Swabbing in the Wild

A longitudinal study of touchscreen input methods for users with hand tremor

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

by
Dennis Kehrig

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Christopher M. Schlick

Registration date: 2012-10-31
Submission date: 2013-06-06

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

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

Aachen, June2013
Dennis Kehrig

Contents

List of Figures

List of Tables

Abstract

Touch screens are less cognitively demanding than indirect input devices. Since cognitive abilities decline with age, touch screens can therefore improve usability for older users. Sadly, the elderly are more likely to be affected by a hand tremor, making tapping very hard. One alternative to tapping is called Swabbing. It attempts to compensate for tremor symptoms in numerous ways.

Previous investigations of Swabbing were limited to lab settings and short term use. In this thesis, a fully functional iPad web browser with optional Swabbing support was used for multiple weeks. For this purpose, Swabbing text entry was enhanced to allow editing text and entering an extended character set as needed for URLs. This thesis also shows how Swabbing could be used to make existing interfaces more accessible by providing Swabbing menus to activate links, buttons and form fields on web pages. This implementation of Swabbing is also the first to provide immediate feedback about the selection. Finally, an approach to double the previous capacity of Swabbing menus is presented.

This thesis shows that Swabbing can be used successfully outside of the lab. For text entry, the corrected error rate when using Swabbing is significantly lower than that of the iPad keyboard, while surprisingly, the uncorrected error rate is low for both techniques despite the user's strong tremor. While text input using two characters per option was faster than with one character per option, text input speed with the iPad keyboard was much higher, contradicting previous studies. More in line with previous studies, the user's link selection performance with Swabbing was eventually on par with his performance when tapping, while consistently requiring fewer touches, indicating fewer errors. For known selection targets, Swabbing outperformed tapping. The user's satisfaction was highest when using the Swabbing web browser with two characters per menu slot.

Überblick

Die kognitive Belastung bei der Benutzung von Touchscreens ist geringer als mit indirekten Eingabegeräten. Da mit dem Alter eines Menschen auch dessen kognitive Fähigkeiten sinken, können Touchscreens zu einer höheren Benutzbarkeit beitragen. Leider sind ältere Benutzer häufiger von einem Handtremor betroffen, der präzises Tippen deutlich erschwert. Eine Alternative zum Tippen ist Swabbing. Diese versucht in mehrfacher Hinsicht die Tremorsymptome auszugleichen.

Die bisherigen Studien zu Swabbing waren begrenzt auf Laborversuche und kurzzeitige Benutzung. Im Rahmen dieser Diplomarbeit wurde ein voll funktionsfähiger Webbrowser mit optionalem Swabbing-Support mehrere Wochen lang benutzt. Zu diesem Zweck wurde die Swabbing-basierte Texteingabe um die Möglichkeit erweitert, Text zu editieren und einen erweiterten Zeichensatz einzugeben, wie für URLs benötigt. Des weitern wird gezeigt, wie Swabbing benutzt werden kann, um existierende Oberflächen zugänglicher zu machen durch Swabbing-Menüs zum Anwählen von Links, Buttons und Formfeldern auf Webseiten. Diese Swabbing-Implementierung ist außerdem die erste mit Echtzeit-Feedback beim Eingabeprozess. Abschließend wird ein Ansatz zur Verdoppelung der bisherigen Kapazität von Swabbing-Menüs präsentiert.

Im Rahmen dieser Diplomarbeit wird gezeigt, dass Swabbing erfolgreich außerhalb des Labors benutzt werden kann. Bei Texteingaben ist die Rate der korrigierten Fehler signifikant kleiner als die des iPad-Keyboards. Überraschenderweise ist die Rate der unkorrigierten Fehler trotz starken Tremors bei beiden Eingabetechniken gering. Während die Texteingabe mit zwei Zeichen pro Menüeintrag schneller war als mit einem, war die Texteingabe mit dem iPad-Keyboard deutlich schneller, im Widerspruch zu vorherigen Untersuchungen. Eher passend zu vorherigen Ergebnissen konnte der User schlussendlich Links mit Swabbing ähnlich schnell anwählen wie mittels Tippen, mit durchgängig geringerer Anzahl an Bildschirmberührungen, ein Anzeichen für eine geringere Fehlerrate. Für bekannte Tippziele ist Swabbing schneller als Tippen. Der Benutzer war mit der Benutzung des Browsers mit zwei Zeichen pro Menüeintrag am zufriedensten.

Acknowledgements

I want to thank the following people:

Chatchavan Wacharamanotham
For great advice sessions and being an impressive role model
Alexander Mertens
For helpful discussions and helping me find users
Prof. Dr. Jan Borchers
For giving inspiring lectures and founding the great Media Computing Group
Prof. Dr. Schlick
For being my second examiner
Johanna Schorn
For proof reading and the most devoted friendship ever
Carl Huch
For being a great friend and equal
Horst and Dietlinde Kehrig
For always being there for me and having my back
Timo Kehrig
For showing me how it's done and being my bro
Ines Färber
For being a steady light in my life and her unsurpassed kindness
Jonathan Diehl
For his mentorship and the occasional push when I needed one
Maren Weissmayer
For finding a bug and sending a yummy thesis survival kit
Alexander Clauss
For thorough advice about iCab Mobile

Conventions

Throughout this thesis we use the following conventions.

Text conventions

Excursus are detailed discussions of a particular point in a book, usually in an appendix, or digressions in a written text.

Source code and implementation symbols are written in typewriter-style text.

```
myClass
```

The whole thesis is written in American English.

Chapter 1

Introduction

Indirect input devices like the mouse require mental translation between the human body and the machine, making them more cognitively demanding than direct input devices like touch screens (McLaughlin et al. [2009]). Since cognitive abilities decline with age (Salthouse [2009]), direct input devices like touch screens can therefore improve usability, especially for older users.

Sadly, the elderly are more likely to be affected by a tremor (Louis and Ferreira [2010]). Tremors cause the body to shake involuntary in a variety of circumstances. They can also affect a person's hands, making the most basic interaction technique on touch screens — tapping — extremely challenging.

Tapping refers to touching a point inside of a target area on the screen for a short time. Tapping is often critical for using a touch screen device successfully — for instance, on Apple's iPad tapping is used to start apps, enter text and navigate the settings menu.

One source of difficulty is the target's size - the stronger the tremor and the smaller the target, the easier it is to miss. In addition, tremors can cause movement even during the short time the finger touches the screen, possibly leading to the gesture being recognized as panning instead of tapping. Therefore, even if a button was hit successfully, it may not be activated. For able-bodied users, this is a welcome safety mechanism to abort accidental touches, making its removal problematic.

Even if buttons could also be activated by panning, target size remains an issue. Recommendations for the size of buttons on touchscreens are usually designed for able-bodied users. For instance, Jin et al. [2007] propose a button size of 19.05 mm square for older adults with poor manual dexterity. This recommendation still does not cover all cases, such as hand tremors categorized as "Severe $>$ 2 cm" (Hurtmanns [2011]), exceeding the proposed size. Consequently, even interfaces that keep these recommendations in mind may not be usable by all tremor sufferers.

Instead of optimizing current user interfaces on touch screens, the problem may be solved by using alternative input techniques. Ideally, such a technique would work in all scenarios that make use of touch screens. Consequently, solutions that require special hardware or the use of other body parts were not considered for this thesis, focusing instead on software-only solutions.

While there are multiple options to choose from, Swabbing is the only one designed specifically with tremor sufferers in mind. Swabbing divides the edge of the screen into an array of targets, each representing an action. Selection works by touching any point on the screen and sliding towards the desired action's target on the screen edge. Crossing the target is not necessary – instead, all touch points are used to calculate a vector, and the vector's direction determines the target.

With Swabbing, tremor symptoms are reduced by the surface friction when touching the screen (Wacharamanotham et al. [2011]) and further compensated for by using linear regression to calculate the vector. Ending the gesture by moving across the edge avoids additional tremor influences that occur when the user has to stop at a certain position (Hurtmanns [2011]). Since the selection is only based on the general direction of the touch points, crossing a different target can still trigger the correct action, allowing the target size to be smaller than the tremor strength.

Hurtmanns [2011] showed that tremor sufferers make fewer mistakes when using Swabbing for selection tasks than when using tapping, while being equally fast. Huck [2012] achieved similar results for text entry tasks using a full 360° layout for the Swabbing menu in comparison with a custom touch screen keyboard. However, due to the short duration of these studies, the learning curve of using either tapping or Swabbing remains unknown. In addition, these studies were restricted to lab settings, using very regular menu layouts, restricting text entry to uppercase letters and using a custom keyboard.

Through a longitudinal study that employs a custom web browser with optional Swabbing support, this thesis aims to overcome these limitations in numerous ways. First, by allowing the user to become familiar with the keyboard layouts, long-term text entry performance can be assessed more realistically. Second, the results are made more meaningful by comparing Swabbing with established interface elements as present on the iPad, in particular its sophisticated touch screen keyboard. Third, by adding a cursor menu, Swabbing's text entry capabilities are enhanced to allow editing existing text, such as the URL of a web site. This also requires the ability to insert special characters. Both features are critical for Swabbing to be usable outside of the lab.

The previous investigations focused on using Swabbing solely as an alternative to traditional interfaces, exemplified here by making the web browser itself controllable via Swabbing. However, this thesis also investigates how existing interfaces can be made more accessible to tremor users without modifying or replacing them. Here, these interfaces are provided by the web sites the user accesses: In addition to tapping, the user can use Swabbing to activate known tapping targets, such as links, buttons and form fields. This principle could be applied to other application interfaces as well, though further studies will have to show the limitations of this approach.

The Swabbing implementation used for this thesis is also the first to provide closed loop feedback to the user, i.e., information about the selected target before the user lets go of the screen. Feedback is provided in two ways. First, by changing the background color of the option the user is aiming at, the user can end a touch as soon as the intended target is confirmed, possibly shortening selection times. Second, a dot at the screen edge indicates the exact target point, allowing the user to notice when they are in danger of accidentally selecting a neighboring option. This could accelerate learning, and reduce the error rate by improving the input accuracy. However, this is not investigated as part of this thesis.

Since Swabbing gestures are intended to start in the middle of the screen, the user's fingers usually only pass through one side of it. This can be used to offer two actions per menu slot. Through linear regression, every gesture indicates a line crossing the entire screen. While one bit of information is provided by the side of the screen that is used to indicate the line (e.g., left or right), another bit is provided by the direction the user is moving in (e.g., away from the center or towards it). The line itself usually provides two menu options to choose from. The first bit could be used to decide which one to use (e.g., if the gesture was performed on the left side, use the left item), while the second bit can be used to decide which of the two actions to perform (e.g., if the item has an inner and an outer character, insert the inner character if the user moved towards the center). This approach was used to increase the opening angle of the options in the text menus by combining pairs of characters into single options, expecting a reduced error rate and/or increased selection speed since the selection can be less precise.

Chapter 2

Related work

While there are many approaches to generally increase usability of computers for motion-impaired users, the focus here is on techniques designed for or possibly applicable to touch screens and their potential in helping people with hand tremors, without requiring additional hardware, or using alternative body parts. This will ensure that the solutions are usable whenever touch screens are.

2.1 TRABING

Proposed by Mertens et al. [2010], TRABING is a "Touchscreen-based Input Technique for People Affected by Intention Tremor". Intention tremors occur at the end of an intentional movement towards a target if the person is looking at it. They therefore make tapping unreliable. TRABING exploits that the tremor symptoms are lower before reaching the target by letting the user touch the screen and slide off it towards an imagined target off the screen (see figure 2.1). The measured movement is therefore less affected by the tremor and can be used to trigger an action by comparing the direction of the movement with the location of targets. Tremor symptoms also were suspected to be reduced by the surface friction, a finding later confirmed by Wacharamanotham et al. Wacharamanotham et al. [2011].

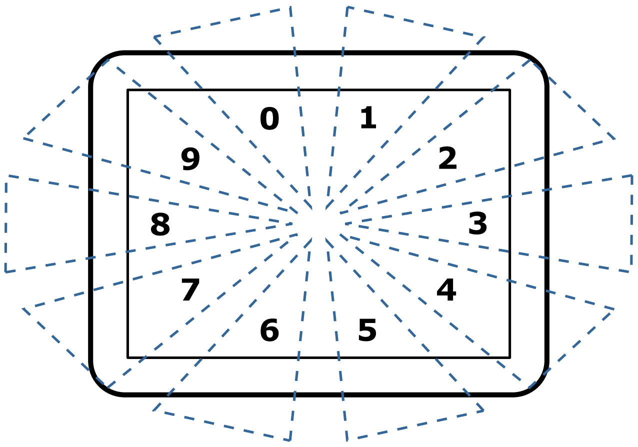


Figure 2.1: TRABING

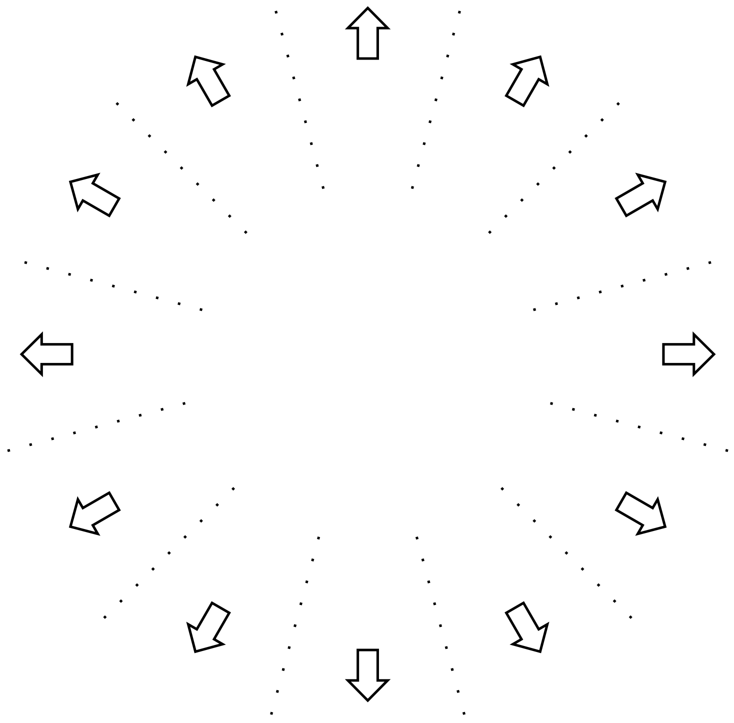


Figure 2.2: Swabbing

2.2 Swabbing

TRABING is described as using a "swabbing interaction movement" (Mertens et al. [2010]), a term that Jan Hurtmanns picked up when he investigated various basic designs for menus using this technique (Hurtmanns [2011]). His work was adapted to text input by Huck [2012].

Swabbing generalizes TRABING by using linear regression to determine the direction of the gesture from all recorded touch points. This makes it usable for people with tremors that affect any part of the movement, not just the end as with intention tremor. In addition, crossing the target is no longer strictly necessary. Instead, a threshold distance of about 4 cm has to be covered to provide enough coordinates to reliably calculate a direction (Huck [2012]). Since the selection is only based on the general direction of the touch points, crossing a different target can still trigger the correct action, allowing the target size to be smaller than the tremor strength. Since tremor sufferers sometimes accidently lift the finger while sliding over the surface, a swabbing movement is complete only when the finger has stopped touching the surface for 250 ms (Huck [2012]).

Keates et al. [2002] did not detect a significant difference in movement offsets (average distance from the optimal line) between able-bodied and motion-impaired users when using a mouse. Frett et al. found that users with Parkinsonian tremor and essential tremor had the lowest movement variability (MV) when using a FingerWorks MultiTouch Surface (MTS), and higher MV when using a mouse, trackball or joystick (Frett and Barner [2005]). Hurtmanns [2011] found that Swabbing produces lower errors rates than tapping and comparable selection times when used by tremor sufferers. These results suggest that Swabbing is a viable input technique for users with hand tremors.

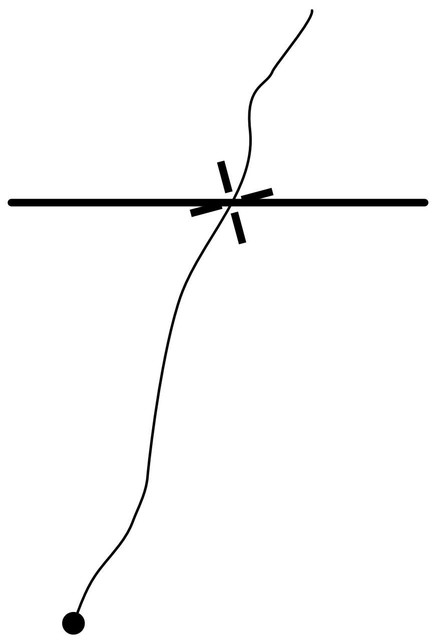


Figure 2.3: Goal Crossing

2.3 Goal Crossing

Instead of clicking a point within an area on the screen, goal crossing (Wobbrock and Gajos [2007]) lets the user merely cross a line that represents an action (Figure 2.3). Adapted to touchscreens, goal crossing would share one advantage with Swabbing: reduced tremor symptoms due to surface friction. The important difference is that only the short distance covered when crossing a potential target is used for target selection, again opening up the possibility for selection errors due to uncontrolled movement as caused by tremor. The goal lines have to be as wide as the tremor is strong, essentially causing the same problems as when merely adjusting the button sizes.

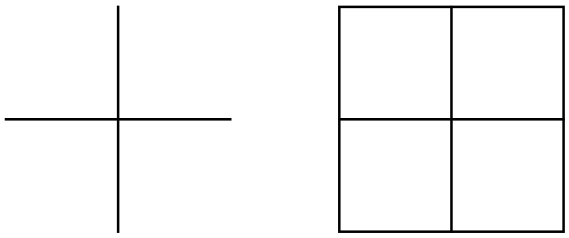


Figure 2.4: Point cursor (left) and area cursor (right)

2.4 Area Cursor

Normal mouse cursors have a so-called hot spot that is just one pixel wide. When clicking, exactly one pixel is selected on the screen, preventing any ambiguity about which target was aimed at. Area cursors increase the size of the hot spot to a wider area (Figure 2.4). Consequently, a click may target multiple items at once.

The implementation of Worden et al. [1997] resolves the ambiguity of area cursors by reverting to a normal cursor if multiple targets are within the cursor's area. This implementation therefore is effectively a normal mouse cursor except when clicking on a point with no associated target. In that case, it searches for the target in a predefined area around the cursor and clicks it if only one was found. Consequently, this approach would not decrease the chance of accidentally clicking a wrong target and is therefore not suitable for tremor sufferers. On the contrary, it essentially enhances the tremor strength by making it more likely to trigger a different target when missing the intended target, even if the other target was not hit directly, but happened to be the only target in the area around the hit point.

2.5 Bubble Cursor

Similar to the Area Cursor above, the bubble cursor (Grossman and Balakrishnan [2005]) is effectively a normal cursor except when no target is selected. Then it looks for the tar-

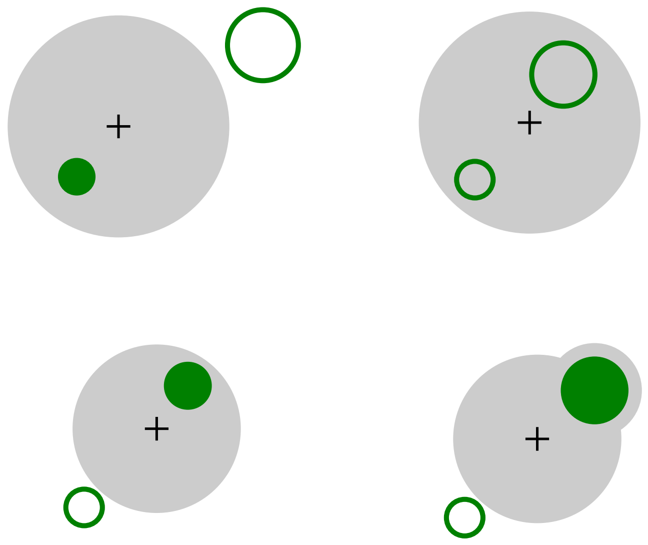


Figure 2.5: Bubble Cursor

get closest to the center and would activate that on click. It therefore poses the same problems as the area cursor described above. The difference is that the size of the cursor area is gradually reduced in case of conflict, instead of directly reverting to area pointing (Figure 2.5).

2.6 Enhanced Area Cursors

(Findlater et al. [2010]) proposed four versions of enhanced area cursors, all of which divide the task of clicking a target into two parts: activation and selection. Targets within the cursor's area are activated on click in the Click-and-Cross design. In the Cross-and-Cross design, the cursor area is moved by moving the point cursor to the area's edge, thus dragging it along. In the direction opposite of the movement, part of the circle's outline forms a trigger arc. Activation is performed by reversing the direction of movement, crossing the trigger arc, and stopping the movement for 300ms.

Then, both designs present a circular menu around the cur-

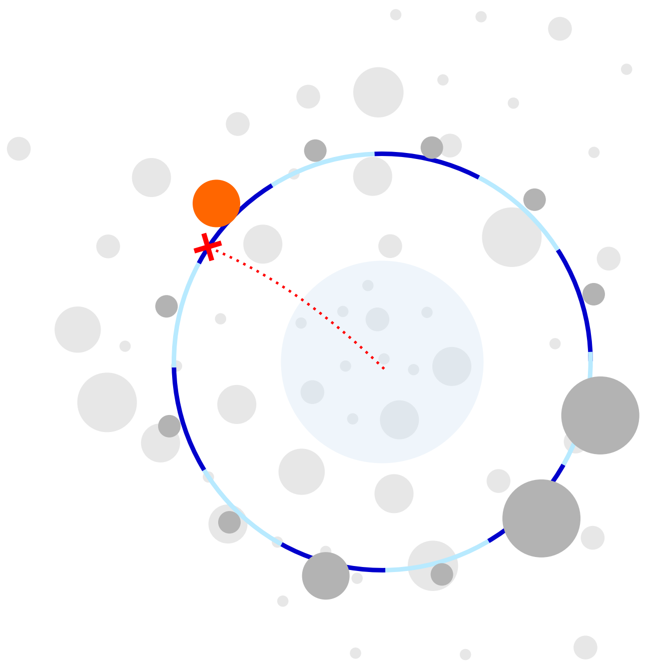


Figure 2.6: Selection step in the Click-and- Cross and Cross-and-Cross designs

sor area (Figure 2.6) and position the point cursor in the center of that circle. Each selected target is assigned an equally sized arc of the circle. Activation is performed by crossing the arc belonging to the intended target, and stopping for 300ms.

On first glance, the activation phase seems similar to Swabbing. The most important difference is that with the area cursors the direction of the movement is irrelevant, the importance lies on the arc that was crossed. For people with strong tremors it is very easy to generally move towards one arc, but finally cross a different arc due to the tremor (see figure 2.7). With Swabbing, crossing a different target is unproblematic as long as the general direction of the movement still points towards the intended target. Unlike Swabbing, the circle will usually be smaller than the screen. Even if the direction of the movement were to determine

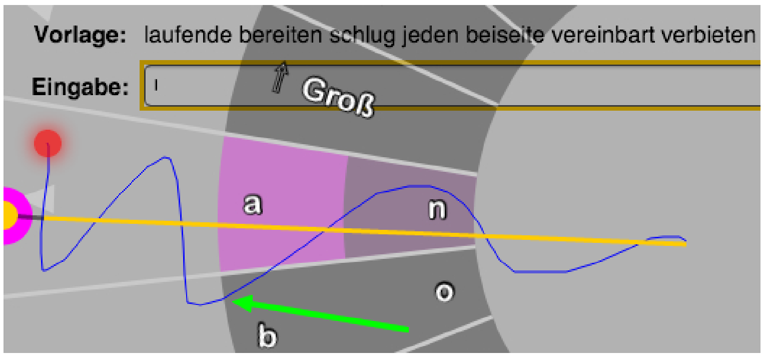

Vorlage: laufende bereiten schlug jeden beiseite vereinbart verbieten
Eingabe:
Groß
a
n
o
b

Figure 2.7: Gesture path that triggers the correct action with Swabbing, but would trigger an incorrect action if this were arc crossing

the target, fewer cursor coordinates can be recorded until activation, resulting in lowered accuracy.

However, a similar division of activation and selection can be found in the aiming menu that I propose. Activation is performed by opening the aiming menu, essentially activating all targets on the screen. Selection is performed by using the resulting menu.

Activation in the Motor-Magnifier design and the Visual-Motor-Magnifier works the same way as with Click-and-Cross. In the Motor-Magnifier design, selection is eased by lowering the mouse gain. This cannot be adapted to touch screens. In the Visual-Motor-Magnifier-Design the activated area is simply enlarged (by default, to four times the size) and activation is then performed by using a bubble cursor. This could be adapted to touch screens and would certainly improve many situations.

However, one problem remains. Despite being designed for the case of small targets tightly packed together, advanced area cursors still have a weakness when there are many such targets close together. Similar to the Bubble Cursor, the size of the area will shrink to encompass at most ten targets. To activate these targets, then, movement has to be more precise as a smaller area needs to be hit. Since the Swabbing based aiming menu considers the whole screen in all cases, the activation step is not affected by the number, size, or clustering of targets.

Chapter 3

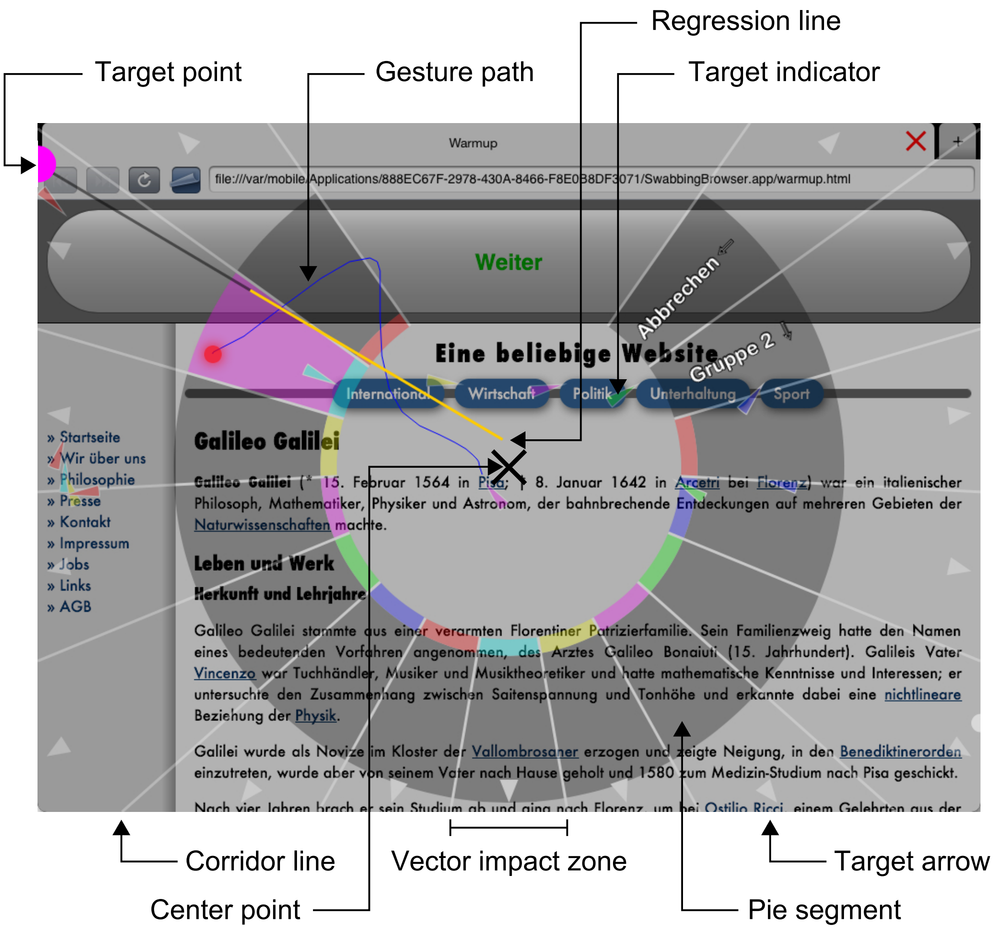
Swabbing implementation

For Swabbing-related definitions, see figure 3.1.

3.1 Visual design

The basic approach of Swabbing is to divide the screen edge into segments (called “vector impact zones”), and assign menu options to those segments. Hurtmanns [2011] tested several visual designs and finally chose to display the options as equally sized segments of a circle (see figure 2.2). A large area in the center of the circle would remain empty to invite the user to place a finger there. Each option is represented by an arrow pointing from the circle’s center towards the appropriate vector impact zone. In addition, dotted “corridor lines” provide information about the size of the vector impact zones, and allow the user to see when they are about to aim away from the intended target.

For this thesis, the design had to be changed in order to work as an overlay. A black background with 30% opacity shades the application to make the menu more visible. A precomposed color layer, also with 30% opacity, contains the pie segments (black), corridor lines (white), target ar-



Galileo Galilei

Galileo Galilei (* 15. Februar 1564 in Pisa; † 8. Januar 1642 in Arcetri bei Florenz) war ein italienischer Philosoph, Mathematiker, Physiker und Astronom, der bahnbrechende Entdeckungen auf mehreren Gebieten der Naturwissenschaften machte.

Leben und Werk

Herkunft und Lehrjahre

Galileo Galilei stammte aus einer verarmten Florentiner Patrizierfamilie. Sein Familienzweig hatte den Namen eines bedeutenden Vorfahren angenommen, des Arztes Galileo Bonaiuti (15. Jahrhundert). Galileis Vater Vincenzo war Tuchhändler, Musiker und Musiktheoretiker und hatte mathematische Kenntnisse und Interessen; er untersuchte den Zusammenhang zwischen Saitenspannung und Tonhöhe und erkannte dabei eine nichtlineare Beziehung der Physik.

Galilei wurde als Novize im Kloster der Vallombrosaner erzogen und zeigte Neigung, in den Benediktinerorden einzutreten, wurde aber von seinem Vater nach Hause geholt und 1580 zum Medizin-Studium nach Pisa geschickt.

Nach vier Jahren brach er sein Studium ab und ging nach Florenz, um bei Ostilio Ricci, einem Gelehrten aus der

Figure 3.1: Swabbing-related definitions

rows (white) and target indicators (various colors). The precomposition prevents a target indicator from blending with the colored part of a pie segment since this would make the color hard to recognize, a problem the prototype had. Finally, a text layer with 100% opacity contains the labels with their outlines.

Previous implementations of Swabbing ordered items in clockwise direction. For this thesis, the order was reversed to achieve a more natural mapping of the items in the tabs menu to the tabs of the traditional interface. For this thesis, several menu entries needed to be longer than one character. For the initial version of the prototype, an attempt was made to only rotate items when necessary, but in later iterations, a more consistent approach was chosen by rotating all labels that are longer than one character. In addition to labels, many entries feature icons. However, these are merely unicode characters. In some cases the iPad replaces the unicode characters with colored icons, e.g., for the arrows used in the back and forward options.

During the main study, the full opacity of the text layer turned out to be problematic since some of the words were covered by a label, and therefore partially unreadable. In most cases, using a web browser will not require the user to transcribe a string, although there are exceptions, such as captchas. Consequently, adding enough transparency to still allow reading text below a label may be advisable. Due to the application in the background, visual noise was a concern, so the corridor lines are not dashed or dotted.

While previous Swabbing-related studies cropped the screen to a square, this would have wasted valuable screen real estate for the browser. Therefore, with the iPad's aspect ratio of 4:3, this design is the first non-square implementation of Swabbing. If the user starts the selection gesture off center, it is vital that they aim towards the vector impact zone, and not to the pie segment. Otherwise, the wrong option may be activated, which is more likely in non-square designs. To emphasize the importance of the vector impact zones, the corridor lines were extended all the way to the edge of the screen. In addition, the target arrows were added, providing a visible goal to aim at. See chapter 6 "User study 2: Swabbing feedback" for a preliminary study investigating the effectiveness of this approach.

For the feedback visualization, the prototype simply reduced the transparency of the active pie segment. But due to the precomposition, this was not possible in the final design - the grey pie segments in figure 3.1 are actually completely black. Instead, magenta was chosen as a highlighting color, as it is not often used in web sites and would therefore usually stick out. The target point was also colored magenta to indicate the shared nature with the pie segment highlighting.

Similar to the 270° and 180° designs by Huck [2012], the top of the menu contains a gap, in response to findings by Moscovich [2009] about fingers feeling stuck when moving in the upward direction. The design here is 290° wide, a size that grew organically. In an attempt to make a well founded choice, it was considered to use the full 360° since that produced the best results (Huck [2012]). However, by then the text entry menus already included an area to show the text the user is entering, since the actual input field may be covered by the menu. Removing the gap would have meant moving the text area into the center of the screen. This might inhibit users when touching the center to start a selection gesture, thereby impacting text entry speed negatively. Furthermore, the typed text would often be covered by the user's hand, possibly leading to more errors. Lacking a qualified alternative, the decision was made to keep the design as is.

3.2 Gesture detection

In the simplest case, the user touches the screen, moves the finger into the direction of the vector impact zone that belongs to the intended target, and lifts their finger off the screen. If the distance between the first and last touch point is at least 40.63mm (the same threshold Huck [2012] used) the intended target is triggered after a delay of 250ms. During this time, the user could touch the screen again to continue the gesture - this counteracts the accidental lift-offs that tremor sufferers are prone to.

