Universität Konstanz
FB Informatik und Informationswissenschaft
Bachelor-Studiengang Information Engineering

Bachelorarbeit

Interaction Concepts for Multi-Touch User Interfaces: Design and Implementation

zur Erlangung des akademischen Grades eines
Bachelor of Science (B.Sc.)

Studienfach:	Information Engineering
Schwerpunkt:	Computer Science
Themengebiet:	Angewandte Informatik

von

Toni Schmidt

Matr.-Nr.:	01/589138
Erstgutachter:	Prof. Dr. Harald Reiterer
Zweitgutachter:	Prof. Dr. Marcel Waldvogel
Einreichung:	05. Dezember 2008

1 Abstract

Interactive tables operated by direct touch (tabletops) are getting more and more common. Such tables offer great opportunities for novel user interfaces that base on direct touch interaction. As part of this thesis we illuminate the particular advantages of direct touch interaction as well as we discuss possible constraints. User interfaces attempting to exploit most benefit from direct touch interaction have to be designed specifically for touch interaction's needs. In contrast, legacy user interfaces are designed for mouse interaction and lack conformance to touch interaction's requirements. Yet, such legacy user interfaces take the largest part of today's user software, being essential in a great number of tasks. Nevertheless, little research has been done in making legacy applications usable on tabletops. For this reason, we designed and implemented interaction techniques empowering users to benefit from direct touch interaction when using legacy (WIMP) user interfaces. We identified three dimensions of interaction tasks in WIMP user interfaces as being essential. Dimension 1 is focused on the tradeoff between selection efficiency and effectiveness for differently sized targets. We introduce ZoomTap, a technique enabling users to rapidly select large targets as well as to select small targets with high precision. Entering text by touch input is another task being essential for using WIMP user interfaces. When using physical keyboards, users benefit from tactile feedback. Virtual keyboards cannot offer such feedback. In dimension 2 we address this issue with the design and the implementation of two alternative text input techniques. Our Qwerty virtual keyboard lets users benefit from its familiarity to a physical keyboard. The second design, Column Typing, supports users by providing an easily recognizable letter history. An informal user study with six participants showed that users benefit from Qwerty's layout and utilize Column Typing's feedback component. For effectively operating WIMP user interfaces, users have to be supplied with techniques for dragging and scrolling as well as for invoking contextual menus. In the third dimension we introduce interaction techniques allowing users to perform these tasks in a natural and ergonomic way. We implemented the designs of all three dimensions in a prototype program, making our techniques actually usable on tabletops. In combination, all interaction techniques form a coherent interaction concept, enabling users to perform all vital parts of WIMP user interfaces while gaining benefit from direct touch interaction.

2 Zusammenfassung

Interaktive, berührungssensitive Tische (Tabletops) verbreiteten sich immer mehr. Durch den Einsatz von direkter Touch-Interaktion auf Tabletops eröffnen sich vielfältige Möglichkeiten für eine natürlichere Gestaltung visueller Benutzeroberflächen. In dieser Arbeit beleuchten wir die Vorteile direkter Touch-Interaktion, genauso wie wir mögliche Einschränkungen behandeln. Da die Eingabe durch Touch-Interaktion besondere Eigenschaften aufweist, müssen Benutzeroberflächen speziell auf deren Bedürfnisse zugeschnitten werden. Die meisten traditionellen Anwendungen sind jedoch auf Maus-Interaktion ausgerichtet. Diese traditionellen Anwendungen sind in einer großen Zahl von Alltagstätigkeiten unverzichtbar. Trotzdem wurde bisher nur wenig Forschungsaufwand für die Entwicklung von Touch-Interaktionstechniken für traditionelle Benutzeroberflächen aufgebracht. Die Zielsetzung dieser Arbeit war, geeignete Interaktionstechniken zu entwerfen und zu implementieren, welche die Vorteile von direkter Touch-Interaktion nutzen, um traditionelle WIMP-Benutzeroberflächen zu bedienen. Hierbei haben wir drei Dimensionen von Interaktionsaufgaben identifiziert, die für WIMP-Nutzerschnittstellen essentiell sind. Dimension 1 konzentriert sich auf die Abwägung zwischen Effektivität und Effizienz bei der Selektion von verschieden großen Zielen. Wir stellen ZoomTap vor, eine neue Interaktionstechnik, die die schnelle Selektion großer Ziele genauso wie die präzise Selektion kleiner Ziele ermöglicht. Ein weiterer essentieller Teil von WIMP-Anwendungen ist die Texteingabe. Hierbei muss das Fehlen des taktilen Feedbacks einer echten Tastatur berücksichtigt werden. In Dimension 2 adressieren wir diese Problemstellung durch den Entwurf und die Implementierung von zwei alternativen Texteingabetechniken. Unsere virtuelle Qwerty Tastatur lässt Nutzer von bestehender Erfahrung mit dem Qwerty-Layout profitieren. Unser zweiter Entwurf, „Column Typing“, bietet explizites Feedback über die zuletzt eingegebenen Buchstaben an. Eine informelle Nutzerstudie mit sechs Teilnehmern zeigte, dass Nutzer sowohl von dem gewohnten Qwerty-Layout profitierten, als auch Column Typing's Feedback nutzen konnten. Um WIMP-Interfaces effektiv nutzen zu können, müssen Techniken für Dragging und Scrolling, so wie zum Aufruf von Kontextmenüs bereitgestellt werden. In der dritten Dimension stellen wir Interaktionstechniken vor, die eine natürliche und ergonomische Durchführung dieser Aufgaben ermöglichen. Alle Entwürfe wurden in einem funktionsfähigen Prototyp implementiert und auf unterschiedlichen Tabletops verwendet. Zusammengefasst bilden unsere Techniken ein konsistentes Interaktionskonzept, das Nutzer, neben der Kontrolle aller wichtiger Bestandteile von WIMP-

Benutzeroberflächen, von den Möglichkeiten direkter Touch-Interaktion profiteren lässt.

Contents

3 Introduction

3.1 Motivation

> *"Interactive, direct-touch digital tables are an emerging form factor with largely immature user interface design" [SRF+06]*

Such interactive, digital tables operated by direct touch input are commonly called tabletops. Tabletops provide benefits over traditional displays in various ways [SRF+06]:

1. Tabletops unify display and input surface and thus are direct input devices [Hin02]. Direct input devices allow interface elements being manipulated by directly touching them. Such ability aids users in accomplishing tasks. In detail, direct touch interaction reduces the "gulf of execution" [HHN85], "the gap between a user's goals and the means to execute these goals" [JGH+08]. With direct touch interaction this gap is considerably lower than e.g. with traditional mouse interaction.

   For example, a user attempting to hit a button with a mouse would have to grasp the mouse, detect the cursor on the screen, move the cursor to the button and click the left mouse button. Thereby the user also has to map two different planes (vertical display and horizontal mouse base) and different movement speeds (mouse's and cursor's). With direct touch interaction the user would merely have to touch the button with a finger.

2. Besides having an interactive display surface, tabletops benefit from being tables. In order to communicate, discuss and collaborate, people tend to gather around tables [SRF+06]. Tables are found in "homes, offices, command-and-control centers, cafés, design centers, showrooms, waiting areas, and entertainment centers" [SRF+06] and, amongst others, have the purpose of brining people together. This can be exploited:

   > *"... a horizontal tabletop surface provides opportunities for building and enhancing colocated collaborative environments" [SRF+06]*

3. Tabletops are usually larger than desktop displays (the tabletop prototypes at the

Human-Computer Interaction Group of the University of Konstanz have a screen diagonal of 34" [RSF+08] and 70", Microsoft Surface is sized 30" [Mic08], the MERL DiamondTouch is sized approx. 40" [FWSB07]). This size "can positively influence working styles and group dynamics", be employed as "external physical memory" and can "serve as an external cognitive medium for new forms of visual representation" [SRF+06].

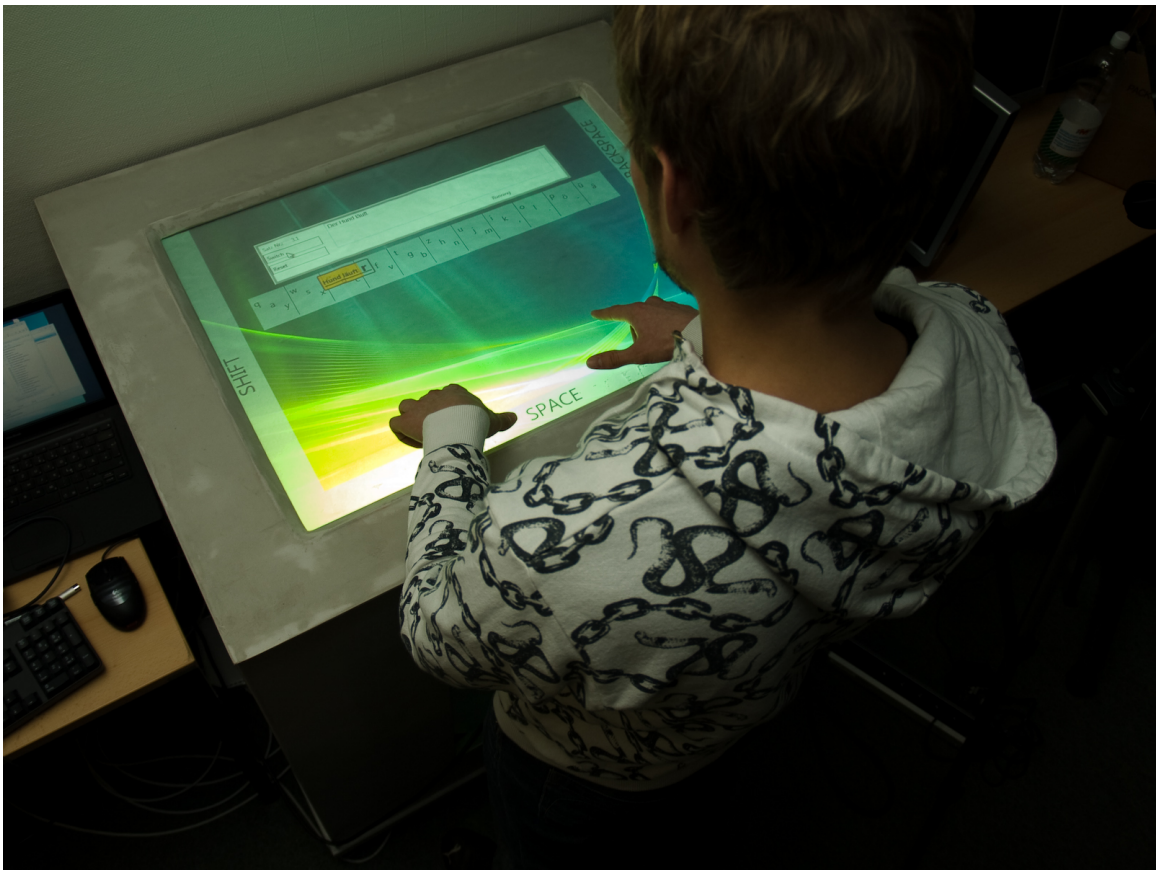

Figure 3.1: The tabletop prototype at the Human-Computer Interaction Group of the University of Konstanz. The user is currently entering text using our Column Typing technique (see section 5.2.2.3)

Tabletops provide opportunities for fascinating novel user interfaces benefiting from direct touch interaction. However, touch interaction techniques differ from traditional mouse interaction considerably in matters of motor skill required:

- User are enabled to select interface elements directly with their fingers rather than by positioning the mouse.
- User interfaces may exploit the human's ability to use both hands parallel. With mouse interaction, users act purely unimanual.

- Interfaces attempting to gain benefit from transferring real-world behavior to interaction techniques might require specific motor skills. These skills may already be existing, yet it is unclear to what extent they can be transferred into the touch interaction domain.

Hence, insight into the human's motor skill of controlling body parts relevant for touch interaction is required. In section 4 we deal with the topic of direct touch interaction as well as we provide insight into issues related to bimanual input.

As the tabletop domain emerges, more and more user interfaces designed specifiably for touch input are likely to be developed. However, applications whose interface is designed for traditional mouse interaction techniques still take the largest part of existing user software. Supporting such existing applications is essential for making a diverse application landscape in the tabletop domain possible:

> *"Still, legacy applications are widely deployed and many are indispensable for real-world tasks. A digital tabletop environment must therefore address issues related to using preexisting applications on a horizontal workspace" [SRF+06]*

Most legacy applications are WIMP (i.e. windows, icons, menus and pointing device) user interfaces. Current operating systems (Microsoft Windows, Apple MacOS, Linux GUIs) follow the WIMP paradigma. Consequently, the majority of applications running on these operating systems are WIMP user interfaces, too.

Yet, little research has been done in the field of transferring WIMP user interfaces into the tabletop domain. The only existing compound framework developed for this purpose, Fluid DTMouse [ER06], suffers several drawbacks:

- Fluid DTMouse provides no technique of entering text. Yet, "text entry is a vital part of day-to-day computing familiar to most people" [HHCC07].

- Fluid DTMouse requires users to learn two modes for placing the cursor. In normal mode the cursor is set directly beneath the finger. In precision mode the cursor is set in the middle of two fingers. Hence, users have to rethink from an absolute technique to a relative technique when changing modes.

- Fluid DTMouse's dragging technique depends on the mode for placing the cursor. In normal mode a dragging is initiated as soon as a single finger is placed on the the display. In precision mode the user needs to tap with a third finger in between the two already placed on the display. This inconsistency is likely to irritate users. Moreover, starting a dragging directly at the moment a finger touches the display is not conform to the standard VDI/VDE 3850 Blatt 3 - User-friendly design of

useware for machines - Design of dialogues for touchscreens [Dah06].

- Fluid DTMouse provides no visual clues about the way of performing its techniques to the user. Thus, users are likely to need extensive training time until they are able to use Fluid DTMouse "blindly".

For these reasons we designed and implemented novel interaction techniques for controlling WIMP user interfaces by touch input. In detail, we designed and implemented techniques for the following tasks:

- Selections: Selections are one of the most basic tasks WIMP user interfaces require the user to perform. We review existing techniques and present ZoomTap, a technique allowing user to magnify a small screen portion for enlarging a target and thus enhancing selection precision. The topic of selection tasks is dealt with in section 5.1.

- Text input: Entering text is essential for operating WIMP user interfaces:

  > *"Text entry is one of the most frequent actions we undertake when working on desktop computers. Entering text is necessary for activities that require elaborate text compositions such as coding programs, authoring articles, or writing emails, as well as in situations that demand taking notes,typing in commands, or annotating content" [HHCC07]*

  We designed, implemented and evaluated two alternative text input techniques:

  - Our Qwerty soft keyboard provides users a text input environment they are accustomed to.

  - Our Column Typing design supports users in reducing typing errors by providing explicit feedback.

  A qualitative user study with six participants showed that users prefer the Qwerty layout they are accustomed to but profit from Column Typing's feedback in matters of error recognition (see section 5.2.3).

- Dragging, Scrolling and Contextual Menus: These three aspects are further parts essential for controlling WIMP user interfaces. We designed and implemented interaction techniques allowing users to perform dragging and scrolling tasks as well as invoking contextual menus.

Summarized, our interaction techniques enable users to control all essential aspects of WIMP user interfaces. Selection precision for small targets is enhanced while main-

taining high efficiency for selecting large targets. Text can be entered aided by several feedback components compensating the lack of a physical keyboard's tactile feedback. Dragging and scrolling tasks can be performed in a natural way. Contextual menus are invokable with only a small and ergonomic finger motion, supported by visual clues. Our techniques make WIMP user interfaces actually *usable* by touch input, overcoming mouse interaction's hurdles and letting users benefit from the advantages of tabletops and direct touch input.

3.2 Terminology

Tap
A finger's touch immediately followed by the lifting of a finger at one position. During this activity the finger must not move.

Bimanual
Both hands are used simultaneously for the same task.

Multi-touch
More than one finger can be identified and processed by hardware device and user interface.

Tabletop
A table whose surface is an interactive display, operated by direct touch input. Display and input surface are assumed to be unified.

"touch input", "touch interaction" or "touch-operated"
In this thesis, theses terms imply the possibility of multi-touch input by means of both the hardware device and the interaction technique.

Virtual keyboard
A virtual (or soft) keyboard is an on-screen keyboard, allowing entering letters by selecting them from the keyboard map.

3.3 Structure of the Thesis

This thesis is structured into two main parts.

In the chapter "The effectiveness of touch interaction" (4) we give an overview about topics relevant when designing touch interaction techniques. Such topics are the human's

motor abilities to control its hands, fingers and arms (see 4.1), bimanual interaction (see 4.2) factors from the theories of embodiment (see 4.3) and issues concerning direct and absolute interaction (see 4.3.2). We provide a more detailed insight in the human's ability of parallel controlling both hands (see 4.4). Several relevant studies are summarized in order to provide empirical evidence for the described issues (see sections 4.2, 4.4.2 and 4.3.3).

In the chapter "Designing touch interaction techniques for WIMP" (5) we present interaction techniques for controlling WIMP user interfaces by touch input. This chapter is structured into three dimensions:

- Dimension 1 is concerned with interaction techniques for performing selection tasks (see 5.1). In this section we present ZoomTap (see 5.1.3), a novel technique for precisely selecting small targets while still being able to rapidly select large targets. Prior to our novel design, we sum up existing touch interaction techniques for performing selection tasks (see 5.1.2).

- In Dimension 2 we deal with text input in the domain of tabletop displays (see 5.2). We present two alternative soft keyboard designs for text entry by touch input (see 5.2.2.2 and 5.2.2.3). An qualitative user study with six participants was run in order to compare the two designs and get insight into the users's subjective ratings and preferences (see 5.2.3).

- Dimension 3 is about interaction techniques for further tasks essential for WIMP user interfaces (see 5.3). We present interaction techniques for performing Dragging and Scrolling tasks (see 5.3.2.2 and 5.3.2.3). Furthermore, we present an interaction technique for triggering contextual menus (see 5.3.2.4).

4 The effectiveness of touch interaction

> *"Direct physical interaction with the world is a key constituting factor of cognitive development during childhood"* [KHT06]

Touching things and working with our hands is one of the first actions a human learns as a child.

We learn through doing, express ourselves through gestures, understand the context of a problem better through epistemic actions [KHT06], write, type, grasp and do numerous more actions involving our hands and fingers. Thus, the human's abilities to control arms, hands and fingers is highly sophisticated.

When reviewing and designing interaction techniques for touch input, insight into these abilities is essential, since touch input utilizes fingers (and consequently also hands and arms) as main input "device".

The first aspect to pay respect to when exploring the human's ability to control its hands is the human motor system. Motor abilities define how fast and precise the hands can be moved. For example, grasping an object is a basic task depending on motor abilities.

4.1 The human motor system

When attempting to understand the properties of touch interaction, the human motor system that is responsible for controlling coordinated movements of arms, hands and fingers is the first part we look at. Knowledge about this motor system is crucial, since touch interaction requires hitting targets with a finger. Hitting a spot with a finger is a combination of movements in the arm, the hand and the finger.

A model describing the motor process of hitting a spot with a finger was provided by Meyer et al. in 1988 [MAK+88]. This optimized initial impulse motor control model describes the task of hitting a spot with a finger as follows:

1. A initial high velocity arm movement is performed towards the target. If the target is hit with this movement, the task is complete. In figure 4.1, three possible processes of this movement are shown by the drawn though lines. The leftmost undershoots the target, the middle hits the target, the rightmost overshoots the target.

2. Otherwise, a submovement towards the target is performed. Note that this submovement begins before the finger actually makes contact. The submovement is of finer granularity and of lower speed than the initial high velocity movement. In figure 4.1 two possible submovements are shown by the dashed lines.

3. This process is continued until the target is reached

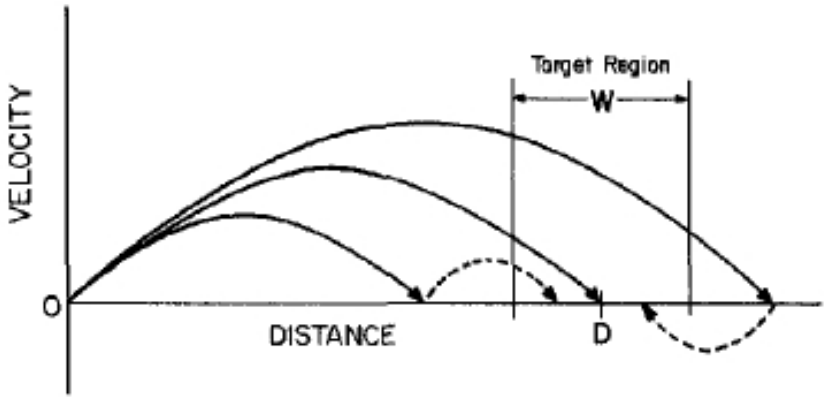

Figure 4.1: Possible sequences of movements towards a target [MAK+88]. The horizontal axis indicates a movement distance, the vertical axis velocity. The drawn through lines indicate three possible initial high velocity movements towards the target. The dashed lines indicate two possible corrective submovements.

The optimized initial impulse motor control model gives clue about the motor process used for hitting a target with a finger. The process of hitting a target is of direct relevance for the domain of touch interaction techniques. With touch interaction we use fingers to touch interface elements. Hence, interface elements we want to touch need to be hit by the finger.

We can learn from Meyer's model that large targets are hit requiring less (or even none) corrective submovments. Since submovements are of lower speed than the initial high velocity movement, they raise the time required for hitting an element. Summarized this means that small targets need more time to be hit than large targets. With ZoomTap, our interaction technique for performing precise selections (see section 5.1.3), we utilize a local zoom to enlarge targets and thus lower hitting times. The Qwerty soft keyboard we designed, implemented and evaluated as part of this thesis (see section 5.2.2.2), consists

of key buttons considerably larger (they are sized 20mm x 20mm) than the minimal effectively hittable size of approx. 10mm x 10mm [VB07]. Hence, users are able to hit these keys faster than smaller keys, even if such smaller keys could be hit effectively.

4.2 Bimanual interaction

A further part to consider when reviewing the human's motor abilities for controlling arms, hands and fingers is the ability of bimanual actions. Bimanual actions involve both hands at the same time. If a person benefits from bimanual actions in the real world, a transfer to touch interaction techniques could also aid the user in interacting with a system.

Bimanual actions are part of our everyday life:

> *"Every day we turn pages with one hand while writing with the other. We steer our car with one hand while changing gears with the other. We hold a ruler or drafting machine with one hand and use a pencil in the other." [Bux]*

A study to prove the hypothesis that bimanual interaction leads to an improved performance in accomplishing a task was conducted by Buxton and Myers in 1986 [BM86]. They ran two experiments to investigate the differences between uni- and bimanual interaction.

The first experiment consisted of a compound positioning and scaling task. A rectangle had to be fit into another one. Thereby, size and position of the original rectangle ad to be altered. The size was scaled around the rectangle's center. The input took place via a puck on a graphical tablet and a treadmill-like slider, each controlled by one hand.

The first experiment was to show a possible correlation between total trial time and the time in which the hands were engaged in parallel activity. Training for each task was done independently. Moreover, subjects were not motivated to use bimanual input in the actual experiment. Fourteen subjects participated. Independant variable was the interaction technique. Dependant variables were trial time and percentage of bimanual activity with respect to total trial time.

Results showed that the simultaneous usage of both hands was generally used. "The most important result was that all but one subject used both hands simultaneously in performing the task." In average, the subjects were involved in parallel activity in 40,9% of the total time.

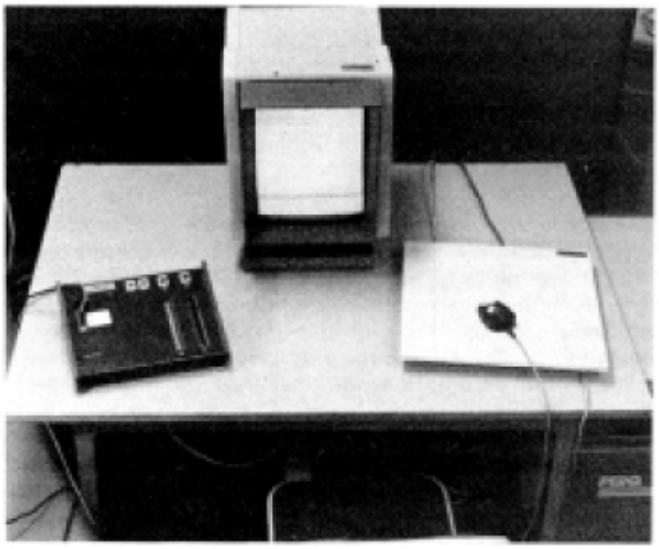

Figure 4.2: The experimental environment [BM86]

The second experiment was to show possible improvements in performance in task completition time when using both hands parallel. Subjects had to select a word in a text document. The document was divided into three columns, so that a subject had to select either the leftmost, the middle or the rightmost word (see figure 4.2). Interaction consisted of scrolling in the document and selecting the word.

24 subjects participated in the second experiment. For their experiment, the authors have split subjects equally into two groups. One group only used their right hand to do selection as well as navigation. The other group used their right hands to select and their left hands to navigate in the document. Each group consisted to equal parts of experts and novices with respect to mouse usage. Input devices were a graphical puck on a tablet (right hand) and a touch-sensitive slider (left hand).

Note that the input device used in this experiment was a puck that absolutely controlled the cursor. In contrast, a mouse is a relative input device (see section 4.3.2 for more information about absolute and relative input devices). However, the motor skill required for using a puck is principally the same as for a mouse. Hence, experts with respect to mouse usage are likely able to transfer their experience to the usage of a puck. Yet Buxton and Myers do not address this issue.

Results showed that two-handed interaction outperforms one-handed interaction in matters of trial completition time.

- *"Experts: the two-handed group outperformed the one handed group by 15%.*

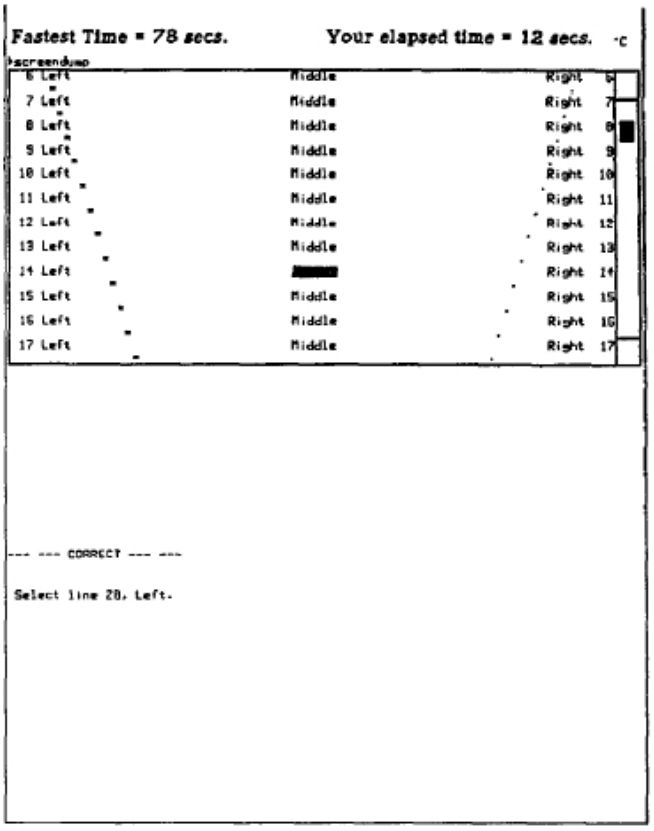

Figure 4.3: The screen layout of experiment two. Users had to select a word from the screen's upper area. A word was defined by its row (1 to 60) and its column (either Left, Middle or Right). In the lower section, subjects were presented feedback for their last trial ("CORRECT" in this example) and the next trial's description ("Selece line 28, Left" in this case) [BM86]

- *Novices: the two-handed group outperformed the one handed group by 25%." [BM86]*

Furthermore, Buxton and Myers showed that novices can decrease their performance gap to experts when using two-handed input.

- *"One-handed: experts outperformed novices by 85% (p =0.05).*
- *Two-handed: experts outperformed novices by only 32% (p= 0.02)*
- *Experts using one hand outperformed the novices using two hands by only 12%, and this difference has no statistical significance. "[BM86]*

This study proves the naturalness and efficiency of bimanual input in the context of the carried out tasks. The first experiment showed that bimanual input was highly

used although subjects were not motivated to use both hands at the same time. The second experiment showed that bimanual execution resulted in a higher trial time for the concrete task.

This means that bimanual interaction can be used to enhance a system's performance without stressing the user too much. That knowledge is vital for the design of bimanual interaction techniques. However, Buxton and Myers utilize the hands in a narrow manner. In their study both hands are part of the same task and work on the same item. Yet this is not the only kind in which the two hands can be used. For example, the hands could be appointed to independent tasks. Buxton and Myers only concentrate on one interaction style and therefore a need for additional studies exists. The section "Issues on bimanual input" (see 4.4) deals with such studies and provides a fundament on which actual bimanual interaction techniques can be designed.

4.3 Reality-Based Interaction and Embodiment

Besides the human's motor abilities, knowledge about the theories of embodiment can be useful for designing touch interaction techniques. A central thesis of embodiment is that "Our physical bodies play a great role in shaping human experience in the world, understanding of the world, and interactions in the world." [KHT06]. We use our body not only as a manual tool for activities like grasping but rather use our body as an extension of our brain. For example, bodily activities support fundamental tasks like learning, understanding and communicating [KHT06].

Of special interest is the behavior of how our hands are integrated in such tasks. In touch interaction we use our hands to directly manipulate interface elements. In real-life we also manipulate objects directly with our hands. Hence, letting the user transfer real-life behavior to touch interaction is only consequent.

4.3.1 Gestures

Gestures are of particular interest, since "Gesture plays a role in pre-linguistic communication for babies as well as aids cognition and fully linguistic communication for adults" [KHT06] Moreover, "gesturing has been shown to lighten cognitive load for both adults and children"[KHT06]. Transferring the human's natural ability to gesture to touch input concepts has various advantages.

- First, input gestures can be used in a metaphoric way in order to take advantage of pre-existing knowledge.

