03*, pages 159–178, 2003.

S.D. Scott, M.S.T. Carpendale, and S. Habelski. Storage bins: Mobile storage for collaborative tabletop displays. *Computer Graphics and Applications, IEEE*, 25(4):58–65, 2005.

A.M. Seto. Designing discoverable digital tabletop menus for public settings. 2012.

C. Shen, K. Everitt, and K. Ryall. Ubitable: Impromptu face-to-face collaboration on horizontal interactive surfaces. In *UbiComp '03*, pages 281–288. Springer, 2003.

C. Shen, F.D. Vernier, C. Forlines, and M. Ringel. Diamondspin: an extensible toolkit for around-the-table interaction. In *Proc. CHI '04*, pages 167–174. ACM, 2004.

C. Shen, K. Ryall, C. Forlines, A. Esenther, F.D. Vernier, K. Everitt, M. Wu, D. Wigdor, M.R. Morris, M. Hancock, et al. Informing the design of direct-touch tabletops. *Computer Graphics and Applications, IEEE*, 26(5):36–46, 2006.

B. Shneiderman. 1.1 direct manipulation: a step beyond programming languages. *Sparks of Innovation in Human-Computer Interaction*, page 17, 1993.

D.C. Smith, C. Irby, R. Kimball, and E. Harslem. The star user interface: An overview. *Proceedings of the June*, pages 7–10, 1982.

O. Ståhl, A. Wallberg, J. Söderberg, J. Humble, L.E. Fahlén, A. Bullock, and J. Lundberg. Information exploration using the pond. In *Proc. CTS '02*, pages 72–79. ACM, 2002.

D. Stanton and H.R. Neale. The effects of multiple mice on children's talk and interaction. *Journal of Computer Assisted Learning*, 19(2):229–238, 2003.

N.A. Streitz, J. Geißler, T. Holmer, S. Konomi, C. Müller-Tomfelde, W. Reischl, P. Rexroth, P. Seitz, and R. Steinmetz. i-land: an interactive landscape for creativity and innovation. In *Proc. CHI '99*, pages 120–127. ACM, 1999.

N.A. Streitz, P. Tandler, C. Müller-Tomfelde, and S. Konomi. Roomware: Towards the next generation of human-computer: Interaction based on an integrated design of real and virtual worlds. *Human-Computer Interaction in the New Millenium, Addison Wesley*, pages 551–576, 2001.

S. Strothoff, D. Valkov, and K. Hinrichs. Triangle cursor: interactions with objects above the tabletop. In *Proc. ITS '11*, pages 111–119. ACM, 2011.

K. STUART, K. William, and J. BETTY. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a crt. *Ergonomics*, 21(8):601–613, 1978.

A. Tang, M. Tory, B. Po, P. Neumann, and S. Carpendale. Collaborative coupling over tabletop displays. In *Proc. CHI '06*, pages 1181–1190. ACM, 2006.

J.C. Tang. Findings from observational studies of collaborative work. *International Journal of Man-machine studies*, 34(2):143–160, 1991.

Luyten K. Hoven E. Vermeulen, J. and K. (in press) Coninx. Crossing the bridge over norman's gulf of execution: Revealing feedforward's true identity. In *Proc. CHI '13*. ACM, 2013.

S. Voelker, M. Weiss, C. Wacharamanotham, and J. Borchers. Dynamic portals: a lightweight metaphor for fast object transfer on interactive surfaces. In *Proc. ITS '11*, pages 158–161. ACM, 2011.

X. Wang, Y. Ghanam, and F. Maurer. From desktop to tabletop: Migrating the user interface of agileplanner. *Engineering Interactive Systems*, pages 263–270, 2008.

M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, 1991.

D. Wigdor and D. Wixon. *Brave NUI world: designing natural user interfaces for touch and gesture*. Morgan Kaufmann, 2011.

D. Wigdor, G. Perm, K. Ryall, A. Esenther, and C. Shen. Living with a tabletop: Analysis and observations of long term office use of a multi-touch table. In *TABLETOP '07*, pages 60–67. IEEE, 2007.

R. Wimmer and F. Hennecke. Everything is a window: Utilizing the window manager for multi-touch interaction. In *Proc. EICS '10*, 2010.

J.O. Wobbrock, M.R. Morris, and A.D. Wilson. User-defined gestures for surface computing. In *Proc. CHI '09*, pages 1083–1092. ACM, 2009.

P.C. Wright, R.E. Fields, and M.D. Harrison. Analyzing human-computer interaction as distributed cognition: the resources model. *Human-Computer Interaction*, 15(1):1–41, 2000.

C. Wu, Y. Suo, C. Yu, Y. Shi, and Y. Qin. uPlatform: a customizable multi-user windowing system for interactive tabletop. *Human-Computer Interaction. Design and Development Approaches*, pages 507–516, 2011.

T. Yoshikawa, B. Shizuki, and J. Tanaka. Handywidgets: local widgets pulled-out from hands. In *Proc. ITS '12*, pages 197–200. ACM, 2012.

N. Yuill and Y. Rogers. Mechanisms for collaboration: A design and evaluation framework for multi-user interfaces. *ACM Trans. Comput.-Hum. Interact.*, 19(1):1:1–1:25, May 2012.

A. Zanella and S. Greenberg. A single display groupware widget set. In *Western Computer Graphics Symposium 2000*, pages 94305–6004, 2000.

Index

Typeset March 29, 2013