As the user moves their finger, the system records the touch coordinates and adds them to a path - a sequence of coordinates. Every time coordinates are added, the system analyzes the whole path using linear regression. Consider the formula used for linear regression, n being the number of touch coordinates, with x_i and y_i being the x- and y-coordinate, respectively:

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \overline{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

$\beta = \frac{\sum_{i=1}^{n}(x_i-\overline{x})*(y_i-\overline{y})}{\sum_{i=1}^{n}(x_i-\overline{x})^2}$

$\alpha = \overline{y} - \beta * \overline{x}$

$y = \alpha + \beta * x$

Assume that the user moves downwards, and also a little bit to the right. The sign of $x_i - \overline{x}$ will always be opposite that of $y_i - \overline{y}$, consistently leading to $(x_i - \overline{x}) * (y_i - \overline{y}) \leq 0$ and therefore to a gradual decrease of β's dividend. Since $(x_i - \overline{x})^2 \geq 0$ always holds, the line's slope will be negative.

Now assume that the user continues their movement, moving downwards just as fast as before, but a little bit to the left instead of to the right. Gradually, the calculated line's orientation should change from south east over south to south west. Instead, the line will rotate in the opposite direction. When moving to the left, $x_i - \overline{x}$ will eventually become negative when $y_i - \overline{y}$ is negative, too. In those cases $(x_i - \overline{x}) * (y_i - \overline{y}) > 0$ holds, leading to a gradual increase of the dividend where previously it would consistently decrease. Since $(x_i - \overline{x})^2 \geq 0$ always holds, the slope will gradually increase, leading to the observed effect.

Since this effect is not observed when moving mostly horizontally, a simple remedy is rotating the coordinate system to calculate the linear regression. In the current implementation, whether to rotate or not is determined for every analysis depending on the first and last touch point: if the horizontal distance is smaller than the vertical distance, the movement is considered vertical and the coordinates are rotated before calculating the vector.

When moving roughly diagonally, this can cause repeated switching between using rotation and using the coordinates as-is. In some cases, the target point calculated this way can differ significantly between the two methods. If the target point is visualized, it will seem to jump back and forth when crossing the diagonal as the user continues to move their finger. Therefore, it may be useful to stick to one method once the target point is first visualized, though this will re-introduce the original problem if the user adjusts the direction dramatically.

As long as the first finger touches the screen, additional touches are simply ignored. If the user lifts a single finger off the screen and touches it again within 250ms, the new touch point will be considered in the same way as if the user had moved the finger there without lifting it off the screen — the path used to calculate the Swabbing vector is simply continued. For accidental lift offs, this is appropriate, but occasionally the user would just briefly touch the screen again by accident. To see how this can cause errors, see figure 3.2.

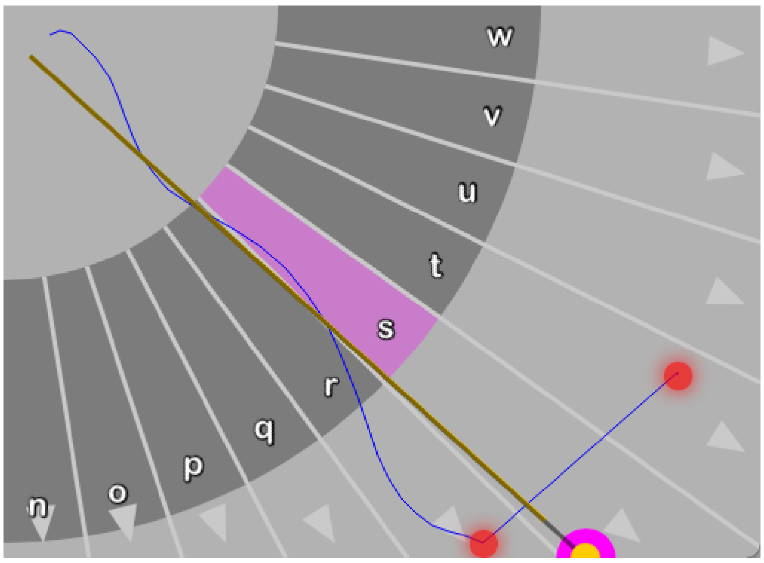


Figure 3.2: Incorrect selection due to an accidental extra touch within 250ms after letting go of the screen

Now assume that the touches interleave, i.e., the first finger leaves the screen while other fingers are still touching it. If any of the other fingers move within 250ms, the recognition path is continued with the location of the closest finger still touching the screen[1]. Note that on the iPad all active touches are reported if any one of them move, though some may be marked as "stationary". On other platforms, the lo-

[1]The requirement for the remaining touches to move was unintentional — the path should continue with the closest touch right away, even before it moves again (if ever). This way, the gesture would end when all fingers have left the screen, not before. However, since tremor users are unlikely to keep a touch stationary for 250ms, this will probably not have caused any problems in this thesis.

cation of the other fingers may have to be retrieved from memory. The closest finger will become the lead finger, i.e., as long as it touches the screen, other touches will be ignored.

3.3 Toggling the overlay

The Swabbing overlay can be turned on and off by tapping the screen with five fingers. This gesture is recognized as soon as five fingers are simultaneously touching the screen, i.e., they do not need to begin simultaneously. The gesture is blocked from being recognized again for one second in order to prevent accidental double tapping from undoing a previous performance of the gesture.

The iPad has global gestures using four or five fingers to switch between running apps by swiping left or right, bring up the task bar by swiping upwards, or show the home screen by moving the fingers inwards. While it is not enough to just tap the screen for those gestures to be recognized, a user with hand tremor will likely move his fingers enough for the system to recognize the movement as swiping, thus triggering the global gestures. On the iPad used for this thesis, these gestures were simply deactivated.

To prevent other parts of the software from becoming less responsive, the gesture recognizer for the five finger tap runs in parallel to all other gesture recognizers. Consequently, the touches can have other effects until the five finger tap is recognized, such as activating links on the page. A Swabbing-only software would not have this problem.

This exact problem does not occur when the Swabbing layer is visible, as single touches have no effect before crossing the threshold distance. However, since for Swabbing gestures a second finger is only ignored while the first finger is still touching the screen, briefly lifting the first finger off the screen would cause the system to continue the recognized path at the location of the second finger. If the resulting path is longer than the threshold distance, a Swabbing entry may be activated. In the next paragraph, a related issue is discussed and the same remedies apply.

3.4 Improving multitouch handling

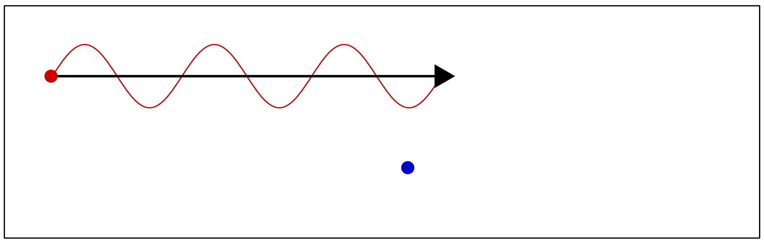


Figure 3.3: Red: first touch's path, blue: a second touch began, black: calculated vector

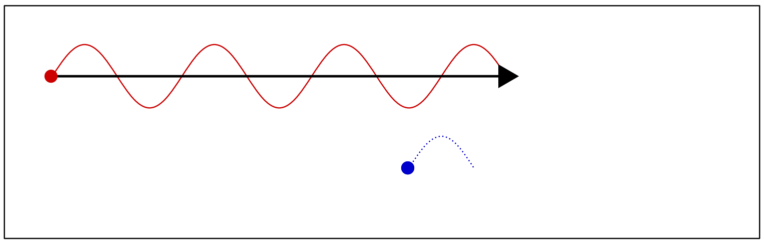


Figure 3.4: Since the first touch has not yet ended, the second touch is ignored

In a small spontaneous experiment, the user was asked to first put one finger on the screen deliberately, and then put up to three additional fingers on the screen before starting to move towards the intended menu option. This way, the user's hand rests on the surface and has more physical support, possibly reducing the tremor effects. By placing one finger on the screen first, the user can determine the finger that the system uses to recognize the Swabbing gesture. This worked rather well[2], but revealed a problem with the way additional fingers (see figure 3.3) are handled: when moving multiple fingers (see figure 3.4) towards the edge of the screen, not all of them will leave the screen at the

[2]The user was able to perform the gestures well, but mentioned added cognitive effort due to consciously having to choose a finger to start with. Consequently, this technique should not be a requirement.

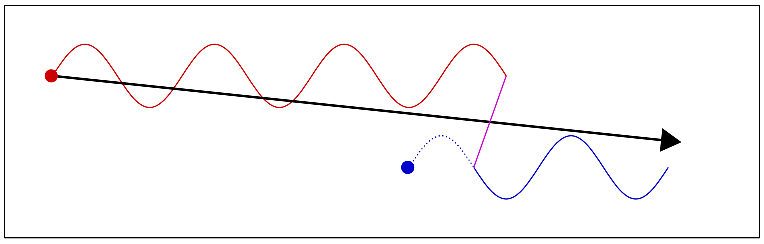


Figure 3.5: The first touch has ended and the recognized path is immediately continued by the second touch. Even though the touches' paths are parallel, the vector's direction is altered

same time. If the lead finger is not the last finger to cross the screen edge, the system will continue the recognition path with the location of the closest finger still touching the screen (see figure 3.5), possibly changing the recognized direction so much that the wrong target is selected even though the lead finger left the screen when the intended target was selected.

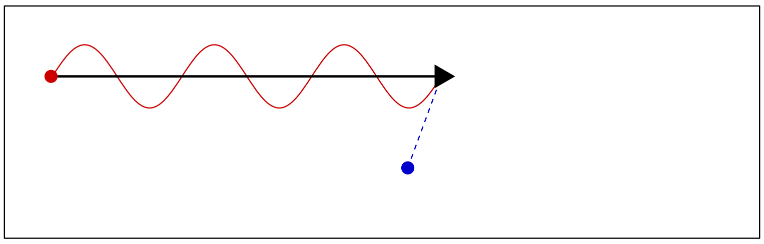


Figure 3.6: Red: first touch's path, blue: a second touch began, black: calculated vector. The dotted line is the offset of the second finger's initial touch point from the current location of the first finger

The problem described in the previous paragraph results from respecting the origin of the second finger's touch points instead of just their direction and distance from the second finger's initial touch point. When the second finger is touching the screen (see figure 3.6), it can be assumed to belong to the same hand and therefore should generally move in the same direction as the first finger, just from a different starting point (see figure 3.7). By translating the touch coordinates of the second finger with the initial off-

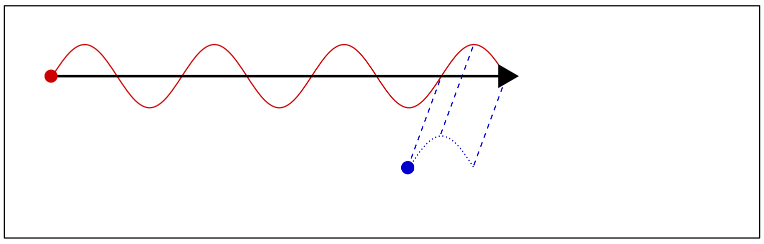


Figure 3.7: Red: Since the first touch has not yet ended, the second touch is ignored

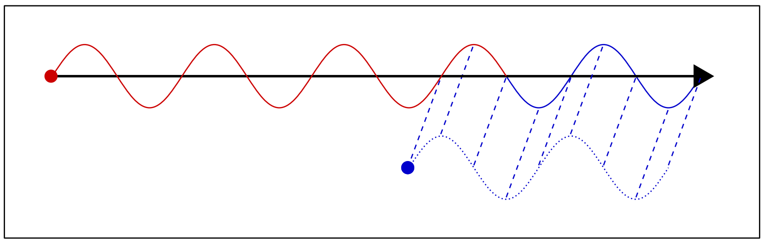


Figure 3.8: The first touch has ended, and the recognized path is continued with the second touch, but translated using the initial offset from the first touch. As long as the touches' paths continue to be parallel, the vector's direction remains the same

set from the first finger's location, the second finger's touch points would influence the vector exactly as if they had come from the first finger (see figure 3.8). This would effectively enhance the touchscreen's recognition area since the second finger would continue the first finger's path long after it has left the screen. Technically, this could be continued indefinitely - when the first finger has left the screen, the second finger's **translated** coordinates would be used to calculate the offset of a third finger, resulting in a cumulative extension of the recognized area.

3.5 What is the target size?

In order to maximize the reliability of using Swabbing, all options should generally be equally difficult to select. On the surface, this seems to have been the case in all current Swabbing implementations, since all options have the same opening angle. But is that enough?

Usually, to perform a Swabbing selection, a user would touch the screen in its center and move towards the screen edge in the direction of the intended target. The more touches can be recorded to calculate the regression line, the more stable the target point, increasing reliability. Since the screen edge is further away when moving diagonally (and for non-square layouts also in the direction of the wider side), more touch coordinates can get recorded, which should make these more reliable to hit.

Now assume the user were to start the gesture as close to the screen edge as when moving downwards from the center. While this removes the advantage of additional data for the calculation, the opening angle between the starting point and its connections to the corners of the vector impact zone is bigger than from the center, indicating a target size that is bigger than when moving downwards - a geometrical advantage.

But just how big is this advantage? We can draw some inspiration by looking at how Fitts' law (see Fitts [1954]) is used to assess the difficulty of moving a hand to a specific location, as required for clicking tasks. To calculate the index of difficulty, Fitts' law defines the width of the target as the length of that part of the line in direction of movement that is enclosed by the target area. In other words, for non-circular targets, the target width differs depending on the starting position.

Since Swabbing gestures can be started anywhere on the screen, a useful definition of the target size should depend on the origin as well. To highlight that importance, estimate the difficulty of selecting the letter "s" in figure 3.9, depending on the gesture's origin.

Except in rare cases, such as a 360^o menu with eight options, all current Swabbing implementations include this bias. It is present most strongly in the implementations for this thesis due to the non-square layout. However, without

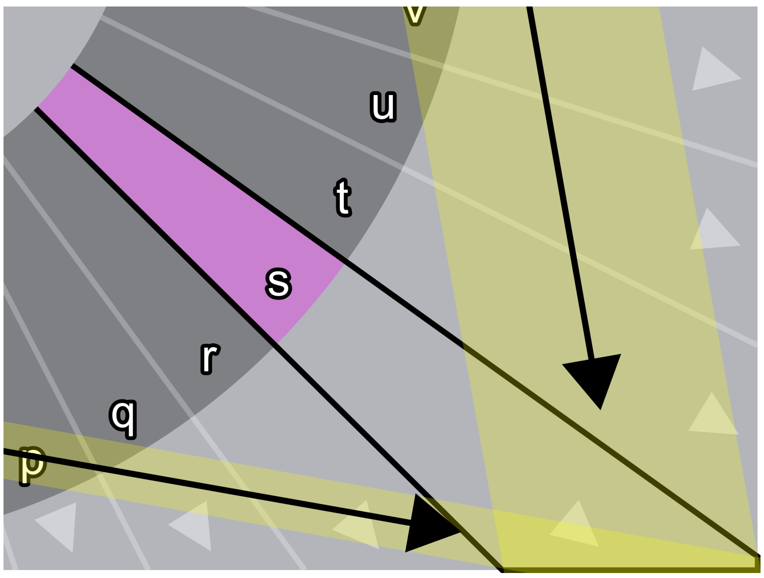


Figure 3.9: The selection difficulty depends on the origin of the selection gesture

a clear definition of a target's size it is unclear how to design menus that are objectively fair. The simplest solution would be to clip the gesture recognition area to the menu's circle. Sadly, this would simply reduce overall accuracy by shortening most gesture paths.

Figure 3.10 shows candidate definitions for measuring the target size. From the starting point of the gesture, the angles towards the corners of the vector impact zone are calculated. In the example the starting point is the center of the menu, therefore the angles coincide with those of the corridor lines. The mean of the angles determines the angle of the ideal gesture path, shown as the gray line in the middle. A, B and C are lines perpendicular to this ideal gesture path, clipped by the lines through the start point and the corners of the vector impact zone. They all represent a possible way to assess the target size relative to the starting point of the gesture.

Figure 3.11 reveals two flaws of these approaches: As the distance to the screen edge becomes shorter, the target size changes, which is counterintuitive. Worse still, the angle of

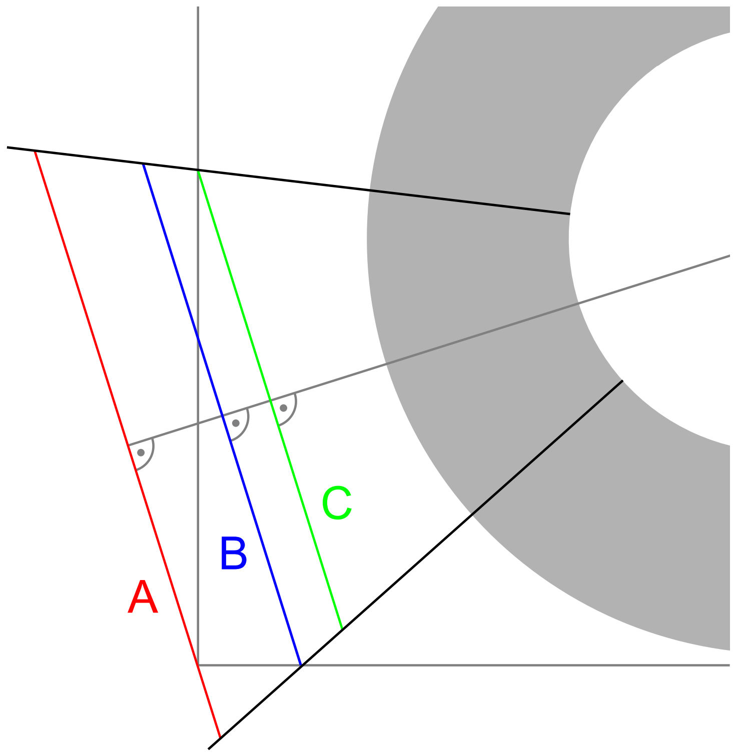


Figure 3.10: Candidates to determine the target size

the ideal gesture path changes as well.

The definition of the target size should only depend on the points that intersect with the screen edge and the angle between the target and the starting point of the gesture. Compare this to playing pool: the size of the pocket does not change as the ball gets closer, although it is widest when the line of sight is perpendicular to the line between both ends of the pocket. A possible definition for the target size therefore would be the diameter of the largest circle that could fit between the corner points when moved towards the target in a straight line from the gesture starting point (see figure 3.12).

Assume (g^x, g^y) is the gesture starting point, while (c_1^x, c_1^y) and (c_2^x, c_2^y) represent the corner points. Then, $m^x = c_1^x +$

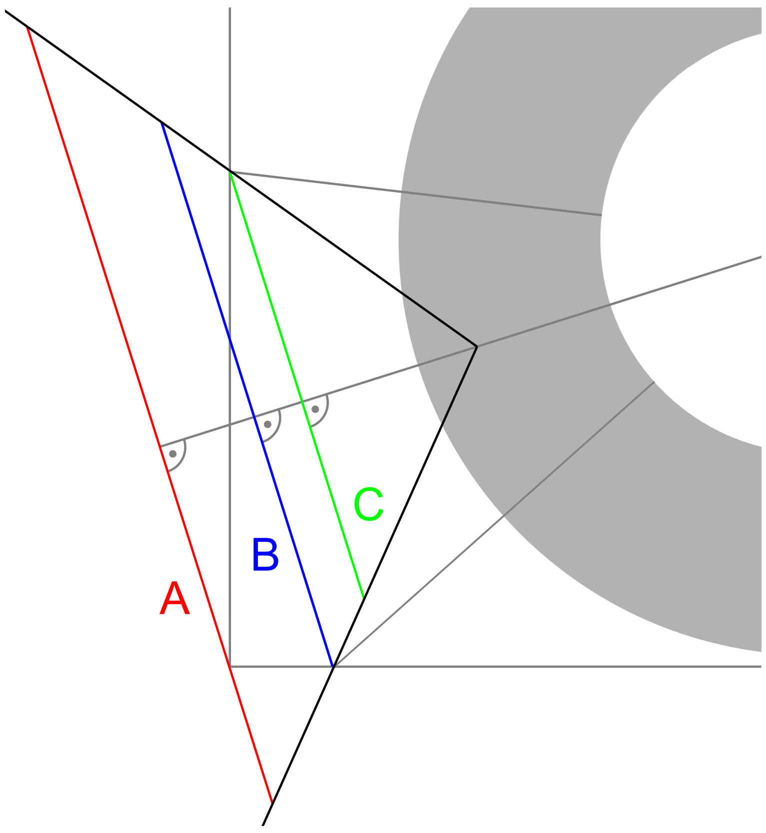


Figure 3.11: The definitions have two flaws: target size and ideal gesture path would change with the distance to the target

$\frac{c_2^x - c_1^x}{2}$ and $m^y = c_1^y + \frac{c_2^y - c_1^y}{2}$ represent the point in the middle of the line connecting the corner points.

The line for the ideal gesture path can then be described by $g(x) = a_1 x + b_1$ with $a_1 = \frac{g^y - m^y}{g^x - m^x}$ and $b_1 = g^y - a_1 g^x$. Note that for simplicity, special cases like vertical lines are ignored.

Next, we need the line perpendicular to g, described by $s(x) = a_2 x + b_2$ with $a_2 = -\frac{1}{a_1}$ and $b_2 = c_1^y - a_2 c_1^x$. We can use this to calculate the point where g and s intersect: $i^x = \frac{b_2 - b_1}{a_1 - a_2}$ and $i^y = s(i^x)$. Finally, the target size is the doubled distance between i and c_1: $t = 2\sqrt{(c_1^x - i_x)^2 + (c_1^y - i_y)^2}$.

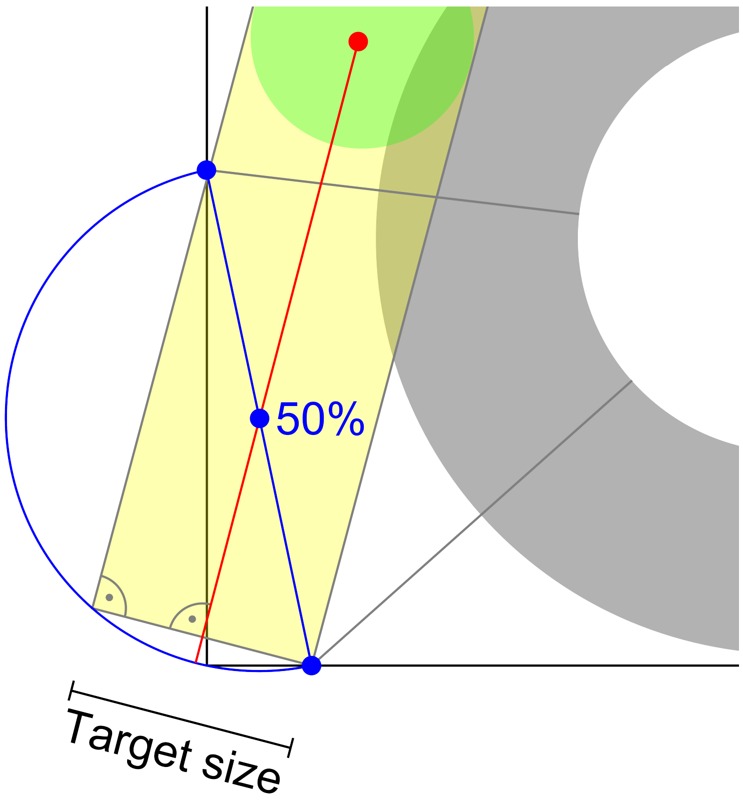


Figure 3.12: Target size as the width of the widest possible corridor that still fits between the corner points of the vector impact zone, starting from the gesture starting point (red) and moving towards the mid point between the corner points (blue)

In order to get an equivalent of Fitts' law for Swabbing, we also need to take into account the distance between the target and the gesture's starting point. However, there are several difficulties when considering the distance. On the one hand, it is harder to aim correctly right away at targets that are further away. When starting the gesture at the center point, this problem does not exist due to the equal opening angles. But the further away the starting point from the target, the harder it is to see with the naked eye how accurately the finger is moving in the direction of the target.

On the other hand, due to the greater distance, the length of the gesture path can be increased, improving accuracy by providing more touch coordinates to balance out Swabbing symptoms. Due to the linear regression, the reduction in difficulty is probably not strictly linear - doubling the gesture length may increase the accuracy more or possibly also less than twofold. Finally, in contrast to tapping tasks, the target does not have to be reached. Rather, the user determines themselves how far they move. Therefore, when attempting to calculate the difficulty of selecting a certain target from a specific point, the maximum direct distance to the target should only be used if the user plans on moving all the way to the target - otherwise, using the average length of their gestures may be more appropriate.

Note that Huck [2012] grouped recorded gestures into eight distinct directions and analyzed their differences in the error rate in order to assess whether users have more trouble moving in one direction than in another. According to the findings above, the error rates should be systemically lower in the diagonal directions. However, no such trend was found - whatever bias equal opening angles produced is far exceeded by variances in the tremor strength depending on the direction of movement. For right handed users, moving to the left had the lowest error rate of 4.44% while moving diagonally in the lower right direction produced the second highest error rate of 12.5%. The second lowest error rate was 5.04% for the upper right direction — clearly, the main contributor to errors is not simply whether the movement is diagonal.

Consequently, while an objective definition of target size is desirable, an objectively fair menu might not be the best option when it comes to increasing Swabbing reliability. If the directions of movements with higher error rates are stable over time for a given user, the size of the options most afflicted could be increased instead. Since the user hardly made any mistakes during the Swabbing-based text entry tasks of the main study, it is unknown whether this is the case.

Another option would be to discard the circular layout of Swabbing menus entirely, and adopt list-like layouts that stick to the directions with the highest accuracy (see 8

"Summary and future work"). Note that this would require a redesign of the aiming menu and its target indicators.

For this thesis, constant angles were used, maintaining visual uniformity. As a result, the distance between the options' labels is always the same. When the user tries to estimate the position of an option that is part of a sequence (such as a letter in the alphabet or a digit), the uniform distribution may lead to more accurate results than with varying label distances, as produced by using an ellipsis instead of a circle or unevenly spaced menu options. It may also be easier for the user to scan the available options. Further studies are necessary to verify either hypothesis.

Chapter 4

Software Testbed: Touchscreen Web Browser

Originally it was considered to implement Swabbing in the browser as a bookmarklet, i.e. a bookmark that, when opened, injects JavaScript code into a web page to transform it. This would still have required tapping to activate the bookmarklet, and so an extension that would load automatically seemed more appropriate.

Sadly, only one browser for the iPad with support for extensions was found — iCab (see Clauss) — and extensions were executed only once the entire page was loaded. On many web pages this would have meant waiting for several seconds until Swabbing support became available, which was unacceptable.

Consequently a custom app needed to be developed, or an existing one modified. Chrome for iOS was one candidate as a starting point that was rejected due to a warning in the build instructions saying that "it is not currently possible to build the actual browser binary". The most promising choice was Foxbrowser (see Grätzer) since its code is openly available on github.com. Unfortunately most of the more complex features depended on Firefox Sync and would therefore have needed to be rewritten.

The final decision then was to write the iPad browser from scratch.

4.1 General requirements

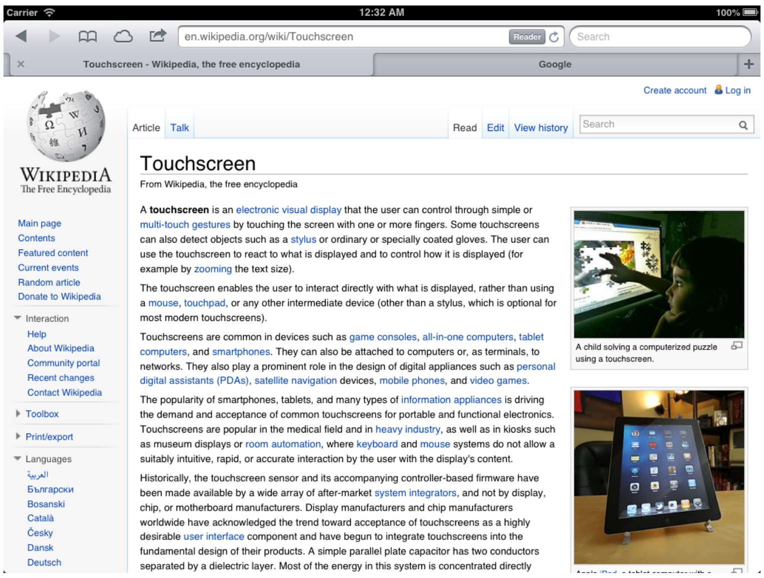
The most fundamental functionality of a web browser is navigation: displaying websites at arbitrary locations and following links to other pages at the request of the user. Users should be able to enter and open arbitrary URLs, as well as activate links and buttons on web pages. For the user's convenience, a browser should record a history of visited locations and allow jumping back and forth between those. Users should be able to branch out by opening a new tab, switching between tabs and closing old ones. If users access a website by mistake, they should be able to stop the page from loading. In case a website did not load properly, or is suspected to be outdated, a reload function should be offered. For convenience and memory support, the browser should also offer the ability to add, load and delete bookmarks.

When using search engines and websites that require user credentials before granting access, support for filling out forms is vital as well. Among other things this requires the ability to focus input fields, toggle checkboxes, activate radio buttons, selecting options from a drop down list, and submitting the filled out form. Filling in text input fields requires the ability to enter and edit text.

Finally, the user needs to be able to scroll and adjust the zoom level. This is especially important for users with bad eyesight.

4.2 The traditional interface

Loosely modeled after the iPad's default web browser Safari (see figure 4.1), the custom browser's traditional interface (see figure 4.2) features a horizontal list of tabs, each showing the title of the page in the tab and a button to close the tab. On the right of the tab bar, there is a button to open new tabs. Opening and closing tabs works by tapping the respective buttons while tapping a tab's title will make this tab the current one.



Touchscreen

From Wikipedia, the free encyclopedia

A **touchscreen** is an electronic visual display that the user can control through simple or multi-touch gestures by touching the screen with one or more fingers. Some touchscreens can also detect objects such as a stylus or ordinary or specially coated gloves. The user can use the touchscreen to react to what is displayed and to control how it is displayed (for example by zooming the text size).

The touchscreen enables the user to interact directly with what is displayed, rather than using a mouse, touchpad, or any other intermediate device (other than a stylus, which is optional for most modern touchscreens).

Touchscreens are common in devices such as game consoles, all-in-one computers, tablet computers, and smartphones. They can also be attached to computers or, as terminals, to networks. They also play a prominent role in the design of digital appliances such as personal digital assistants (PDAs), satellite navigation devices, mobile phones, and video games.

The popularity of smartphones, tablets, and many types of information appliances is driving the demand and acceptance of common touchscreens for portable and functional electronics. Touchscreens are popular in the medical field and in heavy industry, as well as in kiosks such as museum displays or room automation, where keyboard and mouse systems do not allow a suitably intuitive, rapid, or accurate interaction by the user with the display's content.

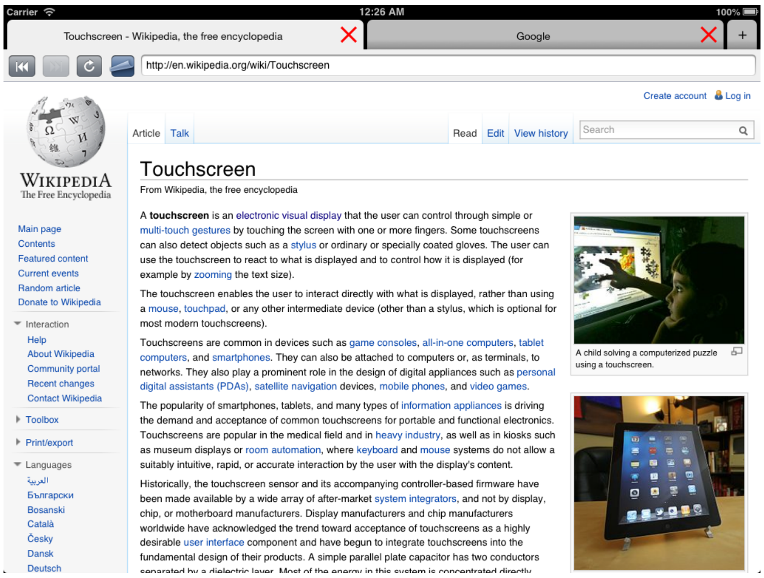
Historically, the touchscreen sensor and its accompanying controller-based firmware have been made available by a wide array of after-market system integrators, and not by display, chip, or motherboard manufacturers. Display manufacturers and chip manufacturers worldwide have acknowledged the trend toward acceptance of touchscreens as a highly desirable user interface component and have begun to integrate touchscreens into the fundamental design of their products. A simple parallel plate capacitor has two conductors separated by a dielectric layer. Most of the energy in this system is concentrated directly

A child solving a computerized puzzle using a touchscreen.

Figure 4.1: Safari - the default browser on the iPad

Every tab has its own history that can be accessed by tapping a back button or a forward button. These buttons are disabled when the respective functions are not available. A combined stop/reload button will allow stopping an ongoing page load, or trigger a reload otherwise. There's also a button to change study related settings. It is normally not used.

In addition, every tab features a URL bar — a text input field showing the current page's URL. When tapping the URL bar, a cursor appears and a keyboard interface is displayed in the bottom half of the screen (see figure 4.3). Ad-



Touchscreen

From Wikipedia, the free encyclopedia

A **touchscreen** is an electronic visual display that the user can control through simple or multi-touch gestures by touching the screen with one or more fingers. Some touchscreens can also detect objects such as a stylus or ordinary or specially coated gloves. The user can use the touchscreen to react to what is displayed and to control how it is displayed (for example by zooming the text size).

The touchscreen enables the user to interact directly with what is displayed, rather than using a mouse, touchpad, or any other intermediate device (other than a stylus, which is optional for most modern touchscreens).