Jacob et al. identified that "people have common sense knowledge about the physical world" [JGH+08] and refer the term Naïve Physics to such knowledge. For example, if an object is thrown, the thrower has implicit knowledge about the path of flight and the possible point of impact. If transferred to the touch interaction domain, physical properties could be attached to virtual objects, making them behave similarly to physical objects. Since the virtual objects would behave similar to physical objects a user could utilize existing knowledge about the physical world and hence be better able to control the virtual environment.

- Second, barring the user from gesturing might decrease the user's performance: "systems that constrain gestural abilities are likely to hinder the user's thinking and communication" [KHT06]. Traditional mouse/keyboard interaction techniques bind the user's hands to the input devices and therefore prevents the user from gesturing freely. I.e. a user has to continuously move a mouse to position the cursor, and thus the mouse has to be grasped all the time. In touch interaction the user directly manipulates an interface element. Hence, such an element can be chosen by direct touch without going the detour over a relative and indirect input device. Thus the user's hands are less bound to manipulating the system.

- Third, humans are in possession of a motor memory. "We are able to sense, store and recall our own muscular effort, body position and movement to build skill" [KHT06]. Since gestures are actions that involve muscular effort, they can be stored in our motor memory. When a user recalls such an action, the motor memory helps through supplying the actual movement pattern without having the user to think about the gesture's performance pattern.

4.3.2 Direct and Absolute Interaction

Before illuminating benefits of direct and absolute interaction, we will briefly introduce the characteristics of direct and absolute input devices. According to Hinckley, an absolute input device is "position sensing" [Hin02]. This means that such a device enables identifying its position in a definite reference frame. A reference frame requires a fixed origin and fixed dimensions. The issue of requiring a fixed origin for absolute input devices was referred to as the "nulling problem" by Buxton [Bux83]. Hence, an absolute input device needs to be calibrated. For example, a laserpointer whose spot on a screen is used to position a cursor, is an absolute input device [KBSR07]. In this case the screen has to be calibrated in advance, in order to let the tracking system know where the screen's origin is and what dimensions the screen has. Absolute input devices have the advantage of providing users a fixed mapping between input and output areas.

In contrary to absolute devices, relative input devices are characterized as being "mo-

tion sensing" [Hin02]. This means that when moving a relative device, the movement difference is reported by the device. However, the device is not capable of identifying its position in a reference frame. The mouse is the most prominent example for a relative input device. Relative input devices have the advantage of not requiring any calibration. Note that differing between relative and absolute devices is somewhat indistinct. For example, a mouse can also be used as an absolute device, if provided an origin once and then is never lifted or moved beyond the screen's bounds.

Beyond differing between absolute and relative input, input devices can be classified as being direct or indirect. A direct input device is described as having a "unified input and display surface" [Hin02]. Normally direct input devices, such as touchscreens, are also absolute. However, such absoluteness is not a prerequisite for direct input devices. With direct touch input, users are able to control interface elements by directly touching them.

This ability to directly controlling a system when using touch input supports the transfer of real world behavior to touch interaction. In touch interaction the user is able to work on and manipulate the desired object directly by placing her fingers on the object. This close connectivity of system and user can be benefited from in various ways:

- One of these possible benefits is that the learning and understanding of complex relations is supported by physically working on the problem: "Humans learn about the world and its properties by interacting with it" [KHT06]. When working with a direct touch interface users can be enabled to manipulate interface elements by "grasping" them and thus can interact with such elements similar as if they were physical objects.

- Another relevant subject is the manner how we manipulate environments in order to accomplish a task. "Body engagement with physical and virtual environments constitutes another important aspect of cognitive work" [KHT06]. Concretely, epistemic actions are of interest. Epistemic actions describe the behavior of "manipulating artifacts to better understand the task's context" [KHT06]. For example, rotating a Tetris block in all ways in order to see where it could fit is an epistemic action. Transferring this idea to a touch interaction scenario, a user could benefit from the existing familiarity with epistemic activities. Given the opportunity to directly manipulate objects with her hands and immediately see the effects triggered by different alternatives, a user's understanding of a system will likely be improved.

- A further argument for direct input is motivated by the optimized initial impulse motor control model [MAK+88]. As explained in section 4.1, this model proposes that humans initially perform a high-speed and imprecise movement towards the target. If the target is hit, movement stops. If the target is missed, small corrective

movements are iteratively performed towards the target. When applying mouse usage to this model, the user not only has to do the corrective movements needed to compensate the initial imprecise high-velocity movement, but also has to map the corrective movements to the Control-Display ratio induced by relative input. In touch interaction, motor and display space are the same. Hence, the CD-ratio is one and no mapping has to be done. This means that a user is more likely to hit a target with less corrective movements. This supports effectiveness as well as efficiency of a system.

Summarized, direct touch interaction reduces the "gulf of execution" [HHN85]. The gulf of execution can be described as "the gap between a user's goals and the means to execute these goals" [JGH+08]. Reducing this gulf is very helpful for better integrating a user into a system. It can be achieved through a combination of a user's previously described knowledge of our world and a touch interaction concept that pays respect to such knowledge.

However, there are tradeoffs to be looked at. Interaction techniques based completely upon metaphors taken from real world activities might not always enhance a system's performance. Jacob et al. identify six dimensions in which reality-based interaction needs to be checked against [JGH+08]:

- Expressive Power: Important functionality must not be cut off by reality.
- Efficiency: I.e. an expert might prefer pure a command-line system.
- Versatility: Different application domains require different levels of versatility.
- Ergonomics: User should not experience injuries or fatigue when using the system.
- Accessibility: "Realistic actions may not be ideal for the disabled" [JGH+08].
- Practicality: Cost, space, size, etc have also to be taken into account.

4.3.3 Experimental evaluation of relative vs. absolute input

One more thing to be considered is the performance of absolute interaction through direct touch versus relative interaction via a computer mouse. Touch-operated systems have to enable users to perform his tasks efficiently.

Efficiency in this context means, that the user is able to perform his tasks as least as quick as he would be able to with a traditional input device. Inefficient input devices generally narrow the possible application areas of systems. As a consequence, this holds

also and particularly for touch-operated systems.

A study treating this topic was carried out by Forlines et al. in 2007 [FWSB07]. Two experiments are part of this study. Both were performed on a multitouch-capable DiamondTouch [DL01] tabletop display.

In the first experiment, the subjects had to select a rectangle and then move this rectangle to a target location. This task was either carried out by direct touch or by mouse. In the first case, a subject had to place its finger in the rectangle and then move the finger to the target. In the second case, the same task was done using the mouse cursor.

Twelve subjects participated, the design was within-subjects with the independent variables being input device, target width and target distance.

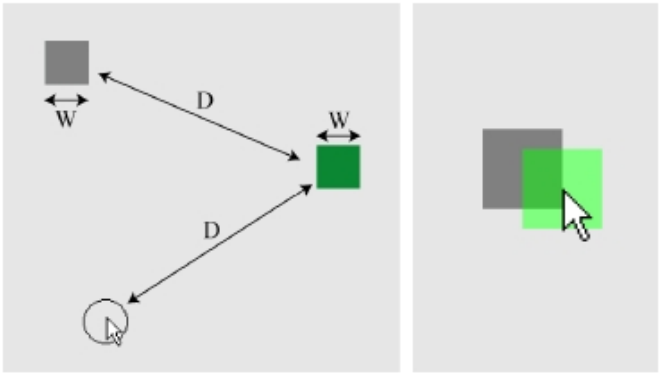

Figure 4.4: "Task details. (left) Round home location, the target on the right, and the dock on the left. (right) After selecting the target, participants dragged it to the dock" [FWSB07]

Results show that in the selection part, touch interaction was significantly faster than mouse interaction (1,01s to 1,19s). However, the docking task was performed significantly faster by the mouse (0,92s to 1,09s). Average trial times were almost identical (2,12s to 2,13s). The error rate in the selection part was significantly higher when touch interaction was used (8,5% to 4,1%).

The second experiment was also conducted by twelve participants and designed within-subjects. Independent variable was input device. Again two rectangles had to be fit into each other. However, the source rectangle also had to be scaled so that it matched the target rectangle. To accomplish this, bimanual interaction was used. The subjects had to grab the source rectangle at opposing corners. Now the rectangle would scale if both corners were moved away from each other. A movement was accomplished by parallel moving both corners. One group of subjects used direct touch. In this case, one finger controlled one corner in absolute and direct manner. The other group used two mice,

whereas one cursor controlled on corner.

Selection times were significantly faster with touch interaction (1,45s to 2,43s). Docking times were almost identical (2,1s to 2,07s).

A further finding of this study is that selection error rates increased significantly with rising target distance. This matter has to be considered when dealing with interaction techniques for touch input on tabletop displays.

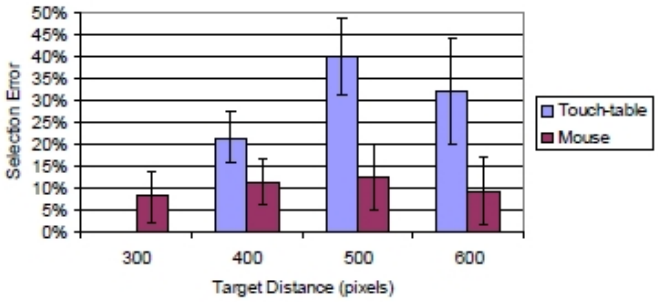

Figure 4.5: "Selection error rates for each target distance and input device for targets with a width of 16 pixels." [FWSB07]

These results show that absolute interaction is not necessarily faster than relative interaction. Advantages in touch interaction showed up when the subjects had to select two corners simultaneously. Here, the subjects benefited from their ability to quickly grasp an item with two hands at the same time. In the case of mouse interaction, the subjects had to map the relative movements of both mice to screen coordinates which caused a significant slowdown. Error rates show that touch selection is more likely to fail than mouse selection. This fact has to be taken into account in the design of future concepts.

4.4 Issues on bimanual input

In their study on bimanual input, Buxton and Myers showed that bimanual action is both natural and efficient. However, as stated in the section 4.2, their tests did not vary in matters of the amount of stress put on each hand. In their design the right hand was utilized for precise selection and positioning tasks. The left hand was stressed less, since it only had to roughly control a rectangle's size and to scroll to a line in a document respectively. The results of Buxton and Myers's experiment indicate that such a division of stress supports the user in accomplishing a task. Yet, it is arguable if this particular division is the only one applicable. For example, a violin player uses each hand in a very complicated manner [Gui87]. Buxton and Myer's study doesn't grant any insight into such a division of burden on each hand.

4.4.1 The Kinematic Chain Theory

In order to approach the topic of bimanual interaction systematically, more general patterns than those provided by Buxton and Myers are needed. In this context, the Kinematic Chain Theory by Guiard can be placed [Gui87]. The Kinematic Chain Theory looks at the two hands as motors creating motion. Moreover, "the two manual motors cooperate with one another as if they were assembled in series, thereby forming a kinematic chain" [Gui87].

When taking a closer look at this model, a classification of human manual activities is helpful. Classification supports abstracting between groups of activities stressing the hands on different levels in matters of bimanual motor activity. Guiard puts human motor activities into three categories.

First there are unimanual asymmetric activities. Examples for such activities are handwriting or dart throwing. These unimanual activities are nevertheless called asymmetric, since "it is impossible to demonstrate that the other hand plays no role" [Gui87] in performing the activity. For example, a study has shown that "writing speed of adults is reduced by some 20% when instructions prevent the non-preferred hand from manipulating the page." [Gui87] Second, "there are activities that must be termed bimanual and symmetric, as the two hands play essentially the same role" [Gui87]. Thereby the hands either work in phase or out of phase. In-phase activities would be rope skipping or weight lifting. An example for an out-of-phase symmetric activity is rope climbing.

Finally, there are activities that are "markedly asymmetric but bimanual" [Gui87]. Those activities employ each hand in different manner. The violin player from the previous example uses such an asymmetric and bimanual division of labor.

When considering bimanual touch input, all three kinds of human manual activities are of interest. An unimanual task could be the selection of an object with one finger. A symmetric bimanual action can be thought of as a zoom gesture in which both hands are moved symmetrically towards each other. However, the most intricate part with respect to motor activity is asymmetric bimanual action. The violin player has to exercise for years until she reaches the skill of coordinating both hands precisely enough to master the instrument. Yet, hammering with one hand while holding the nail with the other hardly needs practice. Both activities are clearly asymmetric and bimanual but obviously require different levels of motor skill to perform. When designing bimanual touch interaction concepts, it is desirable to benefit from already existent motor behavior patterns rather than requiring the user to learn complex motor motions.

Guiard provides us with such patterns. In the following explanations, a dominant right hand is assumed.

4.4.1.0.1 "Right-to-Left Spatial Reference in Manual Motion" [Gui87] First, Guiard identifies that "motion of the right hand typically finds its spatial references in the results of motion of the left hand". This means that the left hand sets the frame of reference in which the right hand operates. "For example, in sewing, no accurate activity could be performed by the hand carrying the needle if the fabric were not kept steady" [Gui87]. Another illustration of this spatial dependency was shown by a hand-writing experiment. Subjects were asked to write down a dictated text. The writing was both recorded on paper and on the table. Results showed that the text on the paper was aligned normally. However, the text recorded on the surface of the table was highly deformed. This happened due to the fact that the left hand constantly realigned the sheet of paper in order to support the right hand's writing.

Transferring this pattern to touch interaction, the left hand should support the right hand in terms of supplying the appropriate frame of reference. Consider a painting task in which the right hand has to draw a figure into a window using touch input. In this case, the left hand should be able to align the window. So the user would be able to define his own preferred frame of reference for the painting task and hence wouldn't have to move his right hand in uncomfortable manner.

4.4.1.0.2 "Left-Right Contrast in the Spatial-Temporal Scale of Motion" [Gui87] Second, Guiard states that the left hand's granularity of action is smaller than the right hand's. That means that the left hand can carry out tasks less precisely and slower than the right hand is able to. For example, a painter uses her right hand to perform fine strokes on the canvas while the left hand holds the palette.

Transferring this into the touch interaction domain, the right hand should be assigned tasks of fine granularity. For example, precise selections should be done by the right hand. The left hand should analogically perform tasks of coarser granularity, like positioning a window.

4.4.1.0.3 "Left-Hand Precedence in Action" [Gui87] Third, "the left hand's contribution to current action starts earlier than that of the right" [Gui87]. This means that an action is initiated by the left hand while the right hand follows in action. This pattern is a logical result of the two previous principles. Since the left hand sets the spatial reference, this reference has obviously to be set before the right hand starts acting. Furthermore, due to left hand's lower granularity, the left hand is likely to need more time to perform its action and hence has to start earlier.

4.4.1.1 Asymmetric and dependent motor behavior

Guiard concentrates actions behaving accordingly to these three models intto the group of asymmetric and dependent activities. The two hands still act asymmetric but they support each other in order to accomplish a single task. Contrasting the violin player's complex asymmetric independent motions, asymmetric dependent activities are more appropriate for designing touch interaction concepts. Such concepts are likely to be performable without much practice since they base on everyday motions and do not require the user to learn new complex motion patterns.

4.4.2 Experimental evaluation of bimanual interaction

A study which is related to Guiard's findings was run by Kabbash et al. in 1994[KBS94]. The study was to show differences in user performance when putting different levels of stress on each hand. Twelve subjects participated in this study. The design was within-subjects.

Subjects had to draw lines between points. A line was drawn by a drag-and-drop procedure from a start point to an end point. Furthermore, the line's color had to be chosen through a menu. The same color never appeared twice in a row. The menu was moveable by dragging its header. Input was done via standard computer mice. The experiment was divided into four conditions, each consisting of a different interaction technique.

- "Right-tearoff Menu" terms standard unimanual mouse input. The mouse controlled a cursor. This single cursor was responsible for both drawing the line and selecting the color.

- "Left-tearoff Menu" supplied the subject with two mice. Each mouse controlled one cursor. Each cursor had exactly the same functionality. Hence, the subjects had to split the effort of both hands on themselves without guidance through an interaction concept.

- In "Palette Menu" the left hand's mouse only controlled the menu position. The right hand was responsible for drawing the lines and selecting colors.

- The last condition is "Toolglas Menu". Hereby color selection is done by moving the menu over the starting point of a line. Now, a color was selected by clicking "through" the menu. The corresponding color field of the menu had to be over the starting point. With this technique, color selection and the start of the drawing were done during the same step of interaction. The toolglas concept was presented by Bier et al. in 1993 [BSP+93].

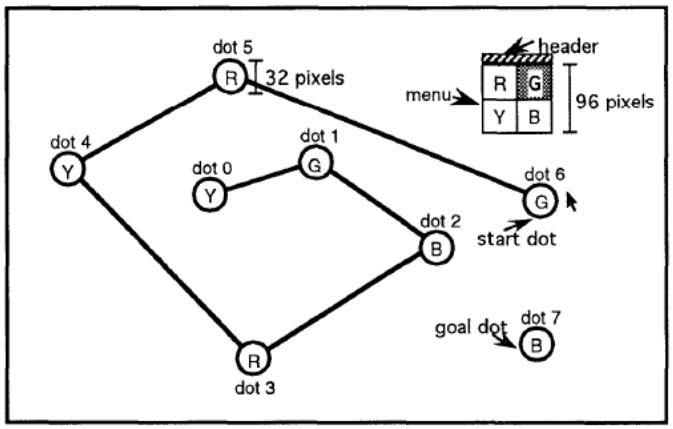

Figure 4.6: "Experimental Task. Having completed six dots, the subject must draw a line segment from dot 6 to dot 7, first selecting the color "blue" from the menu. The menu could be repositioned by clicking and dragging its header" [KBS94]

4.4.2.1 Relation to Guiard's principles

Bringing these conditions to Guiard's paradigms, a partly connection can be identified.

"Left-tearoff Menu" is closely related to asymmetric independent motor action since the user has the same power on each hand. However, the user could still adapt her motor labor in such way that an asymmetric dependent division would be induced. For example, nothing would hinder the user from using her left hand in the same way as in "Palette Menu", which clearly divides motor labor in an asymmetric dependent way. However, the study brings knowledge about how important user guidance with respect to bimanual action is. "Left-tearoff Menu" supplies the user hardly any guidance, whereas "Palette Menu" and "Toolglas Menu" limit the left hand's possibilities. Next, "Toolglas Menu" and "Palette Menu" also stress both hands on different levels.

"Toolglas Menu" was the condition in which motor labor was most divided in an asymmetric and dependant way. The menu controlled by the left hand has to be controlled much coarser than the right hand's cursor. The menu also set the spatial frame of reference since it had to be placed over the starting point of a line. Furthermore, the menu had to be placed over the starting point before the actual drawing, since there could be no drawing without color choosing.

"Palette Menu" lacks the supply of a spatial reference. The menu was not translucent and so had to be kept apart from the starting point.

4.4.2.2 Experiment Results

"Toolglas Menu" resulted in the fastest mean trial completion time (2,43 seconds). This was significantly faster than "Palette Menu" (2,90s), "Left-tearoff Menu" (2,96s) and "Right-tearoff Menu" (2,89s). In average left-hand use referring to the total time, "Toolglas Menu" (83%) also was significantly better than "Palette Menu" (47%) and "Left-tearoff Menu" (49%).

This can be interpreted as an argument for Guiard's theory. However, since color selection and the beginning of the drawing were one single action in "Toolglas Menu", it is questionable how much of this technique's temporal advantage was actually caused by the asymmetric dependent division of motor labor. Yet, the finding that "Toolglas Menu" employed much more left-hand use is speaking for Guiard's paradigms. Subjects obviously found it natural to use their left hand as a direct support in matters of the Kinematic Chain Theory. (Noch zu schwammig)

Kabbash et al. also measured sequencing errors and waiting times during a trial. A sequencing error was logged if a subject did an error in the order of execution. For example, trying to draw a line before choosing the color was a sequencing error. In case of this study, sequencing errors give information about how well users are able to divide their manual labor to each hand in order to accomplish the task. Results show that "Left-tearoff Menu"(4,31%) had a significantly higher error rate than "Toolglas Menu"(1,04%), "Palette Menu"(1,29%) and "Right-tearoff Menu"(0.789%). Hence, a guidance of the user with respect to Guiard's models resulted in a much lower error rate.

Analysis of waiting times during trials gives information about how much a user needs to think in order to correctly control each hand. Waiting times recorded in this study were the time after the drawing of a line and the time after the selection of a color. "Left-tearoff Menu" resulted in 0.489s cumulated average waiting time during each trial. "Toolglas Menu" had significantly less waiting time (0.037s). That shows that a division of motor labor according to the Guiard demands less thinking from the user to coordinate her hand movements.

4.4.2.3 Discussion

A further issue with this study is that input is performed relatively and indirect by mouse interaction. In contrast, touch interaction is direct and absolute. It is doubtful if the results of this study would be analogue to a study performed by touch input. The previously outlined study by Forlines et al. gives some clue to that issue [FWSB07] (see section 4.3.3). The task setting of Forlines et al. was most similar to "Left-tearoff

Menu" since both hands/mice had the same functionality. Since their results show that bimanual direct touch input is superior to indirect mouse input, a transfer to Kabbash's test setting would probably have shown an enhancement in user performance. Yet a study to prove this hypothesis is missing.

The study by Kabbash et al. showed that basing interaction concepts on Guiard's findings can speed up task performance, lower error rate and enhance a user's certainty. However, the experiment was not only based on different levels of divisions of motor labor but also on concrete interaction techniques. Hence a precise conclusion how asymmetric dependent division of labor is superior to asymmetric independent counterpart cannot be drawn. Yet the finding that interaction concepts that divide motor labor after Guiard's principles are superior to letting the user to the division herself is vital for designing novel touch interaction concepts.

4.5 Conclusion

This chapter showed how touch interaction can be used to support the user. Buxton and Myer's study demonstrated that bimanual input is natural and can enhance efficiency.

Furthermore, the directness and absoluteness of touch input helps the user in transferring already existing knowledge about the physical world to a virtual environment. That way existent gestural behavior patterns can be benefited from. Also, epistemic actions can support the user in learning more about the context of a problem. Moreover, direct touch input supports the user in hitting a target according to the optimized initial impulse motor control model.

Section 4.4 highlighted constraints that have to be taken into consideration when designing bimanual interaction techniques. Guiard's Kinematic Chain Theory states that an asymmetric and dependent division of the two hand's motor labor is used in a large number of everyday tasks. Hence, such division seems promising for bimanual interaction concepts. Kabbash's studies partly prove Guiard's findings in the context of a mouse-interaction environment [KBS94] (see section 4.4.2). Forline's study [FWSB07] can partly be used to transfer Kabbash's results to touch input. Summarized, Guiard's Kinematic Chain Theory provides a solid theoretical background for designing bimanual interaction techniques.

5 Designing touch interaction techniques for WIMP

In the following sections, touch interaction techniques for controlling WIMP user interfaces will be introduced. These include existing techniques as well as designs that were developed and implemented as part of this thesis.

WIMP As stated in section 3.1, traditional WIMP applications take the largest part of today's user software. WIMP stands for windows, icons, menus and pointing devices. Operating systems like Microsoft Windows, Apple MacOS, Linux and most applications running on them are WIMP user interfaces.

Many of these legacy applications are indispensable in everyday tasks. Therefore, as tabletops are more and more emerging, it is essential for these devices to enable the user to control WIMP user interfaces. Web browsing, e-mailing, instant messaging, watching pictures and videos are just few examples for the vast number of frequently used tasks in WIMP user interfaces.

Touch input offers the advantage of letting users manipulate by directly touching them. This is more natural than mouse interaction as well as it lowers the gulf of execution (see section 4.3.2). However, touch interaction offers less hardware modalities than a mouse does: The mouse is used for positioning the cursor by moving, performing selections with the left button and access further tasks with the right button. The only hardware modality a touchscreen offers is the reporting of finger positions. This would be sufficient e.g. for positioning a cursor. However, there are numerous more tasks users have to perform frequently when using WIMP user interfaces.

In order to enable the user to operate WIMP user interfaces by touch, we first have to abstract from concrete mouse interaction techniques. Rather, we have to look at what the user has to "tell" a system in order to get a certain operation done. Foley et al. provide a model that describes the different logical layers of interactive graphical user interfaces in matters of interaction [FWC84].

According to Foley et al. a user controls a system by prompting commands [FWC84]. An example for a command would be "Open the folder named 'HCI' that lies on the

desktop". A command now is split into parts, each representing an interaction task. For example, the tasks for opening the folder would be selecting the folder and executing the imperative 'open'. An interaction task now is "performed by means of an interaction technique"[FWC84]. Furthermore, "each task can be implemented by many different techniques."[FWC84]. An interaction technique is bound to a specific input device. For example, the folder's selection could be performed by positioning the mouse cursor over the folder followed by a left button. However, it could also be performed by using keyboard interaction techniques, like tabbing with the ALT-key.

This means that interaction tasks are on a lower logical level than concrete interaction techniques are, and thus can be isolated and transferred between environments.

When designing touch interaction techniques for WIMP user interfaces, the identification of the most elemental interaction tasks in WIMP is crucial. Foley et al. identified six such elemental tasks for interactive graphics. These tasks "are independent of application and hardware, form the building blocks from which more complex interaction tasks, and in turn complete interaction dialogs, are assembled." [FWC84] In detail, the six tasks are: Select, Position, Orient, Path, Quantify, Text. With respect to WIMP, Select, Position and Text are most relevant. Selecting an item from a set of alternatives is elementary for using WIMP. The P in WIMP stands for Pointing Device. This pointing device is used to control "a cursor as a point of entry to the windows, menus, and icons on the screen." [SRP07]. Hence, every time a user wants to execute a command, she has to position the cursor over the corresponding interaction object. When the user now additionally wants to manipulate the object, a selection of this object has to be performed. For example, for opening a menu, the cursor has to be positioned over the menu. After that, the menu has to be selected.

Text input is another vital part of every system based on the WIMP paradigm. Text input is used in a vast number of highly used applications, such as word processing, Web Browser control, E-Mailing or Instant Messaging. Moreover, not even a login into a system could take place without text input.

Beyond Selection and text input as fundamental tasks for graphical user interfaces, there is a number of interaction tasks especially important in the context of WIMP.

Dragging is the first such task to be mentioned. Dragging is the process of selecting an item and moving that item with the cursor to a different location. Dragging is a fundamental part of direct manipulation interfaces. Sketchpad [Sut64], the first direct manipulation interface [Mye98], already supported "grabbing objects" and "moving them"[Mye98]. Since WIMP is based on direct manipulation, dragging is a basic task for manipulating the interface. WIMP is comprised of windows "that could be ... moved around the screen using the mouse" [SRP07]. Moreover, dragging is used for moving scroll bars and icons. Beyond moving objects, dragging is facilitated for selecting a group

of objects, for example a set of folders or a passage in a text document. Summarized, an interface based on the WIMP paradigm would hardly be usable without dragging.

Another task is the evocation of contextual menus. Contextual menus are very helpful for controlling WIMP, since they "provide appropriate commands that make sense in the context of the current task" [SRP07]. More detailed, they only "provide a limited number of options associated with an interface element" [SRP07]. This means that contextual menus provide a list of commands that are actually applicable to the current interaction object. Without such contextual menus, the user would have to search the corresponding command from an explicit menu or she would have to use a keyboard shortcut. Another aspect is the direct manipulation principle contextual menus take advantage from: "It [the contextual menu] allows the user to point directly to an object of interest, with the underlying system performing the desired operation" [KS86]. Put in the context of touch interaction this is even more important than in mouse interaction, since the user not only points the cursor to an object but really touches the object with her finger.

A further task that is a fundamental part of WIMP is scrolling. Scrolling's importance roots in the way documents are presented in WIMP. Examples for documents would be Word Files, PDF documents or web pages. In WIMP, documents are displayed within windows. Since a screen does not have unlimited amount of display space, only a limited amount of content can be displayed in a window at one time. That means that content may be cut off the window if it doesn't fit in completely. Hence, the user has to be supplied with commands that allow navigating in windows so that the full amount of content can be accessed. In WIMP this is facilitated through scrolling. In particular, vertical scrolling is of importance: "Computer users spend a large amount of their time reading and editing documents " [MH04]. Documents are normally presented analogically to physical sheets of paper. That means that they expand vertically while having a fixed horizontal size. Hence, "Since documents are generally too long to fit on one screen, users must frequently scroll to other parts of the document" [MH04].

The following chapters deal with the design and implementation of touch interaction techniques for the previously listed interaction tasks. The chapters will be structured into three dimensions, according to the importance of the separate interaction tasks:

- The first dimension deals with the selection task. More detailed, the goal is to provide the user with an interaction technique that allows selecting objects precisely. Providing precise touch selection is vital for efficient touch usage of WIMP systems. WIMP systems are typically designed for mouse input and lack an adjustment to the requirements of touch interaction.

  As part of this thesis, we designed and implemented ZoomTap, a selection tech-

nique that allows precisely selecting targets, triggering precise draggings (see section 5.1.3). Moreover, it offers the user the possibility of selecting targets by direct touch, which is the fastest way of selecting large targets [VB07].

- The second dimension focuses on text input through touch interaction. We designed and implemented two alternative designs of text input techniques for tabletops (see section 5.2.2) as part of this thesis. We ran a qualitative user study to evaluate theses techniques (see section 5.2.3).

- The third dimension is engaged in how a user can be supplied with touch interaction techniques for the tasks dragging, scrolling and the evocation of contextual menus. Solutions for these issues were designed and implemented as part of this thesis (see section 5.3.2).

All three dimensions are essential for using a WIMP system by touch input. Concentrated, the interaction techniques we will introduce serve as a compound set of techniques. This set allows users to benefit from direct touch interaction while being empowered to control all essential parts of WIMP user interfaces.

5.1 Dimension 1: Precise Selection

In this section, we illuminate the domain of selection tasks. As stated in section 5, selections are one of the most fundamental interaction tasks for controlling graphical user interfaces.

In detail, we focus on selection techniques performed by touch interaction. Moreover, we focus on the domain of tabletop devices (see 5).

We describe and discuss existing techniques (see section 5.1.2 and present ZoomTap, a technique we designed and implemented as part of this thesis (see section 5.1.3). ZoomTap is a selection technique for performing precise selections. It was designed with specific relation to factors induced by WIMP user interfaces.