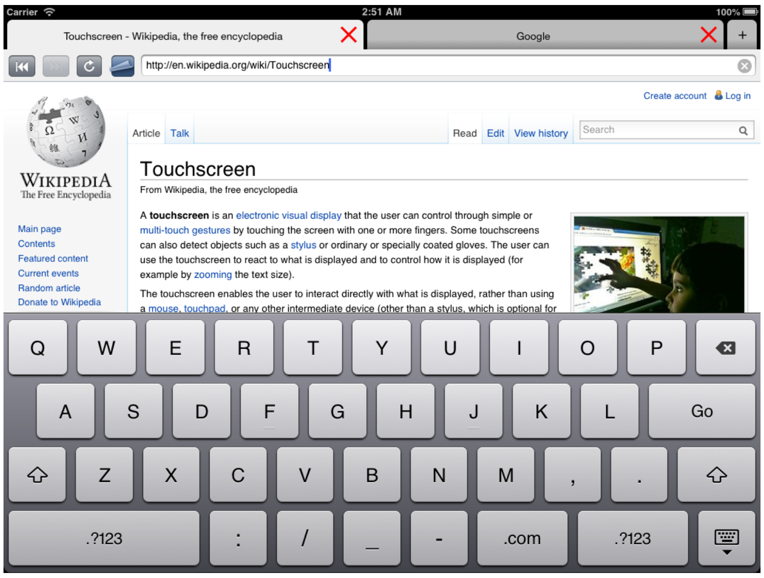
Touchscreens are common in devices such as game consoles, all-in-one computers, tablet computers, and smartphones. They can also be attached to computers or, as terminals, to networks. They also play a prominent role in the design of digital appliances such as personal digital assistants (PDAs), satellite navigation devices, mobile phones, and video games.

The popularity of smartphones, tablets, and many types of information appliances is driving the demand and acceptance of common touchscreens for portable and functional electronics. Touchscreens are popular in the medical field and in heavy industry, as well as in kiosks such as museum displays or room automation, where keyboard and mouse systems do not allow a suitably intuitive, rapid, or accurate interaction by the user with the display's content.

Historically, the touchscreen sensor and its accompanying controller-based firmware have been made available by a wide array of after-market system integrators, and not by display, chip, or motherboard manufacturers. Display manufacturers and chip manufacturers worldwide have acknowledged the trend toward acceptance of touchscreens as a highly desirable user interface component and have begun to integrate touchscreens into the fundamental design of their products. A simple parallel plate capacitor has two conductors separated by a dielectric layer. Most of the energy in this system is concentrated directly

A child solving a computerized puzzle using a touchscreen.

Figure 4.2: The browser's traditional interface



Touchscreen

From Wikipedia, the free encyclopedia

A **touchscreen** is an electronic visual display that the user can control through simple or multi-touch gestures by touching the screen with one or more fingers. Some touchscreens can also detect objects such as a stylus or ordinary or specially coated gloves. The user can use the touchscreen to react to what is displayed and to control how it is displayed (for example by zooming the text size).

The touchscreen enables the user to interact directly with what is displayed, rather than using a mouse, touchpad, or any other intermediate device (other than a stylus, which is optional for

Figure 4.3: Changing the URL using the browser's traditional interface

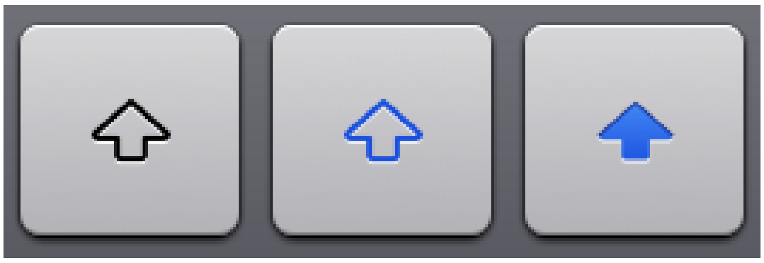


Figure 4.4: Shift states on the iPad: Shift Off, Shift for letters only, Full Shift (iPad native)

ditionally, if the URL bar is not empty, a tappable button to clear its contents is offered on its right side. The keyboard can be closed by tapping the button in its lower right corner or by tapping the “Go” button to confirm the URL. Alternatively, tapping outside of the keyboard area will also close the keyboard.

The keyboard has two basic layouts, one for entering letters and some punctuation marks, and another one for special characters. Both basic layouts feature a shift mode that changes some of the keys. The shift key can be tapped to act as a mode, or dragged on other characters to insert their uppercase variant without otherwise changing the keyboard state. The buttons to switch between the basic layouts work the same way. However, the letter key’s Shift key has a third state that only applies to letters, not to punctuation marks (see figure 4.4). For the studies, the automatic shift key mode for letters only when starting a sentence was deactivated. However, this mode made an appearance anyway: when pressing the shift key, typing some letters (the first being uppercase) and later deleting the typed characters again, the shift state is automatically set to “letters only” again. Caps Lock was deactivated for the studies.

There are some slight differences depending on the focused element. For URLs, the space bar is replaced with buttons to enter a colon, slash, underscore, dash or a top level domain like “.com”. For numeric input fields, the special character layout is shown immediately since it features digits. Also depending on the context, the confirmation button may have a different label (such as “Go” or “Return”).

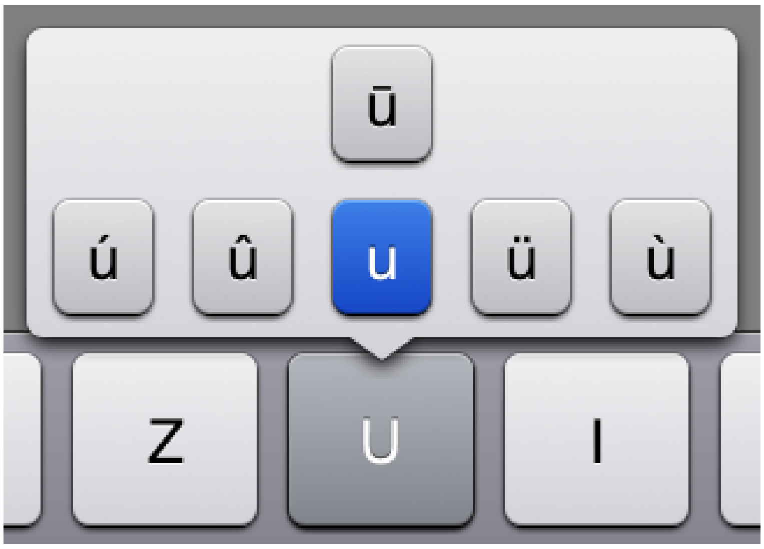


Figure 4.5: Long tapping a letter in iPad's native keyboard will open a pop over for accented characters

Screenshots of the different keyboard layouts can be found in appendix A "iPad keyboard layouts".

The characters on the keyboard are activated by tapping them. For some keys, touching them continuously will reveal more buttons with accented or otherwise related versions of the selected key (see figure 4.5). To choose such a version, the user has to slide a finger over the corresponding button and lift the finger off the screen. Some keys can also be inserted more quickly by sliding upwards (see figure 4.6).

The user can control the text cursor by touching and holding the screen at the intended location of the cursor. The cursor can be moved by moving the finger accordingly (see figure 4.7). When the URL bar is focused, tapping it again will open a popover menu (see figure 4.8) with options to select the word at the current cursor position or the whole content of the URL bar. A paste button will allow pasting the clipboard[1]'s contents. Once a part of the text is selected,

[1]The clipboard stores data in memory in order to facilitate copy & paste functionality. On Apple's operating systems it is usually called pasteboard.

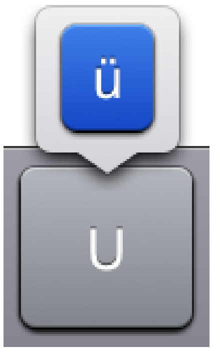


Figure 4.6: Sliding upwards over a key is a shortcut for some accented characters in certain iPad keyboard layouts

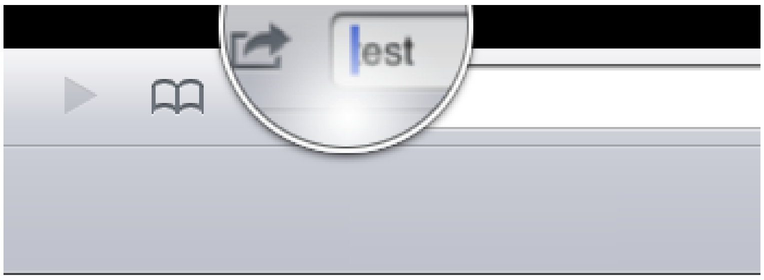


Figure 4.7: A lens appears above the touch point when moving the cursor

the popover menu changes to allow the clipboard operations "cut", "copy" and "paste", as well as a "define" option that would open a dictionary app (see figure 4.9). All these buttons are activated by tapping them. Furthermore, the bars that mark the ends of the selection are augmented by dots that can be held and dragged to adjust the selection range, similar to moving the cursor.

To scroll the page, the user can touch the content area of the screen, then move the finger to move the page accordingly. For zooming, the same gesture is performed with two fingers moving in opposite directions - outward to zoom out, inward to zoom in. Alternatively the user can double tap the page to change the zoom level — depending on the context, this will zoom in or out.

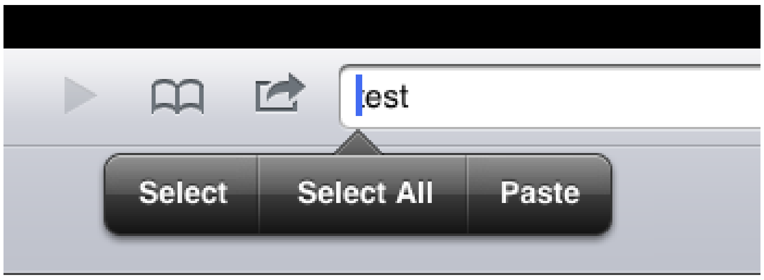


Figure 4.8: Cursor context menu without selected text

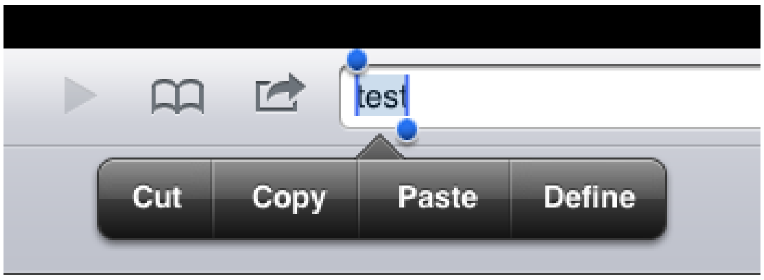


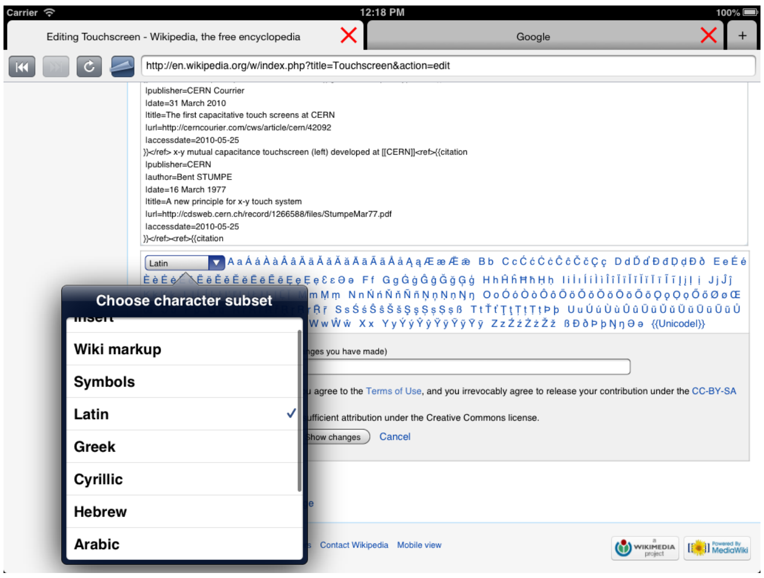
Figure 4.9: Cursor context menu with selected text

Simple tapping is used to activate links and buttons, focus text fields, toggle checkboxes, select radio buttons and open drop down lists. Doing the latter will display a list of items that the user can select by tapping them (see figure 4.10). If there are more items than fit on the screen, the list can be scrolled in the same way as the web page.

Bookmark support was only implemented in the web-based prototype to study the concept. The iPad version for the main study does not offer bookmark support, allowing more data to be collected regarding special character input.

4.3 Swabbing interface

Swabbing support is implemented via a translucent overlay that covers the entire application (see figure 4.11). When the overlay is visible, it captures all touch events. It can be shown or hidden by tapping the screen with five fingers.



```
|publisher=CERN Courrier
|date=31 March 2010
|title=The first capacitative touch screens at CERN
|url=http://cerncourier.com/cws/article/cern/42092
|accessdate=2010-05-25
}}</ref> x-y mutual capacitance touchscreen (left) developed at [[CERN]]<ref>{{citation
|publisher=CERN
|author=Bent STUMPE
|date=16 March 1977
|title=A new principle for x-y touch system
|url=http://cdsweb.cern.ch/record/1266588/files/StumpeMar77.pdf
|accessdate=2010-05-25
}}</ref><ref>{{citation
```

nges you have made)

u agree to the Terms of Use, and you irrevocably agree to release your contribution under the CC-BY-SA

ufficient attribution under the Creative Commons license.

Figure 4.10: Drop down list support in the browser's traditional interface

The gesture is recognized as soon as five fingers are touching the screen simultaneously. After recognizing the gesture, it is blocked for one second to prevent its effects from being undone by accidentally tapping the screen again, a problem common with users who have hand tremor.

Immediately after activation, the overlay will always show the main menu, allowing to control the browser. When submenus are opened, they are pushed onto a stack in memory, while only the top one is being shown. All submenus offer a cancel entry as the last option to remove them from the stack and return to the previous menu. However, some submenus replace the current one instead of being pushed on the stack.

The active menus can also change depending on events on the website: if a navigation event occurs, "content sensitive" menus are popped from the stack to keep the state intact. The three text menus and the aiming menu are considered content sensitive. The aiming menu will be closed and reopened every time the page changes within the visible area or when it is scrolled. For the text menus, the fo-

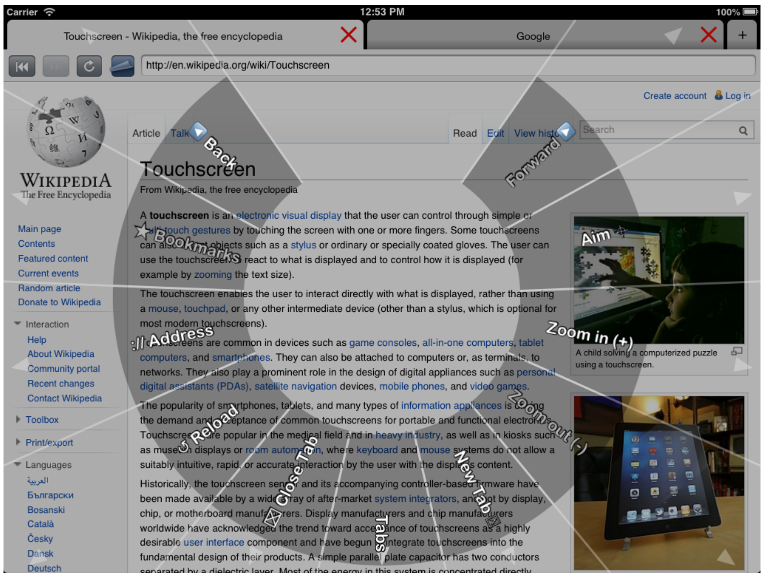
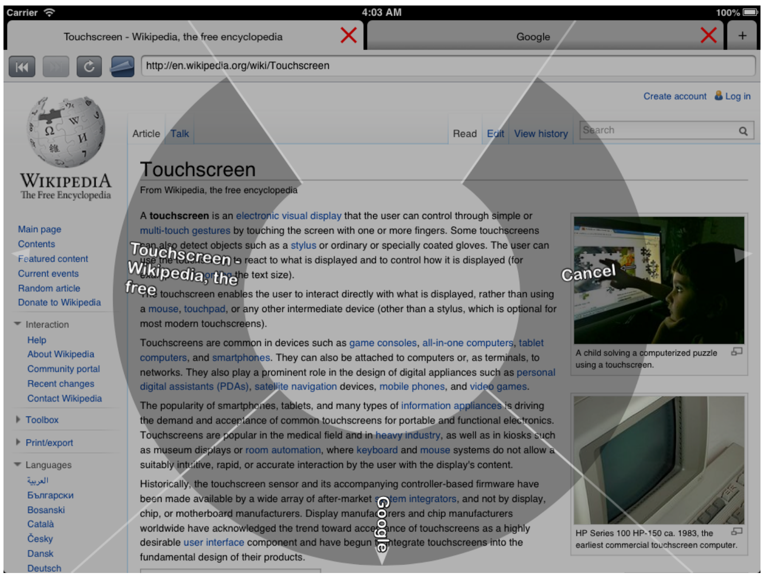

Touchscreen - Wikipedia, the free encyclopedia
Google
http://en.wikipedia.org/wiki/Touchscreen
Back
Forward
Bookmarks
Aim
Address
Zoom in (+)
Reload
Zoom out (-)
Close Tab
New Tab
Tabs

Figure 4.11: The main menu of the browser's Swabbing interface

cused element is monitored to keep the state in sync: if the text is changed by a script on the web site, the change is propagated to the Swabbing representation of the content. If the element disappears, the current text menu is removed from the stack.

The main menu features entries that directly correspond to the back and forward button, the stop/reload button, the close button of the current tab and the button to open a new tab. Two more entries allow zooming in and out, respectively. Furthermore, there's an entry to focus the URL bar, followed by a switch to the keyboard menu. Another entry will switch to a menu representing the currently open tabs. A deactivated entry to manage bookmarks was left in the main menu for consistency with the prototype. For scrolling, there is no Swabbing alternative as the usual panning gesture works well enough even for users with hand tremor.



Touchscreen

From Wikipedia, the free encyclopedia

A **touchscreen** is an electronic visual display that the user can control through simple or multi-touch gestures by touching the screen with one or more fingers. Some touchscreens can also detect objects such as a stylus or ordinary or specially coated gloves. The user can use the touchscreen to react to what is displayed and to control how it is displayed (for example, zooming the text size).

The touchscreen enables the user to interact directly with what is displayed, rather than using a mouse, touchpad, or any other intermediate device (other than a stylus, which is optional for most modern touchscreens).

Touchscreens are common in devices such as game consoles, all-in-one computers, tablet computers, and smartphones. They can also be attached to computers or, as terminals, to networks. They also play a prominent role in the design of digital appliances such as personal digital assistants (PDAs), satellite navigation devices, mobile phones, and video games.

The popularity of smartphones, tablets, and many types of information appliances is driving the demand and acceptance of common touchscreens for portable and functional electronics. Touchscreens are popular in the medical field and in heavy industry, as well as in kiosks such as museum displays or room automation, where keyboard and mouse systems do not allow a suitably intuitive, rapid, or accurate interaction by the user with the display's content.

Historically, the touchscreen sensor and its accompanying controller-based firmware have been made available by a wide array of after-market system integrators, and not by display, chip, or motherboard manufacturers. Display manufacturers and chip manufacturers worldwide have acknowledged the trend toward acceptance of touchscreens as a highly desirable user interface component and have begun to integrate touchscreens into the fundamental design of their products.

A child solving a computerized puzzle using a touchscreen.

HP Series 100 HP-150 ca. 1983, the earliest commercial touchscreen computer.

Figure 4.12: The tabs menu to switch between the currently open tabs

4.3.1 The text menus

There are three text menus that the user can switch between. Similar to the iPad, the default text menu allows entering normal letters (see figure 4.13), while the special character menu offers digits, punctuation marks and other special characters (see figures 4.14 and 4.15). The third text menu offers cursor and selection control, as well as access to the clipboard (see figure 4.16). The Shift mode has to be manually toggled by the user and is set to off when switching between text entry menus. In the letters menu, if a letter is typed, the shift mode is reset as well. Menu entries show the characters they would insert in the given mode, whereas the iPad always displays uppercase letters.

When URL fields are focused (such as the URL bar, or `<input type="url">` tags on websites), the space key's position is switched with that of the submit option (otherwise located in the special characters menu). For multiline text field, the submit option is replaced with an option to start a new line.

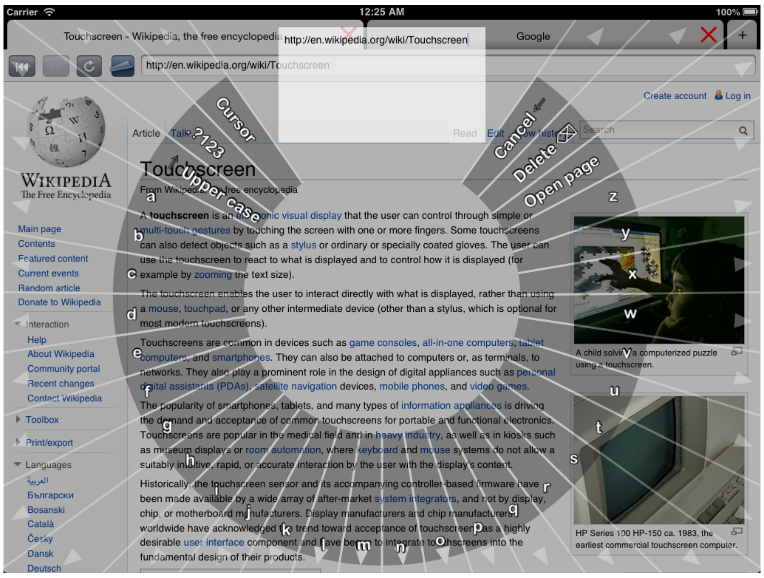


Figure 4.13: Text menu for entering letters with Swabbing

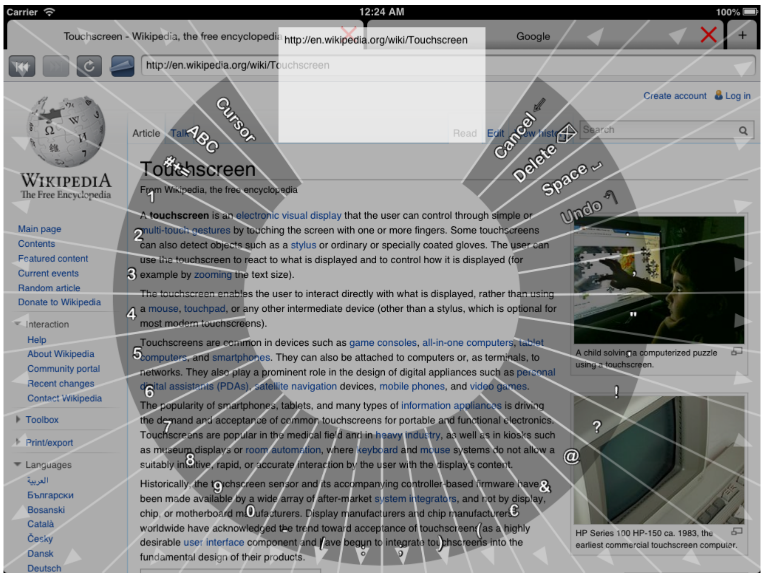


Figure 4.14: Text menu for entering special characters with Swabbing, shift off

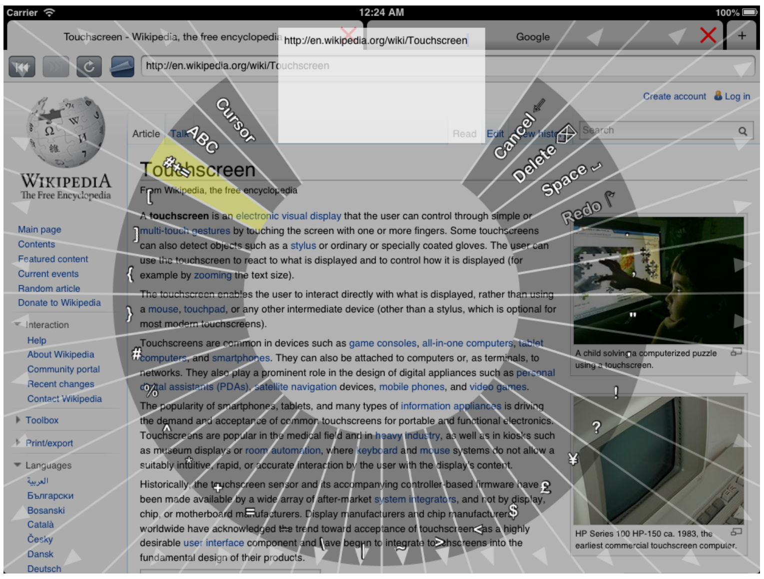


Figure 4.15: Text menu for entering special characters with Swabbing, shift on

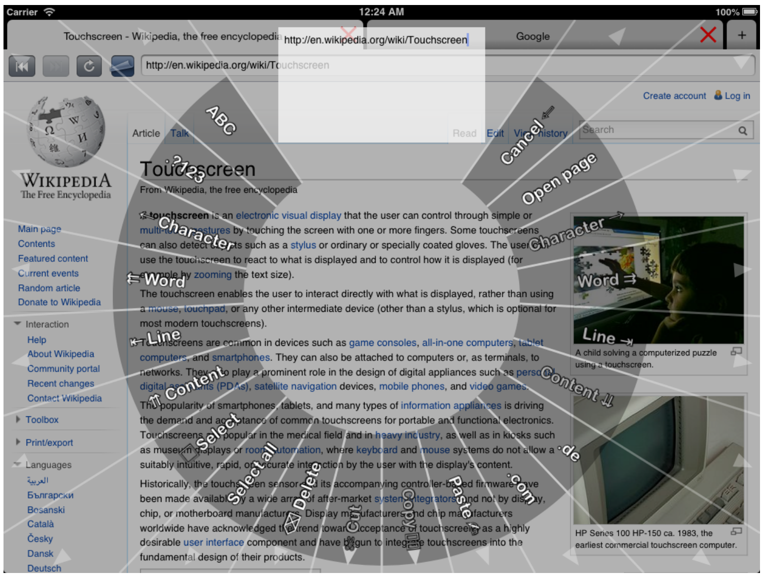


Figure 4.16: Text menu for controlling the cursor position, selection and accessing the clipboard

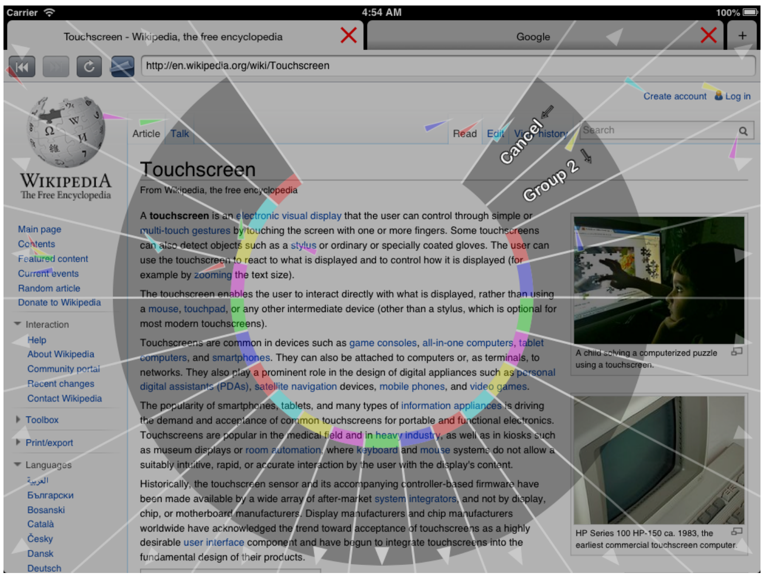
The text menus show the current value, cursor position and selection of the edited field in a small area in gap of the Swabbing menu. The text menus do not feature accented characters, including German umlauts. In most cases ä/ö/ü/ß can be replaced with ae/oe/ue/ss and other accented characters are rarely needed. However, how to properly support accented characters other than by simply adding more text menus remains an open question.

4.3.2 The cursor menu

The cursor menu allows moving the cursor to next/previous character, word, line or content boundary. If the select mode is active, moving the cursor selects text like the cursor keys on a desktop keyboard would when holding down the shift key. For convenience, the cursor menu also features the backspace entry and and entry to quickly select all text. Cut, Copy and Paste options provide access to the normal iPad clipboard. For consistency with the prototype, the cursor menu initially also featured deactivated entries for undo/redo. At the request of the user in the main study, those were later replaced with entries for the top level domains ".de" and ".com". Finally, the cursor menu features an entry to submit the current input, i.e. open the respective web page or submit the form the edited field is in.

4.3.3 The aiming menu

Finally, there is the aiming menu (see figure 4.17). It scans every eighth pixel of the visible area of the web page in reading direction for tapping targets like links, buttons and form fields. Hit testing is performed using the JavaScript function `document.elementFromPoint(x, y)`. The first position an element is found at is recorded and later used to display an arrow shaped target indicator pointing at this position. This approach was chosen over positioning the indicator in the middle of the found element to prevent problems with multiline links. If a link starts at the end of



Touchscreen

From Wikipedia, the free encyclopedia

A **touchscreen** is an electronic visual display that the user can control through simple or multi-touch gestures by touching the screen with one or more fingers. Some touchscreens can also detect objects such as a stylus or ordinary or specially coated gloves. The user can use the touchscreen to react to what is displayed and to control how it is displayed (for example by zooming the text size).

The touchscreen enables the user to interact directly with what is displayed, rather than using a mouse, touchpad, or any other intermediate device (other than a stylus, which is optional for most modern touchscreens).

Touchscreens are common in devices such as game consoles, all-in-one computers, tablet computers, and smartphones. They can also be attached to computers or, as terminals, to networks. They also play a prominent role in the design of digital appliances such as personal digital assistants (PDAs), satellite navigation devices, mobile phones, and video games.

The popularity of smartphones, tablets, and many types of information appliances is driving the demand and acceptance of common touchscreens for portable and functional electronics. Touchscreens are popular in the medical field and in heavy industry, as well as in kiosks such as museum displays or room automation, where keyboard and mouse systems do not allow a suitably intuitive, rapid, or accurate interaction by the user with the display's content.

Historically, the touchscreen sensor and its accompanying controller-based firmware have been made available by a wide array of after-market system integrators, and not by display, chip, or motherboard manufacturers. Display manufacturers and chip manufacturers worldwide have acknowledged the trend toward acceptance of touchscreens as a highly desirable user interface component and have begun to integrate touchscreens into the fundamental design of their products.

Figure 4.17: The aiming menu listing tapping targets on a web page

one line of text and ends in the next, the bounding box returned by `node.getBoundingClientRect` wraps both lines of text entirely. Positioning the arrow in the middle of that bounding box would make it very hard to find. A click at that position would also usually not hit the link.

All elements using the following HTML tags are included: `<area>`, `<button>`, `<select>`, `<textarea>`. In addition, `<input>` tags are included they are not of the type “file” since the iPad does not support file fields. `<a>` tags are included if they contain an “href” attribute or have direct handlers for the “click” or “mousedown” events. A direct click handler means that the element’s `onclick` property is set. Sadly, event handlers register via `element.addEventListener()` cannot be listed programmatically, a current technical implementation. It is possible that other operating systems than iOS offer more elaborate APIs that include access to those event handlers as well.

To support Google Maps, `<div>` elements of class “gsq_a”, “mb-icon-btn” or “mmh-btn” are also listed. Finally, every element's "cursor" property is inspected. If the value is "pointer", the element is assumed to be clickable - this value is used on systems with a mouse to indicate that the element under the cursor is clickable by changing the mouse cursor to a pointing hand. If a web site's designer added this visual hint, then likely because some click related event handlers were registered as well. Sometimes this heuristic fails — for instance, the iPad version of Google sets the pointer cursor for the `<html>` root object, thus affecting all elements on the page. In the iPad version of the browser, this curious choice is detected and the cursor value changed to "auto".

If more than twenty tapping targets are found, they are grouped to prevent the menu from becoming unusable. For even distribution, the number of chunks is calculated using $c = \lceil \frac{|T|}{20} \rceil$, with T being the set of tapping targets. The number of items per group is calculated using $n = \lceil \frac{|T|}{c} \rceil$.

To indicate which menu item belongs to which link, the target indicator that points at the link is shaped like the continuation of the option's pie segment towards the center. Therefore the indicator's angle is the same as that of its menu item. With many options, it can be hard to tell neighboring items apart when using the orientation only. To compensate, the target indicators and corresponding menu items are colored the same way. The order of colors is chosen so that neighboring colors are easy to tell apart even for colorblind users. However, only one of the participants in the user study was colorblind, so more testing is required. Please refer to chapter 5 "User study 1: Link assocation" for more details.

The big advantage of using these target indicators instead of representing the targets via their labels is that the user can find the appropriate menu entry without having to look at all options in sequence. The shape and color of the indicator basically provide an index to the menu. In addition, screen clutter is reduced since the options to not need any labels, making the web page easier to see through the overlay.

Since the user did not use any dropdown lists during the sessions with the traditional interface, Swabbing support for dropdown lists was not ported over from the prototype. All elements that the aiming menu shows behave as when tapped, except that when text fields are focused, text entry is performed using Swabbing.

4.4 The web-based prototype

Developed as a web application, the prototype runs in modern web browsers on the desktop or as a full screen web app on the iPad. It is written in HTML 5, SVG and JavaScript, using jQuery for the interaction and Raphael.js to generate the SVG based menus.

While the iPad browser offers Swabbing as an enhancement, the prototype was developed with the intent of providing two entirely distinct browsers. The screenshots therefore only show tabs and a menu bar since no buttons were needed.