5.1.1 Introduction

A general issue in touch operated systems is the fact that the ability to hit a target with a finger is limited in terms of precision. This is caused by several circumstances. On the one hand, a finger has a large contact area with a surface when pressed against (approx. 1x1cm). This area obviously is occluded and prevents the user from detecting

what he has hit precisely. As stated in section 4, a pointing movement is performed by first moving the hand fast and imprecise, followed by small correcting movements convergent towards the target. If a user now over- or undershoots the target with the first imprecise movement and lacks the ability to identify how he would have to correct his movement, an eventual hit of the target is unlikely.

A further problem induced by the large contact area between finger and surface is a user's incapability to figure out the precise point of contact. This point is crucial since a selection only takes places in one point, not in an area. One could argue that this point lies in the geometric center of the contact area. Moreover, a user might as well expect this point to be close to the fingertip rather than in the geometric center. So a user might be irritated if he expects to have hit the target but has actually missed. This problem could be approached using heuristics, placing the point of contact above the geometrical center. Yet a more robust solution is the providing of a precise selection technique, since different users might expect different points of contact. Furthermore, the point of contact varies depending how steep the finger is put on the display. Considering a tabletop display, a finger placed closely to the body is more likely to be put on vertically than would a finger placed on the far end of the display (see section 5.3.1.1.1). This causes different areas of contact and consequently differently sensed points of contact.

One method of overcoming these issues is providing screen elements of sufficient size. However, this is impractical in a number of contexts. For example, mobile applications lack the space for displaying items in a size large enough to always be hit by a user. Moreover, existing WIMP applications ported to touch input areas are unlikely to be designed specifically for the needs of touch selection.

In this chapter, existing touch selection techniques will be shown and discussed. Furthermore, a novel selection technique will be introduced. This novel technique was, along with two existing techniques, implemented in a prototype.

5.1.2 Related work

A number of studies investigated interaction techniques leading to enhanced touch selection precision. These studies are outlined in this chapter. The studies are divided into single- and multitouch interaction techniques. Single-touch selection techniques offer a starting point for the design of more evolved multi-touch techniques and thus have to be reviewed before designing novel selection techniques.

Since the concepts presented in this thesis are to run independent of a specific application domain, only selection techniques requiring no alteration of the application's user interface are taken into account. Such alteration would require the possibility of adapt-

ing the application itself to the requirements of a specific selection technique. However, most applications do not offer such opportunity.

5.1.2.1 Baseline selection techniques

This study by Potter et al. evaluate three single-touch selection techniques [PWS88]. These techniques are of particular interest since they supply a baseline representing the most elementary selection techniques. Moreover, these techniques are regularly used as control conditions in later studies.

In the experiment, subjects had to select pairs of letters, each pair representing a US postal code (e.g. CA for California). These pairs were aligned in a Matrix of 5 columns and 10 rows. One target had an area of approx. 1,61 square cm.

Twenty-four people participated in this within-subjects experiment. Independent variable was touch strategy.

AK	HI	ME	NJ	SD
AL	IA	MI	NM	TN
AR	ID	MN	NV	TX
AZ	IL	MO	NY	UT
CA	IN	MS	OH	VA
CO	KS	MT	OK	VT
CT	KY	NC	OR	WA
DE	LA	ND	PA	WI
FL	MA	NE	RI	WV
GA	MD	NH	SC	WY

Figure 5.1: Arrangement of letter pairs [PWS88]

The first technique presented by Potter et al. is named “Land-On”. A selection was registered at the location the finger first touches. If no target was under this location, an error was logged. An error is also registered if the wrong target was hit. I.e. the users were not allowed to perform corrections after placing the finger on the screen.

“First-Contact” is the second technique. A selection was registered at the first target a finger makes contact with. Thus, subjects were allowed to perform corrections after having initially placed their finger besides a target. An Error was marked if the wrong target was selected.

As the third technique the authors present “Take-off”. Thereby a selection was made when the finger was lifted from the surface. Moreover, the cursor was not positioned directly under the finger but 12 mm above the fingertip. So subjects were able to see

the exact cursor position. The selection took place at the last cursor position. Error registering was equal to “First-Contact”.

Mean trial times were significantly smaller for First-Contact (16,93s) than for Take-off (20,92s). Land-on (17,73s) didn’t differ significantly from the other techniques. Error rates were significantly smaller for Take-off (2,25%) than for First-Contact (4,08%) and for Land-on (5,08%).

These results clarify that touch selection is liable to a tradeoff between error robustness and speed. Take-off clearly supported the user most with respect to precision. However, the user lost the advantage of directness. Subjects always had to consider the displaced cursor before performing the selection. The other two techniques did not have this disadvantage. However, Land-on and First-contact suffered from the large contact area between surface and finger explained previously. Summarized, none of the techniques tested by Potter et al. is sufficient for the needs of precise touch selection.

Furthermore, this experiment gives no clue about how large a target has to be in order for being able to be hit reliably.

5.1.2.2 Shift

A main issue with Take-off is the displaced cursor. A technique dealing with this issue was presented by Vogel and Baudisch in 2007. Their “Shift“-technique places the cursor directly under the finger [VB07]. However, the screen content occluded by the finger is displayed once more beside the finger. Furthermore, this content was enlarged and a crosshair was displayed at the current finger position. A selection is triggered by lifting the finger.

That way the user is able to profit from the touch screen’s directness without suffering from the disadvantages caused by occlusion and contact area size.

In the evaluation, subjects had to select targets of different sizes. Sizes reached from six to 96 pixels, with six gradations. One pixel had a width of 0,436 mm. Twelve participants were tested. The design was within-subjects.

The Shift technique was tested against direct touch and Offset Cursor. direct touch didn’t supply the user with any support, a selection was made simply triggered by lifting the finger at the position of the target. Offset Cursor is the same as the Take-off technique only named differently (see section 5.1.2.1).

Analysis of error rate shows that direct touch is significantly more error-prone than the two other techniques for targets of a size of six and twelve pixels.

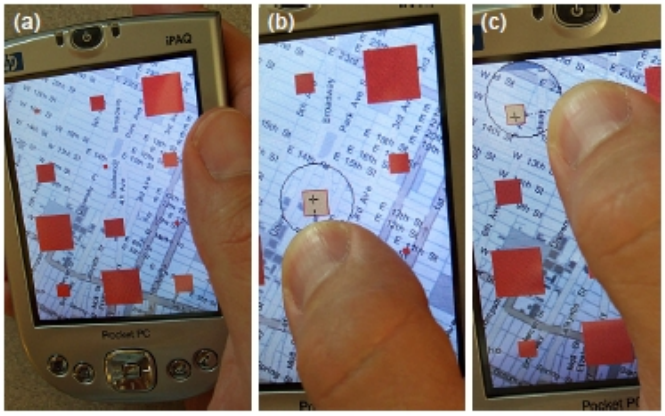

Figure 5.2: "(a) Small targets are occluded by a user's finger. (b) The proposed Shift technique reveals occluded screen content in a callout displayed above the finger. This allows users to fine tune with take-off selection. (c) By adjusting the relative callout location, Shift handles targets anywhere on the screen." [VB07]

These results indicate that Shift both lets the user profit from the directness of a touch screen while reducing the error rate of direct touch. However, the Shift area could cover relevant parts of the screen. Transparency or a movable Shift area could be solutions for this issue.

Analysis of mean trials times reveals direct touch performs fastest with all target sizes. For targets sized 24 pixels and above, Shift performed similarly fast to direct touch. For targets smaller than 24 pixels, Shift performed similar to Offset cursor.

A further interesting result of this study is the error rate of direct touch selection. Error rate of Direct touch was similar to Shift and Offset cursor for targets of 24 pixels size and above. This corresponds to a size of approximately 10x10 mm, which is similar to the area of contact between finger and surface. For targets of 18 pixels and smaller, direct touch was significantly more error-prone than Shift and Offset cursor.

5.1.2.3 TapTap

A selection technique basing on Zooming was presented by Roudaut et al. in 2008 [RHL08]. For selecting an object, a user has to first tap into the object's proximity. Next, the area around the tapped point is zoomed in. The final selection is done through tapping on the actual target in the zoomed area.

TapTap was evaluated against a number of other selection techniques. The design was

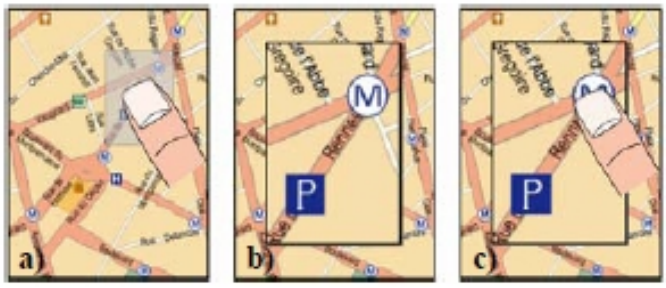

Figure 5.3: TapTap Design [RHL08]

within-subjects. Twelve people took part. Independent variables were Technique and Target Area.

The most interesting of techniques TapTap was evaluated against were direct touch, Shift and Offset Cursor. The other techniques, MagickStick [RHL08] and Thumbspace [KB07] are only applicable on mobile devices and thus drop out of consideration for finding selection techniques for a tabletop scenario. Subjects had to select targets with a width of 3 mm. This is only slightly larger than the smallest target in the Shift-study (2,6 mm).

Selection time analysis shows that "TapTap is about to 2.3 times faster than Offset Cursor, 2 times faster than Shift" [RHL08]. There was no significant effect in matters of time between direct touch and TapTap.

Error rate analysis "showed that TapTap (6.7%) has the lowest error rate and direct touch (59.9%) the highest in comparison to all other techniques" [RHL08]. There was no significant difference between TapTap, Shift and Offset cursor.

Summarized, TapTap performs best for targets sized 3 mm. However, when using TapTap, a user would always have to do two taps for one selection. One could argue that a selection could be triggered instantly if a target was hit and only zoom if no target was hit. However, this would be impractical if targets were located close to each other. Furthermore, knowledge about the underlying visualization would be required. Yet this knowledge has to be considered inexistent when designing interaction concepts for WIMP-systems.

5.1.2.4 Multi-touch selection techniques

The previously covered selection techniques all utilized only one hand for performing a selection. Yet there are several indications that bimanual interaction supports a user in accomplishing a task (see 4.2).

A study by Benko et al. introduced several selection techniques taking advantage from bimanual input [BWB06]. Twelve volunteers participated, the design was within-subjects. Three novel designs are shown. All three techniques are designed to split motor labor in an asymmetric dependant way, according to Guiard's Kinematic Chain Theory [Gui87] (see section 4.4).

The first technique, X-Menu, allows the user to adjust Control-Display ratio of cursor positioning. The Control-Display (or CD) ratio is the quotient of the distance put back by the input device (e.g. the finger) and the distance interpreted by the system (e.g. the cursor movement distance). In direct touch interaction, this ratio is one and the cursor always is beneath the finger. If CD-ratio is raised, the cursor moves slower than the finger, but still has the same movement vector (i.e. moves in the same direction).

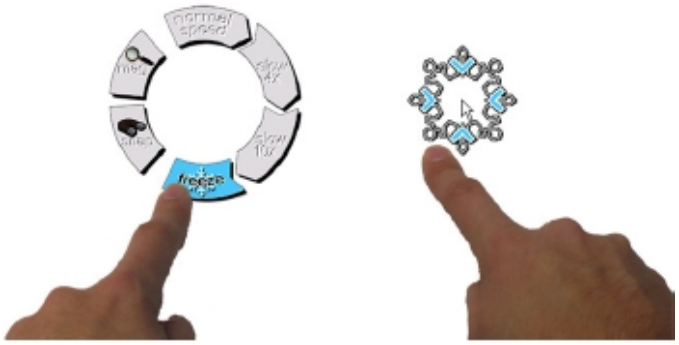

Figure 5.4: Dual Finger X-Menu. The Control-Display ratio can be altered by selecting the corresponding button from the circular menu [BWB06]

X-Menu works in the following way: If a single finger is placed on the display, a selection behavior is similar to direct touch. However, when placing a second finger on the display, a circular widget menu is displayed. This menu has several buttons allowing changing the CD-ratio. The CD-ratio can be increased in two steps, 4x and 10x. Additionally the cursor can be frozen in place. When changing the CD ratio, the cursor moves slower than the finger, but still in the same direction. That way a user is able to hit a target more precisely.

The second technique, Slider, works after the same principle. However, the CD-ratio is not altered by selecting buttons on a widget. With Slider, an alteration of the two finger's distance changes the CD-ratio in discrete steps. The divisions are the same as in X-Menu, including a cursor freeze after passing a certain threshold. Concentric circles around the cursor indicate how much the CD-ratio has been altered. One circle means 4x increase, two circles mean 10x increase. A cursor freeze is shown as a crystal-like shape around the cursor.

Stretch uses a different approach. Hereby, a user can trigger a zoom of an area around

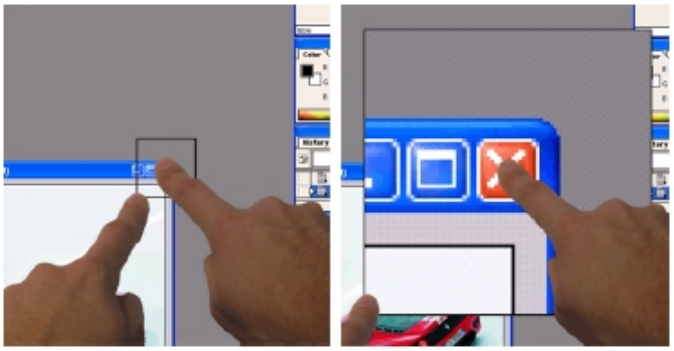

Figure 5.5: Dual Finger Stretch. One Finger controls the cursor, the other specifies a square zooming area [BWB06]

the cursor. This is triggered by placing a second finger next to the first. The square defined by the first finger (midpoint) and the second finger (corner) indicates the area that will be zoomed into. Moving the second finger away from the first enlarges the square and thus causes a zoom. Once triggered, the zoom square stays in place, even if the first finger moves. In this way, small items can be hit by enlarging them to such extent a reliable hit is possible.

Benko et al. evaluated these three techniques against Offset, which is simply a different name for Take-off. Subjects had to hit targets of one, two, four and eight pixels width. All new techniques perform significantly better than offset in terms of error rate. Too, Stretch (20%) is for targets of one pixel size about twice less error-prone than X-Menu and Slider (37%, 38%). Mean trial time was significantly better for Stretch for one pixel targets. In terms of speed, Stretch was about 1s faster than X-Menu and Slider for all target widths.

This study shows that multi-touch selection offers opportunities for enhancing selection precision and speed. The superiority of Stretch proves that a switch from absolute to relative input is less effective and efficient than retention of absolute input. However, a comparison between the most promising single touch selection techniques and Stretch is missing.

5.1.2.5 Apple Multi-touch zooming concept

Apple products feature several multi-touch interaction techniques that are used in various contexts [App08]. Most interesting is the zooming gesture Apple uses in the IPhone as well as on its notebook's track pads. This zooming gesture is triggered by simultaneously moving thumb and index finger into opposite directions. This gesture is normally used to pinch or stretch an object, e.g. a picture. Yet, formal user studies on the ef-

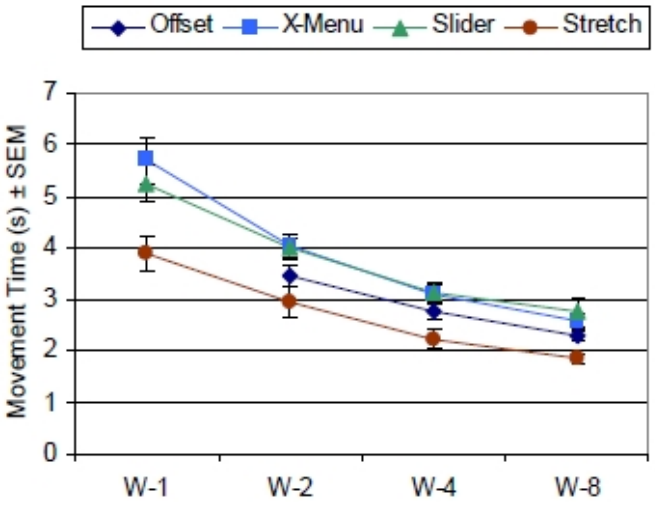

Figure 5.6: "Mean performance time (s) with respect to target widths" [BWB06]

fectiveness and the efficiency of this particular gesture have not been published so far. However, the zooming gesture is being applied in commercial Apple products that are used in large numbers on the consumer market. This raises the assumption that user quickly learn and accept this gesture. Thus, a further consideration of the two-finger zooming gesture is promising.

5.1.2.6 Discussion

Three of the previously presented selection techniques come into consideration when designing selection techniques for WIMP systems.

Shift supplies the user with certainty about his point of contact. This enables the user to accurately correct a selection when missed with the first attempt. However, this way of selecting an item requires the user to learn a new behavior pattern. When selecting an item, a user normally uses a single tap. That means that the finger doesn't rest on the surface but is lifted immediately after making contact. This behavior can be explained by the way we press physical buttons. Considering a keyboard, a key is selected by pressing without any need for further dwelling. Furthermore, when performing a mouse selection, possible corrections are completed before the actual clicking. Findings related to the theories of embodiment show that humans tend to use already learned behavior patterns metaphorically for getting acquainted with new interaction environments [Roh95]. Hence, a transfer of already learned selection behavior is likely.

Shift doesn't support this behavior. If the user misses the target in the first attempt and lifts the finger immediately afterwards, the Shift technique presents no benefit to

the user. Hence, users would have to be trained in order to let their fingers rest on the surface after making contact. Only this way a correction of a selection using the Shift technique would be possible. It is questionable how easy different users would adapt this selection pattern. Existing studies offer no clue about that issue. Yet, if once learned, Shift requires no further discrete interaction step to define the selection.

TapTap always triggers a zoom after the first selection. The second selection is done though tapping on the now enlarged object. That way the user's initial impulse of lifting her finger after making contact is supported. Yet, the fact that the zoom is always triggered might be unnecessary and inefficient. Selectable objects might very well be large enough for being hit reliably with on single tap. In this case the zoom would be irrelevant and only slow down selection time. The authors discuss this issue, proposing to trigger the zoom depending on target size. However, this would require knowledge about the underlying visualization. This knowledge cannot be taken as granted when designing interaction techniques for cross-application useage.

Stretch lets the user choose when a selection is to be rendered more precisely by zooming. However, it suffers from the same issues as Shift. The precision of a selection is enhanced by correcting the position of the finger. Hence, the finger would have to remain on the surface.

A further issue with Shift and Stretch is that the precise initiation of dragging is unreliable. Dragging behavior will be taken into closer consideration in section 5.3.2.2. Yet, in the design chosen, dragging starts at the initial point of contact. The imperative of Stretch and Shift to correct a selection by moving the finger prevents the precise specification of the dragging start point.

TapTap doesn't have that downfall since the user would be able to choose the precise dragging start point from the enlarged area.

5.1.3 Designing selection techniques for WIMP-applications

For overcoming the existing techniques' drawbacks, we have designed a novel technique for performing precise selections. We called this technique ZoomTap. Our technique enables to user to precisely select small targets and precisely initiate dragging tasks. Furthermore, the user is left to possibility of rapidly selecting large targets by direct touch. During design process, two iterations were conducted. Moreover, we implemented ZoomTap in a prototype program.

5.1.3.1 Design principles

As shown, each of the presented techniques has certain advantages and disadvantages. For a more systematic investigation, we first try to establish some reasonable general design principles. An interaction technique suitable for performing selection tasks in the domain of WIMP user interfaces should obey these principles. These principles are subjective in some way, yet they result from our own experiences in using WIMP user interfaces as well as discussions with real-world users of WIMP user interfaces.

1. Support of cross-application usage: The selection technique must be usable without having access to a specific's applications logic. Accessing this logic might be helpful for performing a precise selection. For example, the authors of the TapTap technique propose a triggering of the zoom only if the target undercuts a certain size [RHL08].

   However, it is unrealistic to gain access to the logic of all applications currently running on current WIMP user interfaces in such way. Yet this total access be necessary to provide equal power to every selection task.

2. Effectiveness and efficiency independent from target size: The technique must support selections on both large and small targets in an effective and efficient way. A selection technique that allows the user to select small targets precisely, always requires an additional interaction step and thus lowers efficiency but maintains effectiveness. With Shift, the user has to correct her finger's position. With TapTap, two taps have to be performed. With Stretch, the user has to pull up a zoom rectangle and then correct her finger's position. In contrast to small targets, large targets are best selected by Direct-Touch, like shown by Vogel and Baudisch [VB07] (see section 5.1.2.2). An imposition of additional interaction steps only required for the selection of small targets hence is unnecessary and would only slow down selection speed. Considering this tradeoff between speed and reliability depending on target size is essential for enabling an overall usable system.

3. Support of dragging tasks: The user must be able to set the starting point of a dragging operation with the same precision as a selection. For example, precise dragging is necessary for resizing a window by dragging its corner. A technique that doesn't supply the user with such possibility would narrow the user's possibilities of interaction and thus lower the effectiveness and likely the efficiency of a system.

4. Proximity to already existing behavior patterns: The user should be able to apply already learned behavior patterns for enhancing a selection technique's ease of learning and ease of use. Such patterns are meant with respect to both real world behavior and traditional mouse usage. The necessity of such proximity has been

shown in section 4.

5.1.3.2 Applicability of existing techniques

- Shift: Conflicts with design principle #2, #3 and #4. Detailed explanation is given in section 5.1.2.6
- TapTap: Conflicts with design principle #2 (see section 5.1.2.6)
- Stretch: Conflicts with design principle #3 (see section 5.1.2.6)

5.1.3.3 ZoomTap: A blended approach

Since none of the previously illuminated techniques fulfills all factors required for effectively performing selections in WIMP user interfaces, a novel solution technique had to be found.

For this purpose, we designed a selection technique that takes parts of exiting techniques and putting them together to a blended concept. This concept is to fulfill all of the design principles listed in section 5.1.3.1. We named this blended technique ZoomTap.

With ZoomTap, we considered the tradeoff between effectiveness and efficiency depending on target size by offering adapted functionality for differently sized targets. The technique works as follows:

- Selections of large targets:

  When selecting large targets (larger than 10mm x 10mm), a selection's precision is not required to be enhanced by an interaction technique. Hence, we let users perform a selection with a tap by direct touch. A tap is a touching immediately followed by the lifting of a finger at one position. This means that no aid is provided for hitting a target.

  The selection is triggered when lifting the finger. This is reasoned in the analogy to mouse interaction. When clicking with the mouse, the click is triggered by quickly pressing and lifting the mouse button. A user now can transfer this behavior pattern to touch selection.

- Selections of small targets:

  Since small targets require aid from a interaction technique to be hit effectively,

we provide a local zoom to the user. This zoom enlarges the desired target and thus enhances selection precision. For triggering the zoom the user has to put two fingers on the display. Moving these fingers in opposite directions causes an area to be zoomed into.

This is similar to the zooming gesture used to enlarge images on the Apple iPhone. The zooming is triggered discretely as soon as the two fingers pass a certain movement threshold. This zoomed area is displayed as an overlay widget, pushing the metaphor of a magnifying glass. The area has a size of 13cm x 13cm on the unzoomed display. The zoom factor is two. The zooming reference point [Kön06] is the midpoint of the two fingers when starting the movement. The selection is performed by tapping on the now enlarged target. The zooming itself is animated. After selection, the overlay widget disappears.

The user is supported in various ways. First, zooming has been shown usable for enhancing selection precision [BWB06, RHL08]. Furthermore, zooming also is "an extension of the PHYSICAL WORLD metaphor which draws on the common pattern of feeling when an object approaches us" [Roh95]. The animation of the zooming effect pays respect to the fact that approaching an object also "takes place in and through time" [Roh95]. Such metaphors can be effortlessly transported to other environments [Roh95]. The discrete triggering of the zoom only requires rough finger motion. That way a user does not have to control motor labor in an accurate way. This is crucial since the selection itself already requires accurate motion. The zooming merely is a tool to enhance precision and thus should not require precise motion.

ZoomTap also supports the precise definition of a dragging start point. After zooming, a dragging can be started with the same precision as a selection. This is done by placing the finger on the desired element followed by moving the finger (and thus also the element) to the target location. Our dragging technique is described more detailed in section 5.3.2.2.

ZoomTap utilizes the advantages of both TapTap and Stretch. TapTap supplied the possibility of setting an accurate dragging start point. Stretch supplied the concept of letting the user choose when to zoom in. However, letting the user choose when a precise selection is necessary might not be the perfect way. A user unaware of the hand's limited precision might try to hit small targets initially by direct touch. A utilization of the zooming functionality would take place at the earliest during the second attempt. This issue has to be considered in the future development of ZoomTap.

5.1.3.4 Implementation

In order to analyze and compare the different techniques directly, ZoomTap, TapTap and Shift were implemented in a prototype program. TapTap was chosen because of its simplicity and its consistency of concept. The zoom is triggered always which makes the technique's behavior very expectable. The tapping mechanism likely matches the user's expectation.

Shift was chosen because it requires least discrete interaction steps for performing precise selections. Both ZoomTap and TapTap require one further step when doing a precise selection. Moreover the behavior of Shift when performing dragging tasks was to be looked at.

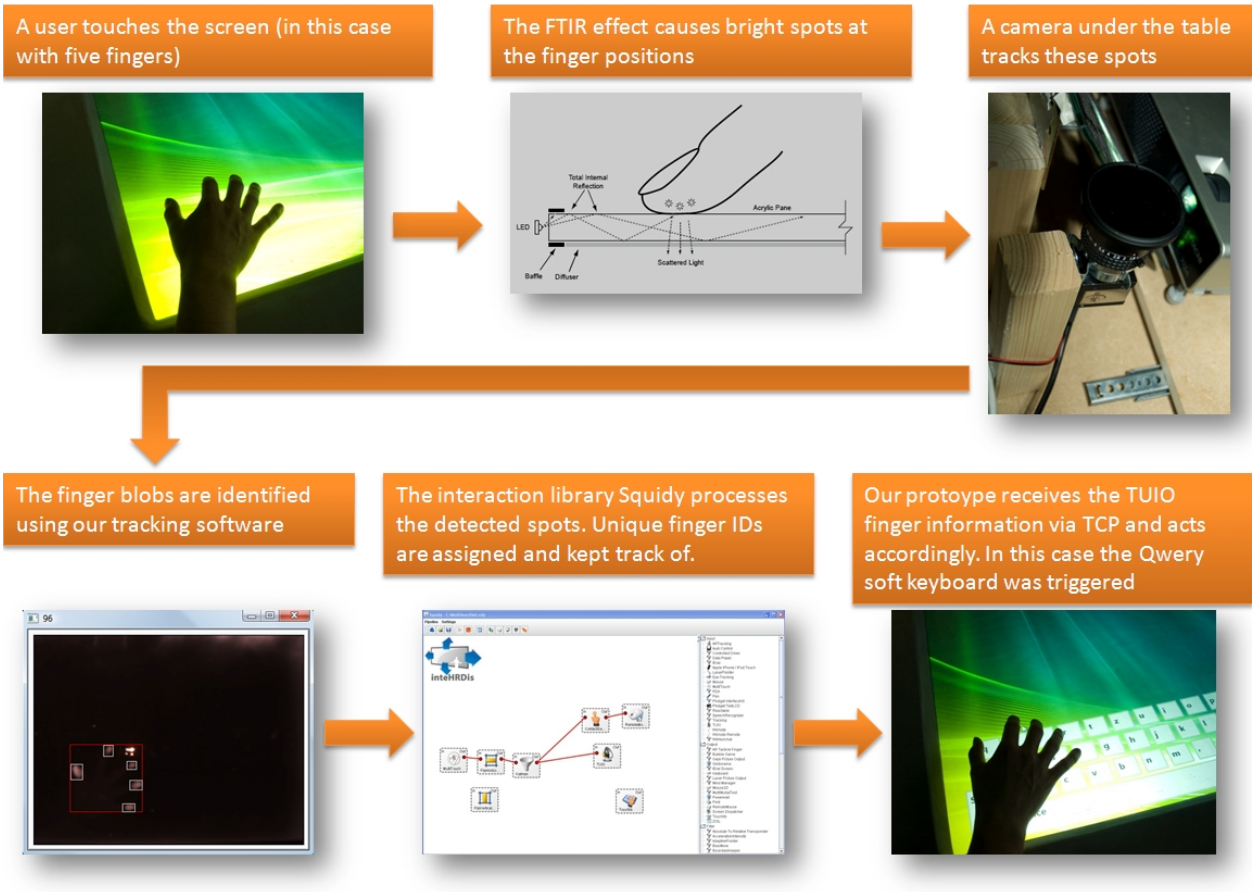

Figure 5.7: Finger recognition and processing. For more information on our tracking software and our hardware device, see [RSF+08]. For more information on Squidy, see [JKR09]. For more information on the FTIR (i.e. frustrated total internal reflection) effect and how it is used for multi-touch, see [Han05].

A self-built 34" multi-touch-capable tabletop display served as testing environment (see the term paper [RSF+08] for further information on hardware setup and tracking algorithms). The prototype is implemented in the programming language C#, using the Windows Presentation Foundation (WPF) as framework for visualizations. As operating system Windows Vista is used. The program receives tapping events and finger positions from the interaction library Squidy [JKR09]. The finger data is received via TCP. TUIO is used as protocol for encoding finger information [KBBC05]. The TUIO protocol is commonly used for transferring multi-touch finger data. The finger's five most recent movement vectors are stored for further use.

The implementation of Shift is as follows: The screen content from beneath the finger is copied into a bitmap container using a Windows API function. This content is displayed in a separate window.This window represents the Shift-area. The window has none of the controls, headers or borders that are normally part of a Vista window. Thus, the user only sees the Shift-area's content. The content is enlarged by a factor of two. Moreover, a crosshair is displayed in the center of the Shift-area. This crosshair corresponds to the finger's point of contact on the actual display. This way the user is able to clearly see the content occluded by her finger and identify the exact point of contact. The content is continually updated as long as the finger moves. Furthermore, the Shift-area is translucent and hence the underlying content is visible. By default, the Shift-area is displayed approx. 7 cm above the finger. The area itself has a size of approx. 4x4cm. A black border helps in distinguishing between Shift-area and the actual display. The Shift-area is moved to the sides of the finger if the display's corner is approached. That way every point on the display is accessible.

Figure 5.8: **Left:** The user has missed the cross initally. The Shift area above the finger provides feedback about the exact point of contact.
Right: The user has corrected her finger position to hit the cross.

TapTap is implemented similarly. When a user taps the display, the area around the tap center is loaded into a bitmap structure (see figure 5.1.2.3. This bitmap is copied into a Window. This window is display centered on the position of the first tap. An animation

displays the transition between the zoomed and unzoomed state. When the user taps onto the enlarged area, the point is recalculated so that the tapping point corresponds to the undistorted point. Again, a border around the area is displayed.

ZoomTap uses the same functionality as Shift and TapTap to display the zoomed area. However, the zoom is triggered by a gesture. This gesture is identified using movement thresholds and vectors. Finger vectors are used to determine their relative movement. In detail, two vectors point in opposite directions, if their components signs differ from each other. For example, two vectors A(-1, -2) and B(2, 3) point into different directions. The vectors C(-2, 2) and D(2, 4) do not point into different directions with respect to the definition used here. Figuring out if the vectors point into opposite directions is achieved by taking their component's products. That means, that the two X-components are multiplied with each other. The same is done for the Y-components. If the product of both the X components and the Y components is smaller or equal zero, the vectors point into different directions.

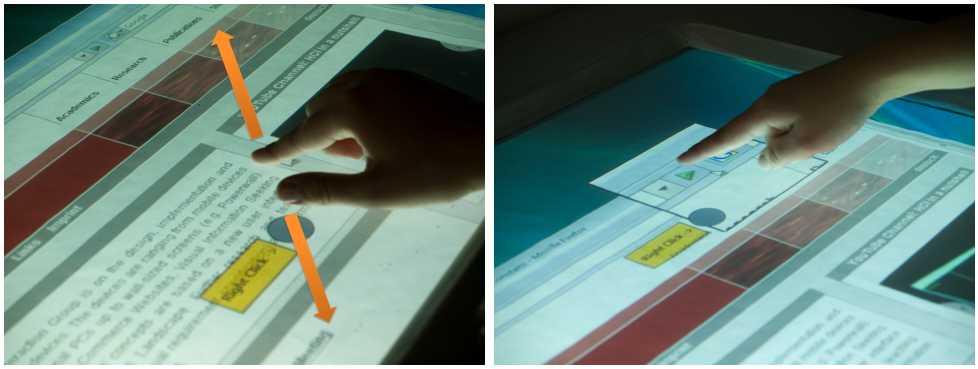

Figure 5.9: **Left:** ZoomTap's local zoom is triggered by moving two fingers into different directions.
Right: The local zoom has enlarged an area on the screen.

After figuring out the finger directions, the movement distances are taken into account. A zooming only takes place if both fingers have moved at least 30 pixels.

The combination of both approaches is to ensure a reliable triggering of the zooming technique. The final tap in ZoomTap is done in the same way as in TapTap.

5.1.3.5 Observations

ZoomTap and the other implemented techniques were practically used by"real-world" users. On the one hand, the selection techniques were tested by members of the Human-Computer Interaction Group at the University of Konstanz on several occasions. Moreover, the tabletop was exhibited at the do it.konferenz in Stuttgart (13.10.2008) [Med08].

ZoomTap was used by a limited number of people (approx. 30, no systematic counting was done) to control a web-browser.

One of the behaviors regularly observed was the way how users placed their fingers on the display when intending to initiate a zoom. Users normally put their index finger and thumb on the display. However, they regularly put on these fingers very close to each other. Very close means that index finger and thumb touched each other. When initially placing the two fingers on the display in such manner, the zoom gesture following is unergonomic. This is because of the following motor process: First, the hand has to be lifted a small way up, in order to allow expanding of the fingers up to a certain point. This point is normally reached when the distance between the two fingers is approx. 3 cm. After that point, the hand is being lowered down so that the fingers are able to expand further. When taking a look at the complete process, the hand performs a wave-like up- and down motion. Since users have to keep contact with the display through all the time, unergonomic motor motion is the result. That way the zoom gesture no longer is the imprecise, rough and relaxed movement originally indented. As a consequence of the described drawbacks, a revision of ZoomTap was performed (see section 5.1.3.6).

Once triggered, the zoom was normally understood instantly and the selection was also performed with no further incidents

Observation of the Shift-technique confirmed its previously stated disadvantages. Users tended to do a hit-and-release tap for selecting items. Since most observations were done when users had to control a web browser, most browser controls and hyperlinks were hit reliably. However, these controls are normally larger than 10mm x 10mm and thus are most effectively selected by direct touch. Yet, users also used this hit-and-release behavior for selecting smaller targets. An example would be the closing button of a Tab in the Firefox web browser. This button has a size of approx. 5mm x 5mm. Without extra advice, users performed several taps until they finally hit the target. An exploitation of the Shift's advantages only took place after several remarks. Another issue with Shift was the inability of Shift in starting a precise dragging operation. Such precise dragging is required e.g. for resizing a window or moving the scroll bar. Since users not instantly noticed that they missed the dragging target, much time was needed until the next try was undertaken. Thus, users were quickly frustrated.

5.1.3.6 ZoomTap: Second iteration

Since observations showed that the existing zooming gesture not always allowed ergonomic hand movement, a redesign of the ZoomTap technique was conducted. Goal was to provide a guidance of the user in such way that thumb and index finger are initially put on the display in relaxed state. This means that users optimally should place

thumb and index finger on the display with a distance of 3 cm. Moreover, we wanted to enhance ZoomTap's performance by means of execution time.

The TapTap technique was found to be "about to 2.3 times faster than Offset Cursor, 2 times faster than Shift" [RHL08] for targets of 3mm size. Hence, the subsequent execution of two taps combined with a local zoom enables the user to rapidly select small targets. However, this design has the drawback of always requiring the user to tap twice, even if targets are large enough to be hit effectively without the aid of a zoom.

We redesigned our ZoomTap technique in such way that it benefits from the speed of mere tapping but still leaves the possibility of performing a selection with just on tap on large targets. The idea is that users specify the region they want to zoom into by setting two fingers on the display. The midpoint of these two fingers now is the midpoint of the rectangular area that will be zoomed into. The zoom is then triggered by lifting the two fingers simultaneously. The extent of the zoom area is fixed (100x100 pixels), since we want to enable the user to trigger a zoom without precisely specifying the exact zoom area. We expect such a specification of the zoom area to slow down the overall selection time. In order to trigger the zoom the two fingers have to be more than approx. 1 cm afar from each other. This is to motivate users to put on thumb and index finger in a preferably relaxed state and not clenched together.

In order to allow the user to feel certain about the area that will be zoomed into, the zoom area is visualized by a translucent rectangle with a 3 pixel border. This rectangle clearly lifts off from the underlying visualizations. Using this rectangle, the user clearly sees what area exactly will be zoomed into. Furthermore, we enable the user to move the zoom area on the screen by moving both fingers parallel.

Concretely, the zoom is triggered when the user taps with two fingers simultaneously. As TapTap, our technique requires two discrete tappings for selecting small targets. The evaluation of TapTap has shown that user are able to rapidly perform such tappings [RHL08]. Hence, ZoomTap also is likely to benefit from this ability in terms of overall selection times. Moreover, users are left the possibility of defining and controlling the desired zoom area if they are not completely sure about where they want to zoom in. On the other hand, if users are sure about the desired zoom area, they can trigger the zoom by a rapid simultaneous tapping of thumb and index finger around the target.

The final selection is performed with a single finger tap, like in the first iteration.

In summary, a user can select large targets efficiently by direct touch. For small targets, the user can rapidly trigger a local zoom by tapping two fingers parallel around the desired target. The final selection is then done by a direct touch on the now enlarged target. The evaluation of TapTap indicates that tapping twice for performing precise selections is efficient [RHL08].

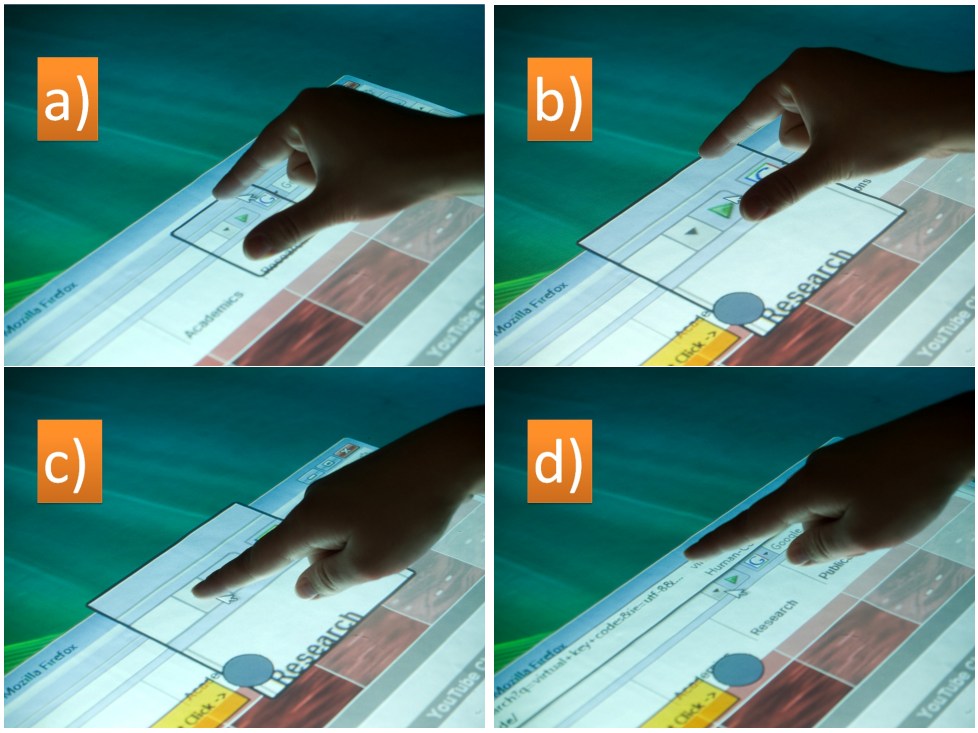

Figure 5.10: a) Two fingers are placed on the display. A rectangle marks the zoom area.
b) The fingers were lifted. Thus, the local zoom was triggered.
c) The user now is able to precisely select the desired target.
d) After performing the selection, the zoom area disappears.

Moreover, the usage of a local zoom still enables the user to precisely start a dragging task. When intending to drag a small target, users can enlarge the area around this target with ZoomTap. Then placing a finger on the target followed by moving the finger initiates the dragging. See section 5.3.2.2 for a deeper insight into our dragging technique.

We implemented ZoomTap's second iteration in our prototype program. The second iteration's implementation built up on the existing implementation described in section 5.1.3.4.

5.1.3.7 Discussion

Since we cannot access information about the target the user wants to select, we cannot tell how large this target is. This leaves the decision if to select the target by direct touch or with aid of ZoomTap completely to the user. However, a user has no way of implicitly knowing how large a target has to be for being able to be hit reliably. Since

direct touch requires the performance of a less demanding interaction tasks, users are likely to prefer direct touch initially. Hence, users will need some time to get a feeling about how well they can hit targets by direct touch and from which target size they utilize the aid of ZoomTap.

There is a time difference of approx. 0.5 seconds between the two-finger tapping and the evocation of the local zoom. This time difference is caused by the technical issues. Despite 0.5 seconds may seem little time, this time difference is clearly noticed. Hence, users may feel that ZoomTap slows down their working stream. However, we expect this time difference to be remediable.

5.1.4 Conclusion

We have presented ZoomTap, an interaction technique for performing selection tasks by touch input. ZoomTap provides the following essential possibilities to the user:

- Small targets can be selected precisely. Such precision is crucial for effectively controlling WIMP user interfaces by touch input. WIMP user interfaces are normally designed for mouse interaction techniques. Yet selection tasks performed by mouse interaction techniques can be carried out much more precise than with touch interaction.

  The selection's precision is enhanced by a local zoom. This zoom is triggered by parallel tapping two fingers around the desired target. Preforming two subsequent tapping tasks has been shown to be faster than correcting an initially misplaced finger position for selecting targets of 3mm size [RHL08] (see section 5.1.2.3.

- Large targets can be selected directly by direct touch, without any additional interaction steps. direct touch is the fastest way of effectively selecting targets larger than 10mmx10mm [VB07].

- ZoomTap enables the user to precisely initiate dragging tasks.

Neither of the existing techniques provides all these possibilities to the user (see section 5.1.3.2 for detailed discussion).

Furthermore, ZoomTap is independent of application domain. Such independence is an essential factor for equally supporting all parts of a system based on the WIMP paradigm.

ZoomTap's current design is the result of two design iterations. The second design iteration was motivated by an regularly observed unergonomic motor process performed

when triggering the local zoom. The design resulted from the second iteration motivates the user to put on her fingers in an ergonomic way. Moreover, we expect the current design to enable users performing precise selection tasks faster than the first design did.

We implemented ZoomTap as a part of the prototype program we developed within the scope of this thesis. We developed the prototype using the Microsoft WPF framework. As oparting system we used Windows Vista.

Further insight is required into the issue in how users can be supported in choosing whether to access ZoomTap's zooming functionality or only use direct touch for performing a selection (see section 5.1.3.7).

A topic in which ZoomTap could be enhanced is letting the user scale the size of the area that will be zoomed into. This means that a user would be able to scale the level of precision by herself. Hence, even pixel-accurate selections might be possible. A drawback would be that the area would have to be scaled before every zooming. Currently users only need to perform a parallel tapping of two fingers with the fingers' distances being mostly arbitrary (see section 5.1.3.6). However, a solution would be to alter the zoom area's size not before the fingers' distance undercuts a certain treshold.

5.2 Dimension 2: Text Input

Already Foley et al. identified text input as a substantial task for interacting with graphical systems [FWC84] (see section 5). Hence, text input also is a task frequently required in WIMP: "Entering text is necessary for activities that require elaborate text compositions such as coding programs, authoring articles, or writing emails, as well as in situation that demand taking notes, typing in commands, or annotating content." [HHCC07]. Thus, users require adequate techniques for entering text when using WIMP. Two alternative designs for text input techniques by touch interaction are presented in this chapter:

- Our Qwerty soft keyboard design reclines at traditional physical keyboards. Letters are entered by touch-typing the corresponding keys (see section 5.2.2.2). In our design letters are sized and positioned to optimize typing speed.

- Column Typing is the second design. With Column Typing, users enter a letter by setting the finger on the letter's horizontal position. The finger's vertical position is arbitrary. Hence, one letter is associated with exactly one column on the display (see section 5.2.2.3).

A qualitative user study with six participants was performed in order to compare both designs for the purpose of future design iterations (see section 5.2.3). In order to compare the designs in "real-life", we implemented both keyboards in our prototype program.

5.2.1 Related work

Before dealing with specific text entry techniques, we will give a brief overview about how text entry effectiveness and efficiency are commonly measured in literature. This will simplify comparing different techniques.

When measuring a text entry technique, subjects are typically asked to enter phrases of text. The system logs the inputted characters with their according timestamps [I. 02]. From this data, the speed in the unit "characters per second" can be extracted. More commonly than the unit "characters per second" is 'words per minute' [I. 02]. "The definition of a "word" for this purpose is "five characters", including spaces or any other characters in the inputted text" [I. 02]. Hence, this unit doesn't name the actual words but rather defines five characters as one word.

The phrases presented to the subjects are typically chosen representative for the english language in terms of character and digraph frequency [ZHS00]. Normally "Tables of singleletter and digram frequency counts for various wordlength and letter-position combinations" [MT65] from Mayzner and Tresselt is taken. Yet this table has the shortcoming of only considering words of the length 3 to 7, which eliminates "some of the most common words such as I, in, on, is, at, to, of, if" [ZHS00]. Moreover, it is unclear if Mayzner and Tresselt's table "still reflects the usage of English in today's digital media" [ZHS00]. For these reasons, Zhai et al. constructed their own digraph table for the evaluation of their metropolis soft keyboard design. This digraph is to be representative for the up-to-date english language corpus [ZHS00].

Morevoer, a set of 500 phrases representative for the english language corpus was published by MacKenzie and Soukoreff in 2003 [MS03].

There are several ways of measuring error rates and the ease of error correction in text entry evaluations. The "minimum string distance error rate (MSD error rate) and keystrokes per character (KSPC)" by Soukoreff and MacKenzie lets subjects commit errors and also perform corrections. [SM03]. In an experiment conducted by Zhai et al. participants had to enter a text phrase correctly before being able to continue to the next phrase [ZSA02].

There are plenty of techniques of accomplishing the task of entering text. Hinrichs et al. have identified six groups to which different text input techniques can be assigned

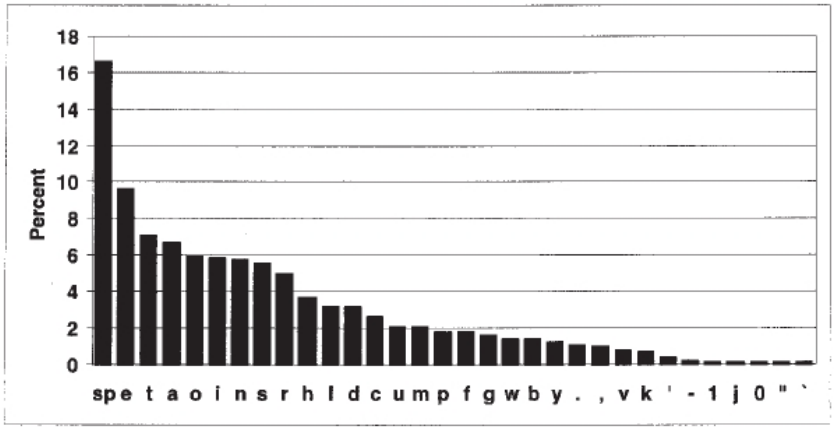

Figure 5.11: Relative letter frequency in the english news corpus, according Zhai et al. [ZHS02]

to. These groups are: Physical Keyboards, Speech Recognition, Handwriting, Gestural Alphabets, Soft Keyboards and Gesture-Based Keyboards [HHCC07].

We exclude Speech Recognition and Handwriting from further consideration:

- In our thesis we only deal with touch interaction techniques. Speech Recognition can not be used as such a technique. Furthermore, Karat et al. have shown that Speech Recognition is significantly slower that typing on a keyboard. [KHHK99]

- Handwriting has a performance of approx. 15 wpm [HHCC07]. This is considerably less than the performance which can be achieved using techniques like soft keyboards (approx. 40 wpm for a QWERTY-Layout used with one finger [MZ99]) or Gestural Alphabets (34 wpm for Unistrokes [GR93]).

Gestural Alphabets allow faster writing than Handwriting (34 wpm for the Unistrokes technique [GR93]), but are still slower than soft keyboard techniques. Moreover, Gestural Alphabets map a gestural representations to each character [HHCC07]. This means that users have to learn the gestural representations before being able to enter text at all. Hence, Gestural Alphabets make sense for personal devices like PDA's, where the user can be expected to invest a certain amount of learning time. In the tabletop domain however users are likely to use a system only for a short amount of time (e.g. in a museum or as kiosk system). Thus, Gestural Alphabets are inadequate as text input technique for the tabletop domain. Most known implementations of Gestural Alphabets (according to [HHCC07]) are Unistrokes [GR93] and Graffiti [Bli95].

For these reasons, the group of Soft Keyboards is most crucial for designing text input techniques basing on touch interaction for tabletop displays. Gesture-Based Keyboards

also fall into this section, since they are principally soft keyboards that do not require the user to lift her finger or stylus while entering text. Hence, they could also be an alternative for text input using touch interaction techniques and will be dealt with in this State-of-the-art analysis.

5.2.1.1 Physical Keyboards

The text entry by physical keyboards is the de-facto standard for text input in desktop computing. Therefore, physical keyboards present the fundament and reference benchmark for designing soft keyboards controlled by touch interaction techniques.

Most physical keyboards are based on the QWERTY design. The first commercial with this key layout was the Sholes-Glidden Type Writer invented in 1867 and manufactured in 1873 [YAM80]. The layout of the QWERTY design is based on the afford to "minimize mechanical jamming" [ZKS05]. This unintentionally also had the effect of a frequent alteration of both hands when typing. Such alteration is a "key factor in rapid touch-typing with two hands" [ZKS05]. An "average, secretarial level typist" [NF82] achieves approx. 55 wpm on a physical QWERTY-keyboard [NF82]. Alternative designs like the Dvorak Simplified Keyboard (DSK, see figure 5.2.1.1) [DMDF36] could not prevail. The DSK performance benefit is reported to be 5% to 10% [NF82], 5% to 15% [ZKS05] and 15% to 20% [YAM80]. Yamada states the the Depression in the 1930s, bad economic decisions of Dvoraok, usage of Qwerty-typewriters in World War II and a disadvantageous report by Strong in 1956 as reasons that Dvorak's design was not established [YAM80].

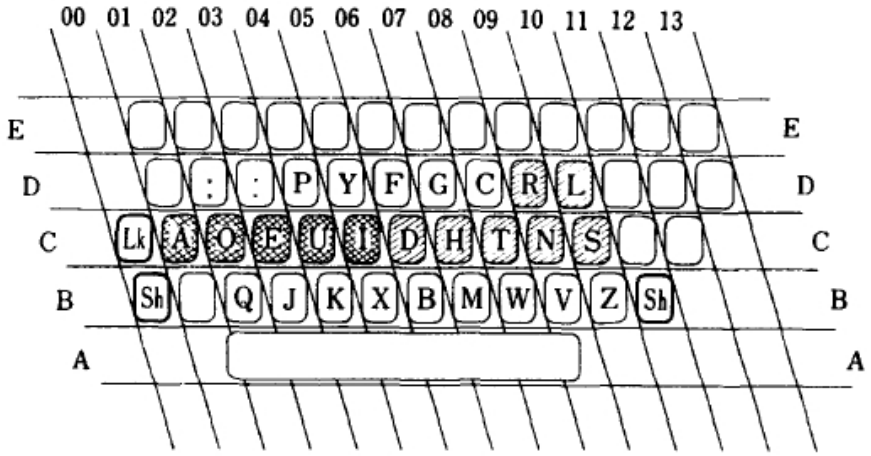

Figure 5.12: The Dvorak Simplified Keyboard [YAM80]

Due to its popularity, it is unlikely that the design of future physical keyboards will alter significantly from QWERTY. Hence, the QWERTY layout is promising as the starting

point for designing soft keyboards for usage in a tabletop environment.

5.2.1.2 Soft keyboards

Soft Keyboards are virtual representations of physical keyboards. A character is entered by selecting this character from the keyboard map. In this section we will give an overview over existing soft keyboard designs. We assume the language of text entry to be english. Thus, we exclude language-specific factors that affect keyboard layout. For example, such a factor is the transposition of the Y and Z characters on the german QWERTY layout.

5.2.1.3 OPTI

The OPTI soft keyboard layout [MZ99] is designed for text entry on a mobile device by a stylus. This means that the text is entered using one finger or a stylus. In contrast, a physical keyboard normally is operated using both hands. This constraint motivates the layout of OPTI: keys are aligned in such way that the finger or stylus has to be moved as little as possible between the entry of two characters. In detail this means that frequent digraphs are as close to each other as possible. A digraph is a pair of two letters.

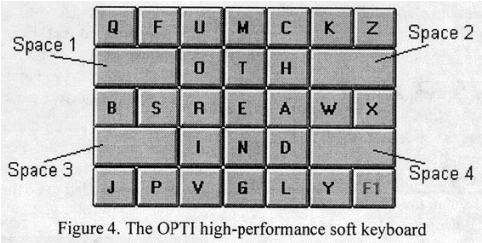

Figure 5.13: The OPTI keyboard layout [MZ99]

MacKenzie and Zhang also propose a theoretical model for predicting the performance of a keyboard in terms of words-per-minute. This model is based on “Fitts’ law to predict the time to tap a key given any previous key” [MZ99]. Each of the 27 x 27 movement times between two characters are then weighted with the corresponding digraph probability. However, this model has been proven to be imprecise [ZSA02, ZHS00]. MacKenzie and Zhang predicted a upper-bound performance of 58,2 wpm. Zhai et al. developed a different prediction mechanism basing on empirical data which predicts OPTI performance to 42,8 wpm [ZSA02].

The authors conducted a longitudinal study to compare their OPTI design to QWERTY. In fact, this is the only longitudinal study in which the QWERTY soft keyboard layout is

compared to a novel design. Five subjects took part in the experiment. The evaluation was a "2 x 20 within-subjects factorial design" [MZ99] with keyboard layout and session being the factors. Each session lasted 45 minutes, divided into two blocks of 20-22 minutes. Each half session the keyboard layout was changed, "in alternating order from session to session" [MZ99]. Subjects were to enter text phrases. Each phrase contained about 25 characters, Furthermore, the phrases "were chosen to be representative of English and easy to remember" [MZ99]. Each subject performed 20 sessions. Subjects were instructed to ignore errors. However, an error was marked by an 'click' sound.

The apparatus was a touch-sensitive WACOM device combining digitizing tablet and LCD display. The text that was to be entered was displayed in the top of the window. The currently typed text was displayed directly underneath. The actual soft keyboard was displayed below the two text lines, but still in close vicinity. Results show that participants performed poorly initially using the OPTI layout (17 wpm). The QWERTY layout performed better initially (28 wpm). However, with ongoing practice, the OPTI layout cought up, reaching the crossover point to QWERTY at the 10th session. This corresponds to four hour of practice.

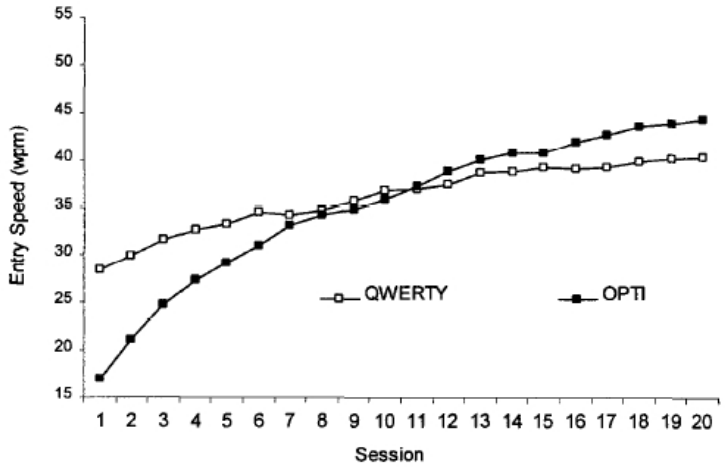

Figure 5.14: Evaluation result: Entry speed to number of session [MZ99]

The OPTI performance reached 45 wpm after 20th session. QWERTY reached 40 wpm. Summarized, this study provides guidance about how much training users need to get accustomed to a new keyboard layout. However, the actual OPTI layout is not transferable to a tabletop environment, since it is designed for one-finger usage. This contradicts the paradigm of frequent alteration of two hands on a physical keyboard. This means that it is unclear if the findings of MacKenzie and Zhang are transferable to a soft keyboard in a tabletop domain.

5.2.1.4 Metropolis

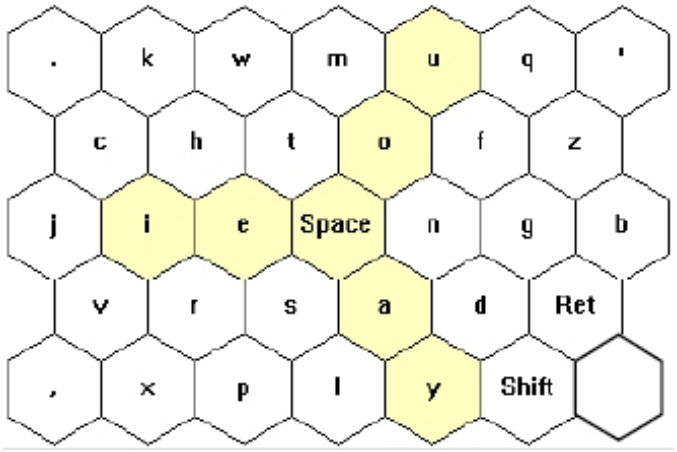

Figure 5.15: A metropolis layout. "Particularly interesting with this layout is that the vowels are connected symmetrically" [ZHS02]

The Metropolis design by Zhai et al. [ZHS00] is another soft keyboard design for use on a mobile device. Like OPTI, the Metropolis design attempts to minimize finger/stylus movement distances during typing. Particularly interesting in the Metropolis design is the way of the keyboard's layout composition. Zhai et al. consider one key as an atom and the finished keyboard layout as a molecule. The goal now is to find the minimum energy state in this 'molecule'. Energy in this case is defined as the mean time for typing a character. This mean time is determined using Fitt's law movement time prediction weighted with the probability of two letters being adjacent:

$$e = \sum_{i=1}^{27} \sum_{j=1}^{27} \frac{P_{ij}}{IP} \left[Log_2 \left(\frac{D_{ij}}{W_j} + 1 \right) \right] \tag{5.1}$$

$Log_2 \left(\frac{D_{ij}}{W_j} + 1 \right)$ is "the Fitts' law prediction of time to move a stylus from key i to key j for a given distance D_{ij} and key size W_j." [ZS01]. IP is the Index of Performance. Zhai et al. assumed IP to be 4.9 [ZS01]. P_{ij} stand for "the frequency of digraph ij in English text" [ZS01].

Basing on this energy function, a 'random walk' though the keyboard design space is conducted. This means that in each step, "the algorithm picked a key and moved it in a random direction by a random amount to reach a new configuration"[ZHS00]. After each step, the algorithm decides wether to keep the new state as starting position for the next step. This decision is done by the Metropolis function:

$$W\left(A \rightarrow B\right) = e^{\frac{-\Delta E}{kT}} if \Delta E > 0 \tag{5.2}$$

and

$$W\left(A \rightarrow B\right) = 1 if \Delta E \preceq 0 \tag{5.3}$$

$W\left(A \rightarrow B\right)$ is the probability of changing from state A to state B(new). ΔE is the energy difference. Furthermore, "k is a coefficient, T is "temperature", which could be interactively adjusted" [ZS01].

The Metropolis function has the advantage of occasionally allowing "moves with positive energy change in order to be able to climb out of a local energy minimum" [ZHS00]. The authors predict the performance of the Metropolis keyboard to 43,1 wpm. However, in a following study that focused on refining the prediction model, the Metropolis keyboard's performance is predicted to be 46,6 wpm [ZSA02].