4.4.1 First iteration

The first iteration didn't provide any functionality. It could only render different menu layouts to speed up the process of designing them. There was no concept yet for transitioning between menus. The visual design was based on Huck [2012], including the clockwise ordering of menu items. Labels were only rotated if they didn't fit into the space between the corridor lines, sometimes producing inconsistent results.

This version of the prototype had a main menu very similar to the final design (see figure 4.18), and a home menu with fewer options for the start page. The main menu allowed directly toggling the bookmark for the current page, and it featured two aiming menus — one for links and one for form elements. There were two different text entry menus, one for normal text and one for URLs. They had the same options as the corresponding native iPad keyboard layouts

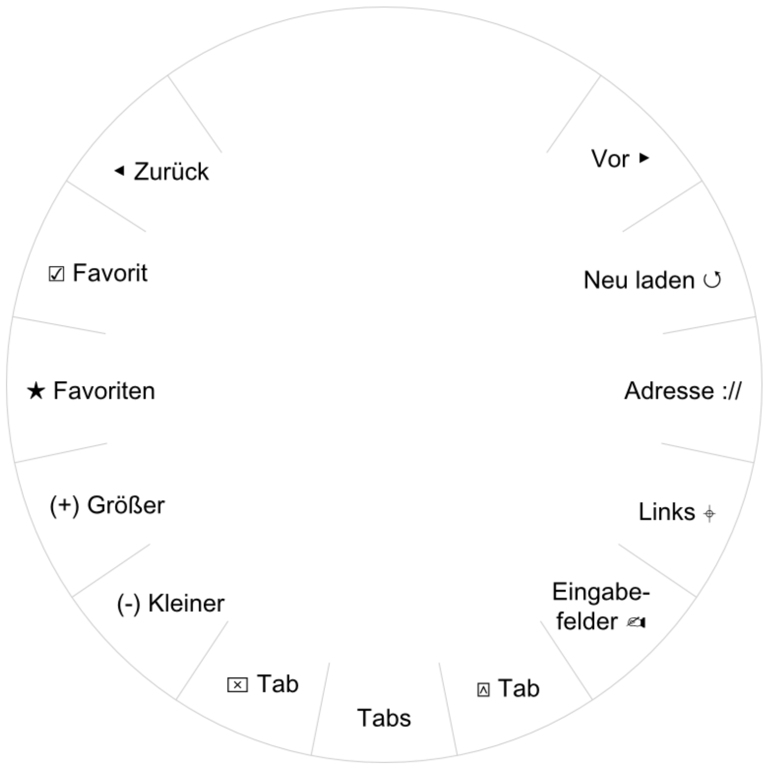


Figure 4.18: Page menu, rendered by the first prototype

for maximum comparability of the native browser and the Swabbing based one. There was no design yet for the tabs, bookmarks and aiming menus.

4.4.2 Second iteration

Some basic interaction was introduced: Swabbing could now be turned on and off by tapping the screen with three fingers or clicking the middle mouse button. While the original idea was to use five fingers for the gesture, the three finger version was quicker to implement since the chances of detecting all fingers at once are higher with fewer fingers. In submenus, the last option was always used to return to the parent menu. For consistency, all labels were now rotated.

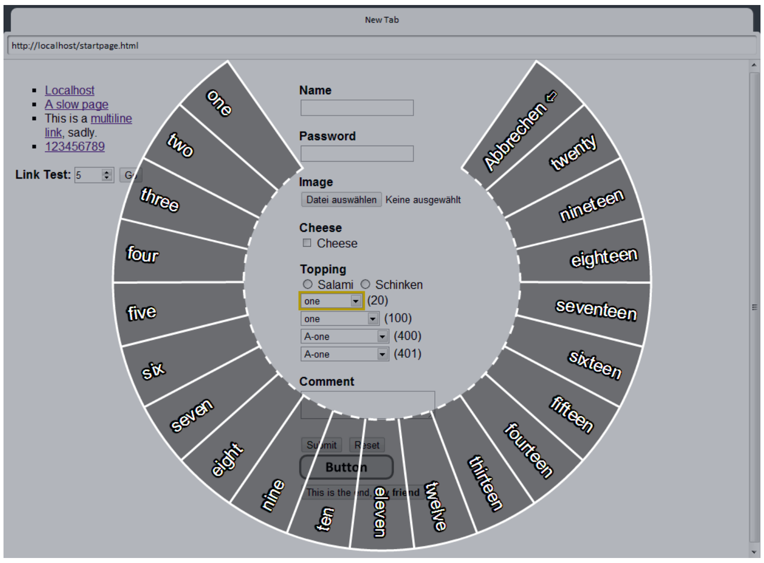


Figure 4.19: Support for drop down menus

In addition to the continuous outline of the circle, a dashed inner circle was added with the same distance from the center as the gesture detection threshold (4.19). The idea was to inspire users to move their fingers at least that far. To improve visibility against the noisy background of web pages, the lines were made thicker, the labels were colored white with a black outline and the background of the option's segments were shaded with a transparent black background.

The tabs menu was introduced, revealing a problem with the clockwise ordering of items. To improve the spatial mappings of tabs to their items, a counter-clockwise order was used instead. To increase consistency and ease of learning, the special home menu was removed, instead showing the page menu on the start page, too.

The number of options on the main text menu was reduced by removing all characters but letters and the space character, favoring reliability of text entry over speed and similarity to native iPad keyboard. To increase ease of learning, the special URL menu was discarded as well, leaving one stable layout for all text input tasks. The menu for letters and the menu for special characters now had the same number of options, allowing for a smooth transition between them.

A first design for the aiming menu was introduced. The markers for targets were scaled versions of the corresponding pie segments. This allowed a direct mapping of indicators to menu options by simply matching their shape. Different colors were used to disambiguate items with similar shapes. Unfortunately it was difficult to tell which element a marker belonged to, especially when the page contained fewer links, resulting in bigger markers. In addition, the markers were positioned in the center of the target element. Since links that contained line breaks were detected as one target, the marker would appear in the middle of the screen, far removed from both ends of the link.

A revised design for the aiming menu used different markers. Instead of using a scaled version of the pie segment, its continuation towards the center of the menu was used, resulting in an arrow shape. The pointy end of the arrow now clearly indicated the corresponding target. It was now positioned to point at the first scanned pixel belonging to the target element, solving the problem with multi-line links. Since the shape of the arrow indicated the position of the corresponding menu item, rapid association was still possible, as the first user study proved.

4.4.3 Third iteration

When not starting the movement off the center of the screen and aiming for the pie segment instead of the screen edge, the user can end up selecting the wrong item. To make it clearer that the target of the gesture is a segment of the screen edge, the corridor lines between the items were extended to the edge of the screen and the inner and outer border lines of the circle were removed. In addition, an arrow was shown in the middle of an item's screen edge to provide a visual clue about where to aim in order to select a certain item.

To limit screen clutter and web page occlusion the corridor lines were made transparent and the thickness of the label outline was reduced. To make the colors of the arrows in the aiming menu easier to recognize when they overlap with a menu pie segment, only inner 10% of the pie segment were colored.

The arrows were enhanced with a shaded background resembling the menu, enabling people to discover the significance of the arrow's orientation on their own (see User study: Link Association). In a second step, the size of that background was lowered to reduce overlap with other backgrounds. The arrow's size was not changed to keep the colors easy to recognize. The color sequence was changed to make green and turquoise harder to mix up.

Lacking context from neighboring characters, some special characters were hard to identify when rotated, so their rotation was removed.

The text menu layouts are slightly different depending on the text field that is being edited. For text fields on a web page, the menu with letters offers the space character, but for the URL bar an "Open Page" option is shown instead. Consequently, the special characters menu shows a space character option for the URL bar, an option to insert a new line for multi-line text fields, and an option to submit the field's form for single-line text fields, mimicking the effect of the Enter key in desktop browsers.

Since Google's auto-complete suggestions were not detected as targets by the aiming menu, adding a menu that allows navigation like with a desktop keyboard was considered (see figure 4.20). The user could have simulated pressing the "Up" or "Down" cursor keys to choose one of the options, and pressing "Enter" to select it. In addition, the menu would have offered the "Left" and "Right" cursor keys, the tab keys to switch between form fields, "Home" and "End" buttons to scroll to the top/bottom part of the page and "Page up/down" buttons to scroll up or down by the screen's height. This idea was quickly discarded since it would venture beyond what the native browser offered. Instead, the markup used for Google's auto-complete options was added as a special case to the detection routine.

The direction of the vertical cursor gestures was changed to move diagonally towards the top left/bottom right cor-

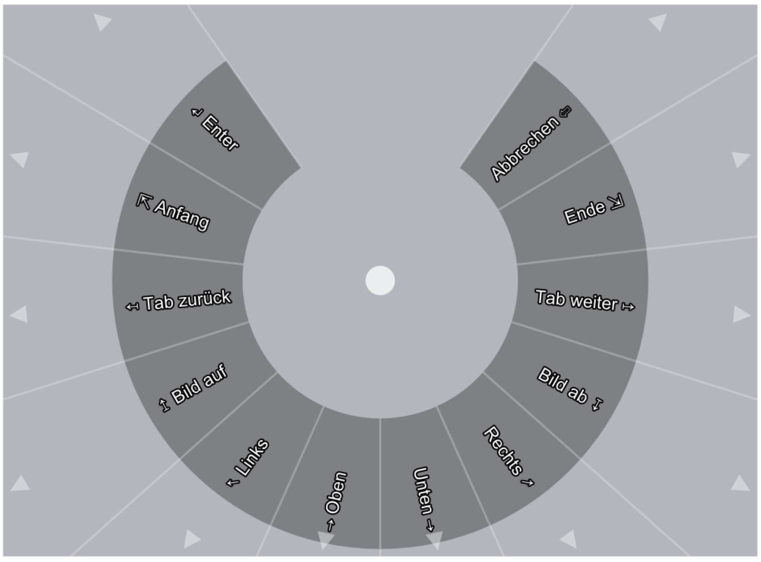


Figure 4.20: Discarded idea for a menu replicating navigation features provided by desktop keyboards

ner. Otherwise the user might expect that the gestures behave like the "Up" and "Down" cursor keys on a normal keyboard, selecting the closest character in the line above/below.

4.4.4 Fourth iteration

The tapping gesture to turn the menu on and off was changed to require five fingers instead of three. This allows distinguishing it from the cursor gestures by simply counting the number of touches rather than analyzing the movement of the touches, possibly induced by tremor only. In addition, if the user accidentally touches the surface with four instead of three fingers, no action is triggered instead of hiding the menu.

A menu for cursor control was introduced as an alternative to the cursor gestures (4.21). It adds an explicit mode for selection rather than subtly using the "Upper case" menu entry for that purpose. It also features access to the clipboard, similar to the iPad's selection menu, and offers undo/redo

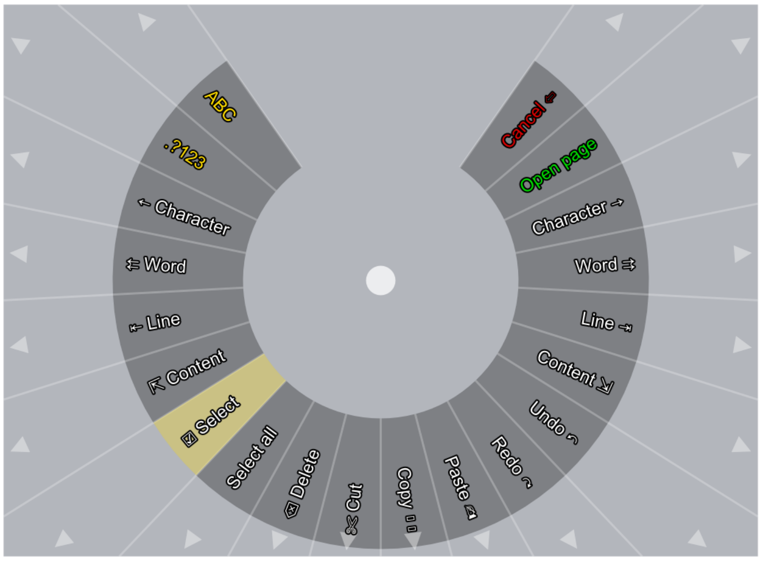


Figure 4.21: Cursor menu with semantically colored labels

functionality. It was placed on the same level as the menus for letters and special characters. All text menus have entries to switch to the other ones.

Since some sites style links to look like buttons (see figure 4.22), providing separate aiming menus for links and form entries turned out to be problematic. Hence they were merged into one general "Aim" entry. In addition, the add/delete bookmark entry was integrated into the bookmarks menu and removed from the main menu. Afterwards, the options on the main menu were rearranged.

The special nature of some options was no indicated by coloring their labels: "Cancel" is red, "Submit" and "Open page" are green, and the options to switch between the three text menus ("Cursor", "ABC" and ".?123") are yellow. In addition, active modes ("Upper case", "Select", "Open in new Tab") are highlighted with a yellow background.

As the screenshots indicate, this was the first version that was also available in English.

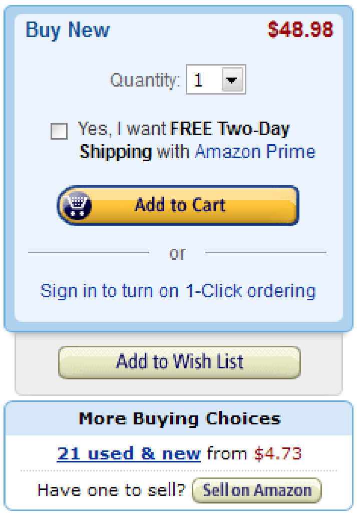


Figure 4.22: Three buttons on Amazon.com, one of which is a link.

Chapter 5

User study 1: Link assocation

A vital piece of functionality in any web browser is link selection in order to navigate from one page to another. On touch screens, links are usually tapped. When links are very small or close to other links, hitting the right one can be challenging, especially for tremor sufferers. Therefore, the Swabbing interface needed to provide a way to activate links.

From the beginning, the idea was to scan the visible area of the page for tappable targets. Originally, the plan was to re-arrange the website itself by pushing targets to the edge of the screen directly. Due to incomplete semantics of websites this approach is problematic. Sometimes the labels of links only make sense in the sentence they are embedded in, e.g., "Find out more here" — merely moving the word "here" elsewhere would remove any clues about the nature of the link. Similarly, the labels describing the purpose of form fields may not use the semantic elements HTML provides to describe their relationship, possibly resulting in unlabeled text fields occupying the screen edge. This would leave the user clueless about what to enter.

Instead, the decision was made to leave the layout of the page intact and provide a different way of associating tapping targets with menu items. Initial ideas included straight or curved lines connecting the targets with their menu items, or even providing a path to follow from the link to the center of the menu, and then to the target. Since the user might want to touch the screen before reaching the center, the change of directions in the gesture path would need to be detected, which is made difficult by the user's tremor. There were also concerns that tracing a visible line might be stressful for tremor sufferers, increasing the intensity of their tremor.

The next idea, then, was to mark the targets with a colored outline such that the color of the outline would match the color of the menu entry. By always using the same sequence of colors in the menu, the user might even learn to associate different colors with certain angles, removing the need to search the menu items in sequence until the right one is found. However, the higher the number of targets, the harder it is to tell individual colors apart. In addition, the outline around multi-line links would span the complete width of the paragraph they are in, possibly overlapping the outlines of other links.

Intrigued by the idea of removing the need for scanning the entire menu in order to locate the wanted item, the next approach was to exploit the unique appearance of each menu item provided by its orientation. Each target would be augmented with a scaled version of its menu item's pie segment. The user would instantly have an idea of the menu item's location, and could simply match the shapes to fully identify it. However, with many targets to choose from, neighboring pie segments would look too similar to be told apart merely by the shape. To allow the user to disambiguate similar items, the idea now was to also color them. While the colors might eventually repeat, neighboring colors should be as dissimilar as possible for maximum effect.

A quick informal study showed that this approach was promising, although it required an explanation. There also was a problem with using the pie segments as indicators. The fewer links there are on the screen, the bigger the (scaled) pie segments would be. For very small clustered targets, like links to page numbers, the indicators would either overlap or become too small to recognize their shape.

Since the pie segments do not have a clearly defined start, middle or end point, this would make it hard to tell which target the indicator actually belonged to.

Instead of using the pie segment itself, its imagined continuation towards the center of the menu was used. This shape looked like an arrow, providing a pointy and a flat end. The pointy end could then be used to clearly mark the indicator's target as well as represent the center point of the menu, while the angle towards the flat end would provide the user with a clue about the corresponding item's location. The user would simply move from the center in the same direction.

To find out how well users understand the concept, the first user study was conducted.

5.1 Study Design

Informal pretests indicated that users tend to notice the matching colors of target indicators and their menu items, but not necessarily the shared orientation. A between-group design was used to find out whether users would form a different mental model if immediately exposed to an aiming menu with duplicate colors or not. This is tested using **independent variable 1:** the order of target counts. Two levels were used — **1A:** 5, then 20 targets and **1B:** 20, then 5 targets. Since the color palette was limited to six colors, the aiming menus for five targets contained no duplicate colors, in contrast to the aiming menus for twenty targets.

In response to the results from testing this design with four users, the target indicators where augmented with a shaded background in the shape of the menu's silhouette (see figure 5.1). This resulted in **independent variable 2:** the visualization type. Two levels were used, namely **2A:** arrow only and **2B:** arrow + menu silhouette.

The **dependent variable** was the mental model the user reported.

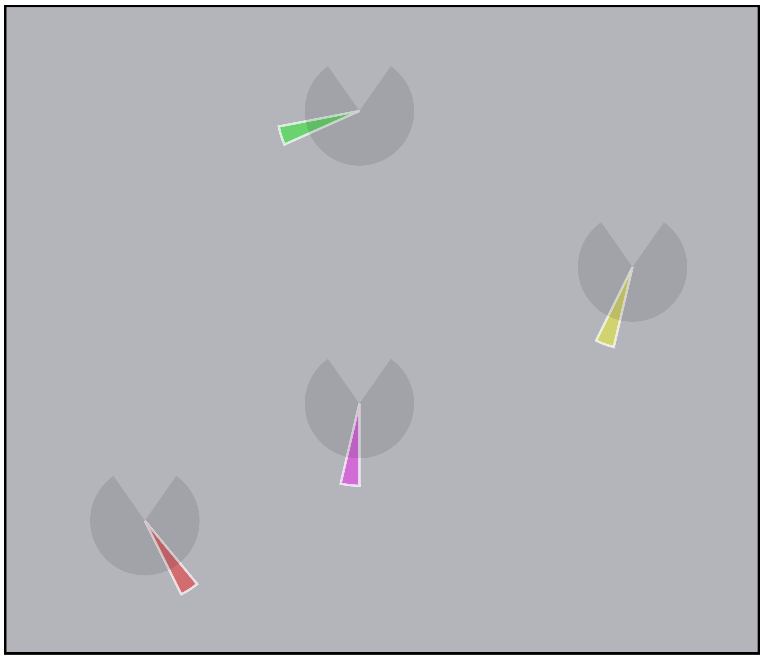


Figure 5.1: Target indicators with menu silhouette

5.2 Participants

All participants were male and had extensive experience with computers.

1A x 2A Two subjects (both age 28)

1B x 2A Two subjects (age 24 and 25)

1A x 2B Two subjects (age 21 and 30)

1B x 2B One subjects (age 31, red/green weakness)

5.3 Measurements

The participants' **mental model** about the aiming menu and its purpose were measured directly through a questionnaire and tested indirectly by asking them to select a predefined target using the Swabbing menu, followed by explaining their choice.

5.4 Procedure

See section C.1 "User study 1: Link association" for material used to perform this study.

1. The instructor explains that the browser is designed for users with hand tremors, and the basic idea behind Swabbing
2. The user uses the system freely for three minutes while the aiming menu is disabled
3. The instructor explains the purpose of the study — to test the aiming menu
4. The instructor opens a sample page with five (1A) or twenty (1B) targets
5. The instructor opens the aiming menu and selects a target for the user
6. The user explains their mental model about the menu and how to select the target using it
7. The instructor opens a sample page with twenty (1A) or five (1B) targets
8. The instructor opens the aiming menu and selects a target for the user
9. The user explains their mental model about the menu and how to select the target using it
10. The instructor opens an article with twenty targets
11. The instructor opens the aiming menu
12. The user announces two targets they want to activate and performs the selection, followed by explaining their choice.

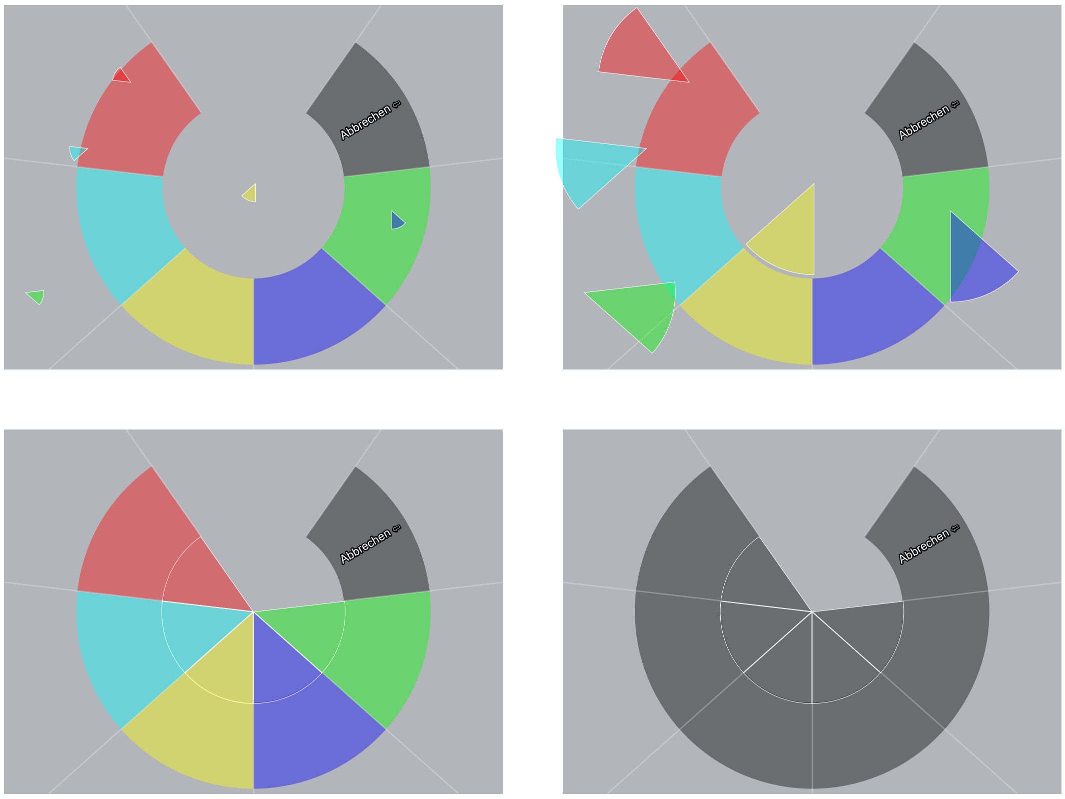


Figure 5.2: Screenshots used to demonstrate the connection between target indicators and menu options

13. The instructor explains how the aiming menu works and the role of the target indicator's color and orientation using a series of screenshots (see figure 5.2)

14. The user explains their mental model about the menu

15. The instructor opens an article with five targets and explains the role of the colors

16. The instructor opens an article with twenty targets and explains the role of the target indicator's orientation

17. The user announces four links activates four links using the menu after announcing the intended target and followed by explaining their choice

5.5 Results

All of the users understood the purpose of the aiming menu after the introduction — to allow users with hand tremors to activate links in web pages. All of the users were able to correctly identify which targets and target indicators belong together. All of the users understood that the color of the menu option is the same as the color of the corresponding target indicator.

However, without the menu silhouette, users only included the arrow's orientation in their decision process after the explanation was given. The three users who were shown the silhouettes were able to use the arrow's orientation before an explanation. One of those users only made the connection after noticing the color ambiguity when being shown twenty targets, but came to the right conclusion without an explanation.

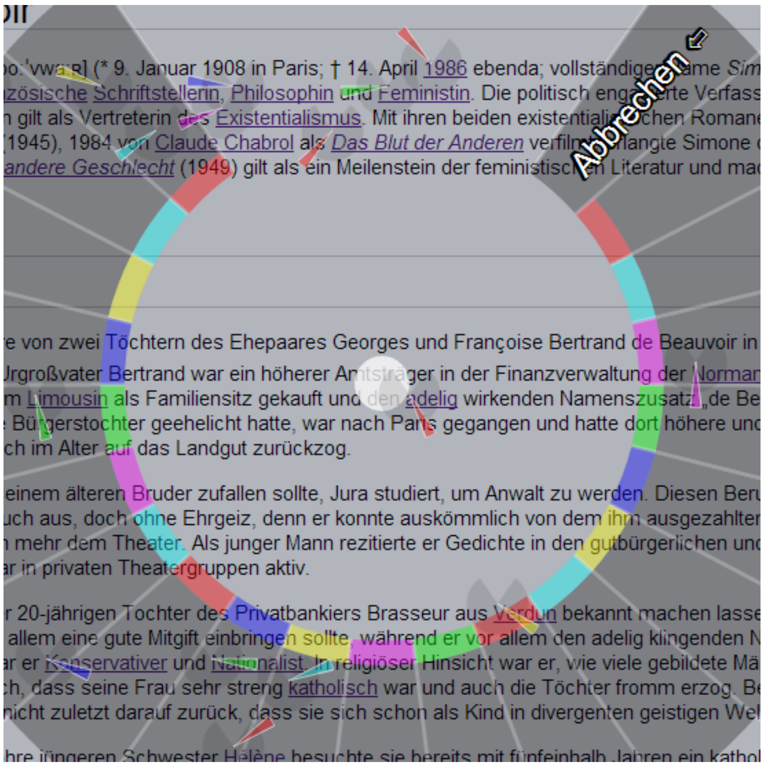
Without the menu silhouette (condition 2A), three of the four users noticed that the order of arrow colors in reading direction corresponded to the order of menu option colors. However, one user was still not sure whether the menu would start on the left or the right side, since both the first and the last item were red.

In most cases, the colors were sufficient to disambiguate neighboring options. However, green and turquoise were occasionally mixed up. In addition, the user with colorblindness had trouble telling magenta and turquoise apart.

Finally, users were not sure whether the menu only shows targets that are currently visible, or would also show targets that can be scrolled to.

5.6 Discussion

The results indicate that most users would be able to use the aiming menu correctly after getting an explanation. Visual clues like the menu silhouette increase the likelihood



ɔII

o:ˈvwaːʁ] (* 9. Januar 1908 in Paris; † 14. April 1986 ebenda; vollständige ame Sim
zösische Schriftstellerin, Philosophin und Feministin. Die politisch eng rte Verfass
n gilt als Vertreterin des Existentialismus. Mit ihren beiden existentiali chen Romane
(1945), 1984 von Claude Chabrol als *Das Blut der Anderen* verfilmt angte Simone
andere Geschlecht (1949) gilt als ein Meilenstein der feministischen Literatur und ma

e von zwei Töchtern des Ehepaares Georges und Françoise Bertrand de Beauvoir in
Jrgroßvater Bertrand war ein höherer Amtsträger in der Finanzverwaltung der Norman
m Limousin als Familiensitz gekauft und den adelig wirkenden Namenszusatz „de Be
Bürgerstochter geehelicht hatte, war nach Paris gegangen und hatte dort höhere und
ch im Alter auf das Landgut zurückzog.

einem älteren Bruder zufallen sollte, Jura studiert, um Anwalt zu werden. Diesen Beru
uch aus, doch ohne Ehrgeiz, denn er konnte auskömmlich von dem ihm ausgezahlten
n mehr dem Theater. Als junger Mann rezitierte er Gedichte in den gutbürgerlichen und
ar in privaten Theatergruppen aktiv.

r 20-jährigen Tochter des Privatbankiers Brasseur aus Verdun bekannt machen lasse
allem eine gute Mitgift einbringen sollte, während er vor allem den adelig klingenden N
ar er Konservativer und Nationalist. In religiöser Hinsicht war er, wie viele gebildete Mä
ch, dass seine Frau sehr streng katholisch war und auch die Töchter fromm erzog. Be
nicht zuletzt darauf zurück, dass sie sich schon als Kind in divergenten geistigen Wel

hre jüngeren Schwester Hélène besuchte sie bereits mit fünfeinhalb Jahren ein kathol

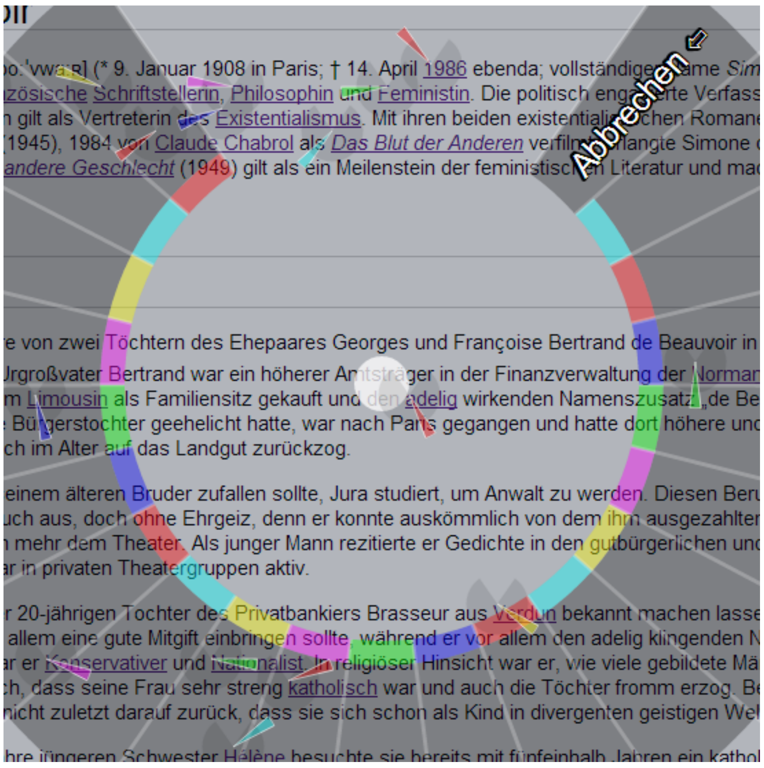
Figure 5.3: Initial color sequence

of understanding the aiming menu correctly even without an explanation. This would be especially beneficial when employing Swabbing in public settings.

Users who note that the target indicators and the menu items are both sorted by the order of targets in reading direction were able to use the menu without discovering the significance of the indicator's orientation. However, their selection time will be lower since they need to resort to looking at the options and indicators in sequence.

To address the color issues, the color sequence was changed. See figures 5.3 and 5.4 for a comparison. Note that a bug in the color selection code made the initial color sequence somewhat irregular. Also, the new color sequence has not been tested again.

Regarding the confusion about the scope of the aiming



o:'vwaːʁ] (* 9. Januar 1908 in Paris; † 14. April 1986 ebenda; vollständiger Name *Sim*
zösische Schriftstellerin, Philosophin und Feministin. Die politisch engagierte Verfass
n gilt als Vertreterin des Existentialismus. Mit ihren beiden existentialistischen Romane
(1945), 1984 von Claude Chabrol als *Das Blut der Anderen* verfilmt, erlangte Simone
andere Geschlecht (1949) gilt als ein Meilenstein der feministischen Literatur und mac

e von zwei Töchtern des Ehepaares Georges und Françoise Bertrand de Beauvoir in
Urgroßvater Bertrand war ein höherer Amtsträger in der Finanzverwaltung der Norman
m Limousin als Familiensitz gekauft und den adelig wirkenden Namenszusatz „de Be
Bürgerstochter geehelicht hatte, war nach Paris gegangen und hatte dort höhere und
ch im Alter auf das Landgut zurückzog.

einem älteren Bruder zufallen sollte, Jura studiert, um Anwalt zu werden. Diesen Beru
uch aus, doch ohne Ehrgeiz, denn er konnte auskömmlich von dem ihm ausgezahlten
mehr dem Theater. Als junger Mann rezitierte er Gedichte in den gutbürgerlichen und
ar in privaten Theatergruppen aktiv.

r 20-jährigen Tochter des Privatbankiers Brasseur aus Verdun bekannt machen lasse
allem eine gute Mitgift einbringen sollte, während er vor allem den adelig klingenden N
ar er Konservativer und Nationalist. In religiöser Hinsicht war er, wie viele gebildete Mä
ch, dass seine Frau sehr streng katholisch war und auch die Töchter fromm erzog. Be
nicht zuletzt darauf zurück, dass sie sich schon als Kind in divergenten geistigen We
hre jüngeren Schwester Hélène besuchte sie bereits mit fünfeinhalb Jahren ein kathol

Figure 5.4: Improved color sequence

menu, it is possible that visualizing the line-by-line scanning process would make it more clear that only the currently visible targets are included in the menu.

5.7 Limitations

Since only three users were shown the menu silhouette, it remains unclear how effective this approach really is. While it seems likely that younger users with extensive computer experience will be able to use the menu after receiving an explanation, further tests should be performed with older users and users lacking computer experience.

The positive result with the colorblind user is encouraging, but since there are many variants of colorblindness, more tests are needed here as well.

Since the sexes score differently in cognitive spatial tests (see Dabbs et al. [1998]), the study should be repeated with female subjects.

Finally, the grouping functionality for more than twenty targets was not tested in this study. The main study indicates that more tests are advisable: When the user first encountered a situation in which the link to select was in the second group, he needed over three minutes to select it while usually he needed less than 10s.

Chapter 6

User study 2: Swabbing feedback

After first assuming that feedback for successful selections would be distracting, Hurtmanns [2011] later implemented a short green highlight for successful activations, mentioning a user who "waited for a long time, because she was wondering, whether or not her selection was correct".