The Metropolis approach focuses on minimizing finger/stylus movement distances, which optimizes a soft keyboard's performance when used with one finger only. Soft keyboards designed for usage in a tabletop environment do not have the constraint of having to fit on a PDA-sized screen and thus are usable by two hands simultaneously, just like physical keyboards. For this reason the Metropolis design cannot be transferred thoroughly to a tabletop domain. However, an inverted Metropolis approach could be promising. Keyboards used by both hands simultaneously should be designed in such way that the frequency of alternating hand sequences is maximized [NF82]. When inverting the Metropolis algorithm, the final state should be that two letters that are probable to be typed in a row are as far from each other as possible. This could lead to a keyboard layout superior to QWERTY.

In a further iteration of the Metropolis design, Zhai and Smith biased the layout in such way that the first letters of the alphabet tend to appear on the upper left corner and the last letters in the lower right corner. A brief user study had the result that the non-biased keyboard had a performance of 8,9 wpm whereas the alphabetically biased keyboard had a performance of 9,7 wpm. Since the study only focused on comparing the two metropolis layouts, no training effect was to be shown. Twelve subjects participated in the experiment, all of them novices with respect to virtual keyboard usage.

The finding that the alphabetically biased keyboard is faster than the non-alphabetically biased is contradictionary to the findings of MacKenzie and Fisher in 1984. They predicated that an alphabetical layout has no positive impact on text entry performance [NF82].

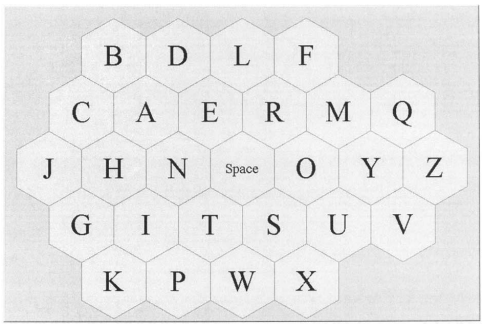

Figure 5.16: ATOMIK: An alphabetically tuned Metropolis layout [ZHS02]

5.2.1.5 Conclusion

Alternative keyboard designs suffer mainly from the large amount of training user require to benefit from a novel design. Like MacKenzie and Zhang have shown, it takes about 10 training sessions or 4 hours of practice to get accustomed to a new keyboard layout [MZ99]. Yet, these numbers are only applicable to stylus or one-finger operated keyboards. It remains unclear to which extent MacKenzie and Scott's findings can be applied to a soft keyboard used bimanually:

> "The motor skill of two-handed, eyes-free touch typing is very different from the simple act of onehanded, eyes-on tapping with a stylus on a soft keyboard" [HHCC07].

The Dvorak keyboard did not prevail as the layout for physical type writers and keyboards. However, the Dvorak keyboard is easier to learn and faster than QWERTY. If users would gain a benefit from this layout in a similar time found by MacKenzie and Zhang, the Dvarok layout could be an alternative as soft keyboard layout.

An inverted Metropolis design could also be promising as an alternative layout for text entry in the tabletop domain.

Existing virtual keyboard designs focus almost completely on mobile devices. Yet is is questionable if these designs are the best way of entering text on tabletops. Tabletops are sized large enough to allow bimanual text entry. Furthermore, tabletops are normally operated by direct touch, not with a stylus. However, tabletops still lack a physical keyboard's tactile feedback. Hence, a need for text entry techniques designed specifically for the tabletop domain is existing.

5.2.2 Designing text input techniques for tabletop devices

In this section we will present two soft keyboards layouts for usage in the tabletop domain. A tabletop environment has various requirements that differ from text input on physical keyboards or mobile devices. For example, a tabletop offers appropriate space for a full-sized QWERTY-layout but lacks the ability to provide the tactile feedback a physical keyboards offer. Not much research has been done to explore text input techniques for application in the tabletop domain. In fact, the only paper focusing on "Text-Entry Methods for Tabletop Displays" [HHCC07] merely deals with applying existing text input techniques to the tabletop domain. Hence, novel text input techniques specifically designed for the tabletop domain could increase text input efficiency compared to the status quo.

We decided for the use of a soft keyboard as the fundament for our novel text entry techniques. Yet not studied specifically for the tabletop domain, soft keyboards for mobile devices perform considerably better than alternative techniques. This has been dealt with in the last section. We now make the hypothesis that a bimanually operated soft keyboards would at least not suffer in terms of performance compared with a stylus-operated mobile-device soft keyboard. Bimanual text entry is used in everyday computing. Therefore users are accustomed to using both hands for typing. This will likely be transferable to the tabletop domain, provided a keyboard sized sufficiently. Hence, a bimanually operated soft keyboard would also outperform techniques like handwriting, speech recognition and gestural alphabets.

Furthermore, we have the hypothesis that bimanual activity leads to a performance benefit compared to a unimanual controlled keyboard. This is due the finding that a frequent alteration of both hands when typing is a "key factor in rapid touch-typing" [ZKS05].

Since the QWERTY layout is the de-facto standard for text input in desktop computing, we also use this layout as a fundament for our designs. Users are likely to benefit from being already accustomed to text entry with QWERTY keyboards. This also was a result of a study by MacKenzie and Zhang that investigated novice experience with soft keyboards:

> "users experienced with desktop systems are expected to benefit from skill transfer when switching to a soft keyboard with a Qwerty layout" [MZ01].

In their study (twelve participants, within-subjects), novices reached 21.17 wpm with a QWERTY keyboard and only 5.34 wpm with a randomly arranged [MZ01]. Note that in MacKenzie's and Zhang's study the soft keyboards were used with a single stylus.

However, this finding is transferable to a bimanually operated soft keyboard, since user would also have to search the desired letters.

We used the german QWERTY-layout in the context of this thesis.

Alternative keyboard layouts could bring a performance benefit, but would also require extensive user training and longitudinal studies. This effort of time is not realistic in the scope of this thesis.

5.2.2.1 Design principles

We find the following design principles to be crucial for text input techniques via a soft keyboard on a tabletop. These principles were developed in relation to [HHCC07] and our own experience in entering text. Moreover, we expect the number of users capable of writing blind to be considerably lower than the number of users needing to search letters from the keyboard at least some of the time. Hence, we will design our techniques in such way that they fit the latter user group's needs.

1. Efficient typing: Efficiency in this context means "the effective text entry speed a person can reach using a certain text-entry method" [HHCC07]. A user is able to hit targets of a size of approx. 1x1cm effectively by touch input [VB07]. However, according to the optimized initial impulse motor control model, we point with our fingers in the following way: "an initial high-velocity ballistic movement is made in the direction of the target. If the ballistic movement ends on the target, the task is complete, but if the movement under- or overshoots the target, a second lower velocity corrective movement is used in the direction of the target" [Cas08]. The smaller a target now is, the more such corrective movements are needed. This means that though a target sized 1x1 cm is hittable effectively, yet larger targets can be hit faster. Brought to our keyboard design, we can use the available screen to supply the user with larger key fields than this is possible on mobile devices. However, the larger a key is, the longer is the distance between the keys and thus the selection time for a key rises. This tradeoff has to be considered.

2. Appropriate feedback: Feedback is one of the most important parts when designing text input techniques for the tabletop domain. This is because of the following reasons: Soft keyboards lack the tactile feedback a physical keyboard offers. Yet tactile feedback is vital for typing text without looking at the keyboard. This is especially relevant for users not capable of writing totally blind. This means that a user operating a soft keyboard on a tabletop would have to permanently look at the soft keyboard in order to hit the keys rapidly. When now having the visual focus on the keyboard, the text that is actually typed is out-of-focus. Hence,

errors are not or later noticed. This is especially relevant for the tabletop domain. Having a display of up to 70", the distances the visual focus has to cross are much higher than for PDA-sized devices. With appropriate feedback, we think that this error-proneness can be reduced.

3. Ergonomic placement: Tabletops differ from mobile devices mainly in one point: screen size. Tabletops are sized from 30" (MS Surface [Mic08]) up to 70" (University of Konstanz Tabletop prototype) or, in special cases, even larger. This means that the far end of the tabletop can well be 1 meter away from the user. If the user now wants to select a target at this far end, she would have to reach out far with her arm. Bringing this issue into the text input context, a keyboard placed at the tabletop's far end would require the user to permanently stretch out her arms. This would probably to arm fatigue and thus would not be accepted by the user. Hence, the user has to be supplied with a text entry technique allowing keeping the arms close to the body while typing.

We developed two text input techniques for direct touch usage on a tabletop. In the next sections we will deal with the design and the implementation of these two techniques. The techniques were implemented for running on a 34" 4:3 tabletop display with a resolution of 1024x768 [RSF+08]. When transferring the keyboards to differently sized environments, the keyboards would have to be scaled accordingly.

5.2.2.2 Qwerty

In this section we highlight our Qwerty soft keyboard design. User enter letters by touch-typing on the letter's visual representations on a keyboard map. We attempt to reduce error rates by highlighting a just-hit key and by presenting a large (approx. 20 cm in height) representation of the last hit character over the keyboard.

5.2.2.2.1 Design The Qwerty design is a soft keyboard in a traditional QWERTY-layout. Its basic principle is that the user keeps her visual focus on the keyboard while typing in order to maximize typing speed. Several types of feedback are to minimize error-proness.

Due to the Qwerty design, experienced users benefit from familiarity to the Qwerty layout and are thus able to predict where one key roughly is. For example, when wanting to type a Q, the user will implicitly know that the letter is somewhere at the upper left corner. We expect this benefit to enhance text input speed since the motor movement patterns are similar to the patterns performed when using a physical keyboard. Movement patterns we regularly perform are stored in our motor memory and can thus be

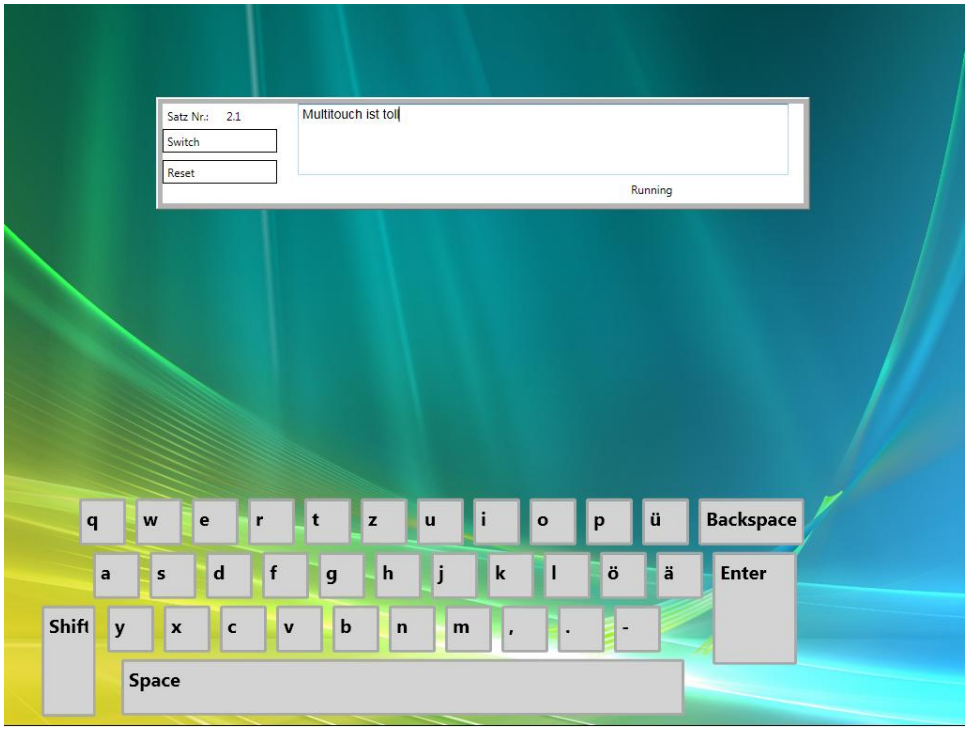

Figure 5.17: The Qwerty soft keyboard. A letter is entered by touch-typing on the corresponding virtual key. The white box was used for our evaluation (see section 5.2.3) and is not part of the keyboard design.

recalled rapidly [KHT06].

Since user are already likely to benefit from accommodation with the Qwerty layout, we wanted to further maximize performance. Concretely, we supply the user with keys large enough to be hit rapidly. One Key is sized approx. 2 x 2 cm. This size is a tradoff between the selection speed of a key and the maximum keyboard size. The size of 2 x 2 cm allows one key to be hit without many corrective movements, according to the optimized initial impulse motor control model. Moreover, the overall keyboard has a width of 43 cm, which is within the spacial bounds of a physical keyboard (33 cm). This means that when using our soft keyboard, the hands do not have to be moved significantly more than during the usage of a physical keyboard. Hence, the hand's and arm's motor load are unlikely to be overstressed.

Letters are inputted via touch-typing. That means that a letter is put in by discretely selecting the corresponding key from the keyboard map. We chose this technique for the following reasons: First, we enter letters on physical keyboards in the same way. Hence, users are likely to expect analog behavior from our soft keyboard. Second, we don't need an explicit state for text entry, since the keyboard in principle is an overlay widget. This means that other interaction techniques such as selection, dragging or scrolling can still

be used. A letter is put in when a finger touches the key, not when the finger lifts off the key. This is analog to typing on a physical keyboard, where a letter is entered by pressing a key down.

User can access capital letters by tapping on the Shift key. After hitting the Shift key, the next letter will be entered uppercase. Afterwards letters are entered lowercase again. This design has that user don't have to tap Shift and the corresponding letter simultaneously. Such simultaneous tapping is difficult on a tabletop due to the lack of tactile feedback. Since for most words only one (the first) letter is capital, we decided to automatically switch back to lowercase mode. This technique is spread in text input on mobile devices. E.g. Windows Mobile Ver. 6 lets users enter capital letters in such way.

The Space, Enter and Backspace keys are aligned analogically to physical keyboards.

The keyboard is triggered by simultaneously placing five fingers on the display. After that gesture, the keyboard gets visible and the feedback text "Switch to Mode: Keyboard" is displayed and faded out at the hand's position. The keyboard is always placed at the lower end of the tabletop. However, its horizontal placement is defined by where the user wants to type. This means that the keyboard always is directly below the position at where the text is entered. By placing the keyboard at the tabletop's lower end, the arms can be kept close to the body while typing. This reduces arm fatigue.

When now typing, we have the hypothesis that user keep their visual focus on the keyboard. This is due to the lack of tactile feedback in contrast to a physical keyboard. Users have to search each key before being able to perform a selection on this key. This distracts the user from the actual text that he types. When he misses a key or selects the wrong key, the user needs longer time to notice the error than he would need if using a physical keyboard. This is especially relevant for tabletops, since the possible distance between the position of the actual text entry and the keyboard can well be greater than 50 cm. Such large distance require the user to switch their visual focus over long ways. An example for such a scenario is the entry of a URL into a web browser. A browser's navigation bar is normally positioned at the top end of the window and thus at the top end of the tabletop.

For this reason, we provide the user with two types of feedback. First, a key flashes in orange when hit. This means that the key's background color is changed to orange and then faded back into the standard gray. The color orange lifts off from the standard gray since the contrast is highly visible. This feedback lets the user know if a key has been hit. Since keys are larger that the area occluded by the finger, the orange flash can be seen when the finger has contact to the key.

However, this kind of feedback is not sufficient for minimizing error rates. The orange

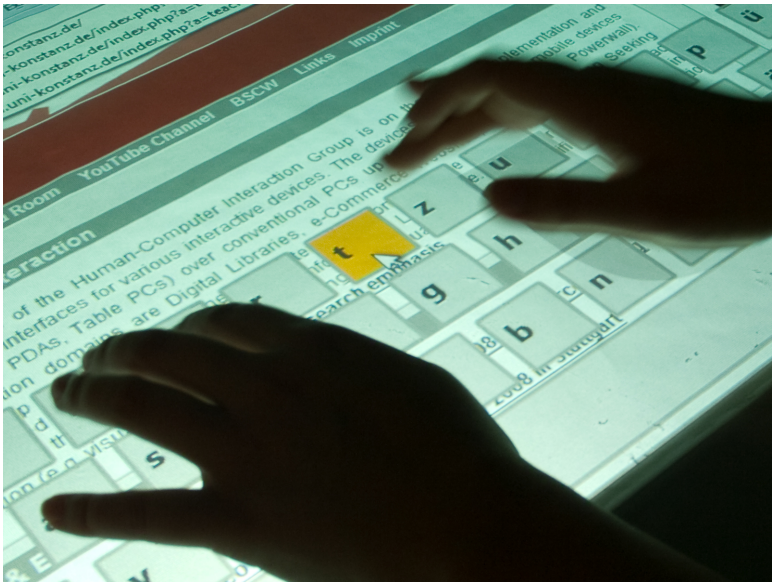

Figure 5.18: A key flashes in orange after being hit.

flash only indicates if a key has been hit, not if the key was the correct one. More important, we observed that users tend to quickly change their visual focus when typing. This means that the visual focus doesn't dwell on a key until the user is sure that the key has been hit. We observed that users tend to directly search the following key after typing one key, without waiting until they are sure if the letter was correct. Hence, the orange flash is not noticed to the extent we expected.

For this reason we built in an additional type of feedback. We display the character last typed in a large format directly above the center of the keyboard. This feedback letter has a height of approx. 10 cm. After getting visible, it is faded out within 0.4 seconds. This large format letter is visible in the peripheral visual field. That means that a user doens't have to exactly focus on the feedback letter but is able to recognize it during the search for the next key that is to be typed. That user benefits from this feedback in two ways: First, she knows whether a key has been hit. If no feedback letter appears, no key was triggered. Second, the user recognizes if the correct letter has been hit. This way a user is able to immediately identify and correct an error.

5.2.2.2.2 Implementation The keyboard is implemented as a transparent WPF window. The keys were aligned manually in this windows. Each key is of the WPF type Border, containing a Label. The content of this Label is the letter corresponding to the key. The evaluation of which key has been hit by the finger is done iteratively by comparing the finger's position to each Border's position and height/width.

The orange flash that appears on a key after selection is done by a WPF ColorAnimation.

Figure 5.19: The user has just hit the letter “k”. This letter is displayed over the keyboard for a short duration and then faded out.

This ColorAnimation fades from a starting color into an end color over an adjustable timespan. For our flash, the key’s color is first set to orange and then faded into gray over 0.4 seconds.

The large feedback letter that appears after entering a key is implemented as a separate WPF Window. That way the feedback letter doesn’t have to be within the keyboard windows’s boundaries. The fading out of the feedback letter is done by a WPF DoubleAnimation over the property Opacity.

An issue with the implementation is the successive entering of the same letter. Each finger is assigned an unique ID after being received from Squidy [JKR09]. If a finger is lifted, our prototype waits a certain time threshold before removing this ID. This way a letter typed rapidly twice in a row is assigned the same ID both times. Thus, only one tap is recognized and unindented successive letter entries are prevented. However, if a user wants to type the same letter successively (e.g. www in a web browser), this threshold prevents her from rapidly entering the letters. A solution for this issue would be to trigger a key at the lifting of a finger, not at its moment of touch. Yet this would delay letter entry and the keyboard would be perceived as being slower by users.

5.2.2.2.3 Discussion

- The feedback only lets the user recognize if the last letter was correct. This way spelling errors that are not recognized immediately are hard to identify. Some

errors are identified not until the user re-reads a sequence of typed words. Such correctional reading requires the user to switch visual focus to the text cursor's position. It is unclear to which extent correctional reading would take place while entering text.

- The Keyboard has to be positioned close to the body in order to enable ergonomic arm postures. If the keyboard is positioned away from the display's border the user would have to stretch out her arms, which would lead to fatigue. Hence, if the text cursor is distant, the user would have to switch visual focus in order to correct-read the typed text. We have the hypothesis that such focus switching is done much fewer than when using a physical keyboard. This is because the user has to identify a key's exact position with her eyes, since there is no tactile feedback. As a result, we expect users to conduct more errors when using a soft keyboard. The feedback we provide might reduce such error-proness, yet it is unclear to which extent errors are reduced. Moreover, we cannot say definitely how well the feedback is perceived by the users.

- Rotatability has not been implemented yet. Such rotatability is needed since users can approach a tabletop from all directions.

5.2.2.3 Column Typing

Column Typing is our second design for text input on a tabletop display. Its basic principle differs from the QWERTY design in the way that the user is to keep her visual focus near to the text typed in and not on a keyboard displayed close to the body. Hence, the keyboard is displayed close to the position where the text is put in. This reduces the distance the visual focus has to cover and thus makes recognizing errors easier. A key is selected by setting a finger on the corresponding horizontal position, while the vertical position can be arbitrary. Hence, one key has exactly one column on the display.

5.2.2.3.1 Design With Column Typing, we wanted to make error recognition as easy and fast as possible. For this reason, the position of the keyboard on the screen is set dynamically 50 pixels below the text cursor's position. Hence the keyboard is close by the typed text. This means that the user's visual focus has to cover only a small distance in order to check the text typed in.

Moreover, we wanted to maintain a ergonomic position of arms and hands. If the user would have to type on a keyboard on the far end of the tabletop, arm fatigue would be probable. So we decided to let the user select a key by only having to set a finger to the key's horizontal position. This kind of input is impossible for a standard QWERTY-

Figure 5.20: The Column Typing design. A letter is defined by placing the finger onto the same horizontal position as the desired letter's. In this example, the finger is below the letter "r". A letter is entered by lifting the finger. The white box was used for our evaluation (see section 5.2.3) and is not part of the keyboard design.

layout, because the keys overlap in vertical direction. For example, the letter A overlaps with the letter Q. To overcome this issue, the key map is stretched in horizontal direction. This means that no key overlaps with a different key in vertical direction. The stair-like layout has nevertheless been inherited from the standard layout. This is to establish a familiar look and thus guides the user when she is searching for a key. We call the letter that will be put in when lifting the finger the active letter.

This design enables the user to use both hands for text input. We expect the motor load to be roughly split by half for each hand. Hence, each hand is responsible for selecting letters from one half of the keyboard. Such split of motor labor is one of the key factors for rapid typing on physical keyboards. With our design, users are unlikely to use more

than their two index fingers. However, we still expect the user to benefit from bimanual motor labor since typing with the two index fingers is also part of bimanual text input via a physical keyboard.

Summarized, the keyboard is displayed close to the text cursor, but the hands can remain close to the body.

This design has the drawback that users cannot directly select a key but rather have to predict a letter's column blindly and the set a finger into this column. This prediction cannot always be correct. We expect a novice's selection precision to be within the two surrounding letters of the desired letter. For this reason, we have to provide the possibility of corrections. This is accomplished by triggering a letter at the lifting of a finger. Thus, in contrast to Qwerty, a user can correct the letter that will be selected by moving her finger to the left or right.

Furthermore, we wanted a column to be able to being hit as predictable as possible. This is achieved by making one column as wide as possible. Like with the Qwerty keyboard, we have to consider the trade-off between keyboard size and finger movement distances. The higher the width of a column, the more probable a user will hit this column with the initial attempt. However, the higher the width of a column, the longer is the way a finger has to be moved from one letter to the next one. We decided to make the keyboard 54 cm wide. This corresponds to approx. 2 cm for each column. This width is about 20 cm wider than a physical keyboard. With our prototype, it covers 90% of the screen width.

For being able to select a letter correctly, the user has to know which letter's column her finger's position corresponds to. We display this key in a larger font size and in bold shape. This visibly lifts off the letter that will be selected by lifting the finger from the other letters.

Yet when testing this design, we quickly noticed that this kind of feedback was not sufficient. The user sees which letter will be triggered by lifting the finger, but typos that are caused by accidentally misspelling the word are not eliminated with that feedback. Moreover, we observed during testing that the visual vicinity of keyboard and text cursor is not sufficient for enabling the user to rapidly spell check her inputted text.

Hence, we decided to not only position the keyboard close to the text cursor, but also display some of the inputted text directly next to the currently active letter. In detail, we display a history of the last ten letters at the left hand side of the active letter. A border surrounds the history as well as the active letter. A line is drawn below the history text and extends up to the active letter. The history area has an translucent orange background. This lifts off the history text from the keyboard map and makes reading the history text easier.

With this design we establish a close connection between the history text and the active letter. The history text and the active letter together form the subsequent text. So the user can see the text that will emerge from selecting the active letter before the letter is triggered. Hence, we expect typing errors to be noticed earlier.

Additionally to entering letters, we also enable the user to rapidly access the Backspace, Space and Enter commands. Space is the letter most used when entering text [ZHS02]. Rapid access to the Backspace command is needed for correcting phrases quickly. The enter command also is used regularly, e.g. for setting the text cursor into a new line or navigating to a entered URL in a web browser.

For executing Backspace, Space, Shift and Enter commands we defined areas at the left, lower and right borders of the display. The left border for Shift, the lower one for Space and the right one for Enter and Backspace. The right area is divided in half vertically, each half responsible for one task.

Each area is 60 pixels wide and high respectively. If a user now wants to access such a command, he has to tap with a finger into the corresponding area. Hence, Backspace, Space, Shift and Enter commands can be executed without having to precisely select a letter from the keyboard map. We expect the border areas to be large enough to be hit rapidly. This way the user can access such often used commands much faster than he would be able to through selecting the commands from the keyboard map. Subsequently, the overall text entry speed will also rise.

Contrary to the Qwerty design, we use a own text entry state for the Column Typing design. This state is triggered by putting five fingers simultaneously on the tabletop.

5.2.2.3.2 Implementation

Like all other on-screen widgets in our prototype, the Column Typing keyboard also is implemented as a transparent WPF window. This window has the extent of the size of the keyboard map. This map is 825 pixels wide on our 34" 4:3 tabletop with a resolution of 1024x768 [RSF+08]. The border areas for Backspace and Enter (left and right) each consume 60 pixels, which is about the rest of the totally available display width, leaving a small gap to the keyboard for preventing unindented hits of Enter, Shift, etc. The area for the Space command at the lower border of the display has a height of 60 pixels. Each Backspace, Space and Enter area is a independent WPF window.

The history text also is displayed in a separate window. This window is positioned as soon as a finger touches the tabletop.

5.2.2.3.3 Discussion

- The technique of selecting a letter by lifting the finger might not be the only reasonable way of selecting a letter when using the Column Typing design. We also developed an alternative design to select letters. This alternative design required users to perform a up- or downwards gesture with a finger to select a letter. We expected this design to help the user in selecting the correct column, since a finger could touch the surface all the time. Hence, a user 'scrolls' through the letters by moving her finger left or right. So the preciseness of selecting the right column is improved. The user always has feedback about the current active latter and doesn't have to initially guess to correct column. Yet we found this design to be inferior to the design in which a letter is selected by lifting a finger. This is primarily due to the gesture. The gesture takes more time to perform than the lifting of a finger. This lowers overall text entry speed. The improved preciseness did not make up the decrease of text entry speed.

- Column Typing requires a distinct text entry state. This is because letters are not selected directly like in the Qwerty design but are selected by only setting a finger onto a letter's horizontal position. Hence, all other interaction techniques have to be disabled as long as the text entry state is active. Otherwise, the user's typing would not only cause letters to be entered but could also execute selection and dragging tasks. However, the user would always have to switch the text entry state on for entering text and back off again for interacting with the system in normal way. This is especially relevant when a user wants to enter many small pieces of text and interact normally with the system in between. In this case the user would have to switch into text entry mode, switch back for setting the new text cursor position, enter the text mode again and so on. The Qwerty design doesn't have this drawback.

  However, the Column Typing keyboard is always displayed below the text cursor. Hence, only the area below the keyboard could be used for selecting letters. The area above the keyboard then could be used normally e.g. for doing selection or dragging tasks. Of course this only partly solves the issue that we need a state of its own.

- The Column Typing design is not rotatable. The user always has to be able to predict where the column of a certain letter is. If the keyboard would be rotated, such prediction would be much harder since the user could not orient himself at the tabletop's borders. However, a tabletop can not only be approached from one direction but rather can be used from every side. Hence, the virtual keyboard should be alignable by the user. However, for using WIMP this is less relevant. WIMP's alignment goes from top to bottom. Hence, a user is likely to use a tabletop from WIMP's bottom side. Nevertheless, you could also imagine that WIMP's windows themselves could be rotateble in a tabletop environment.

Summarized, Column Typing binds the user to one side of the display.

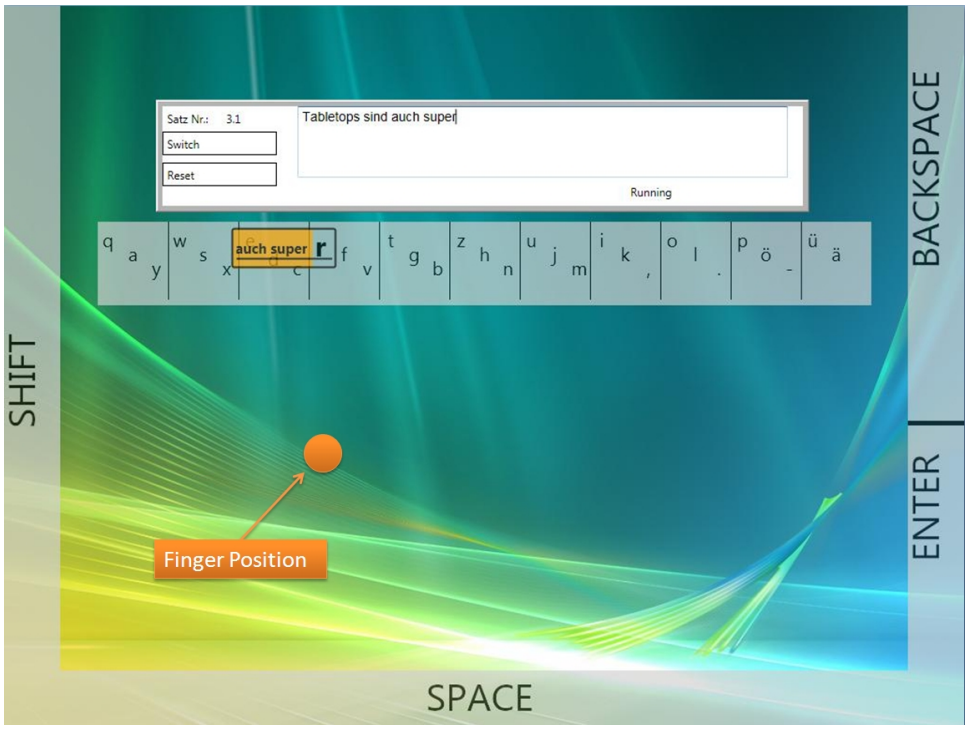

Figure 5.21: Since users have to hit one key's column, rotating the keyboard would decrease selection precision.

- The borders for executing Backspace, Space and Enter commands do not scale for display size. If the tabletop grows in size, the borders depart from the the user. This means that a user standing centered at on side of the tabletop would have to reach out wide to reach the borders. A solution would be to position these areas fixed in relation to the keybaord map.

- A further issue is that we not only need letters for text entry but also numbers and special signs, as well as capital letters. We designed a technique for switching between several sets of keys. In detail, a user has to set two fingers on the display. When now keeping one finger fixed at one position and moving the second finger vertically, the keyboard switches between several sets of keys. Such sets are capital letters, numbers and special signs. One switch is triggered as soon as the second finger has moved beyond a certain distance threshold. However, this technique has not been implemented yet.

5.2.3 Evaluation

We conducted a small user study with six participants. In this study we compared our soft keyboard designs among each other and in reference to a physical keyboard. The study focused on qualitative factors. Factors were the participants' subjective ratings about the feedback components and the keyboard layouts as well as the extent and the speed to which the participants got accustomed to our designs.

Moreover, we informally analyzed performance. Performance was measured in terms of text input speed and the rate of typing errors. These factors give clue about how the different designs perform compared to each other and against a physical keyboard without interference of subjective perceptions. Since there are no empirical studies about soft keyboard performance on tabletops, these factors provide us with a baseline for future comparisons.

In their current design, we expect both soft keyboard designs to have different up- and downsides. In short, the Qwerty layout is expected to perform faster but results overall in more typing errors. The Column Typing design is expected to better support the user in recognizing errors than Qwerty, but also is slower.

However, we cannot definitely say how the designs would be perceived by users. Since the two designs differ in many ways from each other, users are likely to experience the two designs very heterogeneously. For further design iterations, we need to know which design is preferred by users and, more detailed, which parts of which design provide most benefit to the user.

5.2.3.1 Hypothesis

The study's main scope was about finding out how users cope with the different kinds of feedback and layout.

Due to our earlier testings and observations we have found the following hypotheses about the participants' perception and the performance of our soft keyboard designs:

1. Participants will feel that entering text using the Qwerty design is faster and more fluent. We expect this since the Qwerty soft keyboard's layout is closer related to a physical keyboard than the Column Typing layout. In detail, the motor labor a user has to perform when using Qwerty soft keyboard is very similar to the

the motor labor needed for operating a physical keyboard. The motor memory supplies us with the ability to store and recall such movement patterns [KHT06]. Hence, users will likely benefit from typing patterns they are accustomed to from using physical keyboards.

2. Errors that are not noticed immediately after occurring will be recognized more likely when using the Column Typing design. The Column Typing design provides a history of the last 10 letters that is always display directly to the left of the currently selected letter (see 5.2.2.3). We expect this to aid the user in rapidly correct-reading the last typed characters. The Qwerty soft keyboard only presents the last character entered and thus offers no way of correct-reading a sequence of entered characters without looking at the actual typed text.

3. Participants will not get accustomed to the Column Typing design instantly. The Column Typing is in principal a stretched Qwerty-layout. However, the user selects a letter by putting her finger below the desired letter (see 5.2.2.3) and not directly onto the letter. This means that users can no longer perform the direct touch-tapping they are used to from operating physical keyboards. We expect this to require some training time from the users.

Beyond these hypotheses, we focused on the following explorative research objectives:

- How certain do participants feel about the definite entry of a character? This is vital for making the keyboard usable. As long as users would feel unsafe about using a design, they are unlikely to use this design even if it would be faster.

- How well are Space, Backspace, Shift and Enter reachable? These commands are required frequently. Thus, users have to be able to reach these keys with little effort.

- Which strategies do participants use to share motor load between both hands? Rapid alteration of both hands is a major factor for the efficiency of physical keyboards [NF82]. Hence, it would be interesting to see if such behavior is automatically adapted by the users when operating a soft keyboard.

- How reliable are the provided feedback components support users in recognizing typing errors? Since we expect text entry via a soft keyboard on a tabletop to be more error-prone than typing with a physical keyboard, we have to know how well our feedback components are usable and where they could be improved.

- How fast are errors recognized? An error is noticed either at instantly its occur-

rence or later while checking a compound phrase. Qwerty's feedback components supports the first, Column Typing's the latter. We need to know which feedback components fit better to the users.

- Which of the two designs is overall preferred by the participants? This gives us clue about the overall user perception of our designs.
- Do users change their typing behavior after using a technique for a short time? This would give clue about how fast users can get accustomed to a text input technique.

5.2.3.2 Apparatus and participants

The text input techniques were provided as described in sections 5.2.2.2 and 5.2.2.3. Both layouts offered the same possibilities in terms of available characters and keys to the user. As physical keyboard we used a standard Windows keyboard manufactured by Dell.

The experiment was performed on a 34" multi-touch tabletop display [RSF+08]. The resolution was 1024x768. We used an Apple MacBook running the Windows Vista operating system as display computer. Multi-touch tracking was done on a HP Compaq 6190p notebook also running Vista.

A testing framework was implemented in order to provide the visual components required specifically for the test. Moreover, this framework read text phrases from a file, compared these control phrases to those entered by the participant, calculate speed and error rates and write relevant factors into a log file. The comparison was done automatically according to the condition and the trial number.

The visual representation consisted of a single text box for entering phrases and two buttons for switching conditions and resetting a trial.

Six volunteers (three female) were recruited for the experiment. Participants were aged 20 to 25, right-handed and study in the following degree programs at University of Konstanz: 2x Psychology (3rd semester BA/Diploma), 2x History (5th an7th semester BA), Sociology (1st semester MA), Romance studies (1st semester MA). They received a nice hot cup of coffee and much gratitude for their participation. All of them were using a tabletop for the first time but had used techniques for text input by touch on a mobile device before.

For accessing the subjective ratings, participants had to fill out a questionnaire at the end of the experiment. This questionnaire consisted of general questions about age, degree program, semester, gender (see Appendix A). Furthermore, six questions about the participants' subjective preference about the different kinds of layout and feedback were to be answered.

5.2.3.3 Task

Participants were to enter 12 german text phrases and 4 single words into a TextBox with each input technique. The Textbox was positioned horizontally centered and at approx. 2/3 vertical position (see figure 5.2.3.3). Hence, the TextBox was about 40 cm away from the tabletop's edge at which participants were standing.

Figure 5.22: The user test setup. In this picture the participant enters a phrase using the Qwerty soft keyboard. The letter "e" indicates that she just hit the corresponding key. The text is entered into the white TextBox.

Each set of 12 phrases plus the 4 words had about the same amount of characters (374, 373 and 376 characters respectively, including spaces) (see Appendix B). This amount

of letters was found to be appropriate for the total time of a single test to be about 45 minutes.

Text phrases consisted of 3 to 7 words and were in present tense or simple past. The phrases contained no punctuation and no complicated grammatical structures. However, they contained capital letters and umlauts. Capital letters and umlauts are fundamental parts of the german language and thus must not be ignored. Phrases were simple enough to be entered without the need of having the phrase to be dictated again.

Deleting characters was done by the Backspace command. This was necessary for correcting errors. Each trial was finished by the participant via the execution of the Enter command. This granted insight in how well the Enter and Backspace commands can be reached by the users.

Participants were instructed to focus on minimum error rate for the first six phrases of a condition and focus on speed for the second part. In each case participants were motivated to correct errors. Each set of phrases had to be written down by hand, too. This was to eliminate the consideration of errors that had been caused by misspelling rather than by the text input technique.

Participants also ran a small training and filled out a short questionnaire.

5.2.3.4 Procedure

The experiments ran after the following procedure: The participants were greeted and asked to fill out a consent form about being videotaped during the trials. Then, participants were shown each text input technique by the test controller. At this point each participant also conducted a small training, consisting of entering one phrase for each technique.

Before each condition, participants were to write down the 12 text phrases by hand. Text phrases were dictated by the test controller in each case. Instructions were given before participants actually started typing. For each condition, instructions were repeated. 16 trials were conducted for each condition. 12 trials consisted of entering a text phrase. In the remaining 4 trials, participants were to enter a single word. After each trial the TextBox for phrase entry was cleared.

Participants first conducted the trials for the physical keyboard followed by the Qwerty soft keyboard and the Column Typing design.

After performing all 3 x 16 trials, participants were asked to fill out a short questionnaire. When participants had finished with the questionnaire, a half-structured interview was conducted.

5.2.3.5 Design

We used a within-subjects design with the text input technique being the independent variable (physical keyboard, the Qwerty design and Column Typing).

Dependant variables were are the participants' subjective ratings of each technique as well as WPM, KSPC and MSD.

- WPM means words per minute. WPM is a metric for measuring text input speed. As described in section 5.2.1, wpm does not consider the actual words but rather defines one word as five characters, including spaces.
- KSPC means Keystrokes per Character. It is calculated like follows [SM03]:

$$KSPC = \frac{\mid InputStream \mid}{\mid TranscribedText \mid} \tag{5.4}$$

  This metric is especially needed for identifying how many errors were corrected by a participant before finishing a trial.

- MSD means mean Minimum String Distance. It is "is the minimum number of primitives - insertions, deletions, or substitutions - to transform one string into the other" [SM03]. Soukoreff and MacKenzie provide pseudo code of an algorithm for computing the MSD in [SM01]. This metric is needed to measure the rate of errors that have not been corrected by the participants.

Considering the participants' subjective ratings, we wanted to find out more about the participants experiences with the process of entering letters as well as with the perception and usability of our feedback components. These parts are essential for text inputs technique since it depends on them how fast and how secure the text can be entered. Subjects had to fill out a questionnaire at the end of the experiment. The questions were designed in such way that participants had to choose the technique that had better supported in a certain issue for each question. These questions were taken

as a starting point for a semi-structured interview. The questions asked in the interview focused getting a deeper insight in how the participants had been able to cope and had felt when using the different text input techniques.

To get a more controlled view on the participants' error recognition, at predefined positions a wrong character was entered even if the user had hit the right letter. Concretely, the four single words at the end of each condition were for this purpose. Within two of these four words, one character was entered falsely. This way we were able to anticipate an error and thus directly observe how and if the user recognizes this error.

Phrase sets were counterbalanced across participants.

The sequence of text input techniques was physical keyboard - Qwerty soft keyboard - Column Typing for every participant. We decided for this sequence because the physical keyboard and the Qwerty soft keyboard are similar in matters of layout. Hence, the participants will be better able to compare physical keyboard and the Qwerty soft keyboard when these two techniques follow each other. Qwerty soft keyboard was placed second in order to get an insight in how users change their typing behavior when switching between the soft keyboards. Moreover, text input speed was only to be analyzed informally and is strongly expected to be in the order physical keyboard - Qwerty soft keyboard - Column Typing (from fast to slow), see section 5.2.3.1. Thus, we do not require a counterbalancing of input techniques.

Each participant conducted a training of one text phrase per technique. This was to ensure that each technique had been understood by the user. We conducted all trainings directly at the beginning of the experiment. This and the short training duration were to eliminate premature acclimatization to a so far unknown technique.

5.2.3.6 Results

Overall, participants subjectively preferred the Qwerty virtual keyboard's strict layout they were accustomed to. This is mirrored in text entry speed (see figure 5.2.3.6.1). Yet three subjects preferred Column Typing's explicit feedback component.

5.2.3.6.1 Tendencies of measured text entry speed and error rates Informal analysis of text input speed reveals the physical keyboard performs at 47 wpm in mean over all trials and subjects. The Qwerty soft keyboard performs at averagely 21 wpm, the

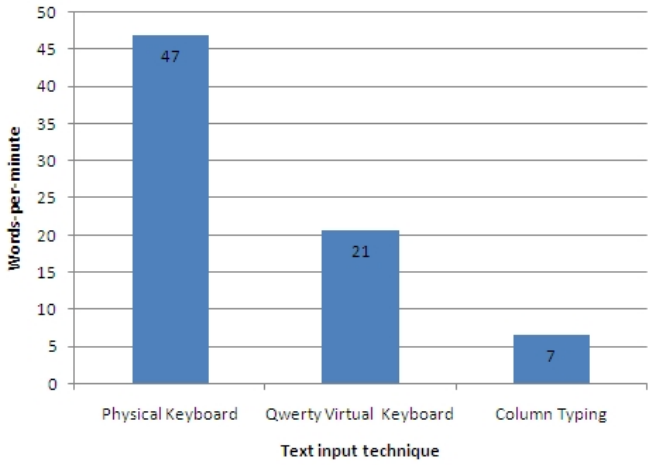

Figure 5.23: Mean text input speed to technique

Column Typing design at 7 wpm (see figure 5.2.3.6.1).

The minimum string distance (MSD, see section 5.2.3.5) did not differ considerably in the part in which participants were to enter the text with least errors possible (the first six phrases of every condition). However, the rate of error corrections performed (KSPC, see section 5.2.3.5) was considerably higher for the Column Typing design for these first six phrases (20% higher, in mean over all participants). This matches our observation of participants being less secure in entering a letter with the Column Typing design but at the same time being able to immediatly recognize and correct errors.

For the part in which participants were to enter text phrases as fast as possible (the last six phrases of every condition), the Qwerty virtual keyboard resulted in considerably less uncorrected errors. The mean Qwerty virtual keyboard's MSD for per fast-typed phrase averagely was 0.6, the Column Typing's 2,17.

These results match with our hypothesis of the Qwerty design being faster handable by users. Column Typing's lower speed and higher error rate require further insight.

5.2.3.6.2 Subjective ratings Participants had to fill out a short questionnaire about their subjective preference of the designs' different components (see figure 5.2.3.6.2. These answers were taken as a baseline for a semi-structured interview digging deeper into the participant's experiences with using our keyboard designs.

Results show an overall preference of the Qwerty layout. Yet, 3 participants felt better supported by the Column Typing layout in checking the typed text and in noticing errors. This partly matches our hypothesis of the Column Typing layout better supporting error recognition. The participants' reportings provide further insight into why each technique

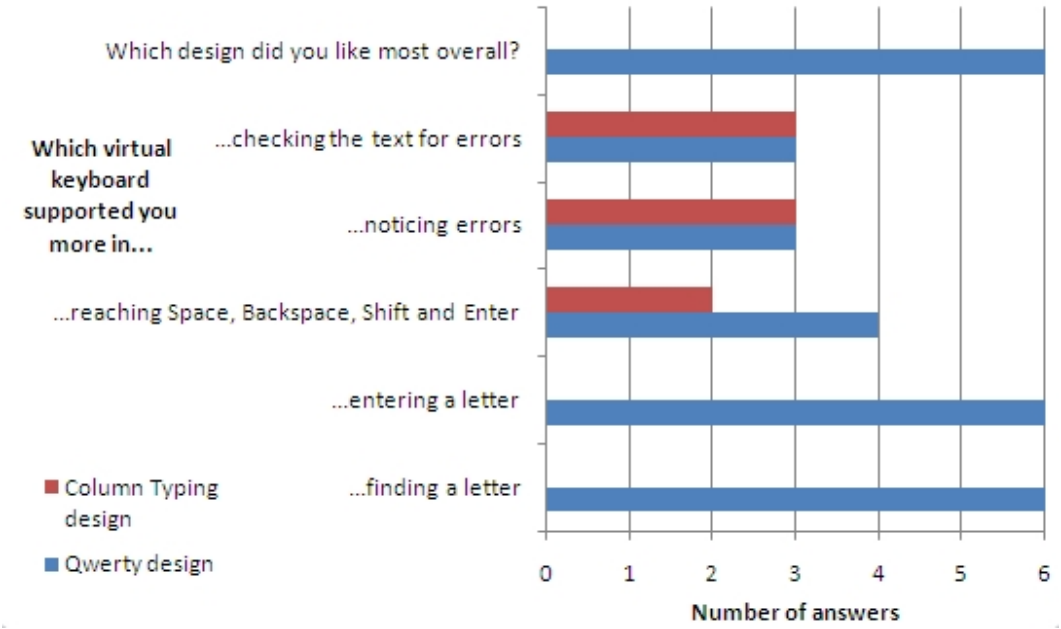

Figure 5.24: Questionnaire results

was perceived its way:

- One major concern all participants reported was the difficulty of selecting a key using the Column Typing layout. Participants stated that selecting a key by positioning the finger somewhere under the key handicaps fast and secure selection. In order to further inspect this issue, we have to take a look at the observations of Column Typing selections during the experiment. These observation have shown subjects used primarily two selection strategies. Note that the users were not biased in what strategy to use when using Column Typing.

    1. Three participants used direct touch-typing and did not check if the letter was correct before releasing the finger. Since users cannot be completely certain if a finger is within a specific key's bounds when placing the finger somewhere under the key, selections without checking if the key is correct, are insecure. Our observations showed that the selection precision was within the surrounding two keys of the desired one. This matches our previously expected initial precision. All three subjects performing this technique used bimanual interaction. Each hand hereby was responsible for about half the keyboard. This issue supports the presumption that direct touch-typing is a very natural way of selecting targets.

       For overcoming this issue, users would have to be trained to perform a touch-correct-release process for selecting keys. Yet this would be impractical in a

scenario in which users need to be able to use a techniques without any training time, e.g. in a public environment. Furthermore, the width of each key's column could be widened. However, our display already used all available space for the Column Typing keyboard. Moreover, three users reported that the long ways their hands had to move when using Column Typing affected them negatively.

In contrast to Column Typing, Qwerty's text entry speed and error rate were perceived all positively by the participants. Observations showed that all participants immediately used bimanual interaction when typing on the Qwerty keyboard. This finding supports the hypothesis that users adapt faster to the Qwerty design.

2. The other three participants used sliding for selecting keys. This means that they placed their finger at an arbitrary position and then moved the finger to the desired key, with the finger having contact to the surface. Hereby one might expect that participants check the letter before releasing the finger, since the desired letter is reached with the hand already having contact to the display. However, we observed that participants tended to release their finger as soon as they expected to have hit the key. This raises error rates because participants cannot be certain whether the finger is safely within in bounds of a key or somewhere at the border to the next key. If now being at the border to the next key, only a small finger movement during lifting can cause the wrong letter to be selected.

   Besides widening a key's column (see above), users could be supplied with feedback providing certanity about the exact finger position. For example, a vertical line could be displayed from the keyboard down to the finger. This could help user recognizing not only if a finger is within a key's bounds but also where exactly in this bounds the finger is. Hence, users would likely set the finger more to the center of a key's column, reducing the number of wrongly selected keys.

- Participants reported difficulties in finding a key when using the Column Typing layout. When being asked if the layout looked farmiliar, no subject could identify the key alignment to be a stretched Qwerty-layout. One participant stated that he somehow 'knew' where a key roughly was before visually searching that key. A similar behavior was observed with the other participants, yet Column Typing did not reach the velocity of key identification the Qwerty layout did.

- Concerning the question of reachability of the Space, Backspace, Enter and Shift keys, participants answered diversly. Three explicitly stated that they felt those keys to be too far away in the Column Typing layout. However, two other partic-

ipants explicitly reported to have benefitted from those keys' size in the Column Typing layout. We can learn from these reportings that those keys should be sized sufficiently, but still should be reachable without too much arm movement.

- Subjective perception of feedback components was a further issue we were highly interested in. All participants stated that they found Qwerty's large feedback letter more distracting than helpful. A key's orange flash when being hit was perceived mostly positively, yet only two subjects stated that this flash had helped them explicitly in recognizing typing errors. Hence, we have to reconsider Qwerty's feedback components. One possibility could be to display a letter history, similar to the history currently used in Column Typing, directly above the keyboard. This would reduce the distance the user's visual focus would have to cross for checking a phrase. However, we observed that users tend to enter keys with having the visual focus on the keyboard most of the time, only rarely changing their visual focus to the text cursor. Since we have to directly focus on a phrase for checking its spelling, it is unclear to what extent such a letter history would aid the user in recognizing errors. Another possibility would be to display the large feedback letter above and not over the keyboard. This way the users would no longer be distracted while searching for keys on the keyboard. On the other hand, the feedback letter would be more distant, raising the question how well it would be perceived.

  Considering Column Typing's feedback component, three participants stated that it had helped them in recognizing errors. Yet they also stated that Column Typing's inferior key selection technique caused them to put focus more on the key selection process rather than on the process of finding a key. As result the text was checked more often. Hence, it is unclear how well the feedback would have been perceived if the key selection process would have been more secure. Moreover, two participants stated that the occlusion caused by the letter history box sometimes made key recognition more difficult. Yet this could be helped by only displaying the letter history box when a finger touches the screen.

- Observation of typing behavior on the physical keyboard showed all participants used the same typing technique for typing. All participants used both hands for entering text on a physical keyboard but had to look at the keyboard most of the time. Mostly the index finger was used for hitting keys, but the other fingers were used casually as well. None of the subjects was capable of ten finger writing.

5.2.3.6.3 Summary Summarized, the Qwerty design was perceived superior to the Column Typing design. Reasons were mainly faster acclimatization to the already familiar design and key selection by direct touch-tapping, being both fast and secure. Column Typing's inferiority is mainly caused by the insecure technique of entering letters.

Column Typing's feedback component is promising for future design since it provides the user the possibility of checking a compound phrase without having to move visual focus to the text cursor. The Qwerty design has room for improvement in its feedback components. The orange flash after hitting a letter was not clearly visible to the participants. The larger letter displayed over the keyboard was perceived as distracting rather than helpful. Solutions could be the usage of a history box, similar to Column Typing, or displaying the feedback letter above the keyboard.

5.2.4 Conclusion

We have presented two alternative designs for entering text on tabletops. Both techniques provide unique opportunities to the user.

The Qwerty virtual keyboard design embraces users with a familiar interface for entering text. The fast and secure hittable keys are selected by touch-typing, just as with physical keyboards. Also the keyboard uses a standard Qwerty layout, the de-facto standard layout for text input in desktop computing.

The Column Typing design provides users an innovative text input environment. A letter is entered by placing a finger merely on the letter's horizontal position with the vertical position being arbitrary. The keyboard is portioned close to the text cursor, diminishing the distance the visual focus has to cover for checking the entered phrase. Moreover, a box containing the last ten letters is displayed directly left to the active key (the key that is going to be entered next). This box makes it easy for users to check recently typed letters and thus recognition is enhanced.

We conducted a qualitative user study with six participants. Results show that the Qwerty virtual keyboard is roughly three times as fast as the Column Typing layout. Participants perceived the familiar Qwerty layout as superior to the unaccustomed Column Typing design in terms of typing speed. As reason participants stated Column Typing's letter entry technique being unfamiliar and insecure. This matches with our observations during the experiment: participants found keys faster using the Qwerty keyboard and hit those keys with higher reliability. However, three participants preferred Column Typing's feedback component for recognizing errors.

In future design iterations, we will focus on making errors better recognizable with the Qwerty design. Possible techniques could be the display of a letter history similar to Column Typing or the display of the feedback letter in some distance to the keyboard rather than over the keyboard. Moreover, we will make the Qwerty design rotatable, enabling users to approach the keyboard from all sides.

Although being perceived largely inferior on the tabletop, different display environments might better exploit Column Typing's design. For example, text on a large display wall could be entered using the Column Typing design in combination with an IPhone as input device. Since keys can be selected be merely defining one dimension, the user would be enabled to enter text by sliding left and right on the IPhone while being able to keep visual focus on the large display. Moreover, one participant mentioned the Column Typing design to be possibly appropriate for disabled people that lack the ability of precisely hitting one spot with a finger. With Column Typing such people could enter text by sliding, being aided by having a reference surface on which their finger is placed.

5.3 Dimension 3: Dragging, Scrolling and Contextual Menus

Precise selection and text input are two of the most basic tasks a user needs to be able to accomplish in order to control WIMP user interfaces. However, there are a numerous additional tasks regularly performed when using WIMP. This chapter deals with supplying interaction techniques for enabling the user handling such tasks.

Concretely, these tasks are Dragging, Scrolling and the evocation of contextual menus:

- Dragging is fundamental for enabling the user to directly manipulate WIMP user interfaces (see section 5). Windows are dragged, files and folders are moved by dragging. Dragging is used to scroll via a scroll bar's handle and to select passages of text in a document. We present a dragging technique allowing users to perform dragging tasks by touch interaction.

- Next, we present an interaction technique for performing vertical scrolling tasks. Vertical scrolling is frequently used in WIMP user interfaces, since documents normally displayed so that they expand in vertical dimension, while having a fixed horizontal size. However, scrolling by dragging the scroll bar is impractical for touch input, since the scroll bar is sized too small to be hittable effectively. Examples for applications that display documents that way are the Adobe Reader or Microsoft Word. Moreover, most websites expand into the vertical dimension and have a horizontal size that fits into a standard screen width. Hence, we designed and implemented a specific interaction technique for vertical scrolling by touch input

- Third, we designed and implemented an interaction technique allowing users to access contextual menus. Contextual menus provide the user a limited number of

options that are directly associated to the interface element the contextual menu was triggered on. Hence, the burden of filtering operations possible for an interface element from operations that are not, is taken from the user. Moreover, the user can access such operations directly on the desired interface element and does not have to walk the detour over an explicit menu. Such directness is fundamental for direct manipulation interfaces, in whose category WIMP user interfaces fall.

All of the previously described techniques were implemented in the same prototype program as the selection and text input techniques were. This way we can compare the techniques and identify possible flaws. Moreover, making the techniques work on a tabletop will also ease actually working with WIMP user interfaces by touch input.

5.3.1 Related work

5.3.1.1 Dragging

As stated in section 5, Dragging is a basic interaction task for every direct manipulation interface and thus needed for WIMP. Using mouse interaction techniques, a dragging task is performed by positioning the cursor over the object, pressing the left mouse button, moving the cursor to the target location and then releasing the mouse button. That means, that the cursor switches between two states. These two states were identified by Buxton as Tracking and Dragging [Bux90]. Tracking refers to the state in which the cursor is only positioned and doesn't manipulate underlying objects. In the Dragging-state, the cursor 'grabs' an object and enables moving or manipulation. Using touch interaction, we can only differ between the states 'Finger lifted' and 'Finger pressed'. This means that we cannot differ between the Tracking and the Dragging state without further assistance of an interaction technique. However, in touch interaction the user sets the cursor directly with his finger. That means that the user doesn't have to relatively set the cursor over an interface element with a mouse but rather can touch this element directly with her finger. Hence, the Dragging state could be entered immediately after touching the object. Yet, this is impractical, since a user has to be able to do small corrective movements in order to perform a precise selection using certain techniques, e.g. Shift (see section 5.1.2.2). Moreover, hovering functionality like Tooltips would be impossible if a dragging would always be started immediately. This robustness against small finger movements is defined in the standard VDI/VDE 3850 Blatt 3 - User-friendly design of useware for machines - Design of dialogues for touchscreens [Dah06].

5.3.1.1.1 Sim Press Within their study on precise multi-touch selection techniques, Benko et al. also presented a technique for distinguishing between the states tracking

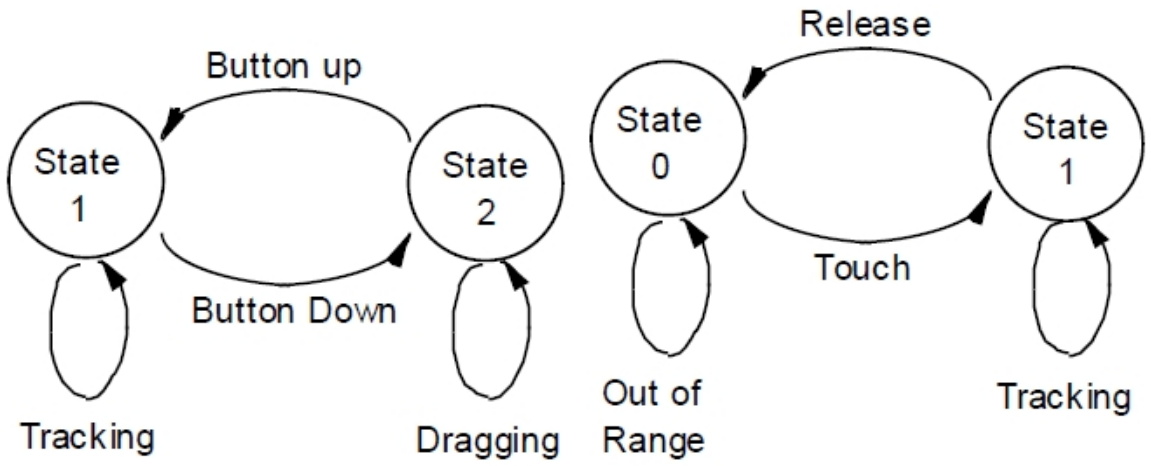

Figure 5.25: **Left:** Mouse input states. Switching between Tracking and Dragging is possible via pressing a mouse button [Bux90]
Right: Touch input states. Only differing between Out-of-range (i.e. finger lifted) and Tracking (i.e. finger has contact) is possible [Bux90]

and dragging [BWB06]. Again, tracking means that the cursor is merely set and not used to manipulate the underlying application. Dragging on the contrary corresponds to a mouse's left button pressed state.

Benko et al. presented SimPress. With SimPress, a finger's area of contact is used to determine if in tracking or dragging state. Concretely, the more strength a user puts on his finger when pressing on a display, the larger the area of contact gets. If the area of contact now surpasses a certain threshold, a switch between the states tracking and dragging is triggered. Yet in the concrete solution, the system immediately changes back into the tracking state. As a matter of principle, that means that a mouse left button click is triggered: "SimPress requires the user to apply a small rocking motion with their finger in order to perform a 'click' " [BWB06].

SimPress enables the user to switch between states using a simple finger rocking motion. However, SimPress is highly dependent on the mechanism used for finger tracking. When using optical recognition, like Benko et al. did, the area of contact recognized by the tracking algorithm is relevant. Yet this area is not only indicated by the finger's strength of pressure but also by other factors. Such a factors is environmental light. Environmental light lowers the contrast in the tracking camera's image and thus also the size of recognized finger contact areas. Furthermore, different people are likely to initially put on their fingers in different ways and with different levels of pressure. Some users might put on their finger steeply, others more flatly. For a robust performance of SimPress, a training or even a calibration for every user would be required.