The only feedback provided by the Swabbing implementation Huck [2012] used was the inserted character — for that purpose, trailing space characters were even replaced with a half open box to make them visible. However, in his future work section, he clearly recommends implementing more feedback mechanisms.

The feedback mechanisms above constitute open loop designs — the feedback occurs after the user has performed their selection. For this thesis, a closed loop feedback design was implemented. As soon as the user crosses the threshold distance, the detected target is highlighted, as well as the exact point the user is aiming at. This allows the user to detect and correct an incorrect selection before it is accepted by the system.

To find out whether this type of feedback is useful, a second user study was conducted. In addition, this study addressed a concern that users might aim for the labels on the circle instead of the actual targets on the edge of the screen. If the starting point is anywhere on the line between the center point and the target arrow, this would not make a difference. But when starting off center — as can easily happen due to the user's tremor — aiming towards a label can mean actually aiming to a different menu option.

However, initially there was no target arrow and the corridor lines did only extend to the outer edge of the circle, leaving no visual clue at all about the actual location of the targets the user should aim at. While the feedback mechanisms might eventually help users to adjust their aim, ideally the visual design inspires them to do aim correctly right away.

The intention of the corridor lines was to make the vector impact zones visible, while the target arrow should serve as a visible target to aim at. Before introducing the feedback mechanisms, the effect of the design changes alone was tested.

6.1 Study Design

To be able to react spontaneously to feedback provided by the user, a single-subject design was chosen. The **independent variable** was the combination of design and feedback shown to the user. The levels were

1. Baseline: no graphic elements beyond the edge of the circle, no feedback
2. Corridor lines and target arrows
3. Corridor lines and target arrows + highlighting the background of the selected option's pie segment
4. Corridor lines and target arrows + revealing the target point at the edge of the screen
5. Combination of 4 and 5

The **dependent variables** were the apparent target of the user's gestures, and the correctness of the selection.

6.2 Participant

The participant was a 60 year old woman with an essential tremor since age 14. The tremor first affected the right hand, and later the left hand as well. Her tremor gets more intense under stress and sometimes also affects her head, mouth and legs. She did not use deep brain stimulation - a brain implant that can greatly reduce tremor symptoms. She wears reading glasses. She was also afflicted by a cataract, but had gotten an operation.

Most remarkably, the user has no prior experience with computers. While she has a mobile phone (with normal key size), she uses it for phone calls only. When she needs information off the internet, she asks her husband.

6.3 Measurements

Before and after the session, her tremor strength was assessed via spiralometry. During the session, the touch coordinates and some screenshots were recorded. Sadly, the screenshot recording code was in its early stages and didn't capture the feedback mechanisms in action.

6.4 Procedures

For all levels of the independent variable, the user was told to select ten options starting at the center point and ten options starting anywhere else. In between, she was routinely asked for her opinion of the software and given explanations when it seemed appropriate.

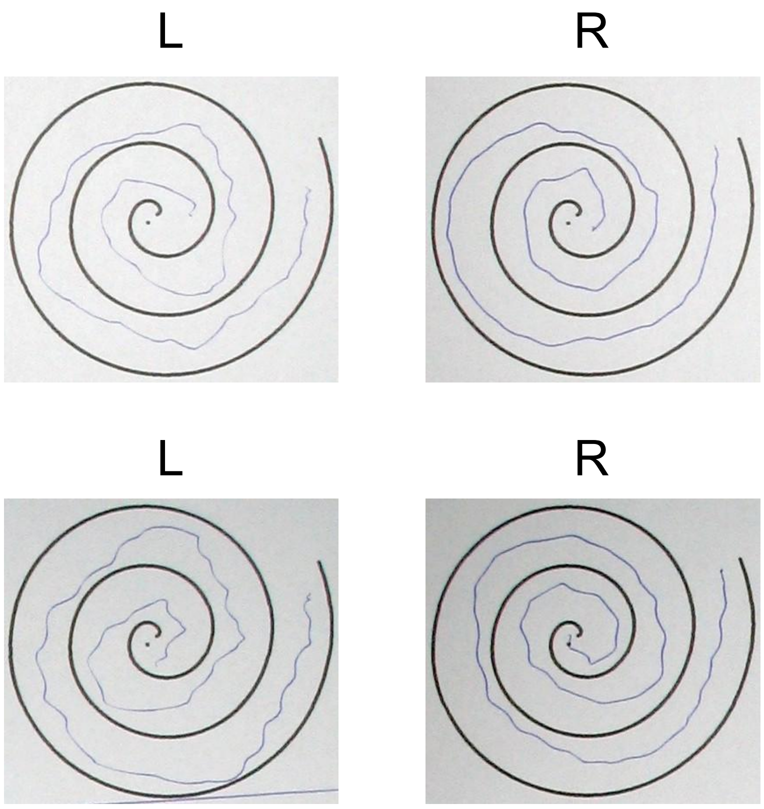


Figure 6.1: The user's spiralometry, before (top) and after (bottom) the session. The distance between the lines is 1cm.

6.5 Results

According to the tremor intensities used by Huck [2012], the user's tremor can be categorized as slight since the peak-to-peak distance of the drawn line in the spiralometry is lower than 0.5cm (see figure 6.1). Due to the dampening effect of the screen (see Wacharamanotham et al. [2011]) the tremor is barely visible when looking at the gesture path in the screenshots.

Initially, the user would have tapped the labels of the menu items. After explaining that the basic input gesture was moving in the direction of the menu option, the user aimed for the labels, as expected (see figure 6.2). When starting off

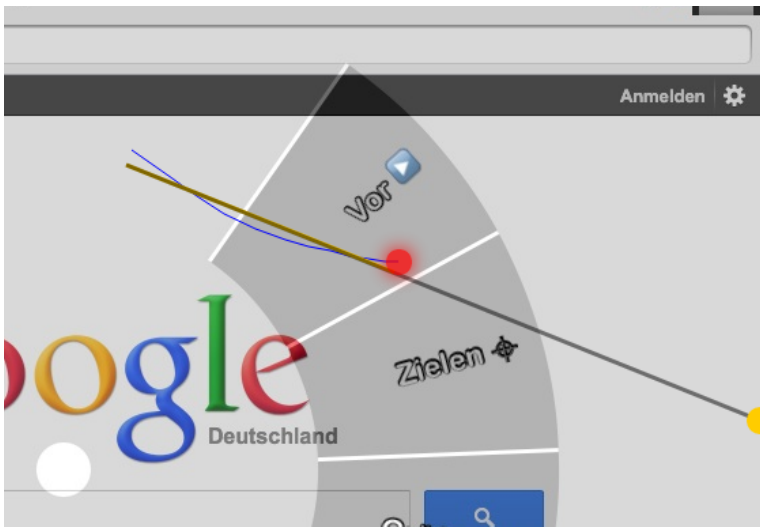


Figure 6.2: The user trying to select the "Vor" option, but not from the center

center, this would often mean missing the actual target area due to the incorrect angle.

In addition, when telling her to start off center, she avoided the center point throughout the whole movement, rather than just in the beginning. The result were curved lines of movement and consequently incorrect selections (see figure 6.3).

When telling her to freely choose a starting point, she would often move from the outside in (see figure 6.4).

After activating the corridor lines and the target arrows, she first did not see the difference. After pointing it out to her, when telling her to start off center, she would first move to the center, and then to the target, to avoid crossing the corridor lines (see figure 6.5). She said she would prefer the lines to not be there, because she felt constrained by them. This is, of course, part of the desired effect.

After telling her that in order to select an option, the movement had to be performed in the direction of the segment of the screen's edge that belongs to an option, she made longer movements to cross the screen edge (see figure 6.6). How-

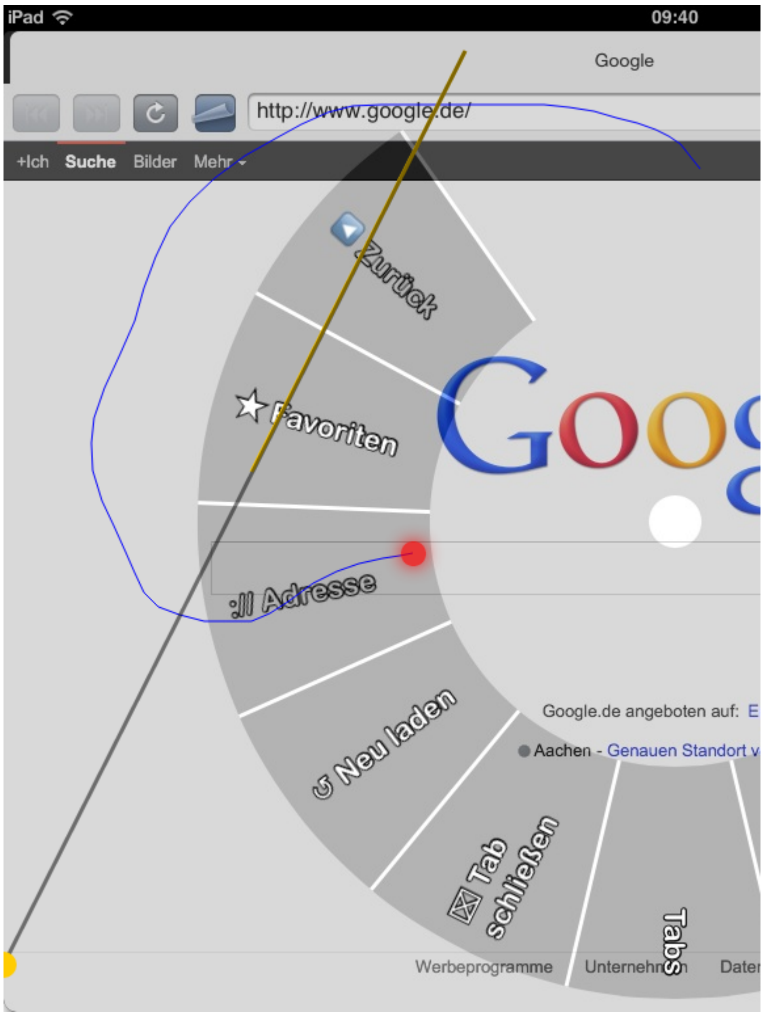


Figure 6.3: The user trying to select the "Adresse" option without crossing the center

ever, when starting off center, she would still first move to the center and then towards the target rather than move towards the target in a single straight line.

In the third condition, the background of the selected menu option is colored differently. This allowed her to first notice that she did not select the intended action, even before completing the gesture. In the fourth condition, a pink dot at the screen edge indicated the exact point the user was aiming at, according to the system. She found this useful.

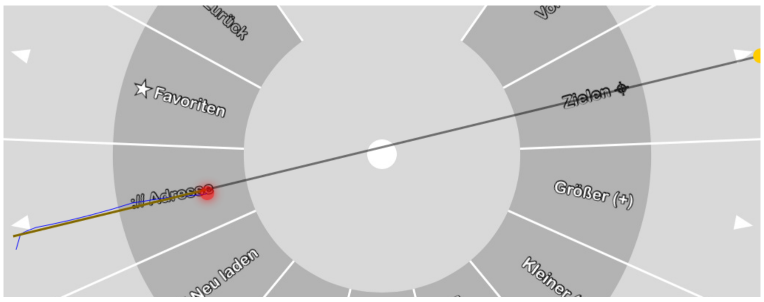


Figure 6.4: The user trying to select the "Adresse" option from the outside in

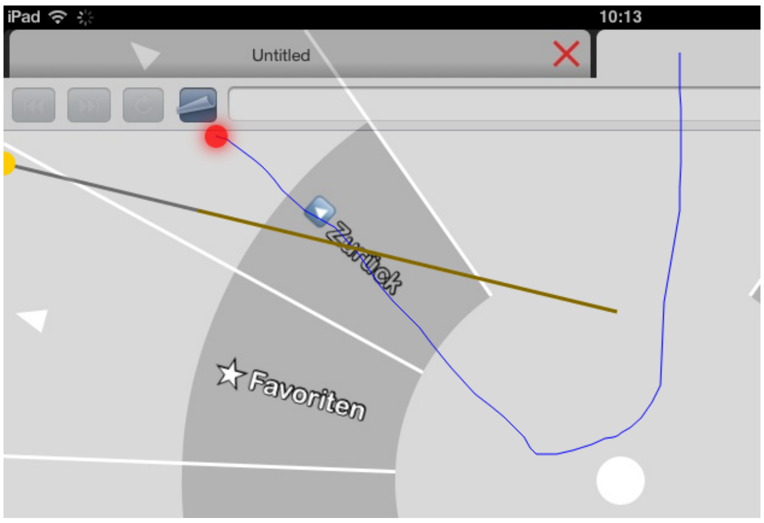


Figure 6.5: The user trying to select the "Zurück" option while avoiding the corridor lines

The fifth condition combined the two types of feedback. As a result she discovered for herself that she did not need to cross the screen edge to select an option. The fifth condition was her favorite. However, even in the end after several explanations, the user would occasionally try to select an option from the outside in, or would first move towards the center point.

Finally, the user commented on how the contrast to the web page was not high enough on visually noisy pages.

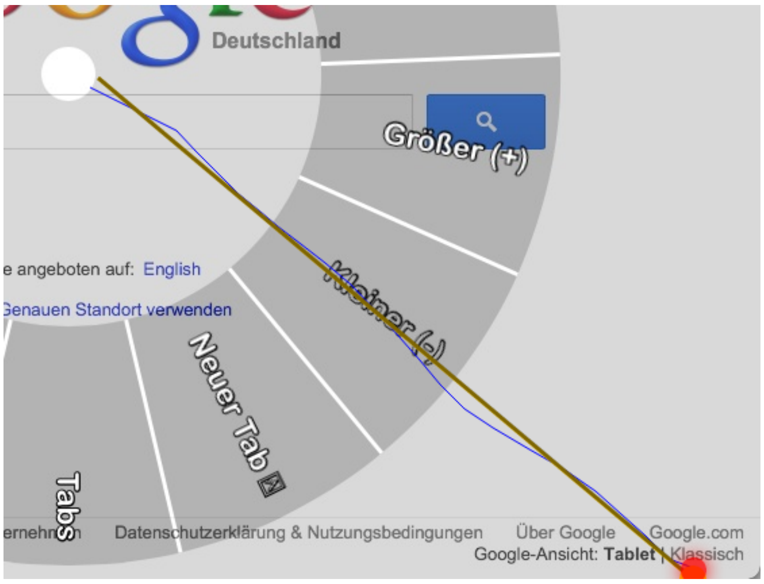

Deutschland
Größer (+)
Kleiner (-)
Neuer Tab
Tabs
e angeboten auf: English
Genauen Standort verwenden
ernehmen Datenschutzerklärung & Nutzungsbedingungen Über Google Google.com
Google-Ansicht: Tablet | Klassisch

Figure 6.6: The user making longer gestures after an explanation about the true location of the targets

6.6 Discussion

Before enabling the feedback mechanisms, the user did not notice that she selected the wrong options. Most likely, the semantics of the menu options were not clear to her due to her lack of computer experience. She therefore was unable to verify the correctness of her choices. It is unclear whether the setting provided sufficient motivation for the user to actually select the correct options since merely responded to instructions and did not expect any particular outcomes from performing the gestures. This might explain why even in the end, after realizing that her gestures didn't select the options she intended to select, she didn't alter her behavior. Nevertheless, the two feedback methods have merit, as they provided the user with the information she would have needed to correct her input.

It is also unclear whether it is possible to inspire users to move towards the screen edge without providing any explanations while keeping the labels and the actual targets at completely different locations. Nevertheless, the corri-

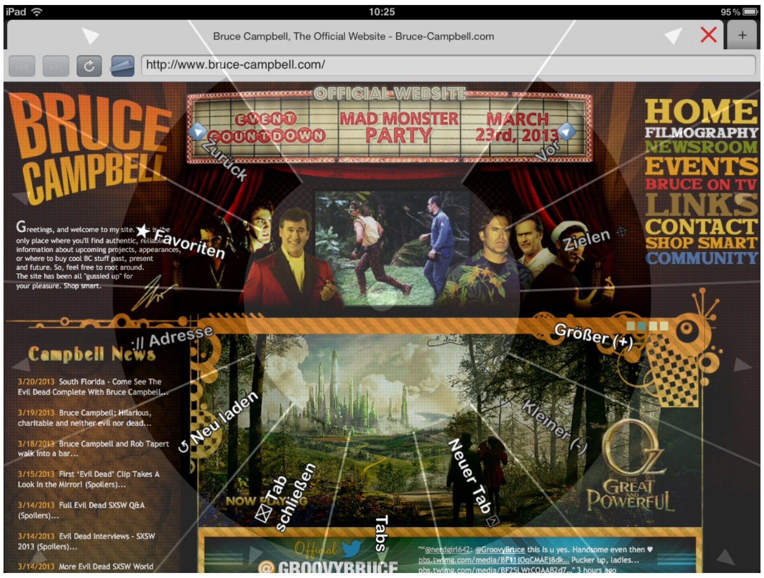


Figure 6.7: Due to the precomposition of the shade and the pie segments, dark web site appeared less shaded in the pie segments than outside, instead of making the labels more readable

dor lines gave the user a strong incentive to move towards the center, and then towards the target, because she did not want to cross the corridor lines. This led to her preferring to start the gesture in the middle in order to spare the effort of first sliding there — a positive outcome. The corridor lines have helped in using the system correctly, more so when they extended all the way to the edge of the screen. Hence, this design was also used in the main study.

In response to the statement about contrast, two changes were made: the shade against the website was made darker, and implemented as a layer that is independent of the background colors of the menu options. For this study, both the shade and the pie segments were precomposed in the same layer. Therefore, on dark backgrounds, the web site was actually more clearly visible in the pie segments than outside of the circle.

6.7 Limitations

Different users may understand the instructions differently, and may understand that the movement has to be performed in a straight line. Users with more computer experience are also more likely to recognize that an incorrect action was triggered as a result of their input. After a few tries, they may discover on their own how the gesture is interpreted.

While the user's tremor intensity was relatively low, this was not a problem for this study. From screenshots like the one in figure 6.2 it is clear that the biggest contributor to incorrect selections were incorrect mental models about Swabbing. In addition, the tremor was still strong enough for her to say that she would prefer the sliding movement over having to tap the small toolbar buttons.

Chapter 7

Longitudinal Study: Swabbing-based web browser

7.1 Study Design

While Hurtmanns [2011] and Huck [2012] had already established the suitability of Swabbing for users with hand tremors in lab settings, it was unknown how Swabbing performed when used on a regular basis for real-world tasks. A **longitudinal single-subject design** was chosen to fill this gap.

The long duration of the study would allow to assess the learning curve of using the tested input techniques, and reveal problems that only occur with regular or prolonged use. For instance, one user of the hierarchical Swabbing menu (Huck [2012]) expected to "go mad" given the prospect of typing a whole letter with this technique. It is possible that with Swabbing the user might become frustrated or fatigued more quickly, limiting its suitability for every-day use. By using Swabbing for real-world tasks over a long period of time, problems specific to certain tasks are more likely to be revealed than during short term use or when defining the task upfront.

Performing studies with multiple participants requires a standardized treatment schedule to maintain comparability of the results. Often this means measuring data that can easily be compared between participants, thereby neglecting effects unique to individuals, even though they may provide valuable insight as well. A single-subject design allows the study to evolve as new information comes to light, so unexpected findings can be explored more easily. Since comparability to other users is not an issue, qualitative analysis can be used more easily to gain a deeper understanding of how the tested input techniques impact the user.

For this study, two **independent variables** were modified: Swabbing support, and the number of characters per Swabbing menu entry. As a baseline for comparison with traditional interfaces, the user first performed six sessions without Swabbing support. Then, Swabbing support was turned on. While in general the user could choose freely whether he wanted to use Swabbing or not, he was required to use it for the training tasks at the beginning of every session (starting with the seventh). After six sessions with Swabbing support, a change request by the user was incorporated into the software.

In order to make the options in the text menu easier to hit, every option should feature two characters instead of one, increasing the opening angle and therefore the target size. The outer character would be selected as before while the inner character would be selected by performing the Swabbing gesture in the opposite direction, i.e., from the outside in. To make it easier to know in which direction the gesture would have to be performed, the alphabet was split in half, using the first half on the outside and the second half on the inside. This way, familiarity with the alphabet might speed up the learning process. This design was used for six more sessions, while the Swabbing menus unrelated to text, as well as the cursor menu, remained unmodified.

See table 7.1 for a schedule of the individual sessions.

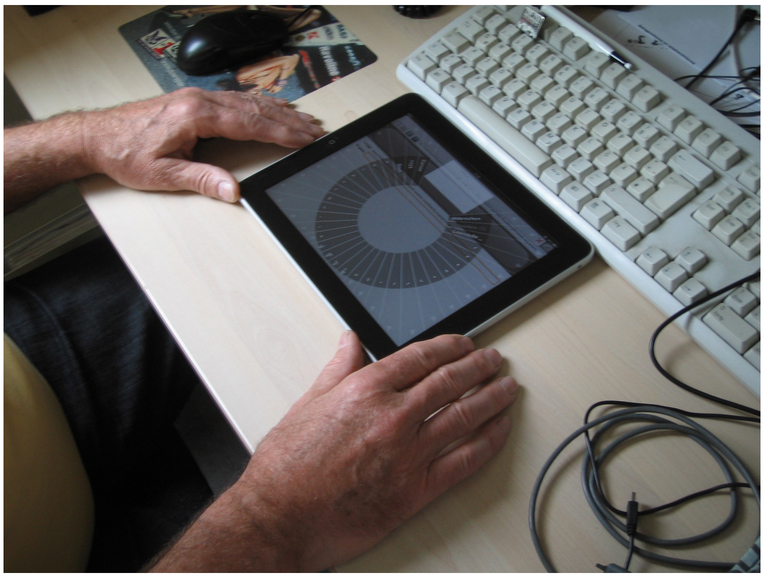


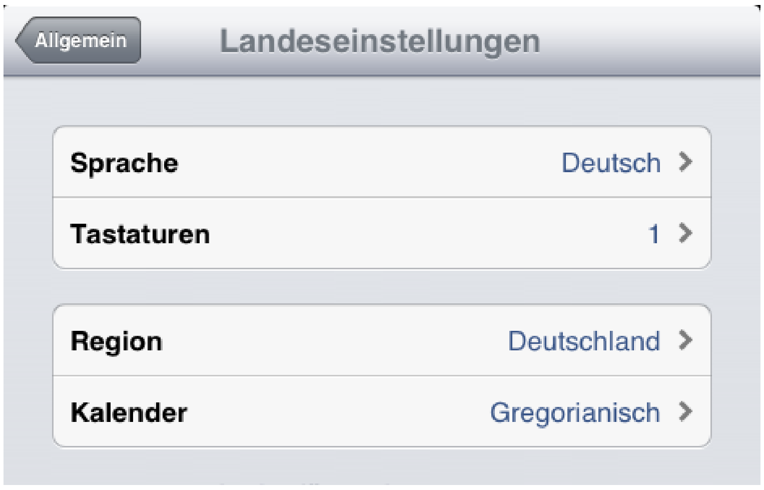
Figure 7.1: The user at his desk, using the browser in the typical configuration

7.2 Setup

For the study, a 16 GB iPad 2 with iOS 6.0.1 was used. Its dual core processor proved essential to process screenshots in the background while keeping the interface responsive. On the iPad 1, the software would frequently crash by exhausting the available memory because the screenshots couldn't be processed quickly enough.

According to the user, the iPad was used in the same sitting position for every session (see figure 7.1). He would put the iPad straight in front of him, while his arms were unsupported.

Since the user is German, the iPad was set to German as well (see figure 7.2). All auto correction features were disabled (see figure 7.3), since only the effect of the mechanics on the user should be tested, not how well the raw input is manipulated. While this makes the Swabbing based keyboard and the native keyboard more comparable, see figure 7.4 for one case in which the iPad still showed special be-



Sprache Deutsch
Tastaturen 1
Region Deutschland
Kalender Gregorianisch

Figure 7.2: iPad Settings — Language: German

havior.

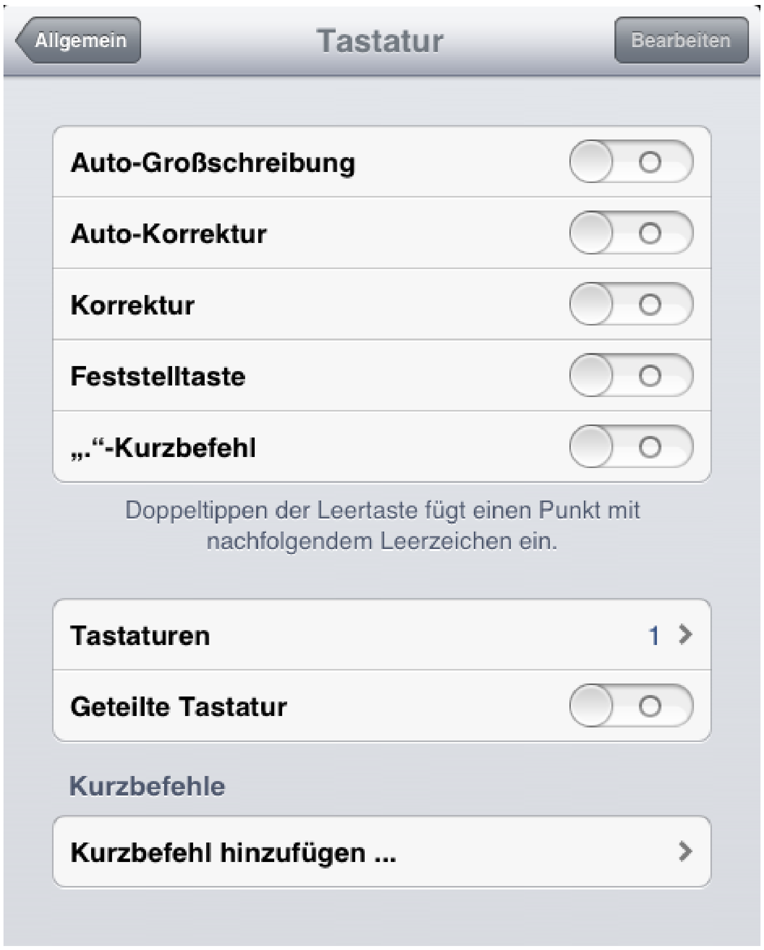
The keyboard layout was set to German, but using the QWERTZ layout (see figure 7.5). Otherwise, the keyboard would have featured keys for umlauts, which the user was discouraged to use for better comparability with Swabbing, lacking support for special characters.

Finally, the multitouch gestures were turned off to prevent conflicts with the five-finger tapping gesture to toggle the visibility of the Swabbing overlay (see figure 7.6).

7.3 Participant

The user was a 66 year old male classic car enthusiast, as made apparent by the web sites he visited. Since age 12 he is near sighted and wears glasses.

He suffers under a severe essential tremor / intention tremor that causes his hands to tremble when deliberately interacting with his environment. It was first noticeable at age 20, and periodically increased in intensity since. He does not use deep brain stimulation. He is right-handed,


Allgemein
Tastatur
Bearbeiten

Auto-Großschreibung: off
Auto-Korrektur: off
Korrektur: off
Feststelltaste: off
„.“-Kurzbefehl: off

Doppeltippen der Leertaste fügt einen Punkt mit nachfolgendem Leerzeichen ein.

Tastaturen 1
Geteilte Tastatur: off

Kurzbefehle
Kurzbefehl hinzufügen ...

Figure 7.3: iPad Settings — Keyboard features: off

and says the tremor is slightly more intense in that hand. His voice is also noticeably affected by the tremor.

The intensity of his tremor increases when he is nervous or stressed, e.g., when he's around people. Alcohol consumption can temporarily decrease the tremor's strength — according to the user, this is true for essential tremors in general. However, the following morning the tremor intensity would usually be worse than otherwise. Drinking coffee increases the tremor strength as well. According to the user, the tremor intensity is not affected by his level of fatigue.

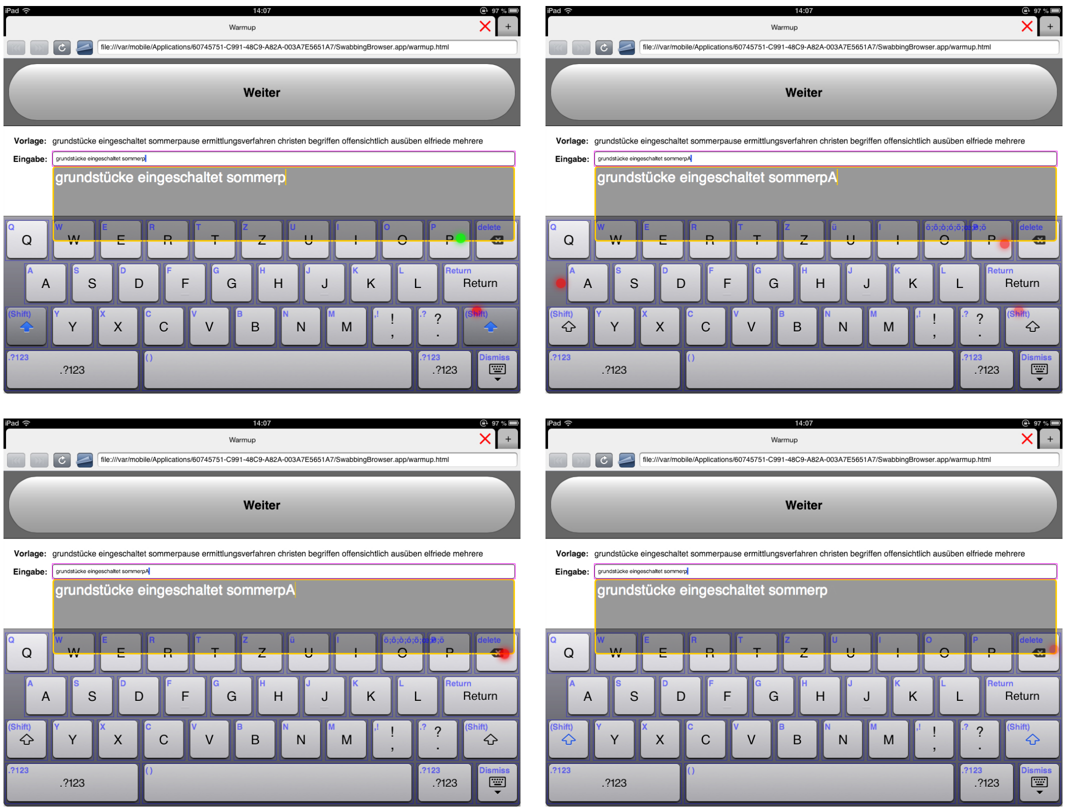

Weiter
Vorlage: grundstücke eingeschaltet sommerpause ermittlungsverfahren christen begriffen offensichtlich ausüben elfriede mehrere
Eingabe:
grundstücke eingeschaltet sommerp
grundstücke eingeschaltet sommerpA
Return
(Shift)
.?123
Dismiss

Figure 7.4: After accidentally pressing the Shift key without noticing (top left), a capital "A" is inserted (top right). After noticing the error and pressing backspace (bottom left), the iPad puts the Shift key in its letter-only state (bottom right). As a result the user repeated the mistake before noticing the behavior.

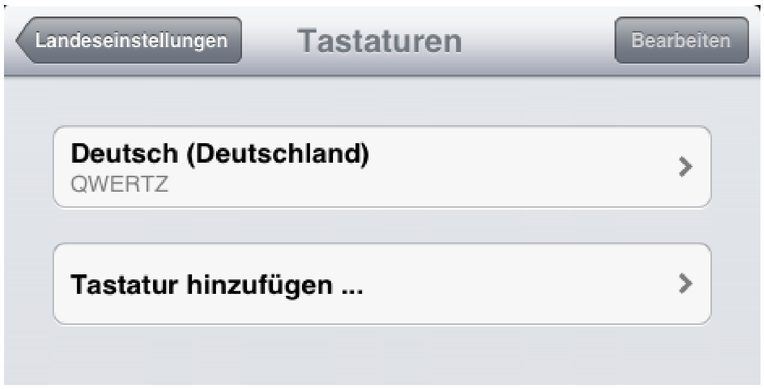


Figure 7.5: iPad Settings — Keyboards: German (QWERTZ layout)

Session	**Date**	**Typing**	**Words**	**Aiming**	**Links**	**Total**
1	2013-04-16	0:07:55	50	0:02:12	10	0:48:31
2	2013-04-17	0:07:59	50	0:01:31	10	0:51:59
3	2013-04-18	0:07:41	50	0:01:10	10	0:13:04
4	2013-04-19	0:07:00	50	0:01:48	10	0:36:42
5	2013-04-20	0:06:47	50	0:01:35	10	1:00:16
6	2013-04-22	0:05:43	50	0:01:14	10	0:28:14
Sum		**0:43:05**	**300**	**0:09:30**	**60**	**3:58:46**
7	2013-04-28	0:12:13	20[a]	0:05:42[b]	10	0:55:09
8	2013-04-30	0:09:24	20	0:08:25[c]	10	0:43:12
9	2013-05-02	0:09:27	20	0:01:45	10	0:31:42
10	2013-05-04	0:09:06	20	0:01:58	10	0:31:47
11	2013-05-05	0:18:58[d]	20	0:01:25	10	0:51:03
12	2013-05-06	0:07:11	20	0:01:13	10	0:32:03
Sum		**1:06:19**	**120**	**0:20:28**	**60**	**4:04:56**
13	2013-05-12	0:08:44	20	0:01:47	10	1:03:38
14	2013-05-13	0:06:16	20	0:01:08	10	0:43:40
15	2013-05-14	0:05:55	20	0:01:41	10	0:54:48
16	2013-05-17	0:06:37	20	0:01:44	10	0:40:30
17	2013-05-19	0:08:19	30[e]	0:01:13	10	1:01:50
18	2013-05-21	0:05:56	20	0:01:31	10	0:45:22
Sum		**0:41:47**	**130**	**0:09:04**	**60**	**5:09:48**

Table 7.1: Session schedule listing the duration of the training tasks with the number of presented words and links. The sessions are grouped by their Swabbing conditions: no Swabbing, one character per entry, two characters per entry.