Another issue is determined by the manner users put on their fingers to select targets. Concretely, people tend to put on their fingers more flatly when having to touch a target that requires stretching the arm out. SimPress now would require the user to put on her finger in an unnatural steep manner.

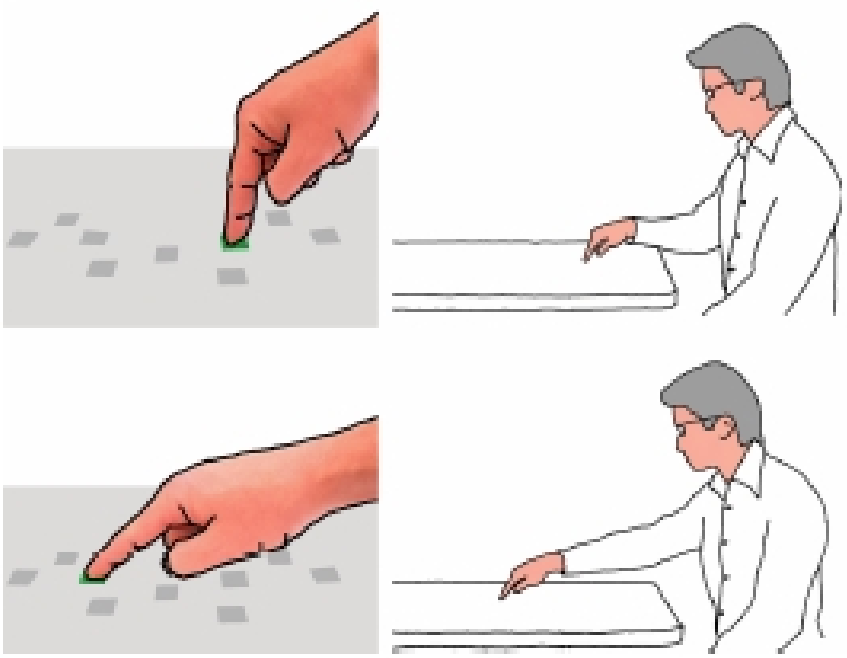

Figure 5.26: The more distant a target is, the more the arm has to be stretched out. Hence, the finger is put on more flatly. This enlarges the finger's contact area. [FWSB07]

In summary, SimPress might be adequate for small devices that are capable of sensing a finger's pressure directly without additional influences. For tabletop-sized devices that use optical finger tracking, SimPress is unlikely to work robustly.

5.3.1.1.2 Fluid DTMouse Dragging technique [ER06] Fluid DTMouse is a another approach that focuses the tracking/dragging issue.

The authors propose multi-finger unimanual techniques for enabling the user to perform selections and dragging. They also propose a technique for accessing contextual menus, but that will be taken into consideration in the section 5.3.1.3. Their techniques allow controlling Windows OS through emulating mouse functionality.

Fluid DTMouse works like follows: When the user touches the display with a single finger, the system changes into left-mouse-button-pressed state. Principally this means that a dragging is started at the finger's first point of contact. When the finger is lifted

without moving, a selection is triggered.

When touching the display with two fingers simultaneously, the mouse cursor is set in the middle between the two fingers. In this case, the system is in Tracking state and no dragging is triggered. In order to switch into the Dragging state, the user needs to tap with a third finger somewhere in between the two other fingers. With Fluid DTMouse, normally the middle finger and the thumb are used for the two-finger technique. The index finger is used for switching between Tracking and Dragging states.

With these techniques Fluid DTMouse covers the essential mouse's spectrum of functionality. However, there are certain flaws in this design. First, there are two different ways to set the cursor's position. For dragging, a direct and absolute mapping is used. For mouse tracking, the cursor is set in between two fingers. This no longer is an absolute mapping since both fingers together define the cursor's position (see section 4.3.2 for further discussion of absolute vs. relative input). Furthermore, for hitting a target precisely, the user has to navigate the cursor through simultaneous finger motion. This is similar to the Take-off selection technique described previously (see section 5.1.2.1). With Take-Off the cursor is set approx. 12mm above the finger and so allows setting the cursor precisely. With Fluid DTMouse, the cursor also is set besides the finger, only that two fingers control the cursor's exact position. Take-off has been shown to be precise yet slow, since the user almost always has to correct his finger's position for hitting a target. Thus, Fluid DTMouse might enable the user to set the cursor precisely, but there are likely alternatives that allow a faster control of the cursor.

Since the authors want to enable the user to fluently switch between tracking and dragging states, the user can enter dragging state though tapping with a third finger. This dragging state now has the same behavior as if the user had touched the display with just one finger. However, the cursor in this case is not under the finger but in between the two previously put on fingers. Hence, two different techniques for one and the same task are generated.

5.3.1.1.3 Apple Trackpad Gestures [App08] Recent Apple Notebooks feature a track pad that is capable of sensing multi-touch input. This kind of input is neither absolute nor direct. Yet, the gestures used by Apple could also be applied to absolute input techniques. For this reason these techniques have to be taken into account when designing novel touch interaction techniques.

Apple also presents a solution for the tracking / dragging issue. For initiating a drag, users have to move the cursor with their index finger onto the desired object and then press their thumb on the track pad. When the index finger is moved, the object is dragged. When the thumb is lifted, the state is switched from dragging to tracking

again. However, this concept requires the thumb to be held in position. Otherwise, the dragging functionality would interfere with the a scrolling technique. With this technique, the index finger's range of movement is very limited, since the thumb has to be held in place (parallel moving two finger causes a scroll). For this reason, Apple's dragging technique is not directly transferable to direct touch interaction on a tabletop. The distance a user has to be able to cover when dragging are considerable higher than one hand span. However, the technique could also be performed bimanually, using one hand as the 'holding' hand, corresponding to the thumb and the other hand for moving the object.

5.3.1.2 Scrolling

Using mouse interaction techniques, scrolling is either facilitated by manipulating the scroll bar of a window or spinning the mouse wheel. There are three ways to interact with the scroll bar in order to scroll. First, the user can click on the up- and down-arrows at the drag bar's upper and lower corners. This results in a scrolling over a discrete distance, normally one line in text documents. Second, the user can click into the open space between the drag bar's handle and the arrows. This also triggers a scrolling over a discrete distance, yet this time the distance is much higher, normally one page in text documents. The third technique to scroll using the scroll bar is dragging the scroll bar's handle. All of these techniques can directly be transferred from mouse interaction into the touch interaction domain, provided the corresponding selection and dragging techniques.

However, a scroll bar may well be smaller than 1cm in width. Like Vogel and Baudisch have shown, a target has to be at least 1x1cm in size to be hit effectively [VB07].

The scrolling technique utilizing the mouse wheel is not directly transferable to touch interaction. However, the concept of carrying out a scroll while only having to specify to document that is to be scrolled seems promising. A user doesn't have to precisely control an interface element like the scroll bar, but rather only has to keep the desired window focused.

5.3.1.2.1 Rate-Based Scrolling [ZSS97] Rate-Based Scrolling is an alternative to the traditional scrolling techniques of dragging the scroll bar or turning a mouse's scroll wheel. Rate-based scrolling "maps displacement of the input device to the velocity of scrolling" [IH00]. That means that the user first has to specify a reference point. This is usually done by pressing the mouse's middle button, e.g. the scroll wheel. After that the scroll speed is defined by the distance of the cursor from the reference point. The greater this distance the faster the scrolling speed. This technique can already be used to scroll

in various applications. For example, the Adobe Reader and the Firefox Web Browser feature this technique. This technique is very usable for continuously scrolling over large distances while always being able to slow down the scrolling speed. However, the user is limited in terms of the maximum scrolling speed. This is "because exceedingly fast scrolling causes disorientation" [IH00]. Igarashi and Hinckley provide a solution for this issue [IH00]. They propose to connect the scrolling to a simultaneous zooming. This means that the faster a user scrolls the more is the document zoomed out. However, this technique would require to alter the underlying application's visualization and is thus not applicable as a general scrolling technique for WIMP.

5.3.1.2.2 Apple Trackpad scrolling technique [App08] As stated previously, the trackpad of recent Apple notebooks features multi-touch detection of finger contacts. Already shown were multi-touch techniques for zooming, dragging and the evocation of contextual menus. Beyond these techniques, Apple also provides a multi-finger scrolling technique.

Apple's scrolling technique allows the user to scroll in windows without having to Drag & Drop the scroll bar. This scrolling technique involves two fingers being moved parallel. The movement can take place into every direction, causing the screen object to move into the corresponding direction. Interesting in this case is that Apple uses a dragging metaphor in this context. This means, when moving the two fingers into one direction, the screen object moves into the same direction. Thus, the user gets the feeling of having a "grip" on the object. This is oppositional to using a scroll bar. When using a scroll bar, the screen object moves into the opposite direction as the scroll bar does.

5.3.1.2.3 The Virtual Scroll Ring [MH04] An alternative to Apple's scrolling technique was presented by Moscovich and Hughes in 2004. Their Virtual Scroll Ring - technique uses a circular gesture to scroll in documents. In detail, a user has to move the cursor in clockwise motion to scroll up and in counterclockwise motion to scroll down. A transparent overlay widget was used as an indicator for the Virtual Scroll Ring. However, the circular gesture's performance is not bound to one spot. In an experimental evaluation the Virtual Scroll Ring (VSR) technique was tested against the mouse wheel technique. VSR was used both with a laptop's touchpad and a computer mosue. Results show that VSR outperforms the mouse wheel when scrolling over long distances. In detail, mouse wheel mean trial completion times were significantly higher for scrolling distances of 192 lines. For the next smaller distance gradation, 24 lines, no significant effect of time and technique was found. Unfortunately, VSR was not tested against the use of a scroll bar. Yet the scroll bar technique is the only one directly transferable to touch interaction. This means that VSR can be interpreted as promising for touch interaction, while lacking the proof of VSR's superiority to the scroll bar

technique. Another issue with VSR is that it requires an own scrolling mode. However, this mode could be triggered using a multi-finger technique, when transferring VSR to touch interaction.

5.3.1.2.4 Summary FluidDT Mouse is the only concept that offers a set of interaction techniques, that allow accessing the most elemental WIMP commands by direct touch interaction. However, additionally to the already stated flaws in desing, Fluid DTMouse lacks any feedback or elements for improving affordance. The user only has the underlying application's reaction as an indication for the effects of her actions. If the user would tap slightly besides a target, she would not know whether the system didn't recognize the selection or if she had missed the target. Moreover, the user is supplied with no hints on how the techniques work. For example, the user has no way of knowing that the tapping of a third finger in between the two other fingers switches between tracking and dragging states. Yet this switching is a vital part of Fluid DTMouse. Hence, training would be required for enabling the user to effectively use Fluid DTMouse's techniques. If the user would be supplied with more indicators on how the techniques work, a better and faster understanding would be likely.

Apple's trackpad interaction techniques offer some promising appendage. Especially the two-finger vertical scrolling gesture is directly applicable to direct touch interaction.

The Virtual Scroll Ring technique is promising, too. Yet the fact that a user would have to enter a specific scrolling mode, makes VSE inferior to Apple's technique.

5.3.1.3 Contextual Menus

Using mouse interaction techniques, contextual menus are normally triggered by clicking the right mouse button on the desired object. This is impractical for touch interaction techniques, since the user can only put fingers on the display and doesn't have additional aid through an input device. Hence, interaction techniques that allow triggering contextual menus have to be found.

5.3.1.3.1 Fluid DTMouse [ER06] Fluid DTMouse's right click functionality engages two fingers. However, there are three steps needed for triggering a right click. First, one finger has to be put on the desired position. Then, a second finger has to be briefly tapped on the display. Third, the first finger needs to be lifted. It is arguable if there aren't any solutions that involve fewer steps for triggering a right click. Furthermore, when being in the mode where the cursor is positioned between two fingers (see section 5.3.1.1.2), a right click is not possible. This disables the user to do precise right clicks.

5.3.1.3.2 Hover Widgets [GHB+06] Grossman et al. presented a technique for triggering so-called Hover Widgets on pen-based interfaces. Hover Widgets are graphical representations (icons) of commands, displayed at the cursor's position. Grossman et al. designed Hover Widgets for four groups of commands: Tools, Edit Dialog, Scrolling and Right-Click. Tools "can be thought of as replacing an icon toolbar, found in most drawing applications"[GHB+06]. That means that commands for activating tools like pen, square or circle. The Edit Widget contains commands "typically found in an application's "Edit" menu: undo, redo, clear, cut, copy and paste"[GHB+06]. The Scroll Widget enables the user to scroll with the pen using a circling gesture. This technique is called the Virtual Scroll Ring [MH04] and will be dealt with later on. Last, the Right Click Widget enables the user to access the command for opening a contextual menu. After having triggered the Widget, "Subsequent pen down events simulate the functionality generally associated with clicking the right mouse button" [GHB+06].

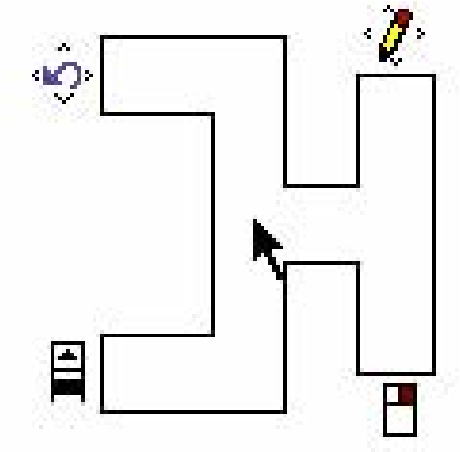

Figure 5.27: The tunnels for accessing the different types of Widgets. Each way though a tunnel requries the pen to be moved in the shape on the letter L

A Widget is triggered through a pen gesture. This gesture is L-shaped for all four types of Widgets. In detail, the pen has to be moved through a virtual tunnel, until it reaches the icon representing the corresponding widget. The design of Hover Widgets requires the hardware to be capable of differencing between Tracking and Dragging-states. The selection of a widget is only possible when in Tracking state. This makes an direct transfer to a touch interaction technique impossible. However, the use of gestures to switch between different commands associated with a certain interaction technique seems promising.

5.3.2 Design and Implementation of techniques for dragging, scrolling and invoking of contextual menus

In this section we present the design and implementation of techniques for performing dragging and scrolling tasks as well as for invoking contextual menus. All these tasks are essential for enabling users to effectively control WIMP user interfaces (see section 5).

5.3.2.1 Design principles

Performing dragging tasks by touch input poses the challenge of having only one gradation for identifying the the finger's level of contact (e.g. you can only tell if a finger touches the display or is lifted). Beyond this major challenge, we postulate the following design principles for our dragging technique. These principles are affected by the standard VDI/VDE 3850 Blatt 3 (see section 5.3.1.1), the significance of selection tasks and our own experiences with using WIMP user interfaces.

- The dragging has to be understood as a 'grasping' metaphor. Hence, the Dragging has to start at the initial point of contact of a finger on the surface. This way the object that is to be dragged is under the finger from the beginning to the end of the Dragging task. The 'grasping' metaphor will lead to a fast understanding of the technique.

- The dragging technique must not interfere with the selection technique. Selections are performed more frequently than draggings. Thus, their performance must not be diminished by the dragging technique.

- A certain robustness against small finger movements after the initial contact has to be ensured. This is to minimize unintended draggings.

Scrolling by touch interaction opens the opportunity of letting users 'grasp' a document. This lets users feel that they have direct control over the desired document. Moreover, an analogy to dragging is established, providing a consistent interaction metaphor. Yet there are certain issues to be considered when designing scrolling techniques. We established the following design principles based on the characteristics of touch interaction, discussion with users and our experiences of using WIMP user interfaces.

- Vertical Scrolling has to be possible to the extent that the user only has to control the vertical dragging movement. This way the user doesn't have to pay attention to her finger's horizontal movements.

- Scrolling must not need a state of its own. Switching between states every time a users wants to scroll would likely decrease performance.

- Users have to be able to scroll without using the scroll bar. The scroll bar was designed for mouse interaction and doesn't fit the needs of touch interaction (see section 5.3.1.2).

Contextual Menus enable users to reach commands relevant for a specific interface element by directly invoking a contextual menu on this element. This way user are able to access such commands more rapidly and are taken the burden of filtering out irrelevant commands from explicit menus. We propose the following design principles to be considered when designing a touch interaction technique letting user invoke contextual menus.

- Users have to be able to rapidly invoke contextual menus when having positioned the system cursor on their desired element. This way users will be able to immediately access the relevant commands.

- The way of how to invoke a contextual menu has to be clarified to the user. The invocation is done normally via a right-mouse click. Touch input lacks such a hardware modality completely. Hence, users cannot transfer a familiar mouse input technique into the touch domain.

5.3.2.2 Dragging

In this section we present a technique for performing dragging tasks by a touch interaction technique. The technique utilizes a distance threshold that has to be crossed by the finger before the dragging is initiated. This prevents unintended draggings and leaves the possibility of performing corrections when doing a selection task (see section 5.1.3.1).

5.3.2.2.1 Design The main obstacle when designing a Dragging technique for touch interaction, is to overcome the impossibility of differencing between Tracking and Dragging states (see section 5.3.1.1). Yet, as stated before, when using direct touch interaction, a Dragging could start directly after touching an object. However, this disables the user to perform small corrective movements to state a selection more precisely. Moreover, the system has to be robust against accidental starts of draggings. This is defined in the Norm VDI/VDE 3850 Blatt 3 (see section 5.3.1.1). Hence, the Dragging must not start immediately after touching an object. Nevertheless, the user has to be able to precisely define the Dragging's start point.

For this reason, we chose a design that allows corrective movements as well as precisely defining the Dragging's start point. Concretely, the Dragging start point is set to the initial point of contact. However, the actual Dragging operation starts not until the finger has moved beyond a certain distance threshold. This way the Dragging's start point can be set as precisely as a selection can be performed. Moreover, small corrective movements are still possible.

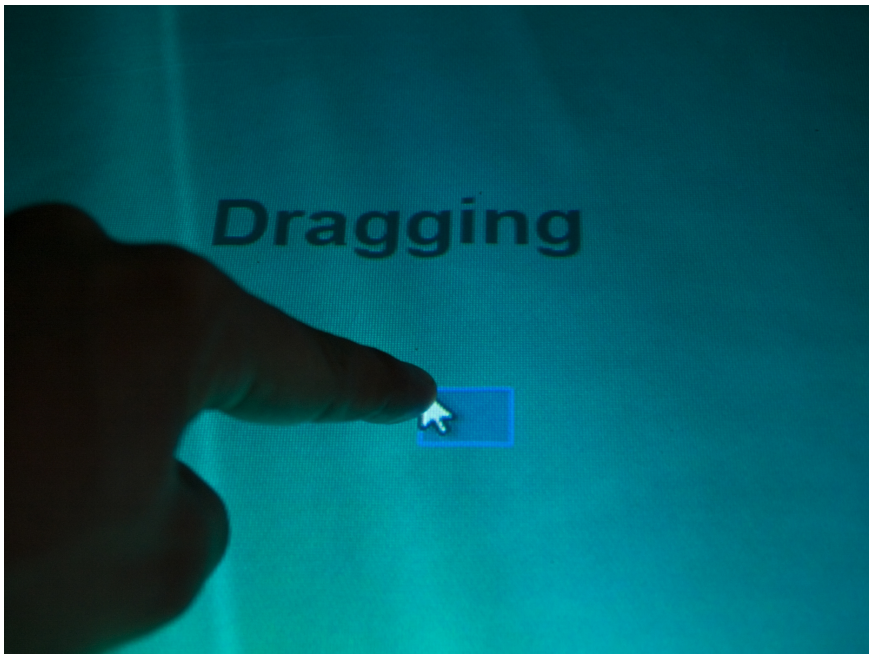

Figure 5.28: Size of the dragging threshold. The text "Dragging" is displayed for a short time as confirmation of the dragging start.

The concrete distance threshold depends on the selection technique used. The selection techniques Shift, TapTap and ZoomTap are implemented in our prototype program (see section 5.1.3.4). When using Shift, the user has to refine his selection by performing a corrective movement with the finger having contact (see section 5.1.2.2). On the other hand, TapTap and ZoomTap utilize zooming as a tool for enhancing selection precision and therefore do not need the user to perform such corrective movements. For this reason, the distance threshold is adaptively changed depending on the corresponding selection technique. For Shift the distance threshold is set to 50 pixels. This corresponds to 3.2 cm in physical space. For TapTap and ZoomTap the distance threshold is set to 30 pixels, which corresponds to 1.9 cm in physical space.

5.3.2.2.2 Implementation The Dragging command is executed by emulating mouse input. This means, that, as soon as a finger has moved beyond the distance threshold, a left mouse button down-command is sent to the Windows OS. Previously the cursor

position is set to the finger's initial point of contact. The mouse button release-command is sent as soon as the finger is being lifted.

For feedback about the Dragging's start, the text 'Dragging' is displayed above the finger. The text is then faded out. This way the user is assured when the Dragging actually started and is not only dependent to Window's feedback.

5.3.2.2.3 Discussion The usage of the distance threshold complicates draggings over a distance smaller than the threshold. However, most Dragging commands are performed in order to move an object a longer way than 30 or 50 pixels. For example, when moving a window or an icon, the user normally wants to alter their position in a large scale. If the user nevertheless wants to drag an object over a small distance, she has to first move the finger beyond the threshold and then back again.

When using the Shift selection technique, the precise start of a Dragging command is not possible (see section 5.1.2.6). With Shift the refining of a selection is performed by corrective movements with the finger having contact. Since the Dragging's start point is set to the finger's initial point of contact, such corrective movements do not alter the Dragging's start point. A solution would be slowly pulling the Dragging's start point towards the finger's contact point. This principally is a dwelling technique and thus would be likely to slow down the execution of a Dragging command.

5.3.2.3 Scrolling

We present a scrolling technique that is performed by parallel moving two fingers up- or downwards. This way a user can scroll in a document from any position on the screen.

5.3.2.3.1 Design As stated in section 5, Scrolling is an elemental task for WIMP. One technique for accomplishing the scrolling task is to drag the scroll bar. However, this technique has several disadvantages when used as a touch interaction technique. First, dragging the scroll bar make users "lose precision in very long documents" [MH04]. This means that navigating to a certain position in a long document requires several corrections when using the scroll bar.

Moreover, the scroll thumb has a width of about 0.6 cm on a 34" tabletop display with a resolution of 1024x768. Hence, the user is not able to hit that target effectively by direct touch [VB07].

Furthermore, using the scroll bar "takes the perceptual, cognitive and motor resources away from the target that the user focused attention on, breaking the work flow" [ZSS97].

This means that the user has to put full focus to the scrolling task, rather than accomplishing scrolling parallel to e.g. reading a document.

Hence, we chose a design that enables the user to scroll without having to drag the scroll bar. Concretely, the user has to put on two fingers of one hand on the display and move them vertically in parallel motion. By that gesture a scrolling in the window that has the system's focus is initiated. The content moves in the same direction as the finger movement. For example, when moving the fingers upwards, the document also moves upwards. This technique is also used in Apple's notebooks [App08].

Horizontal scrolling also is integrated in our design. However, when moving two fingers parallel in horizontal motion, a scroll in time rather than in space is triggered. In the particular case of our prototype program, browser backward and forward commands are executed by this horizontal gesture.

This design supplies the user with several advantages. First, a 'grabbing' metaphor is used for scrolling a document. The document moves in the same direction as the fingers do and thus the user gets the feeling that he can directly manipulate the document's position with his fingers. This further pushes the direct manipulation approach already used in selection and dragging techniques. Moreover, the user no longer has to directly drag the scroll bar. He now can scroll by just setting the focus in the desired window and performing the vertical two-finger gesture. This makes accessing the scrolling task faster. The user doesn't have to precisely select the scroll bar for accomplishing a dragging. This selection is likely to take more time than just moving two fingers vertically. The scroll speed is set in the way that the document moves faster than the fingers. This enables the user to bridge large distances in a document without abuting at a display's border.

The concept of matching a horizontal gesture to a scroll in time is based on the hypothesis that horizontal scrolls in space are rarely used. This was further described in section 5. Backward/forward or undo/redo commands are more frequently used. The horizontal scroll in time is triggered discretely. That means that one gesture triggers one command rather than triggering the command continuously.

5.3.2.3.2 Implementation The technical detection of a scrolling gesture is done be evaluating both finger's moved distances. If each finger has moved more in vertical than in horizontal direction and if each finger has not moved more than 10 pixels in horizontal direction a vertical scroll is triggered. The scroll command itself is executed by sending the keyboard command for the key Page Up and Page Down respectively. After each sent key command the fingers' moved distances are reseted.

The horizontal scroll in time is detected analogically. However, since this scroll in time is discrete, all fingers have to be lifted after a horizontal scroll command in order to

trigger a new one.

5.3.2.3.3 Discussion Our design enables the user to accomplish scrolls in space in vertical direction from any position on the screen. However, there are several issues with our design. First, when scrolling large distances the user will probably hit the upper or lower border of the display and thus has to do clutching. The Virtual Scroll Ring technique would offer a solution for this issue. Yet, then the 'grasping' metaphor would get lost. Furthermore, the Virtual Scroll Ring technique requires an own scrolling state. Switching into a further state would require the execution of further interaction technique. Such switching is not required in our design.

Rate-based scrolling could be another technique applicable as a touch interaction scrolling technique for WIMP. However, rate-based scrolling also requires a state change. Such state changing could be accomplished by a bimanual technique. For example the user could set the reference point with one finger constantly and set the scroll speed by altering a second's finger distance to the first one.

5.3.2.4 Evocation of contextual menus

5.3.2.4.1 Design When designing a touch interaction technique for invoking contextual menus, two main have to be considered:

- First, the contextual menu has to be invoked at the same position as the desired object. This is due to the "direct manipulation approach to controlling the application" [KS86] in WIMP generally and touch interaction in particular. This means that the contextual menu's content is dependent on the object the menu is invoked on.

- Second, users are likely to be used to primarily using mouse interaction techniques for WIMP. This means that contextual menus are normally triggered through clicking the right mouse button. Hence, building upon this already accustomed behavior would probably support the user.

The design we developed handles both issues. Concretely, a context menu is triggered in two steps: First, a finger (normally the index finger) has to be put on the desired interface element. This sets the cursor onto this element. Second, the user has to tap his thumb on a widget. This widget is displayed on the lower left-hand side of the index finger and the cursor respectively. The distance from cursor to widget is each 3 cm to the left and lower direction. The widget is always positioned relatively to the cursor. This means that it moves parallel to the cursor and thus always is beneath the user's thumb.

The widget has the form of a filled circle with approx. 1 cm diameter. Furthermore, the text "Right-Click" is displayed next to the widget. After invoking the contextual menu, the feedback text "Right-Click" is displayed above the cursor.

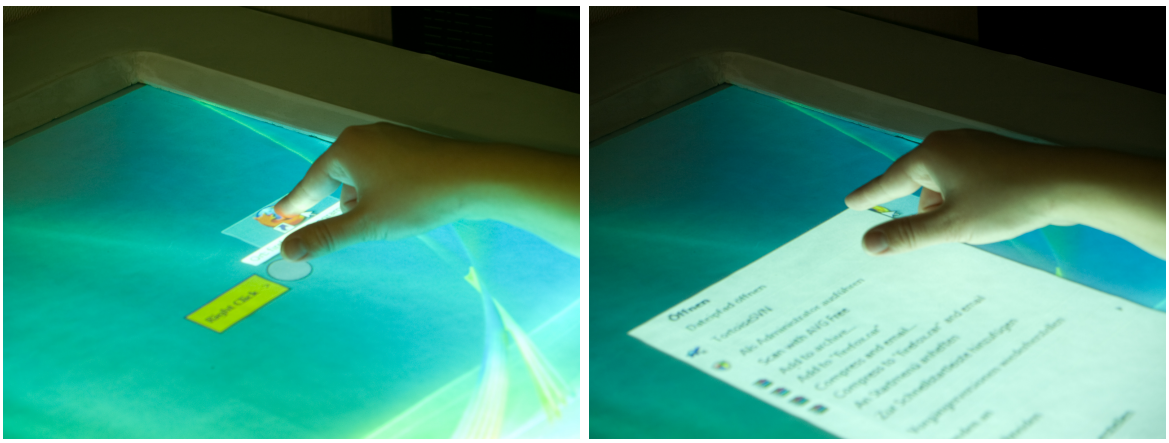

Figure 5.29: **Left:** The user is about to invoke a contextual menu on a desktop icon. **Right:** The user has tapped the circular widget with her thumb and invoked a contextual menu.

This design provides aid to the user in several ways:

- The usage of the index finger for setting the cursor position is carried on, providing the user with the consequent concept that her index finger is used for cursor positioning.
- Next, the contextual menu is invoked with the same hand as the cursor is set with. This is an analogy to mouse interaction and thus is likely to be adapted by the user. The tapping on the widget is an analogy to clicking the right mouse button.
- Since the widget always is positioned relative to the cursor, it is reachable be the right-hand's thumb. The tapping movement required for invoking the menu is then only a small down-and-up movement of the thumb. Observations have shown that this movement can be carried out with little motor effort.

The area in which a tap leads to the invocation of a contextual menu is about twice the size of the visible widget. This way the user doesn't need to precisely hit the visible widget.

The text "Right-Click" that is displayed next to the widget circle offers a certain affordance about the task that can be accomplished by tapping the widget to the user.

The widget only is displayed when a finger has contact with the display. When all fingers are removed, the widget is slowly faded out. This makes clear that tapping the widget

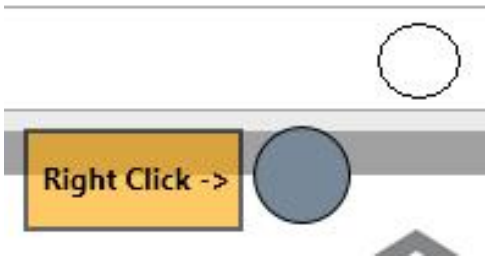

Figure 5.30: The widget for triggeing contextual menus. The single circle indicates the index finger's position (and is not visible in the actual design). The contextual menu is triggered by tapping the gray circle with the thumb.

only is possible when at least one other finger has contact with the display.

5.3.2.4.2 Additional Widget Tasks Triggering contextual menus is one important part the widget enables to user to accomplish. However, there might be tasks that certain users might also need frequently. For example, Copy and Paste commands are likely to be used when editing text. Providing interaction techniques for quickly accomplishing such tasks might provide additional aid to the user. When using mouse interaction techniques, such fast access is not possible, since the mouse's hardware modality of a right click is strictly connected to invoking a contextual menus. This is due to the operating system's interpretation of a right mouse click. This interpretation might be changed, yet this would require profound knowledge of the system's internal design and is thus not handable by an unskilled user.

A virtual widget does not have this close connection to an operating system command. This means that the task accomplished by the technique of tapping on the widget can easily be altered.

The concrete design we chose is that a user can change the task accomplished by the widget by scrolling through a number of possible choices. As soon as the user puts on a second finger (scrolling finger) that is more than 250 pixels (16 cm) off the first finger, a list of choices is faded in above the original "Right-Click" widget. When now moving the scrolling finger up- or downwards, the user can scroll through this list and in this way change the widget's task. The active task is always shown on-translucent. The other possibilities are slightly transparent. This ensures the user about the currently active task. The user can access the currently active task with a thumb-tap. This also is possible when the scrolling finger still has contact and thus the list of alternatives is visible. When lifting the scrolling finger, the list disappears and the last active task stays visible.

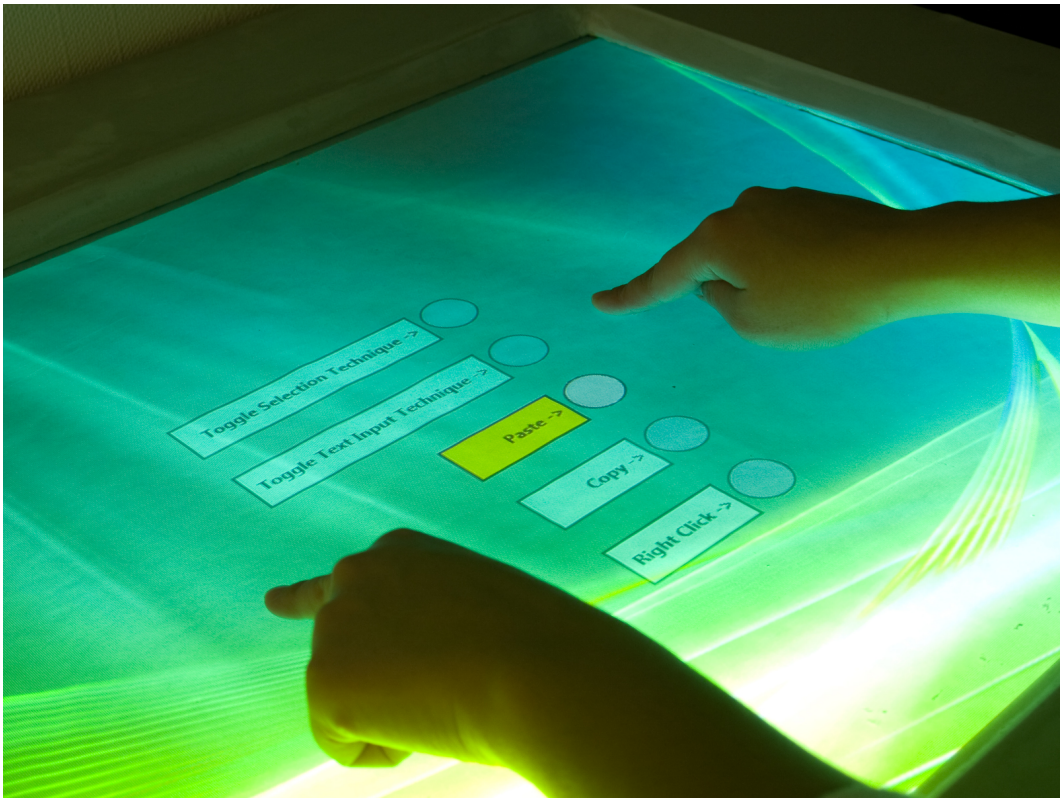

Figure 5.31: Additionally available widget tasks are presented when placing a second finger on the screen. The second finger has to be at least 250 pixels (16 cm) away from the first finger. The user can scroll through the widget tasks by moving the second finger up and down.

5.3.2.4.3 Implementation The mechanism of detecting a thumb's tap is done in the same way as for selection techniques, described in section 5.1.3.4.

Detecting if the thumb's tap is within the widget's boundaries is done through a helper function provided by the WPF framework.

The widget itself is implemented as a translucent WPF window. This window only has the extent of the included widget. When the user moves the cursor, the window itself is moved. This enhances the program's performance, since only a small window has to be painted transparently

The list of possible alternative widget tasks also is integrated in this window. Scrolling through this list is done through changing the relative vertical position of the window to the index finger. The scrolling itself is done in discrete steps so that the user always scrolls one entry up- or down. Concretely, one scrolling step is done as soon as the scrolling finger moves 100 pixels in vertical direction.

The alternative tasks implemented in the prototype are Copy and Paste. Furthermore, tasks for changing the currently active selection technique and text input technique are implemented. However, these last two tasks are only for demonstration purposes.

5.3.2.4.4 Discussion The described design enables the user to perform the task of invoking contextual menus quickly by tapping a widget with her thumb. Moreover, the user can change the task associated with tapping the widget. This might further aid the user in accomplishing her current goal. However, there are several issues that are still to be discussed.

- First, the current design requires the user to use her right hand for accessing the widget in an ergonomic way. Moreover, the user has to stand directly in front of the display. This might be a disadvantage, for example if there is more than one user around the display. This issue could be handled through enabling the user to drag the widget circular around her index finger. The text would also have to be changed, so that is it always readable.

- Another issue is that the user has to tap her thumb to trigger a contextual menu. This means that the widget is left of the user's index finger. In traditional mouse interaction however, the user clicks the right mouse button. This transposition of sides might irritate a user. However, observations have shown that users quickly get accustomed to tapping with their thumbs.

- Supplying additional tasks by making the widget tasks switchable might support the user. However, a user might as well need a task like Copy only once. That means that the user would have to switch to Copy, perform the thumb tap and switch back to Right-Click again. A solution would be that a widget task could have two states: temporary and persistent. If a task is in temporary state, the widget automatically switches back to the persistent task after the execution of the temporary task. This temporary state would be reasonable for tasks like Copy or Paste. Such tasks are likely to be only needed once in a row. When a user now wants to change a tasks state to persistent, she would have to perform a short horizontal movement with her scrolling finger. The metaphor used is closing a padlock. Once a task is in persistent state, the task can be executed arbitrary often in a row.

5.3.3 Conclusion

We have presented interaction techniques for performing Dragging and Scrolling tasks as well as for invoking contextual menus by touch input. These tasks are fundamental

for using WIMP user interfaces (see section 5) In detail, the user is provided with the following essential interaction techniques:

Dragging: A dragging task is performed by placing the finger on an interface element followed by moving the finger while having contact. In order to allow corrections for selection tasks and to prevent unindented dragging invocations (see section 5.3.1.1), the dragging is started not until the finger is moved beyond a certain distance threshold.

This technique was designed analog to the corresponding mouse interaction technique. Hence, users are likely to quickly adapt our Dragging technique. In combination with ZoomTap, our selection technique we presented in section 5.1.3, user are able to invoke Draggings with the same enhanced precision as selections. Fluid DTMouse [ER06], the only existing set of interaction techniques for performing Dragging and selection tasks in WIMP user interfaces, requires the user to use two different interaction techniques for initiating Dragging tasks on large and small targets (see section 5.3.1.1.2).

However, the dragging threshold causes the interface element being dragged to "jump" a certain distance (from its initial position to the finger that has just passed the threshold). Moreover, draggings over small distance require the finger to move beyond the threshold and then back again (see section 5.3.2.2.3. Identifying the seriousness of these issues is subject of further consideration.

Scrolling: The task of scrolling in vertical direction is performed by parallel moving two fingers up- or downwards. Vertical scrolling is particular relevant for using WIMP user interfaces (see section 5). The document moves in the same direction as the fingers (i.e. if the fingers move up the document also moves upwards). This scrolling technique utilizes a grapping metapor. Since the document moves into the same direction as the fingers, the user has the feeling that her hands are "glued" to the document she scrolls in.

Our scrolling technique builds up on the technique used for scrolling with the touchpad in recent Apple notebooks (see section 5.3.1.2.2). Yet with our technique, users are not only able to scroll in space but are also enabled to "scroll in time" when performing a horizontal parallel two-finger stroke. We found that horizontal scrolling in space is less relevant than providing the opportunity of going backward/forward in time (see section 5.3.2.3). Concretely we matched the web browser Forward/Backward commands to this stroke gesture. However, matching this gesture to Undo/Redo commands might be more reasonable in a certain context.

The scrolling technique we utilize has the drawback of not enabling the user to quickly scroll large distances. One possible solution could be a design similar to Rate-Based Scrolling (see section 5.3.2.3.3).

Contextual Menus: With our technique users are able to invoke contextual menus by tapping their thumb on a circular widget. The widget is always positioned relatively to the index finger's position (assuming the index finger was the first finger touching the display after all fingers had been lifted). A small explanatory text phrase ("Right-Click") is displayed next to the widget.

Our technique enables the user to quickly invoke contextual menus. The tapping done with the thumb is an ergonomic movement, since the thumb only has to briefly tap the surface at the position it anyway is. Since the widget's sensible area is about twice as large as the visible circle, the user doesn't need to precisely hit the visible circle. The circle's small visible size (approx. 1 cm diameter) causes only a small area to be occluded.

The invocation is performed with the same hand as selection and dragging tasks are. We expect this analogy to the mouse interaction technique of invoking contextual to support the user in understanding our technique.

Beyond contextual menus, the user is able to switch the task associated with the widget. For switching this task, the user has to put on a second finger on the display and slide this finger up or down. In our implementation, the widget's task can be switched to Copy, Paste and to switch between the different selection techniques we implemented (Shift, TapTap and ZoomTap) as well as between the different text input techniques (Qwerty and Column Typing).

However, the there are several issues still to be discussed. The widget is left of the finger despite it accomplishes the tasks normally associated with a right mouse click. Next, the user has to stand directly in front of the tabletop for the thumb tapping being an ergonomic gesture (see section 5.3.2.4.4).

6 Conclusion and future work

In this thesis we have presented touch interaction techniques for operating WIMP user interfaces on tabletops. Our techniques enable users to perform all essential interaction tasks in the WIMP domain by direct touch input. Thereby, we let users benefit from the advantages of direct and absolute interaction while maintaining quick learnability by analogies to mouse interaction. All techniques are designed to support each other in accomplishing tasks. Summarized, all of our techniques walk hand in hand, presenting users a compound interaction concept for using WIMP user interfaces.

We identified three dimensions of interaction tasks as particular essential for using WIMP user interfaces. In dimension 1 we described the design and implementation of ZoomTap, a novel interaction technique for high precision selection tasks. WIMP user interfaces are commonly designed for mouse input. Yet, direct touch selections are less precise than selections performed by mice. Our technique, ZoomTap, enables users to retain a high efficiency for selecting large targets while providing enhanced precision for selecting small targets. High efficiency for large targets is achieved by letting users perform selections by direct touch, undamped by any further interaction steps. High precision for small targets is attained by allowing users to trigger a local zoom. This zoom enlarges the desired target and thus raises selection reliability.

In our future work, we will address the issue of how to help users in deciding whether a target is large enough to be selected without zooming or needs additional assistance from ZoomTap. Currently users themselves have to decide whether to use the local zoom or select a target by direct touch. We expect users to come mostly to the right decision. Yet, we cannot always assure a right decision with our current design.

Furthermore, ZoomTap only has two zoom levels in its current design. Providing the ability to alter the zoom level continuously would enable users to perform even more precise selections. Moreover, users would be supplied with another technique of directly "grasping" a part of the interface, being able to stretch or clench a certain area. Concretely one could imagine to let the zoom area's size fixed but let users alter the zoom depth by moving their fingers towards or apart each other.

In dimension 2 we have presented the design, implementation and evaluation of two techniques for text entry. Entering text was of particular interest since it is crucial for

a vast number of applications. In order to enable users entering text effectively and efficiently by touch input, we had to overcome the lack of a physical keyboard's tactile feedback. The first design, our Qwerty virtual keyboard, exploits users' familiarization with the Qwerty layout, the standard layout for physical keyboards. With the Qwerty virtual keyboard, letters are entered by touch-typing the corresponding keys. The second design, Column Typing, lets users enter letters by placing a finger merely on a letter's horizontal position rather than discretely selecting a key. Column Typing's feedback component aids users by providing a letter history displayed in the direct vicinity of the next key that is to be entered and thus also is in the vicinity of the user's focus. Moreover, the keyboard is positioned close to the text cursor, minimizing the distance the visual focus has to cover to the text cursor and thus simplifying the proofreading of a whole phrase. A letter is entered by placing the finger to the letter's horizontal position with the vertical position being irrelevant. We conducted an informal user study with six participants in order to compare our two designs. Results showed that participants clearly preferred Qwerty's familiarity and benefited from entering letters by direct touch-typing. On the other hand, participants were able to utilize Column Typing's feedback component for checking their inputted text.

Our Qwerty soft keyboard already performed well and was percepted positively by users. However, the design still is open for improvement. Concretely, feedback components have to be adapted to fit more to the user's needs. Te be precise, our study showed that users benefited from the large letter displayed over the keyboard for recognizing if they had missed a key. On the other hand, participants reported to also be distracted by this feedback. Hence, we have to find an alternative way of providing clearly visible yet not distracting feedback. Moreover, the keyboard design would benefit from being independent to the user's position around the table. Hence, dragging and rotating the keyboard need to be made possible.

A further topic we are concerned with is how to make our text input techniques usable not only on tabletops but on vertical displays as well. For example, in a design room scenario like the Media Room at the HCI group of University of Konstanz (see figure 6), user will have to be enabled to enter text on tabletops as well as on a large display wall. "The Media Room provides a research environment for the design, development, and evaluation of novel interaction techniques and input devices as well as a simulation facility for future collaborative work environments" [RK08]. Hence, text input techniques have to fit such an environment's specific needs. For example, the Column Typing design could be used in combination with an iPhone in order to let the user enter text while maintaining visual focus on the display wall.

In Dimension 3 we have covered further essential interaction tasks for controlling WIMP user interfaces. These were dragging, scrolling and invoking contextual menus. Dragging is essential for every direct manipulation interface, thus also for WIMP user interfaces.

Figure 6.1: The Media Room at University of Konstanz [RK08]

We addressed the problem of not being able to differentiate between Tracking and Dragging states with touch input. Scrolling is used frequently for navigating in documents, yet the scroll bar, the only means of performing scrolling without a mouse or a keyboard, doesn't fit the needs of touch interaction. Being able to invoke contextual menus is another crucial part of WIMP user interfaces. Contextual menus speed up a user's performance and facilitate finding commands relevant to a specific interface element. We have presented the design and implementation of interaction techniques handling these tasks. Our dragging technique works by using only one finger while leaving the possibility of correcting selections and displaying tooltips. Our scrolling technique uses a parallel two-finger vertical movement. Since the document moves the same direction as the fingers do, the user gets the feeling of "grasping" the document. The invocation of contextual menus is done through tapping the thumb on a circular widget. This widget always is displayed under the thumb, assuming the index finger has contact with the display. The tapping is ergonomic, since the thumb has to be moved only a small way while not having to precisely hit a small target. Beyond contextual menus, the widget can be utilized for further tasks. For example, Copy or Paste are tasks regularly needed when operating WIMP user interfaces. We provide users with the ability to switch through a set of tasks with a bimanual technique. Issues in relation to our techniques are draggings over small distance, scrolling large distances without re-positioning the fingers and independence of user position and hand with respect to our technique of invoking contextual menus.

Summarized we have introduced a set of touch interaction techniques consisting of means to precisely select targets, enter text, drag, scroll and invoke contextual menus. Each of these tasks already is vital on its own. Yet, in combination with each other, they

form a powerful consistent interaction concept, enabling fluently controlling WIMP user interfaces and letting users benefit from direct touch interaction.

A Questionnaire

Fragebogen zur Nutzung

Angaben zur Person

Geschlecht: □ weiblich □ männlich

Alter:

Studiengang:

Semester:

Bei welcher der beiden virtuellen Tastaturen viel es dir leichter…

… einen Buchstaben zu finden

□ Design 1 (Qwerty)

□ Design 2 (Column)

… einen Buchstaben einzugeben

□ Design 1 (Qwerty)

□ Design 2 (Column)

… Leertaste, Backspace, Shift und Enter zu erreichen

□ Design 1 (Qwerty)

□ Design 2 (Column)

… einen Fehler zu bemerken

□ Design 1 (Qwerty)

□ Design 2 (Column)

… den Text auf Fehler zu überprüfen

□ Design 1 (Qwerty)

□ Design 2 (Column)

Welche der beiden Tastaturen fandest du insgesamt besser?

□ Design 1 (Qwerty)

□ Design 2 (Column)

B Text phrases

Die Polizei sucht den Schuldigen
Viele wollen wissen was los ist
Das Rettungsprogramm wird aufgestockt
Er trinkt zu viel Kaffee
Es sind keine Korrekturen mehr erforderlich
Er kommt ursprünglich aus Spanien
Doping ist sehr schädlich
Es gibt nur noch wenige Tiger
Sie kauft billiger ein
Viele Köche verderben den Brei
Er schlief zwei Stunden
Das Unternehmen zahlt jetzt mehr Lohn
Igel
Raub
Netz
Kran

Die Menschen kaufen keine Autos mehr
Die Regierung plant eine Konjunkturhilfe
Die Firma hat viele Mitarbeiter
Atmen ist nicht schwierig
Volker hat sehr gute Laune
Eine Haushaltsreform ist ein Muss
Aus Kohle kann viel gemacht werden
Das Leck wurde schnell geflickt
Wir vernahmen ein lautes Geräusch
Sachsen lockt Studenten
Reiche Menschen haben es schwer
Es geht um viel Geld
Kind
Hals
Bein
Ofen

Meine Uhr fiel ins Wasser
Die Bedingungen sind völlig unakzeptabel
Elefanten haben Angst vor Ratten
Peter macht das Fenster auf
Er soll endlich sein Zimmer aufräumen
Peter kann nicht gut schwimmen
Die Lage in der Hauptstadt eskaliert
Sie hat Angst vor Spinnen
Der Trend geht zur Freilandhaltung
Stress wirkt wie Gift auf das Gehirn
Das Material ist robust
Ich gehe nach Hause
Haus
Boot
Auto
Buch

List of Figures

Bibliography

[App08] Apple Inc. MacBook Pro Feature Description. http://www.apple.com/macbookpro/features.html, 2008.

[Bli95] C. H. Blickenstorfer. Graffiti: Wow! *Pen Computing Magazine*, pages 30–31, 1995.

[BM86] W. Buxton and B. Myers. A study in two-handed input. *SIGCHI Bull.*, 17(4):321–326, 1986.

[BSP+93] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and magic lenses: the see-through interface. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 73–80, New York, NY, USA, 1993. ACM.

[Bux] W. Buxton. Human skills in interface design. *Interacting With Virtual Environments*, pages 1–12.

[Bux83] W. Buxton. Lexical and pragmatic considerations of input structures. *ACM SIGGRAPH Computer Graphics*, 17(1):31–37, 1983.

[Bux90] William Buxton. A three-state model of graphical input. In *INTERACT '90: Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction*, pages 449–456, Amsterdam, The Netherlands, The Netherlands, 1990. North-Holland Publishing Co.

[BWB06] Hrvoje Benko, Andrew D. Wilson, and Patrick Baudisch. Precise selection techniques for multi-touch screens. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1263–1272, New York, NY, USA, 2006. ACM.

[Cas08] G. Casiez. The Impact of Control-Display Gain on User Performance in Pointing Tasks. *Human-Computer Interaction*, 23(3):215–250, 2008.

[Dah06] M. Dahm. *Grundlagen der Mensch-Computer-Interaktion*. Pearson Studium,

2006.

[DL01] Paul Dietz and Darren Leigh. Diamondtouch: a multi-user touch technology. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226, New York, NY, USA, 2001. ACM.

[DMDF36] August Dvorak, Nellie L. Merrick, William L. Dealey, and Gertrude Catharine Ford. Typewriting behaviour. 1936.

[ER06] Alan Esenther and Kathy Ryall. Fluid dtmouse: better mouse support for touch-based interactions. In *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, pages 112–115, New York, NY, USA, 2006. ACM.

[FWC84] James D. Foley, Victor L. Wallace, and Peggy Chan. The human factors of computer graphics interaction techniques. *IEEE Comput. Graph. Appl.*, 4(11):13–48, 1984.

[FWSB07] Clifton Forlines, Daniel Wigdor, Chia Shen, and Ravin Balakrishnan. Direct-touch vs. mouse input for tabletop displays. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 647–656, New York, NY, USA, 2007. ACM.

[GHB+06] Tovi Grossman, Ken Hinckley, Patrick Baudisch, Maneesh Agrawala, and Ravin Balakrishnan. Hover widgets: using the tracking state to extend the capabilities of pen-operated devices. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 861–870, New York, NY, USA, 2006. ACM.

[GR93] David Goldberg and Cate Richardson. Touch-typing with a stylus. In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, pages 80–87, New York, NY, USA, 1993. ACM.

[Gui87] Yves Guiard. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *Journal of Motor Behaviour*, 1987.

[Han05] Jefferson Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118, New York, NY, USA, 2005. ACM.

[HHCC07] Uta Hinrichs, Mark Hancock, Sheelagh Carpendale, and Christopher Collins. Examination of text-entry methods for tabletop displays. *Horizontal Inter-*

active Human-Computer Systems, International Workshop on, 0:105–112, 2007.

[HHN85] E.L. Hutchins, J.D. Hollan, and D.A. Norman. Direct Manipulation Interfaces. *Human-Computer Interaction*, 1(4):311–338, 1985.

[Hin02] K. Hinckley. INPUT TECHNOLOGIES AND TECHNIQUES. *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, pages 151–168, 2002.

[I. 02] I. Scott MacKenzie. A Note on Calculating Text Entry Speed. http://www.yorku.ca/mack/RN-TextEntrySpeed.html, 2002.

[IH00] Takeo Igarashi and Ken Hinckley. Speed-dependent automatic zooming for browsing large documents. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 139–148, New York, NY, USA, 2000. ACM.

[JGH+08] Robert J.K. Jacob, Audrey Girouard, Leanne M. Hirshfield, Michael S. Horn, Orit Shaer, Erin Treacy Solovey, and Jamie Zigelbaum. Reality-based interaction: a framework for post-wimp interfaces. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 201–210, New York, NY, USA, 2008. ACM.

[JKR09] Hans-Christian Jetter, Werner A. König, and Harald Reiterer. Understanding and designing surface computing with zoil and squidy. In *CHI Workshop on Multitouch and Surface Computing, held in conjunction with CHI'09*, New York, NY, USA, 2009. ACM.

[KB07] A.K. Karlson and B.B. Bederson. ThumbSpace: Generalized One-Handed Input for Touchscreen-Based Mobile Devices. *LECTURE NOTES IN COMPUTER SCIENCE*, 4662:324, 2007.

[KBBC05] M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Costanza. Tuio: A protokol for table-top tangible user interfaces. In *Proceedings of Gesture Workshop 2005*. Gesture Workshop, 2005.

[KBS94] P. Kabbash, W. Buxton, and A. Sellen. Two-handed input in a compound task. In *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence*, pages 417–423. ACM New York, NY, USA, 1994.

[KBSR07] Werner A. König, Hans-Joachim Bieg, Toni Schmidt, and Harald Reiterer. Position-independent interaction for large high-resolution displays. In

IHCI'07: Proceedings of IADIS International Conference on Interfaces and Human Computer Interaction 2007, pages 117–125. IADIS Press, Jul 2007.

[KHHK99] Clare-Marie Karat, Christine Halverson, Daniel Horn, and John Karat. Patterns of entry and correction in large vocabulary continuous speech recognition systems. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 568–575, New York, NY, USA, 1999. ACM.

[KHT06] Scott R. Klemmer, Björn Hartmann, and Leila Takayama. How bodies matter: five themes for interaction design. In *DIS '06: Proceedings of the 6th conference on Designing Interactive systems*, pages 140–149, New York, NY, USA, 2006. ACM.

[Kön06] Werner A. König. *Referenzmodell und Machbarkeitsstudie für ein neues Zoomable User Interface Paradigma*. mastersthesis, University of Konstanz, Jun 2006.

[KS86] Larry Koved and Ben Scneiderman. Embedded menus: selecting items in context. *Commun. ACM*, 29(4):312–318, 1986.

[MAK+88] D.E. Meyer, R.A. Abrams, S. Kornblum, C.E. Wright, and J.E.K. Smith. Optimality in human motor performance: Ideal control of rapid aimed movements. *Psychological Review*, 95(3):340–370, 1988.

[Med08] Medien- und Filmgesellschaft Baden-Württemberg. do it.konferenz 2008. http://www.doit-konferenz.de/, 2008.

[MH04] Tomer Moscovich and John F. Hughes. Navigating documents with the virtual scroll ring. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 57–60, New York, NY, USA, 2004. ACM.

[Mic08] Microsoft Corporation. Microsoft Surface Datasheet. , 2008.

[MS03] I. Scott MacKenzie and R. William Soukoreff. Phrase sets for evaluating text entry techniques. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 754–755, New York, NY, USA, 2003. ACM.

[MT65] M. S. Mayzner and M. E. Tresselt. Tables of single-letter and diagram frequency counts for various word-length letter-position combinations. *Psychonomic Monograph Supplements*, 1:13–32, 1965.

[Mye98] Brad A. Myers. A brief history of human-computer interaction technology.

interactions, 5(2):44–54, 1998.

[MZ99] I. Scott MacKenzie and Shawn X. Zhang. The design and evaluation of a high-performance soft keyboard. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 25–31, New York, NY, USA, 1999. ACM.

[MZ01] IS MacKenzie and SX Zhang. An empirical investigation of the novice experience with soft keyboards. *Behaviour and Information Technology*, 20(6):411–418, 2001.

[NF82] D.A. Norman and D. Fisher. Why Alphabetic Keyboards Are Not Easy to Use: Keyboard Layout Doesn't Much Matter. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 24(5):509–519, 1982.

[PWS88] R. L. Potter, L. J. Weldon, and B. Shneiderman. Improving the accuracy of touch screens: an experimental evaluation of three strategies. In *CHI '88: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 27–32, New York, NY, USA, 1988. ACM.

[RHL08] Anne Roudaut, Stéphane Huot, and Eric Lecolinet. Taptap and magstick: improving one-handed target acquisition on small touch-screens. In *AVI '08: Proceedings of the working conference on Advanced visual interfaces*, pages 146–153, New York, NY, USA, 2008. ACM.

[RK08] Harald Reiterer and Werner König. Media room - multimodal interaction & user interface design. http://hci.uni-konstanz.de/index.php?a=research&b=mediaroom&lang=en, 2008.

[Roh95] T. Rohrer. Metaphors we compute by: bringing magic into interface design. *Center for the Cognitive Science of Metaphor, Philosophy Department, University of Oregon*, 1995.

[RSF+08] Roman Course Participants: Rädle, Toni Schmidt, Simon Fäh, Harald Lectures: Reiterer, and Werner A. König. Course reader: Interaction design for high-resolution displays. Mar 2008.

[SM01] R.W. Soukoreff and I.S. MacKenzie. Measuring errors in text entry tasks: an application of the Levenshtein string distance statistic. In *Conference on Human Factors in Computing Systems*, pages 319–320. ACM Press New York, NY, USA, 2001.

[SM03] R. William Soukoreff and I. Scott MacKenzie. Metrics for text entry research: an evaluation of msd and kspc, and a new unified error metric. In *CHI*

'03: Proceedings of the SIGCHI conference on Human factors in computing systems, pages 113–120, New York, NY, USA, 2003. ACM.

[SRF+06] Chia Shen, Kathy Ryall, Clifton Forlines, Alan Esenther, Fr?d?ric D. Vernier, Katherine Everitt, Mike Wu, Daniel Wigdor, Meredith Ringel Morris, Mark Hancock, and Edward Tse. Informing the design of direct-touch tabletops. *IEEE Computer Graphics and Applications*, 26(5):36–46, 2006.

[SRP07] Helen Sharp, Yvonne Rogers, and Jenny Preece. *Interaction design : beyond human - computer interaction*. Wiley, Chichester, 2. ed. edition, 2007.

[Sut64] Ivan E. Sutherland. Sketch pad a man-machine graphical communication system. In *DAC '64: Proceedings of the SHARE design automation workshop*, pages 6.329–6.346, New York, NY, USA, 1964. ACM.

[VB07] Daniel Vogel and Patrick Baudisch. Shift: a technique for operating pen-based interfaces using touch. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 657–666, New York, NY, USA, 2007. ACM.

[YAM80] H. YAMADA. A Historical Study of Typewriters and Typing Methods: from the Position of Planning Japanese Parallels. *Journal of information processing*, 2(4):175–202, 1980.

[ZHS00] Shumin Zhai, Michael Hunter, and Barton A. Smith. The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 119–128, New York, NY, USA, 2000. ACM.

[ZHS02] S. Zhai, M. Hunter, and B.A. Smith. Performance Optimization of Virtual Keyboards. *Human-Computer Interaction*, 17(2 & 3):229–269, 2002.

[ZKS05] Shumin Zhai, Per-Ola Kristensson, and Barton A. Smith. In search of effective text input interfaces for off the desktop computing. *Interacting with Computers*, 17(3):229 – 250, 2005. Special Theme - Papers from Members of the Editorial Boards.

[ZS01] Shumin Zhai and Barton A Smith. Alphabetically biased virtual keyboards are easier to use: layout does matter. In *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, pages 321–322, New York, NY, USA, 2001. ACM.

[ZSA02] Shumin Zhai, Alison Sue, and Johnny Accot. Movement model, hits distri-

bution and learning in virtual keyboarding. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 17–24, New York, NY, USA, 2002. ACM.

[ZSS97] Shumin Zhai, Barton A. Smith, and Ted Selker. Improving browsing performance: A study of four input devices for scrolling and pointing tasks. In *INTERACT '97: Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*, pages 286–293, London, UK, UK, 1997. Chapman & Hall, Ltd.