[a] Since the typing speed with Swabbing was expected to be lower, the number of presented words was reduced to leave enough time for free usage of the browser

[b] The user needed over three minutes to select the second link because this was the first time a target was not part of the first group in the aiming menu

[c] The user was interrupted by a phone call for over seven minutes

[d] The user was interrupted by a phone call

[e] After completing one round of the typing task, the user accidentally pressed the home button, ending the recording. He started over right after, resulting in three completed rounds.

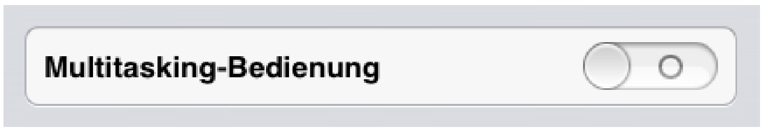


Figure 7.6: iPad Settings — Multitouch gestures: off

The user first came in touch with computers in 1979, mentioning a Sharp computer with cassettes and telling stories about physically huge 25 MB hard drives. Today, he is still actively using computers for three to twelve hours a day. His activities include programming his own website, using desktop publishing software, surfing the internet, writing e-mails, editing videos and more. All his computers (desktop, netbook and web server) are running Windows XP. Aside from using Microsoft Office, he prefers open source software, mentioning Gimp, Scrybus and Firefox as often used programs.

On his desktop PC, he is using a classic IBM keyboard and a normal, symmetrical mouse with two buttons. For his notebook, he has a smaller mobile mouse. Since the tremor intensity is lower in his left hand, he uses it to control the mouse, even though he is right-handed. However, he did not switch the order of the mouse button. To dampen the tremor he holds his left hand with the right one while using the mouse.

While he was taught how to touch type on type writers at young age, he never used that technique. When typing on a hardware keyboard today, he uses multiple fingers, possibly pressing the same key with different fingers or even fingers of the other hand. While typing, he frequently switches between looking at the keys and the screen, occasionally continuing to type while looking at the screen.

The user also has a phone that he uses daily for phone calls, text messages, reminders, listening to music and watching short video clips. The phone does not have a touch screen and the buttons are of normal size. He has been using mobile phones since 1993.

Other than trying the pinch & zoom gesture on a friend's tablet he had no experience with either tablets or touch screens prior to this study. When asked about expected problems, he anticipated problems with typing on the keyboard due to the size of the buttons and accidental double taps.

He started using the internet in 2000 and has a lot of experience doing so. He likes to use it for shopping, home banking, searching for information, discussions in car related sites and maintaining the web site of the car club he is in.

When asked about remaining difficulties of using computers as a tremor sufferer, he mentioned often hitting keyboard keys twice in a row by accident, or hitting the wrong key due to the tremor. While he finds it difficult to hit targets with the mouse, he prefers not to reduce the cursor speed. As one of the most difficult things to use he mentioned drop down menus, especially when they include submenus.

Previously, the user participated in a study where he was asked to move his finger in predefined directions and type on a hardware keyboard. His finger's motions were filmed with a camera for analysis.

7.4 Measurements

After interviewing the user about his demographics, bodily ailments and experience with technology (see the previous section), he was asked about his mental model of the software after seeing it for the first time.

Before and after each session, the user was asked to report his emotional state using self-assessment mannikins, describing emotions through the three axes valence, activation and dominance. Quoting Grimm and Kroschel [2005]:

> *Valence* describes the positive or negative strength of an emotion. *Activation* details the excitation level (high vs. low). *Dominance* refers to the apparent strength or weakness of the speaker.

In addition, the user reported how tired he was on a scale of one to ten, ten being the highest.

Next, his tremor intensity was measured using spiralometry.

After each session, the user was asked to rate the following using a Likert scale:

- Satisfaction with the software
- Input precision
- Input speed
- Difficulty of entering text
- Difficulty of opening links
- Difficulty of controlling the browser

Finally, he was asked to provide additional comments using the voice recorder.

During each session, the user was asked to think out loud while his voice was recorded.

The software saved screenshots after important events that might result in changes on the screen. For backup, additional screenshots were taken one second after all touches ended and at least every five seconds.

In addition, the software created a log file with the following information (and more) along with a timestamp:

- Touch coordinates for all touch phases (began, moved, ended)
- The view and layer at these coordinate (began, ended)
- The layouts of the Swabbing menus
- The Swabbing menu stack
- Results of the Swabbing analysis, like the selected target and the length of the gesture path

- External events of web pages (e.g., URL change, scrolling, zooming)
- Internal events of web pages (e.g., focus, blur, click, mousedown)
- Detailed information about the state of the typing and aiming tasks
- The value, selection range and position of edited text fields

For the text entry task, we define the *presented string* as the words the user was asked to type, separated by a space character. The *transcribed string* is the actual input provided by the user. To analyze the error rate, we also need the *input stream*: the sequence of all keys pressed, including the backspace key. For both the native touch keyboard and for Swabbing, a *key press* refers to a manipulation of the text field, i.e., inserting or deleting a character. Keys that were touched but did not alter the text field (e.g., because the touch ended too far outside of the key's area), are not considered key presses.

Analyzing the log files allowed measuring the number of words per minute (WPM) that the user typed during the text entry task, as well as the uncorrected and corrected error rates. The definitions of these measures, taken from MacKenzie and Tanaka-Ishii [2010], are as follows:

WPM = $\frac{|T|-1}{S} \times 60 \times \frac{1}{5}$ with $|T|$ being the length of the transcribed string and S being the number of seconds from the start of the first touch that resulted in a key press to the start of the last touch that resulted in a key press. Note that this does not include the time it took to type the last character, taken into account by subtracting 1 from the length of the transcribed string.

Uncorrected error rate $= \frac{IF}{C+INF+IF}$ and *corrected error rate* $= \frac{INF}{C+INF+IF}$, with IF being the number of incorrect characters that were inserted, but subsequently fixed, and INF being the number of incorrect characters that were inserted, but not fixed. C is the number of correct characters inserted. For details about this classification, please refer to

MacKenzie and Tanaka-Ishii [2010]. The analysis was performed using StreamAnalyzer (see Wobbrock and Myers [2006]) by creating the appropriate XML files from the log files.

The recorded touches were matched with recorded events like changing the value of a text field or scrolling the web page. This allowed a more precise measurement of the user's performance than merely looking at the logged editing events would have, since random load-induced delays between touching the screen and the software interpreting the touch as a key press are bypassed this way. This also makes other measures produced by the StreamAnalyzer more meaningful, such as gestures per character. Only counting changes in the value of a text field would not take into account extra touches during text entry that did not result in such a change.

For the aiming task, the user needed to select a link chosen at random by the software (selection step), followed by selecting the continue button (confirmation step). By analyzing the log files, selection and confirmation times were measured. The **selection time** describes the time between the software presenting a link and the end of the touch to select it successfully. The **confirmation time** describes the time after successfully selecting a link until the end of the touch to select the confirmation button.

For Swabbing, the continue button is a special case since its location in the aiming menu is always the same. Therefore, the time to confirm a selection does not include matching the target indicator to the corresponding menu entry, except for the first few rounds. In addition, the number of touches per selection and per confirmation were extracted from the log files.

The important difference between the selection and confirmation step is that the first one requires the user to search the page for the link to click and, for Swabbing, find the corresponding menu entry. The confirmation step always referred to the continue button that was always at the same known position on the screen as well as the same known position in the Swabbing menu (first entry in the first group). The confirmation time therefore measures the raw speed of the input techniques, though this can include switching to the first group of links before selecting the continue button.

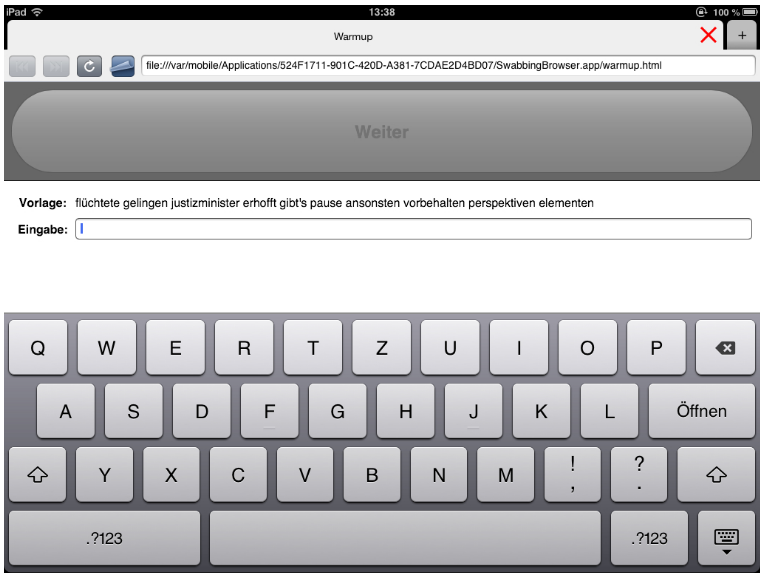
Finally, by listening to the audio logs and investigating some of the session recordings, an attempt was made to identify problems with using Swabbing in the wild, or in web browsers, specifically.

7.5 Procedures

Before each session, the user filled out the first part of the questionnaire, performed a spiralometry with both hands and turned on the voice recorder. The questionnaire can be found in appendix C.3 "Longitudinal Study: Swabbing-based web browser".

Next, the user unlocked the iPad, and started the test software, first showing the text input task. For this purpose, ten words were shown above an input field (see figure 7.7). The input field was automatically focused, triggering either the display of the native touch keyboard (sessions 1-6), the Swabbing text menu for letters using one character per entry (sessions 7-12) or the Swabbing text menu for letters using two characters per entry (sessions 13-18). After five rounds (native touch keyboard) or two rounds (Swabbing), respectively, the text selection task was complete. The user needed to confirm each round by activating the continue/skip button.

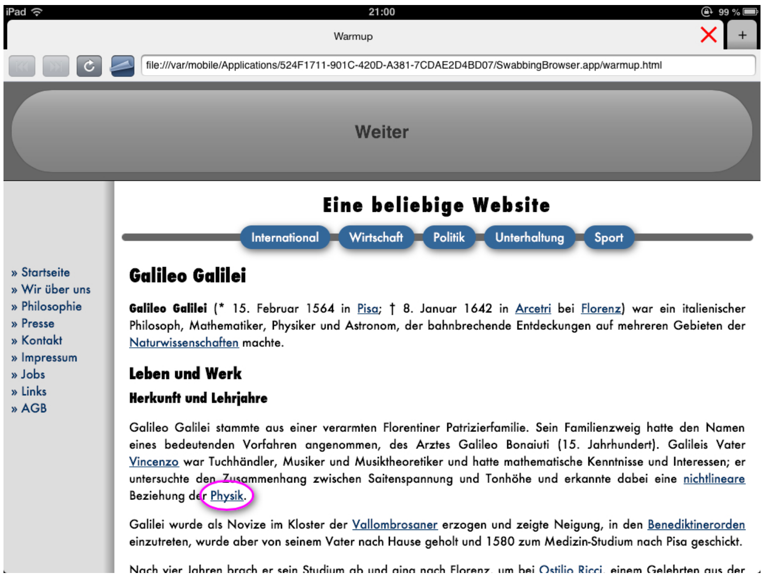
The text entry task was followed by the aiming task. Here, a mockup web page was shown to provide a realistic example for typical distributions of tapping targets on web sites. It featured a vertical and a horizontal list of links, as well as several paragraphs of text containing links. The link select was marked with a circle (see figure 7.8). If the user selected a link, feedback was provided whether the selection was correct or incorrect (see figure 7.9). For other selections, accidental scrolling, etc. no feedback was provided. After ten rounds with one link each the aiming task


iPad
13:38
100 %
Warmup
file:///var/mobile/Applications/524F1711-901C-420D-A381-7CDAE2D4BD07/SwabbingBrowser.app/warmup.html
Weiter
Q W E R T Z U I O P
A S D F G H J K L Öffnen
Y X C V B N M ! , ? .
.?123
.?123

Vorlage: flüchtete gelingen justizminister erhofft gibt's pause ansonsten vorbehalten perspektiven elementen

Eingabe:

Figure 7.7: The text entry task



Eine beliebige Website

Galileo Galilei

Galileo Galilei (* 15. Februar 1564 in Pisa; † 8. Januar 1642 in Arcetri bei Florenz) war ein italienischer Philosoph, Mathematiker, Physiker und Astronom, der bahnbrechende Entdeckungen auf mehreren Gebieten der Naturwissenschaften machte.

Leben und Werk

Herkunft und Lehrjahre

Galileo Galilei stammte aus einer verarmten Florentiner Patrizierfamilie. Sein Familienzweig hatte den Namen eines bedeutenden Vorfahren angenommen, des Arztes Galileo Bonaiuti (15. Jahrhundert). Galileis Vater Vincenzo war Tuchhändler, Musiker und Musiktheoretiker und hatte mathematische Kenntnisse und Interessen; er untersuchte den Zusammenhang zwischen Saitenspannung und Tonhöhe und erkannte dabei eine nichtlineare Beziehung der Physik.

Galilei wurde als Novize im Kloster der Vallombrosaner erzogen und zeigte Neigung, in den Benediktinerorden einzutreten, wurde aber von seinem Vater nach Hause geholt und 1580 zum Medizin-Studium nach Pisa geschickt.

Nach vier Jahren brach er sein Studium ab und ging nach Florenz, um bei Ostilio Ricci, einem Gelehrten aus der

Figure 7.8: The aiming task. The link to click is marked.

was complete. The user needed to confirm each round by activating the continue/skip button.

For each round of both tasks, the continue button was deac-

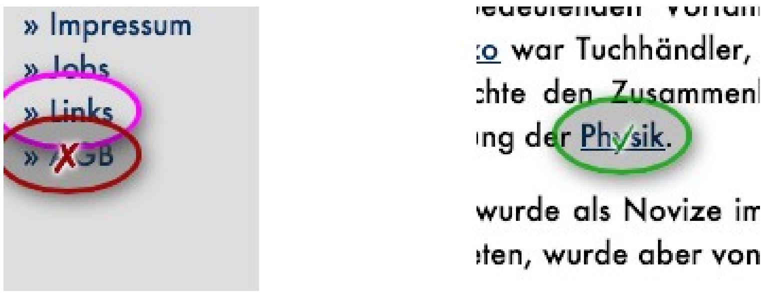


Figure 7.9: Incorrect (left) and correct (right) selection during the aiming task

tivated for five seconds. During aiming tasks, it would contain a red label reading the German word for "Skip" after the five seconds, or a green label reading the German word for "Continue" once the correct link had been selected.

After the aiming task, a thank you note was shown for one second, after which the tab was replaced with a new tab to start with a fresh page history. New tabs automatically load googlecom. The user was now free to use the browser as he wished. He was asked to aim for a session length of roughly an hour, but could end the software at a time of his choosing.

To stop the software, he needed to press the home button. The software was configured to exit in this case, making sure the software was in a consistent state at the beginning of each session.

Finally, the user completed the second part of the questionnaire and another round of spiralometry. He was asked to provide more comments using the voice recorder or the back of the form, and to turn the voice recorder off when done.

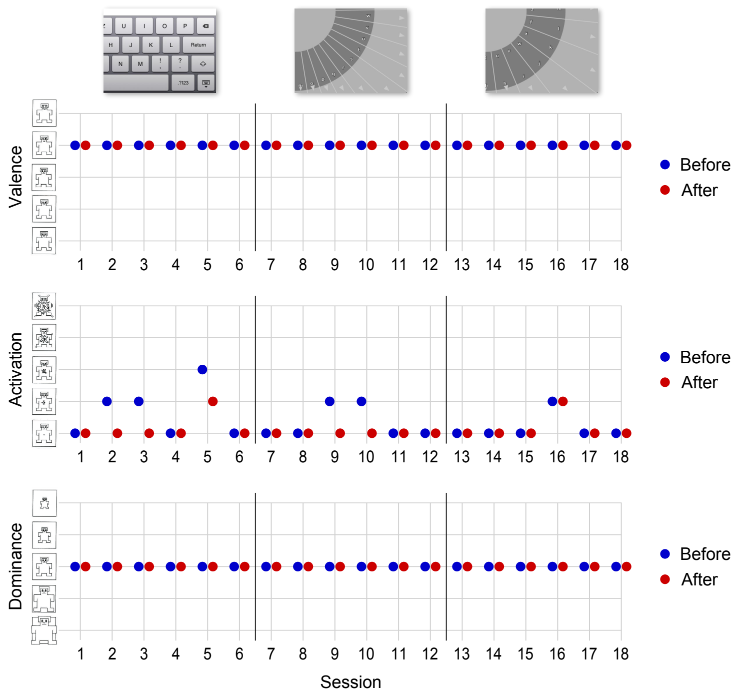


Figure 7.10: The user's emotional State (see Grimm and Kroschel [2005]) before and after each session

7.6 Quantitative Results

7.6.1 Results from the questionnaire

The questionnaire captured the user's emotional state (figure 7.10), level of tiredness (figure 7.11), tremor intensity (figures 7.12 and 7.13), satisfaction with the software and a self-assessment of input precision and speed (figure 7.14), and ratings of text entry, link selection and general browser control difficulty (figure 7.15).

The user's emotional state was usually the same before and after the session. Occasionally he would report having been calmer afterwards by rating his activation lower. Since the user always reported the same value for valence and dominance, it is possible that the meaning of the self-assessment mannikins were not clear to the user. It is also possible that

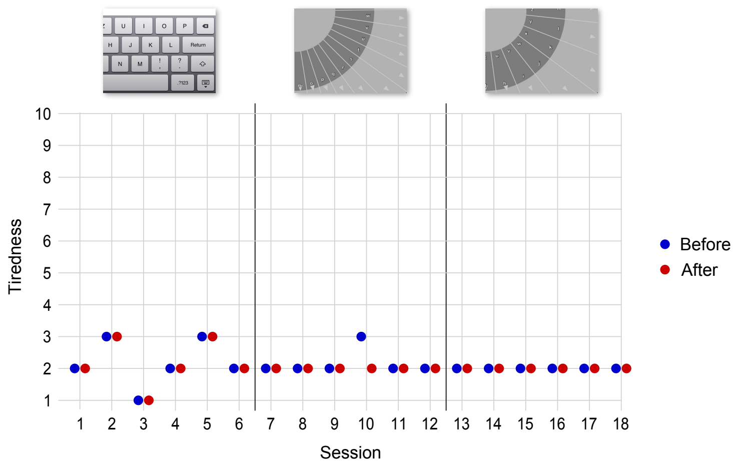


Figure 7.11: The user's tiredness before and after each session (10 being most tired)

he preferred not to report his emotional state in the first place. During meetings with the user, he was always very friendly, calm, patient and understanding, so it is also possible that he was simply not easy to affect. However, the audio log clearly indicated moments of frustration. It is possible that the frustration had subsided by the time the session was over, leaving no impact on the emotional state.

After session 10, the user reported being slightly less tired than before, otherwise the levels reported before and after each session were the same. Given that the user routinely uses computers for several hours a day, it is very possible that he did not find the sessions exhausting. He may also not have considered reporting exhaustion when asked for his level of tiredness.

While the user is right-handed, the measurements regarding his left hand are noteworthy. Here, his tremor intensity varied strongly from session to session, sometimes being moderate (5-10mm) and sometimes being severe (> 20mm). In half the cases, the left hand's tremor intensity also varied before and after the session, sometimes being lower after the session, but more often being higher. Especially after using the browser with two characters per menu

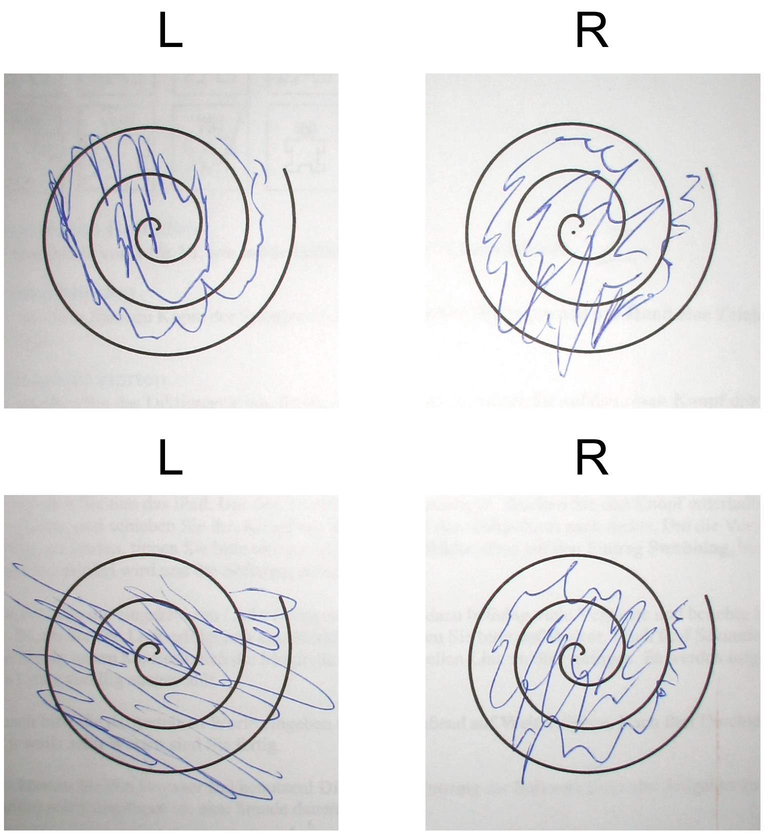


Figure 7.12: Spiralometry of the user before (top) and after (bottom) the first session. The distance between the lines is 1cm.

entry, the tremor was often more intense than before. In general, tremor intensity varies with stress, so this could point to higher stress levels during this condition. Since the user rated was very satisfied with the browser in this condition (see figure 7.14), it is unlikely that frustration was the source of the change in intensity. However, it conceivable that the double character layout increased the cognitive load and thereby affected the tremor intensity.

The right hand, however, showed a remarkably consistent tremor intensity, before and after each session as well from session to session. While the right hand's tremor intensity usually was marked (10-20mm), it never appeared to be severe (> 20mm), as opposed the the left hand with multiple

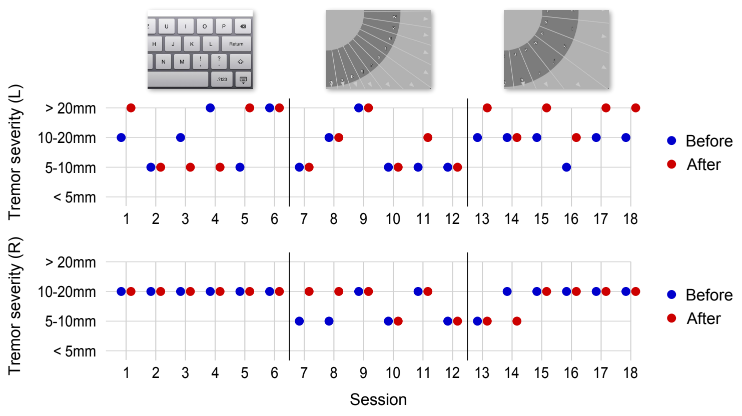


Figure 7.13: The user's tremor intensity before and after each session, measured via spiralometry, using the peak-to-peak value of the drawn line. Left hand on the top, right hand on the bottom. The user is right-handed.

occasions of severe tremor strength. This is especially remarkable since the user originally reported that his tremor is stronger in the right hand.

There is a possibility that the user mixed the spirals up, drawing them with the other hand. It is also possible that the dominant hand is generally more stable for tremor sufferers, or that actively using a hand stabilizes its tremor intensity. Finally, please note that the categorization of the spiralometry results was done by hand, and that other people might disagree with the ratings, possibly changing these findings.

Generally, the user was "relatively satisfied" with the software, but "very satisfied" after the sessions with two characters per menu entry, and "somewhat dissatisfied" after the first session with Swabbing, commenting on it being "partially too slow".

Self-assessed precision was mostly "relatively precise", and "very precise" after four of the six sessions with two characters per menu, again showing that this configuration is the user's favorite. The self-assessed input speed was usually reported as "relatively high", with only three exceptions.

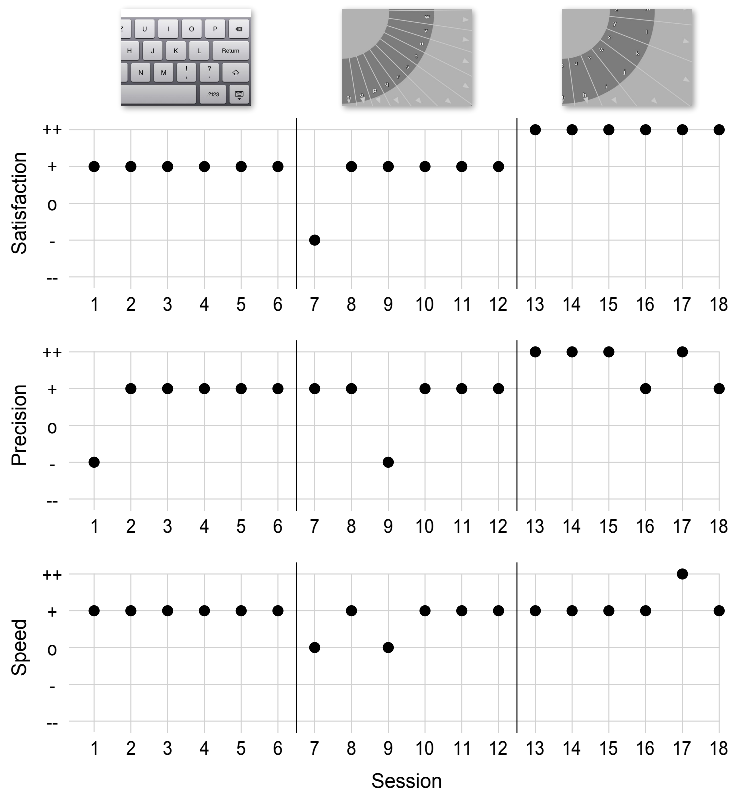


Figure 7.14: The user's satisfaction with the software, and self assessment of input speed and precision, measured on a Likert scale, from very dissatisfied/imprecise/slow (–) to very satisfied/precise/fast (++)

For the single character Swabbing layout, input speed was rated "neutral" two times, whereas the double character layout was rated "very fast" once. At least among the Swabbing layouts, the user's preferences are clear.

As we will see later, the user entered significantly fewer words per minute when using Swabbing for text entry instead of the native touch keyboard. This is hardly reflected in his self-assessed speed. He also specifically claimed at one point that his input speed was on par with that of using the touch keyboard. The most probable explanation for this discrepancy is that the number of words of the text entry task was reduced from 50 to 20 for the Swabbing ses-

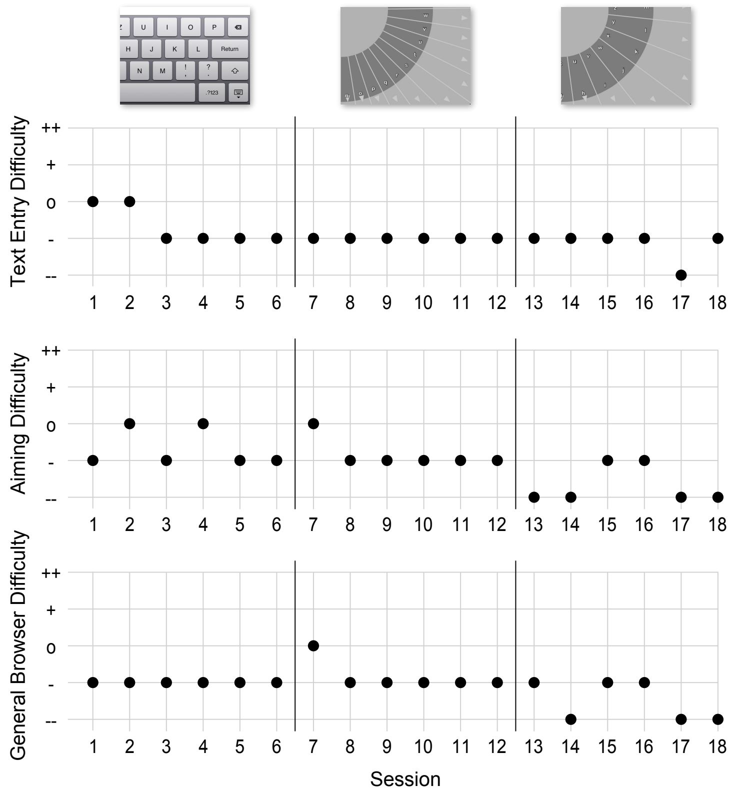


Figure 7.15: The user's rating of the text entry, link selection and general browser control difficulty levels measured on a Likert scale, from very easy (–) to very difficult (++)

sions to leave enough time for free-form browser use. As table 7.1 shows, the text entry sessions ended up being similarly long this way, as intended. Maybe the user was more aware of how long he took to finish the typing task than of his actual input speed. It is also possible that he paid more attention to his speed during the Swabbing sessions, and overestimated his absolute typing speed based on noticing relative improvements.

Finally, the assessment of difficulty levels show that in general, the user found the software least difficult to use during the double character Swabbing condition. However, since his rating of the text entry difficulty changed far less than his rating of the aiming difficulty and general browser control difficulty, it's probably more appropriate to attribute perceived ease of use to having learned to control the Swabbing interface well enough. This indicates that the number of sessions per condition may have been too low to reach the end of the learning curve, but that two weeks or twelve sessions of near daily use could be sufficient. Measuring the link selection time by analyzing the log files confirms this observation, with the learning curve stabilizing during the second half of the Swabbing condition, i.e., from session 13 onwards.

7.6.2 Results from analyzing the log files

All results presented her refer to the text entry and aiming tasks the user performed at the beginning of every session. The data from the free form browser usage was not analyzed quantitatively.

The average selection time per session was initially lower for tapping (see figure 7.16) even though the number of touches needed for this task was higher than one (see figure 7.17), indicating that the user unsuccessfully touched the screen a few times. However, over time the selection time using Swabbing was greatly reduced, reaching as little as 4s compared to 3s for the native condition.

Since the continue button is always in the same place, removing the need to search for it, the gap between the selection and confirmation times when using Swabbing represents the overhead of having to first locate the menu entry that belongs to the marked link. When looking at the confirmation times when using Swabbing, it becomes clear that the technique itself is slightly superior to tapping, at least for the user of this study: with Swabbing, he repeatedly managed to need about two seconds per confirmation step, whereas he consistently needed more than 2.5 seconds with tapping.

Session 16 contains a 40s long outlier in that the user forgot to first switch to the first group of links before confirming the selection, causing him to select an incorrect link (in

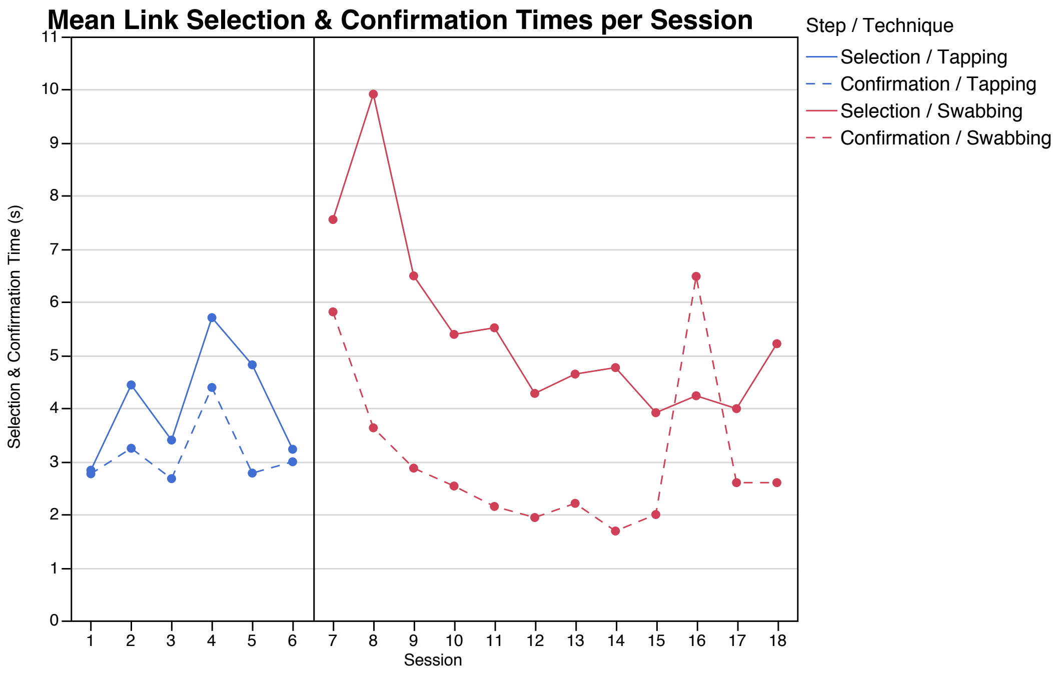


Figure 7.16: Average selection and confirmation times per session for the aiming task

addition to the correct one), followed by waiting and wondering why no new link was shown. Removing this point would result in an average confirmation time of just 1.67 seconds for session 16, the lowest of all. However, since the reason for the outlier can be seen as a weakness of the aiming menu, I decided not to remove it.

Session 8 contains another outlier, this time in the selection time's curve. Here, the user was unable to tell right away which target indicator belonged to the marked link. He assumed there was none, and searched for it in group two. There, he incorrectly decided that the target indicator of the link below was the one he had to use (see figure 7.18). Without this outlier, the average selection time for session 8 would be 6.71 seconds. Since this definitely points out a weakness of the current design, the outlier was kept as well.

Note that a similar gap can be seen between selection and confirmation times during the tapping condition. It is pos-

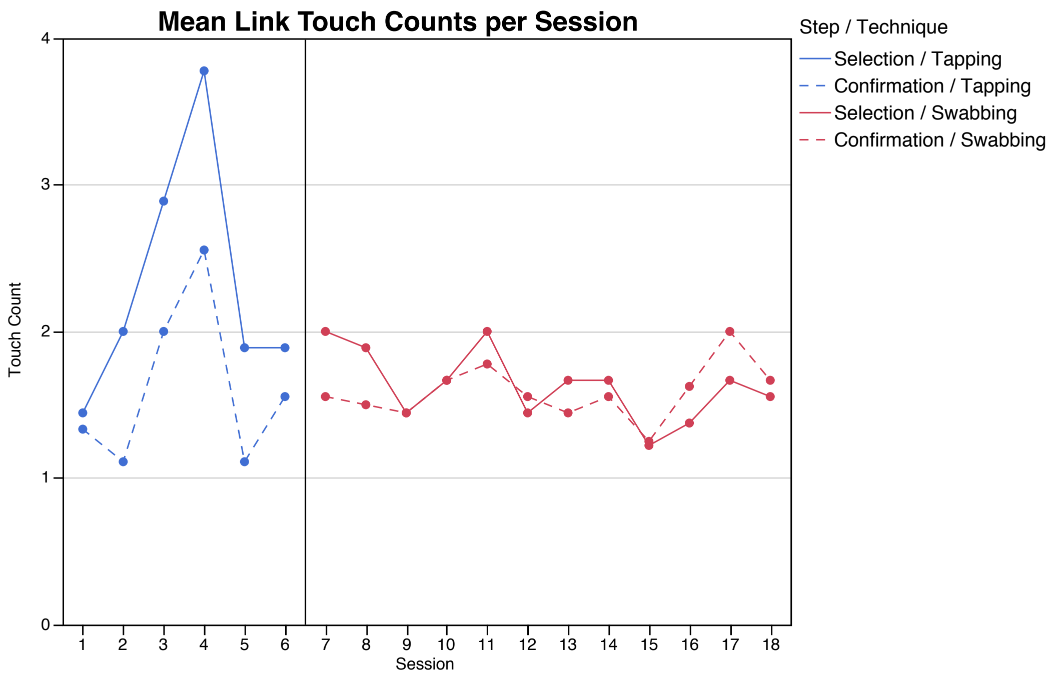


Figure 7.17: Average selection and confirmation touch counts per session for the aiming task

sible that this represents the advantage achieved by increasing the button size. However, the irregular shape of the two curves suggest that there are other factors that have a stronger impact on the tapping performance for tremor sufferers. One candidate for such a factor would be variances in the tremor strength, although the spiralometry data does not support this assumption. However, tremor intensity can change very quickly, so this is inconclusive.

The number of touches needed for the selection and confirmation steps indicates the error rate (see figure 7.17). For tapping tasks, only one touch is needed for each step, whereas with Swabbing two touches may be needed if the link is in a different group. As the graph shows, the average touch count for Swabbing does not exceed two, indicating a low error rate. Further analysis would be needed to distinguish required touches from superfluous touches.

In contrast to the findings by Huck [2012], the user was

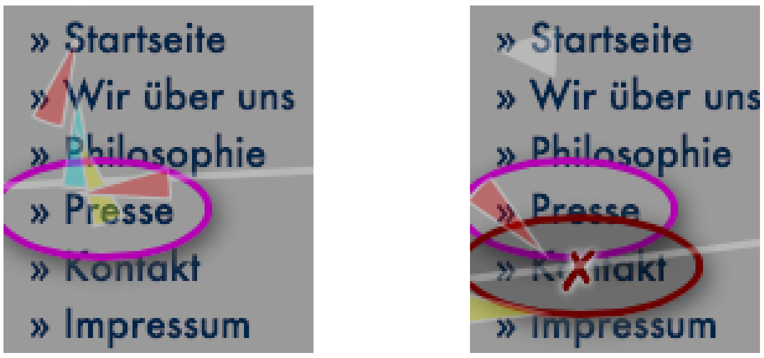


Figure 7.18: Left: The red arrow within the circle would have been the correct target indicator. **Right:** Instead, the user searched for the indicator in group 2, and used the one from the link below.

much faster when using the native touch keyboard than when using Swabbing (figure 7.19). While he also made a lot more mistakes then with Swabbing, he corrected enough of them to achieve an uncorrected error rate that is comparable to Swabbing. This means that for the user of this study, the native touch keyboard is a viable option, despite his strong tremor. Nevertheless he preferred Swabbing over the native keyboard, in particular the layout with two characters per menu. It is possible that the sheer number of errors to correct with the touch keyboard make the typing task more exhausting, as the user would have to watch out for errors much more closely.

Note that the graph for the text input error rate originally looked very different (see figure 7.21). The original data contained numerous problems that would skew the results quite heavily. To produce the graph in figure 7.20, the data was cleaned:

- During the first sessions, the presented string during the text entry task sometimes contained special characters, like German umlauts. However, the user had been instructed to replace ä/ö/ü/ß with ae/oe/ue/ss. Sadly, this was originally counted as an error. Consequently, the presented string used to calculate the error rates was adjusted to contain the

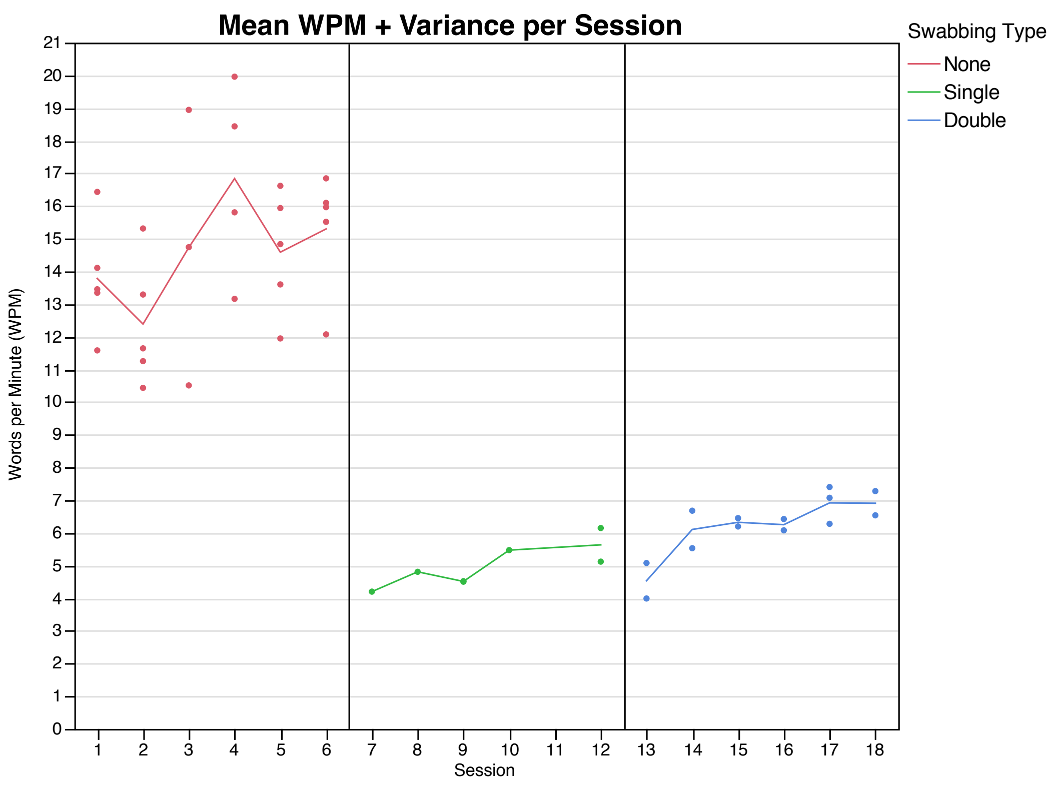


Figure 7.19: Text entry speed, measured in words per minute

replacements the user produced

- Sometimes the user forgot to enter an entire word, resulting in one uncorrected error per character of the missing word. There was no point in testing the user's reading skills, therefore the missing word was removed from the presented string used for the analysis

- When using Swabbing for the text entry task, the word to transcribe was occasionally covered by a label of the Swabbing menu. Initially, the user tried to close the Swabbing menu using the five finger tapping gesture, to read the word and then return to the text field. However, when he first tried this, the continue button was triggered, cutting his typing session short. The words thusly skipped where removed from the presented string. Later, the user would just guess the word to type and frequently guess incor-

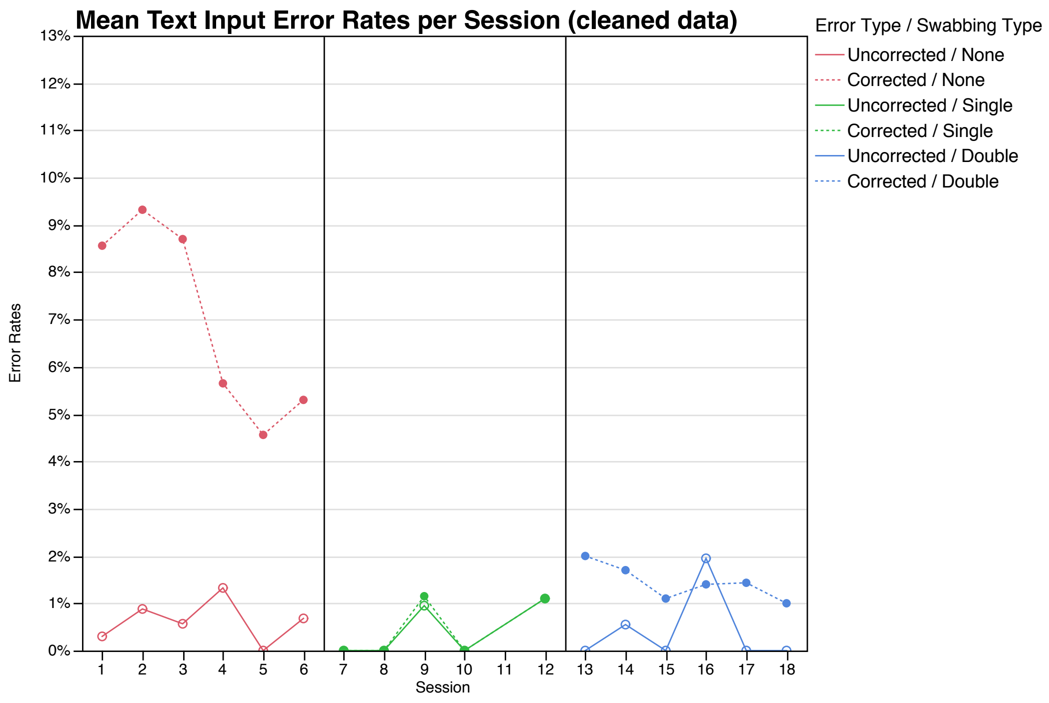


Figure 7.20: Error rate using the cleaned data

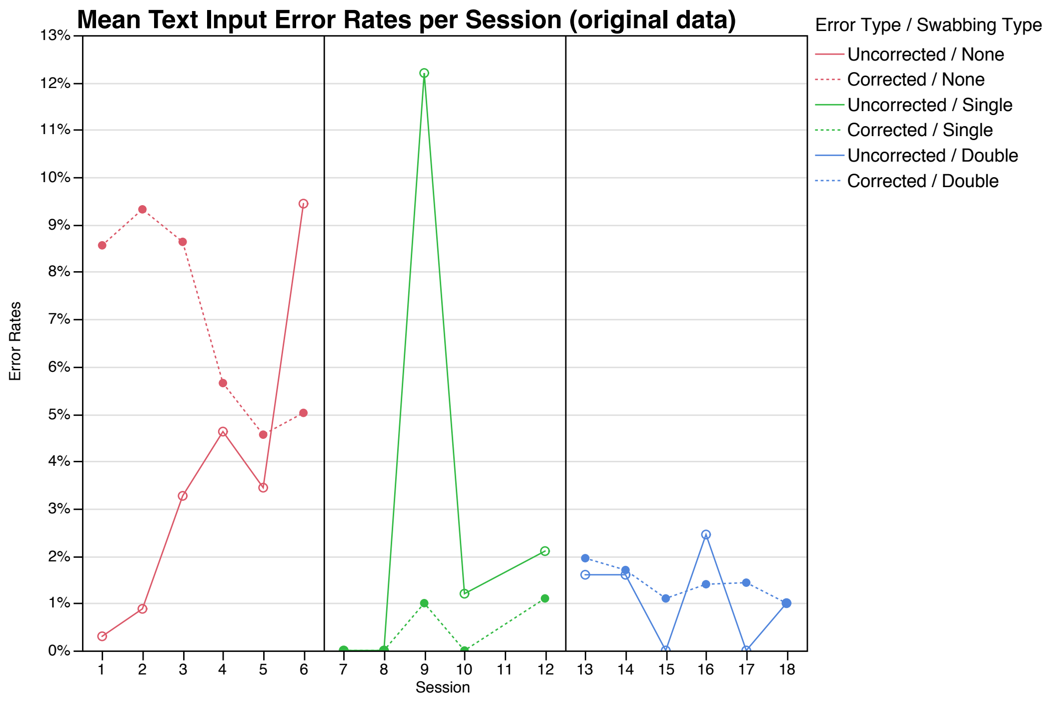


Figure 7.21: Error rate using the original data

rectly. Since he always announced his guess using the voice recorder, it was possible to alter the presented strings used for the analysis such that it would feature the word the user guessed, removing those errors from the results as well.

- Finally, the user got into the habit of adding a space character after entering a word even if it was the last word to be entered. This would also result in an uncorrected error. Since this is no indication of a problem with the input techniques, the final keystroke was removed from the input data, and the space character was removed from the transcribed string used for analysis. Note that this change also slightly affects the number of words per minute. However, figure 7.19 contains the unaltered data.

There are two more things to note about the error rates in figure 7.20. During five of the corresponding text entry sessions, the user decided to correct his mistakes by first moving the cursor using the cursor menu. While it is a positive result that he was able to use the cursor menu successfully, and did so repeatedly, the text input measures used here break down as soon as cursor movement is involved. Consequently, the sessions involving the user of the cursor menu could not be used for the analysis, resulting in just one round of the typing task to contribute to the results for sessions 7, 8 and 9, and no results for session 12. It is obvious that these results would have contained corrected errors if this concept were well-defined for cursor movements. It follows that the corrected error rate for the single Swabbing condition is too low.

Consequently, the seemingly higher corrected error rate for the double character Swabbing condition has to be taken with a grain of salt. Instead of assuming that this condition increased the error, the opposite is conceivable as well: had the user made as many or more mistakes as with the single character Swabbing condition, he likely would have used the cursor menu to correct them as well. Instead, the cursor menu wasn't used at all during this condition, possibly indicating a lower error rate. In addition, an analysis of the errors produced in the last six sessions revealed that only one mistake was made that directly indicates a problem with this layout: the user typed an "r" instead of an "e", and both characters are located on the same menu entry. The user had selected the right option, but in the wrong direction. The other errors were typical Swabbing errors: selecting a character next to the correct one, and selecting the delete option instead of the space option and vice versa. It is likely that the switch from the single to the double character layout contributed to these mistakes, as the user was by then used to one layout and needed to now learn a different one. In addition, the German labels for Delete and Space ("Löschen" and "Leerstelle"), are equally long when taking their icons into account, and both start with an L, possibly making them easier to mix up.

Nevertheless, the user also selected an "f" instead of an "e". This was a frequent mistake with the single swabbing layout, indicating that this direction of movement was particularly troubling for the user (see also Huck [2012] for an investigation into error rates depending on the direction of movement). Since the new layout moved these characters to a different position, it was surprising to see the user make this particular mistake again.

7.7 Qualitative Results

After completing the typing and aiming tasks in the first Swabbing session, the user accessed his personal homepage successfully. To protect his privacy, this session cannot be shown, but figures 7.22, 7.23 and 7.24 show him using the same techniques equally successfully to search for auctions related to his hobby.

In alignment with findings by Huck [2012] according to which movements in certain directions produce more errors than movements in other directions, the user reported that his tremor would usually be lowest if he can perform the gesture out of his shoulder, using his arm as a static appendix. If he needs to use additional muscles for the movement, the tremor would get stronger. This was noticeable quite quickly: the user frequently selected the letter

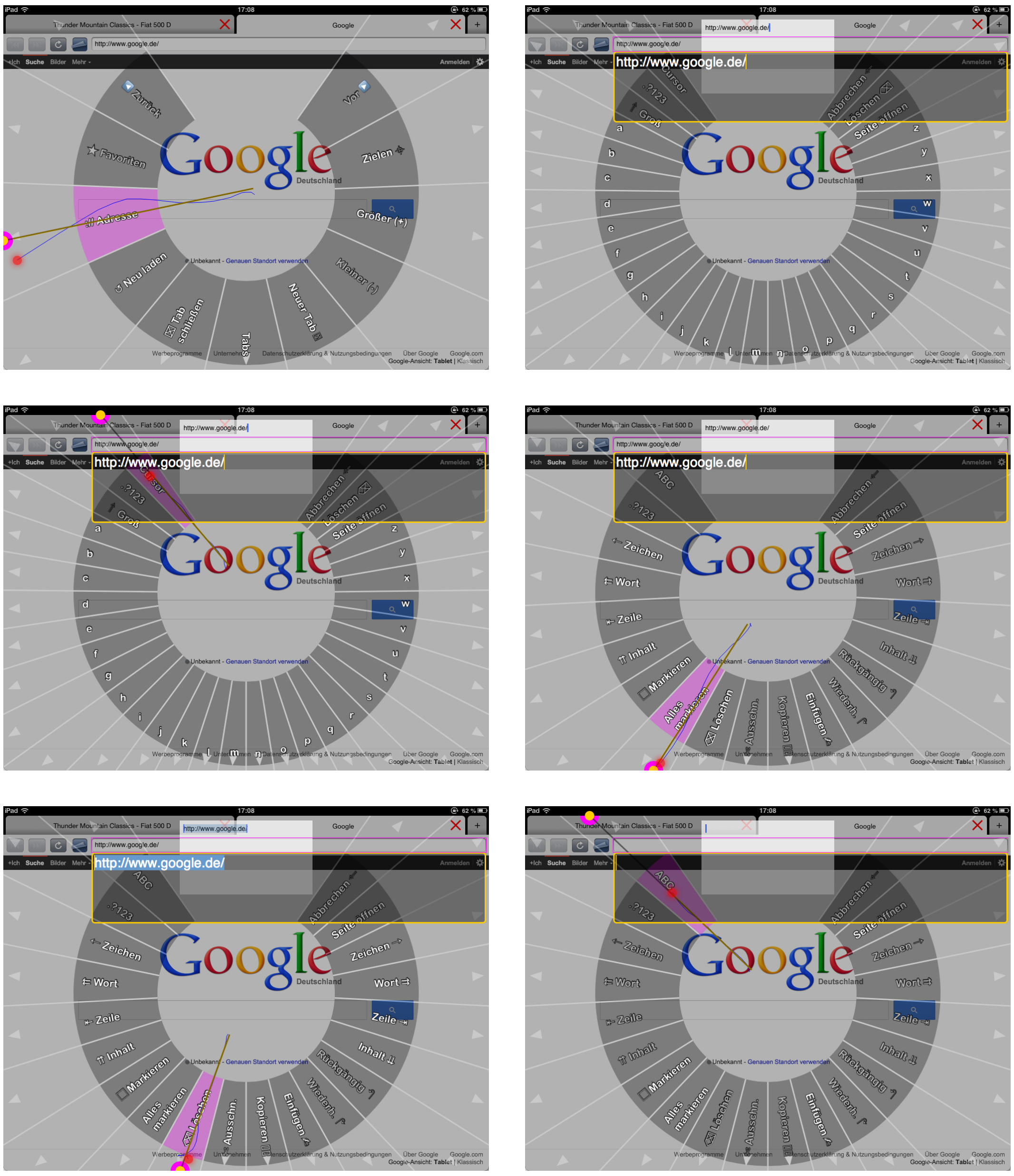


Figure 7.22: Using the cursor menu to clear the address field

"f" when trying to enter the "e", but was able to hit delete very reliably. The delete option, located in the upper right corner, apparently was right on the circle his arm would describe when only using the muscles in the shoulder to perform the gesture, while for the "e", located in the bottom left corner, the user needed to combine various muscles to

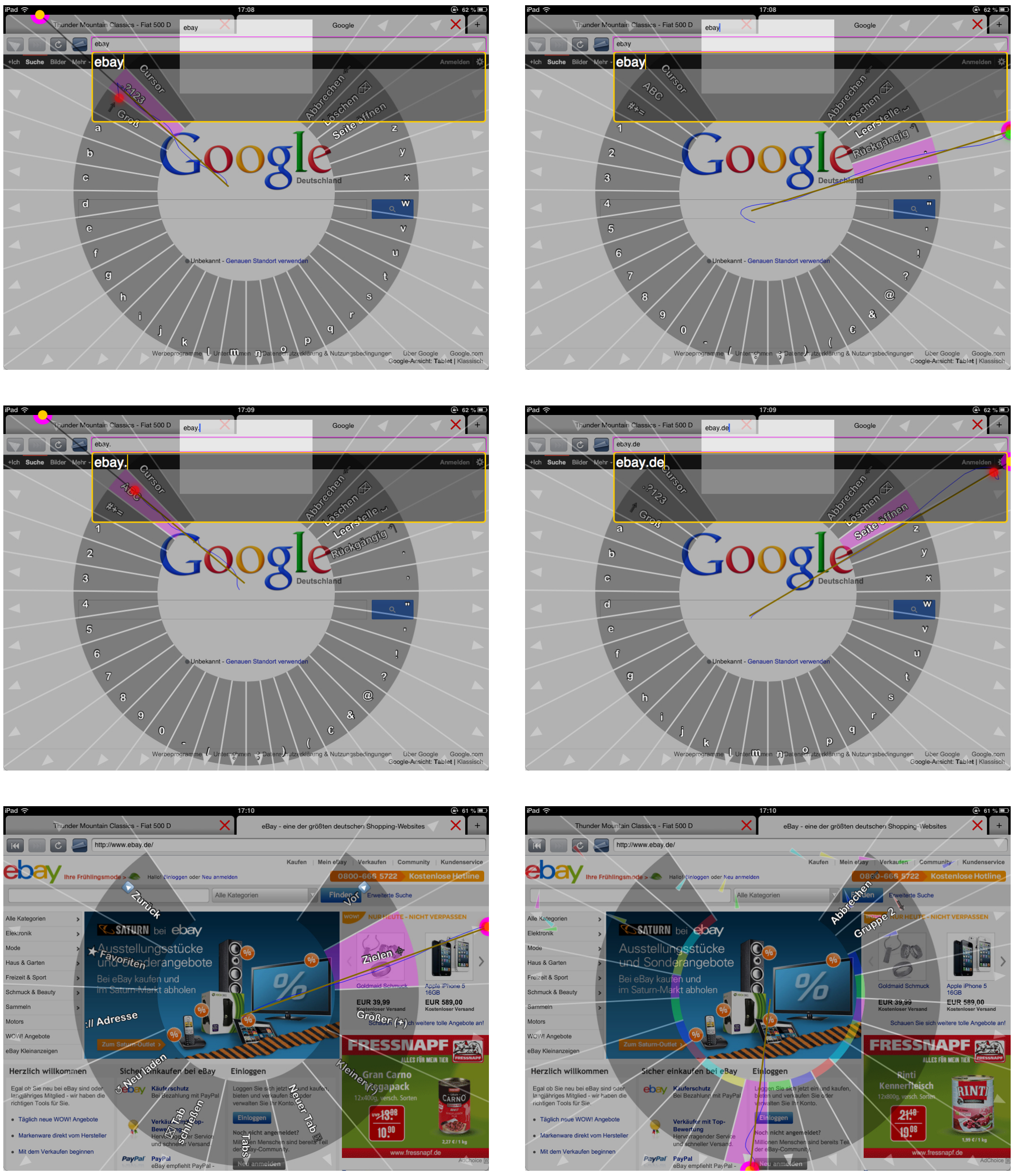


Figure 7.23: Opening a URL and focusing the search field on the page

effect a straight movement in the proper direction.

The user suggested abandoning the circular layout altogether, in favor of a two column, multiple rows layout that would work similar to the double character Swabbing design by offering two options per menu slot.

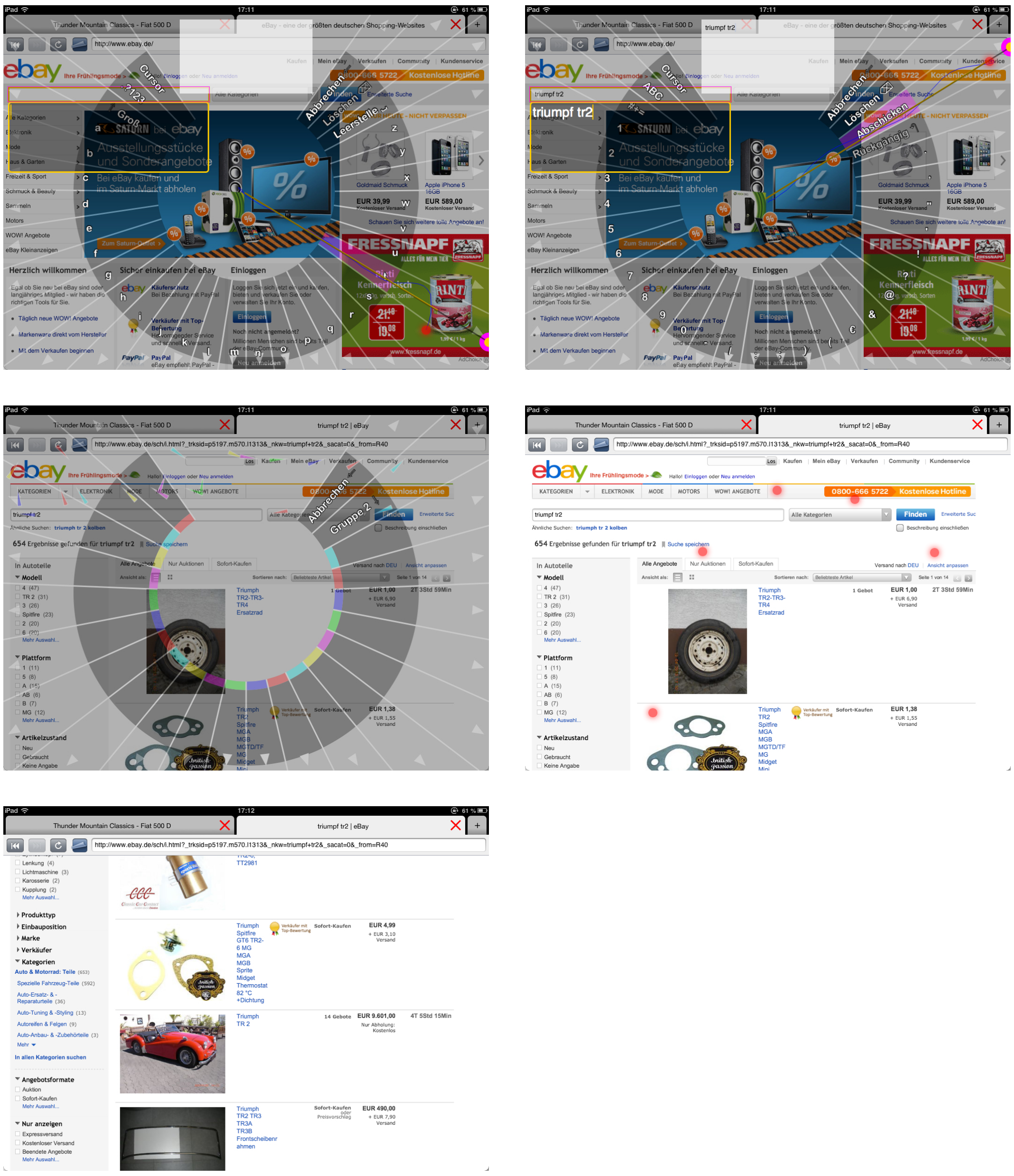


Figure 7.24: Entering a search term, closing the Swabbing overlay and scrolling the page

The user reported having improved his tapping accuracy to 70-80% by not pressing the screen firmly like a hardware button, but as lightly and briefly as possible. When he later demonstrated this technique while filming the motions of his hands, he seemed to employ a yo-yo-like approach, namely almost "throwing" a finger onto the screen while starting to pull back just before actually touching it. In some cases, he would miss the screen entirely this way, but often, this technique allowed him to limit the time during which the finger would actually touch the screen, resulting in better gesture recognition. This approach would likely also reduce the number of accidental double taps and thereby reduce the amount of errors the user has to correct. Looking at the corrected error rate (figure 7.20), there is a sudden drop in the error rate for the native touch keyboard that may be explained by this discovery.

The user also discovered on his own that he could more successfully click links by zooming before clicking. For this, he used the pinch and zoom gesture. He reported that zooming this way felt very uncontrolled, even "nasty" — the tremor symptoms in this case are huge, probably because it required the coordination of many muscles. This could be different for other tremor types than his intention tremor.

He was unaware of the double-tapping gesture for zooming, although he had occasionally used it by accident. However, he could not make sense of the behavior of the iPad in those cases. Nevertheless, after demonstrating the technique to him, he sounded very interested, saying it seemed useful to him.

Another positive discovery the user made was noticing that he could prevent incorrect selections by considering the feedback, deliberately moving in a different direction to then wait until the target point reached the desired option (see figure 7.25).

Some cause for concern is the finding that the user sometimes made mistakes by only checking whether he selected the menu entry he intended to select, but not whether this menu entry would then actually trigger the intended action. Especially for the aiming tasks, this is a problem — for instance, at one point the user was absolutely convinced that he had confirmed a link selection during the aiming task. While he did indeed select the first menu entry, the aiming menu was still in group two. Consequently, a link

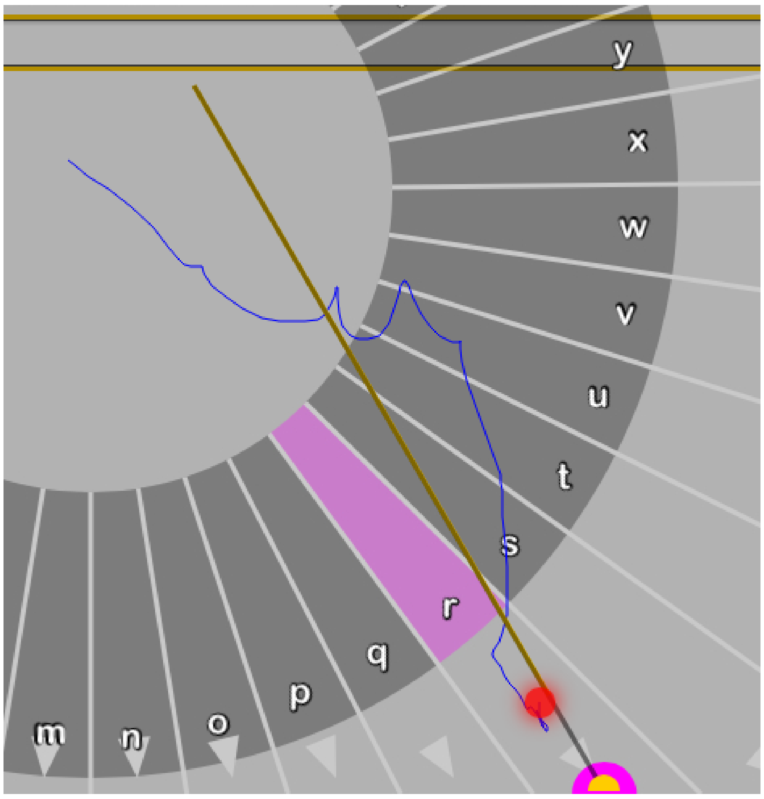


Figure 7.25: Course correction during a Swabbing selection, based on the closed-loop feedback

was selected, not the continue button. Even though incorrect links were marked (see figure 7.9), the user did not notice his mistake, waiting for a new link to be marked by the system, until finally discovering that he could press the continue button. However, he thought that this had been a new round during which, for some reason, no link was shown — rather than realizing that he had never actually ended the previous round in the first place. This problem should be addressed in future work, possibly by extending the closed-loop feedback to also highlight the represented element (i.e., a link on the web page) while the user performs a Swabbing gesture.

7.8 Discussion

The user's high satisfaction with the third condition may be the result of higher input speed, generally becoming more accustomed to Swabbing, or the fact that the main idea for allowing characters to be selected from outside came from the user himself, making him personally involved in the outcome.

Allowing two characters per menu entry seems to have lowered the user's typing speed at first, but this was expected. Not only were the characters in different places than before, he also had to adjust to selecting options from the outside in. However, the user also paused (i.e., did not type for 17 seconds) to remark on how he liked that in the word "attraktiver", the last two pairs of letters (i+v and e+r) were also pairs in the letter menu, easing text entry.

The last finding also indicates that there are three possible sources for the generally higher input speed during the third phase: not every character insertion requires a homing movement, the targets are bigger, allowing the user to be less precise and therefore faster, and general adaptation to Swabbing. The curve suggests that input speed might improve further, possibly due to learning not just the position and direction of individual characters, but even whole sequences of movements for more frequent character combinations.

Typing with Swabbing allows preventing errors in a way not possible with the native keyboard: it is possible to notice one chose the wrong letter (as opposed to having chosen the right one, but not aiming correctly) and still correct (or abort) the gesture before it is final. With the iPad keyboard, the gesture can be aborted by moving far away from the pressed key. Note that Huck [2012] actually implemented his touch keyboard so that the user could first touch the screen and then move towards the intended character.

7.9 Limitations

The main limitation of this study is the single user. Different types of tremor produce problems in different areas, so it is possible that positive results here might be negative for other users. In addition, the user's extensive experience with computers may allow him to understand complex user interfaces better than other users would, possibly masking some existing problems. He also was remarkably tenacious when problems arose, occasionally spending several minutes of trial and error until the system reacted the way he wanted to. Since one motivation for Swabbing in general is that touch screens need to be usable for older users to better cope with cognitive decline, tests with users who show such symptoms might turn out less favorable.

This user also was not interested in more sophisticated web apps, like webmail interfaces or social networking sites. Whether the Swabbing-enabled web browser can be used for more serious tasks, like ordering products in an online shop or doing home banking, remains to be seen.

Another problem is that some of the text input data had to be discarded because of accidental (first condition) or intentional (second and third condition) changes of the cursor position, since the used measures only allow for inserting characters and deleting them from the end of the input. For this reason, the actual error rates for the second condition are unknown.

It would be interesting to see whether users can provide more meaningful ratings of the software after having seen all conditions.

As the learning curves have shown, it may be advisable to let the user grow accustomed to an input technique for more than six sessions. It would also be interesting to see whether the Swabbing condition with one character per menu entry would remain slower than the double letter condition if used last.

Chapter 8

Summary and future work

By making a complete web browser controllable via Swabbing, a lot was achieved.

The closed-loop feedback design is a response to previous work by Hurtmanns [2011] and Huck [2012]. It has been shown to allow users to assess the accuracy of their input while performing the gesture and helps in learning to aim for the edge of the screen instead of the menu labels, if the visual design doesn't achieve this on its own.

Allowing users to enter lowercase letters and special characters is one part of making Swabbing usable in real-world text entry scenarios. The other is controlling a text cursor using Swabbing, including support for selections and access to the clipboard. This allows Swabbing to be used to edit existing text, possibly leading to more interesting research. The user used both capabilities successfully.

One of the most important contributions may be the aiming menu, since it is adaptable to other tapping based interfaces as well. It can augment them without necessarily having to change the underlying application. This could lead to Swabbing being implemented as a general input method on the system level.

This thesis is the first to compare Swabbing with established technology like the touch keyboard on the iPad, revealing that such sophisticated implementations can do surprisingly well even for users with a hand tremor. It might inspire research to directly address the tremor related problems when using the touch keyboard, like filtering double taps, possibly leading to a technique that matches Swabbing's outstanding error rates while surpassing it in term"s of speed. Alternatively, this result can provide a challenge to increase achievable input speeds with Swabbing.

Furthermore, the first longitudinal study using Swabbing provides insight into the learning curve when using swabbing for text entry and aiming tasks. Specifically for aiming tasks, Swabbing provides a reliable and quick way to activate targets on a screen. For known targets such as those in the home menu or for the confirmation step in the aiming task, Swabbing

The double character Swabbing design shows how the previous capacity of Swabbing menus can be significantly increased, though further studies are necessary to show whether the input is equally precise in both directions.

By employing Swabbing in a non-square layout, the need for a useful definition of the target size was brought up, with a suggestion for such a formula. This could eventually help to improve the reliability of Swabbing by providing an objective measure to evaluate Swabbing implementations.

Appendix A

iPad keyboard layouts

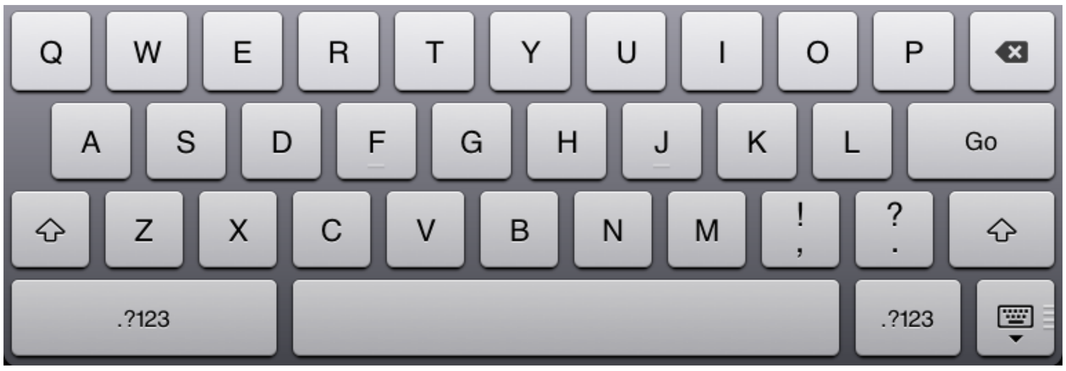


Figure A.1: Letters for plain text

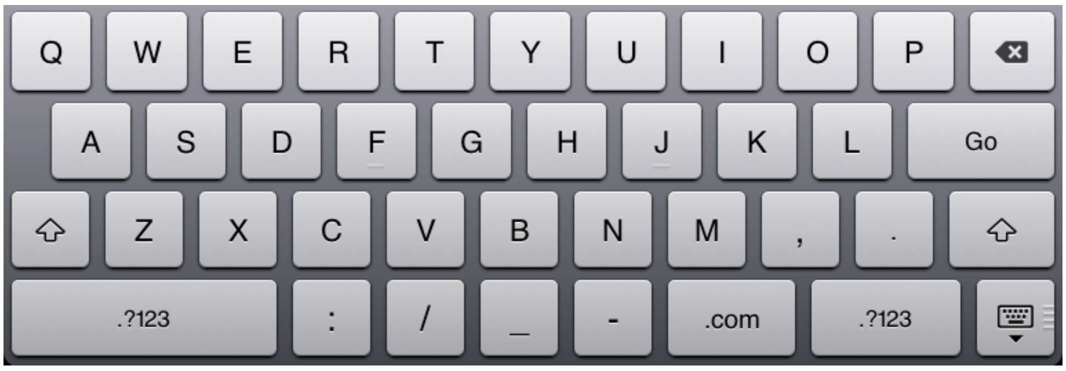


Figure A.2: Letters for URLs

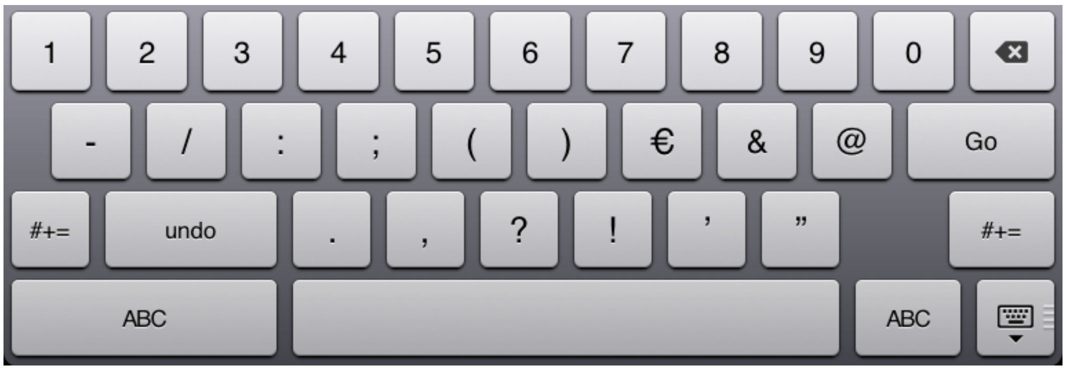


Figure A.3: Special characters for plain text, shift off

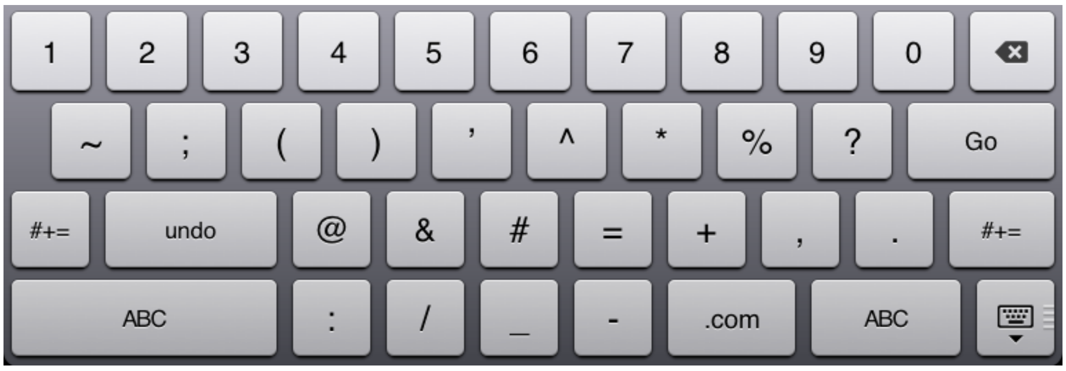


Figure A.4: Special characters for URLs, shift off

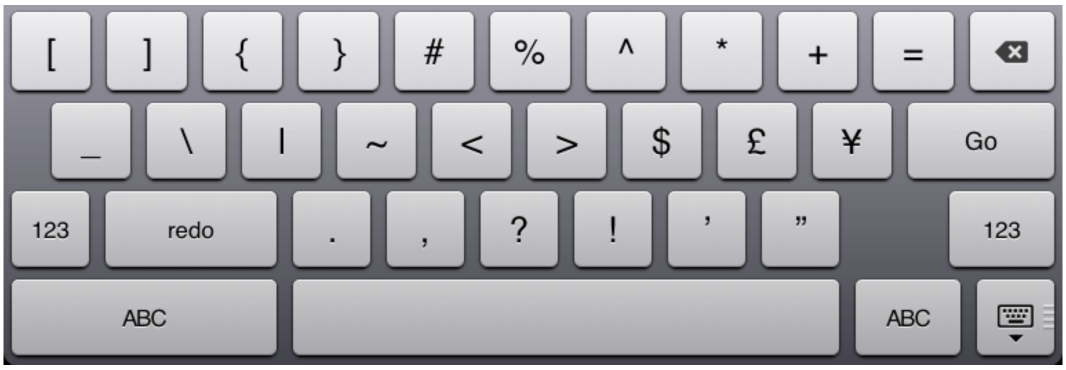


Figure A.5: Special characters for plain text, shift on

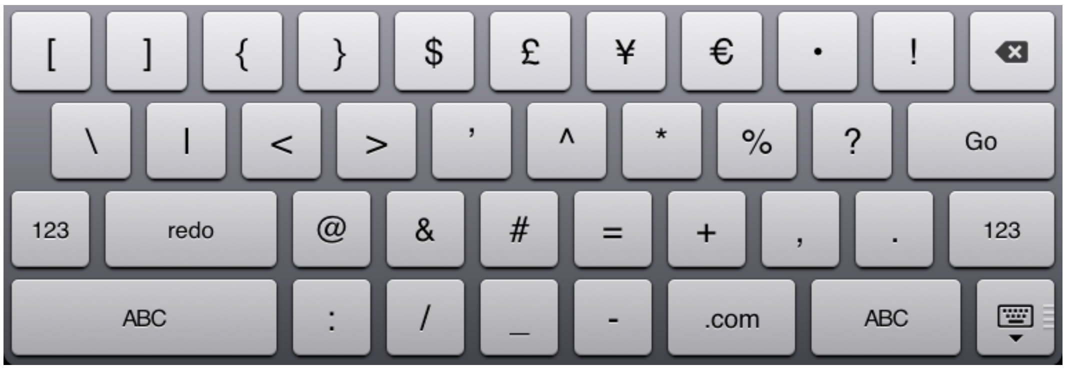


Figure A.6: Special characters for URLs, shift on

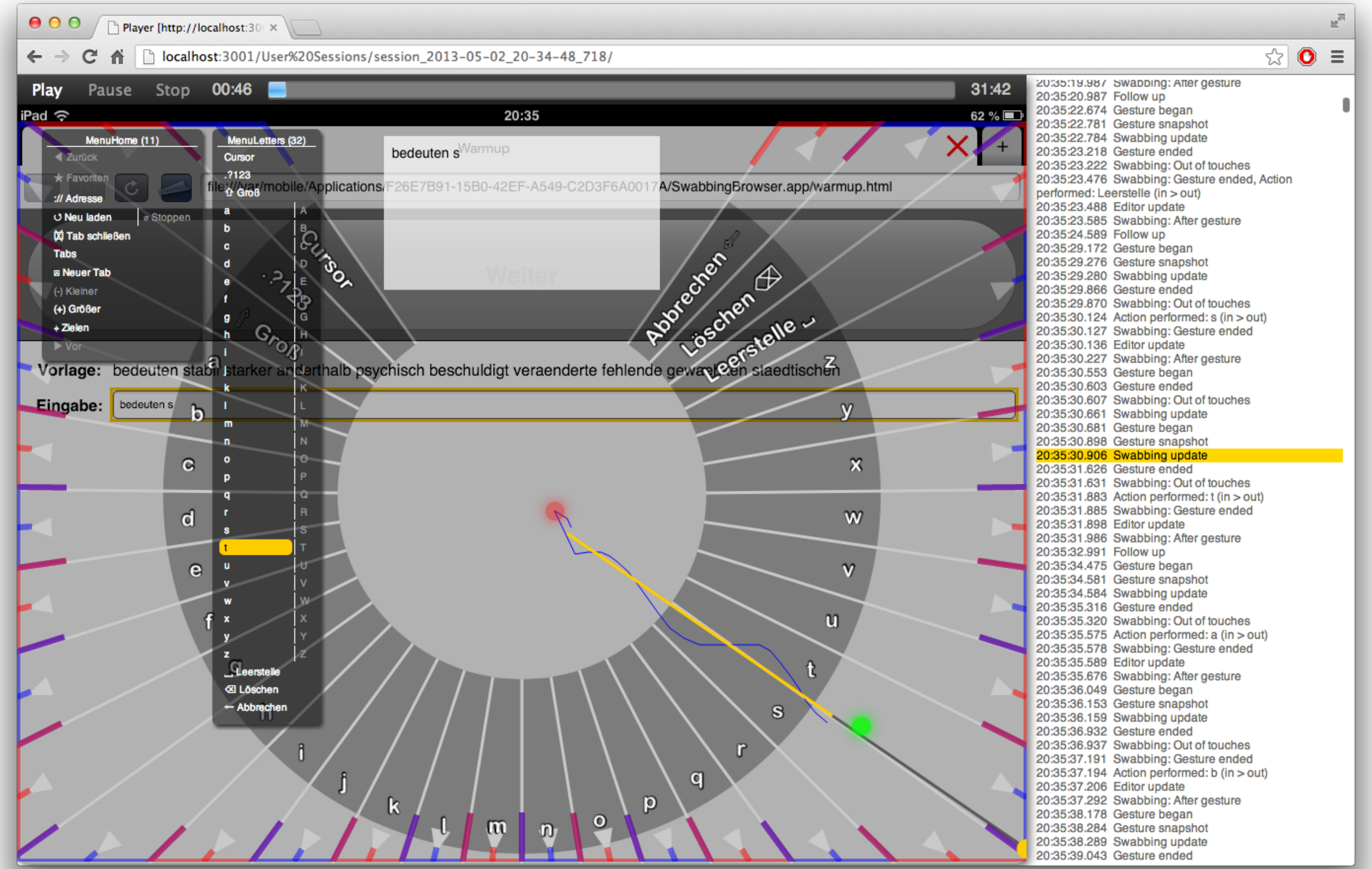
Appendix B

Log file player

To make the recorded data more accessible, a player application was created (see figure B.1). It turns the log file, the screenshots and the audio recording into an augmented video showing the user's session. The player allows jumping to arbitrary points during the session and offers playback control via keyboard controls.

It shows active touches in green, and fades out ended touches after turning them red. It visualizes all the detection results by the Swabbing algorithms and can show the current Swabbing menu hierarchy.

Missing from the screenshot is the player's ability to show the state (value, cursor, selection, location) of whatever text field is focused, be it the URL bar or a text field on web site, regardless of whether it is edited with the native keyboard or via Swabbing. This was useful to validate that text entry by the user is properly captured.


Player [http://localhost:30
localhost:3001/User%20Sessions/session_2013-05-02_20-34-48_718/
Play
Pause
Stop
00:46
31:42
iPad
20:35
62 %
MenuHome (11)
Zurück
Favoriten
:// Adresse
Neu laden
Stoppen
Tab schließen
Tabs
Neuer Tab
(-) Kleiner
(+) Größer
+ Zielen
Vor
MenuLetters (32)
Cursor
.?123
Groß
bedeuten s
Warmup
Weiter
file:///var/mobile/Applications/F26E7B91-15B0-42EF-A549-C2D3F6A0017A/SwabbingBrowser.app/warmup.html
Cursor
.?123
Groß
Abbrechen
Löschen
Leerstelle
Vorlage: bedeuten stabil starker anderthalb psychisch beschuldigt veraenderte fehlende gewarnten staedtischen
Eingabe: bedeuten s
Leerstelle
Löschen
Abbrechen

```
20:35:19.987  Swabbing: After gesture
20:35:20.987  Follow up
20:35:22.674  Gesture began
20:35:22.781  Gesture snapshot
20:35:22.784  Swabbing update
20:35:23.218  Gesture ended
20:35:23.222  Swabbing: Out of touches
20:35:23.476  Swabbing: Gesture ended, Action
performed: Leerstelle (in > out)
20:35:23.488  Editor update
20:35:23.585  Swabbing: After gesture
20:35:24.589  Follow up
20:35:29.172  Gesture began
20:35:29.276  Gesture snapshot
20:35:29.280  Swabbing update
20:35:29.866  Gesture ended
20:35:29.870  Swabbing: Out of touches
20:35:30.124  Action performed: s (in > out)
20:35:30.127  Swabbing: Gesture ended
20:35:30.136  Editor update
20:35:30.227  Swabbing: After gesture
20:35:30.553  Gesture began
20:35:30.603  Gesture ended
20:35:30.607  Swabbing: Out of touches
20:35:30.661  Swabbing update
20:35:30.681  Gesture began
20:35:30.898  Gesture snapshot
20:35:30.906  Swabbing update
20:35:31.626  Gesture ended
20:35:31.631  Swabbing: Out of touches
20:35:31.883  Action performed: t (in > out)
20:35:31.885  Swabbing: Gesture ended
20:35:31.898  Editor update
20:35:31.986  Swabbing: After gesture
20:35:32.991  Follow up
20:35:34.475  Gesture began
20:35:34.581  Gesture snapshot
20:35:34.584  Swabbing update
20:35:35.316  Gesture ended
20:35:35.320  Swabbing: Out of touches
20:35:35.575  Action performed: a (in > out)
20:35:35.578  Swabbing: Gesture ended
20:35:35.589  Editor update
20:35:35.676  Swabbing: After gesture
20:35:36.049  Gesture began
20:35:36.153  Gesture snapshot
20:35:36.159  Swabbing update
20:35:36.932  Gesture ended
20:35:36.937  Swabbing: Out of touches
20:35:37.191  Swabbing: Gesture ended
20:35:37.194  Action performed: b (in > out)
20:35:37.206  Editor update
20:35:37.292  Swabbing: After gesture
20:35:38.178  Gesture began
20:35:38.284  Gesture snapshot
20:35:38.289  Swabbing update
20:35:39.043  Gesture ended
```

Figure B.1: Log file player

Appendix C

Material for the user studies

C.1 User study 1: Link association

Short instructions for preparing the user study (“Preparation”), instructions for performing the study with one participant (“Execution”) and the form used to record the user’s replies (“Survey Response”).

Userstudy: Link Association

Preparation

1. Install the prototype
2. Test the prototype
3. Add bookmarks for the following links:
    1. Galileo Galilei (5)
    2. Galileo Galilei (20)
    3. Jeanne d’Arc (20)
    4. Albert Einstein (5)
    5. Simone de Beauvoir (20)

Userstudy: Link Association

Execution

1. Open three tabs (1: Prototype (**?1,1**), 2: Prototype (no parameters), 3: links.html)
2. Enter fullscreen mode
3. Explain the software: *„This is a prototype of a web browser for the iPad that is designed for people with hand tremor, i.e. trembling hands“*
4. Let the user fill out and sign the consent form
5. Assign a number and variant to the user on both the consent form and the survey response form
6. Explain the Swabbing UI: *„The more your hand is shaking, the harder it is to hit small targets on the screen. As it turns out, your sense of direction when moving longer distances is not affected. The browser uses Swabbing to exploit that fact.*
   You control the browser with a circular menu of commands. To activate a command, you locate the side of the screen that it belongs to, and slide your finger in the direction of that edge.
   When you stop touching the screen and have moved far enough away from the initial touch point, the command associated with the edge of the screen that you pointed to is activated.
   To interact with the web page, you can hide the menu by tapping the screen with three fingers, or in this case by pressing the middle mouse button. The same gesture brings back the menu.“
7. Tell the user to play with the system for up to three minutes
8. Switch to **tab #2** (prototype without any parameters)
9. Tell the user the purpose of this study: *„We also use such a menu to select links and form elements on a web page. The purpose of this study is to find out how users think they have to use the link menu, before and after an explanation.*
   I will now show you two examples of the link menu and I will ask you what you think you have to do to select any of the links.“
10. Open **A1/B2** - Galileo Galilei (**A**: 5 / **B**: 20) and click on Links
11. Ask the user about his mental model (**Mental Model #1**)
12. Open **A2/B1** - Galileo Galilei (**A**: 20 / **B**: 5) and click on Links
13. Ask the user about his mental model (**Mental Model #2**)
14. Open **3** - Jeanne d’Arc (20) and click on Links
15. Ask the user to select two links using the menu - announce, choose, explain (**Selection task #1**)
16. Explain the menu to the user: *„The software scans for clickable elements on the web page by looking at single points on a raster, line by line. For each clickable element, a menu item is shown with a corresponding hint at the first pixel the element was seen at.“*
17. Switch to **tab** #3 (links.html): *„The hint is shaped like the continuation of the menu item's pie segment toward the center of the screen. To disambiguate items with a similar angle, and therefore shape, both the pie segment and the hints are color coded. To find the menu item that belongs to a hit, you have to slide over the screen in the same direction as when starting on the pointy end of the hint and moving towards the round end.“*
18. Switch to **tab #2** (prototype)
19. Ask the user about his mental model (**Mental Model #3**)
20. Open **4** - Albert Einstein (5) and click on Links, then explain: *„There are only five links, so the there is no duplication of colors and you could select the menu item by looking at only the color.“*
21. Open **5** - Simone de Beauvoir (20) and click on Links, then explain: *„Here we have twenty links and therefore every color occurs multiple times. Here you have to look at the shape of the hints first, and only use the color to not choose a neighboring item instead.“*
22. Ask the user to select four links using the menu - announce, choose, explain (**Selection task #2**)

Userstudy: Link Association

Survey Response

Version: **A** / **B** Subject number: ___ Age: ___ Color blind?

Mental Model #1

Purpose of the menu:

How to select link #___:

How does it work:

Mental Model #2

Purpose of the menu:

How to select link #___:

How does it work:

Selection task #1

Mental Model #3

Purpose of the menu:

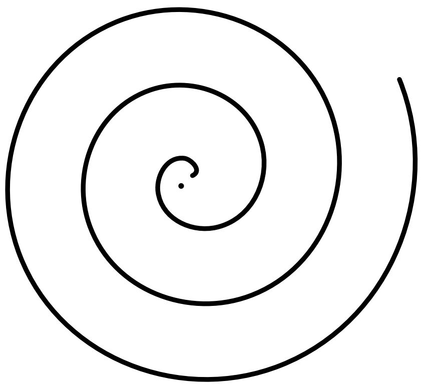
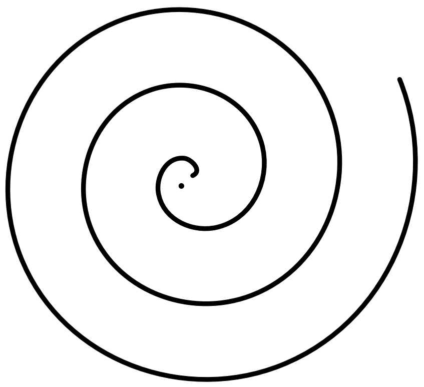
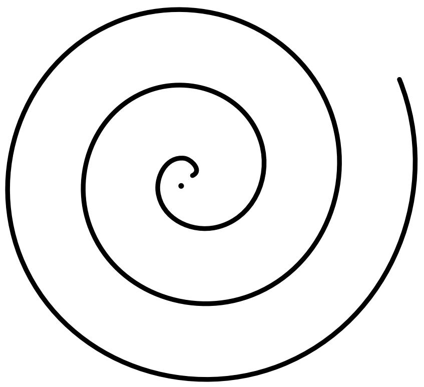
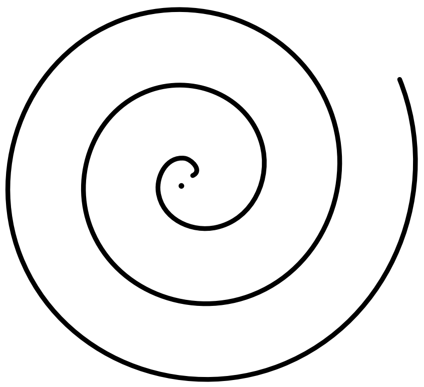
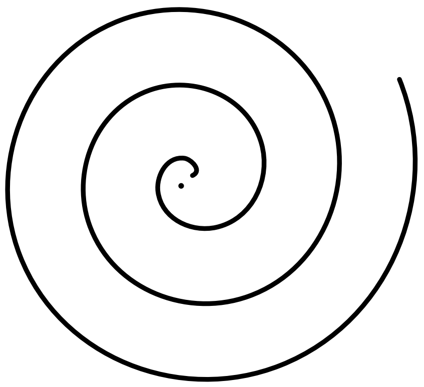
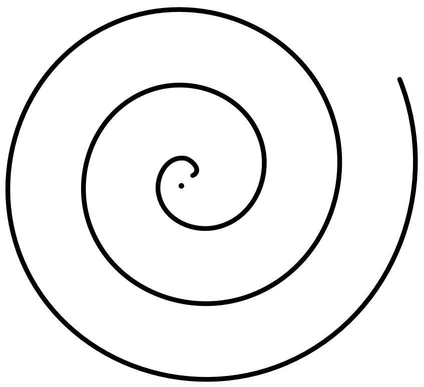
How to select link #___:

How does it work:

Selection task #2

C.2 User study 2: Swabbing feedback

Spiralometry form used for the study.

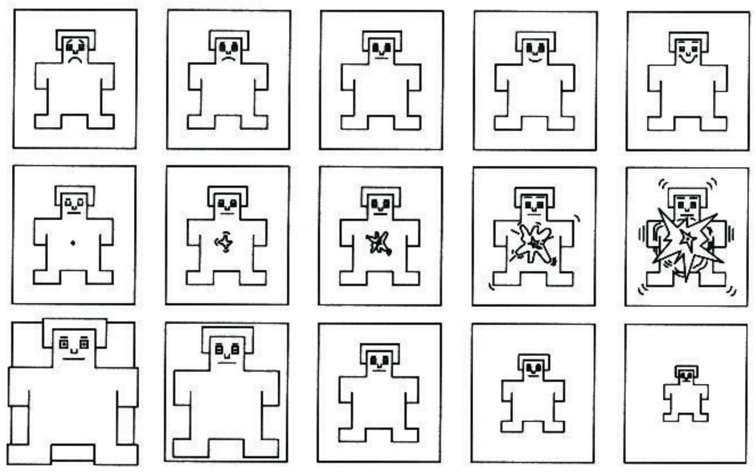
C.3 Longitudinal Study: Swabbing-based web browser

The questionnaire and spiralometry form the user filled out before and after each session.

Vor der Sitzung am ___________

Stimmung

Bitte pro Zeile das Bild markieren, mit dem Sie sich am ehesten identifizieren können:



Körperliches Befinden

Auf einer Skala von **1 bis 10**, wie **müde** fühlen Sie sich (10 = extrem müde)? _____

Spiralzeichnung

Bitte auf einer frischen Kopie der Spiralzeichnungen **das Datum eintragen** und **pro Hand eine Zeichnung** anfertigen.

Diktiergerät starten

Bitte schalten Sie das Diktiergerät ein. Starten Sie die Aufnahme, indem Sie auf den **roten Knopf** drücken. Überprüfen Sie, dass unten im Display des Diktiergeräts **REC** angezeigt wird. Sagen Sie zunächst bitte das **aktuelle Datum und die Uhrzeit**.

Benutzung der Software

Wichtig: Falls Ihnen irgendetwas auffällt (positiv wie negativ), bitte in das Diktiergerät sprechen. Wenn eine Eingabe erfolglos war, reicht auch ein kurzes **Falsch**. Jeder kleinste Kommentar hilft mir!

Bitte nehmen Sie nun das iPad. Um den Startbildschirm anzuzeigen, drücken Sie den Knopf unterhalb des Bildschirms, und schieben Sie den Knopf mit dem Pfeil auf dem Bildschirm nach rechts. Um die Versuchs-Software zu starten, tippen Sie bitte unten in der Mitte des Bildschirms auf den Eintrag **Swabbing**, bis der Bildschirm dunkel wird und die Software startet.

Zunächst bitte die dargestellten Worte eingeben und anschließend auf **Weiter** tippen. Nach fünf Durchläufen mit jeweils zehn Worten sind Sie mit diesem Schritt fertig.

Danach bitte die eingekreisten Links antippen. Sie haben dazu beliebig viele Versuche und beliebig lange Zeit. Wenn Sie den Link erfolgreich angeklickt haben, tippen Sie bitte auf **Weiter**. Nach fünf Sekunden ohne Erfolg haben Sie zusätzlich die Möglichkeit, den aktuellen Link zu überspringen. Es werden insgesamt zehn Links zufällig ausgewählt.

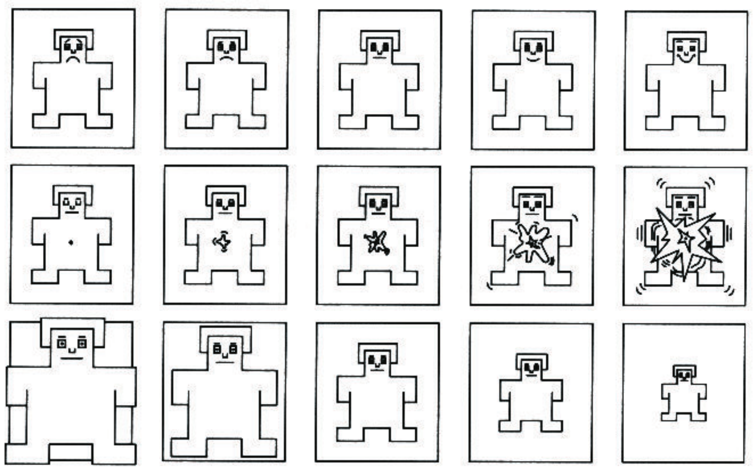
Nun können Sie den Browser frei benutzen! Die gesamte Nutzung der Software (inkl. der Aufgaben zu Beginn) sollte insgesamt ca. eine Stunde dauern.

Nach der Sitzung am ___________

Beenden der Software
Bitte drücken Sie den Knopf unterhalb des Bildschirms, um die Software zu beenden und zum iPad-Startbildschirm zurückzukehren.

Spiralzeichnung
Bitte auf einer frischen Kopie der Spiralzeichnungen **das Datum eintragen** und **pro Hand eine Zeichnung** anfertigen.

Stimmung
Bitte pro Zeile das Bild markieren, mit dem Sie sich am ehesten identifizieren können:



Körperliches Befinden
Auf einer Skala von **1 bis 10**, wie **müde** fühlen Sie sich? _____

Zufriedenheit
Wie zufrieden waren Sie mit der Software?
( ) Stark unzufrieden ( ) Leicht unzufrieden ( ) Neutral ( ) Relativ zufrieden ( ) Sehr zufrieden

Wie präzise fanden Sie Ihre Eingaben?
( ) Sehr unpräzise ( ) Leicht unpräzise ( ) Neutral ( ) Relativ präzise ( ) Sehr präzise

Wie fanden Sie die Ihre Eingabegeschwindigkeit?
( ) Sehr langsam ( ) Etwas langsam ( ) Neutral ( ) Relativ schnell ( ) Sehr schnell

Wie schwer fanden Sie es, Text einzugeben?
( ) Sehr leicht ( ) Relativ leicht ( ) Neutral ( ) Relativ schwer ( ) Sehr schwer

Wie schwer fanden Sie es, Links zu öffnen?
( ) Sehr leicht ( ) Relativ leicht ( ) Neutral ( ) Relativ schwer ( ) Sehr schwer

Wie schwer fanden Sie es ansonsten, den Browser zu steuern?
( ) Sehr leicht ( ) Relativ leicht ( ) Neutral ( ) Relativ schwer ( ) Sehr schwer

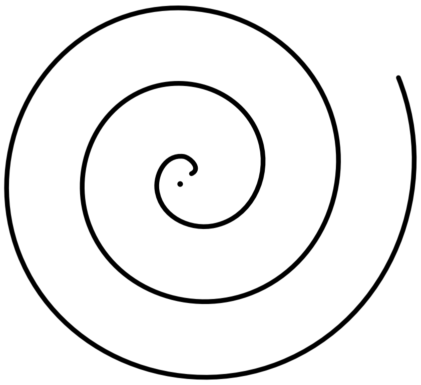
Kommentare
Falls Sie einen der obigen Punkt erleutern möchten oder Verbesserungsvorschläge haben, sprechen Sie diese bitte ins Diktiergerät oder schreiben Sie auf die Rückseite dieses Blattes. Vielen Dank!

Diktiergerät stoppen
Sagen Sie zunächst bitte das **aktuelle Datum und die Uhrzeit**. Danach bitte die Aufnahme stoppen, indem Sie auf den roten Knopf drücken.

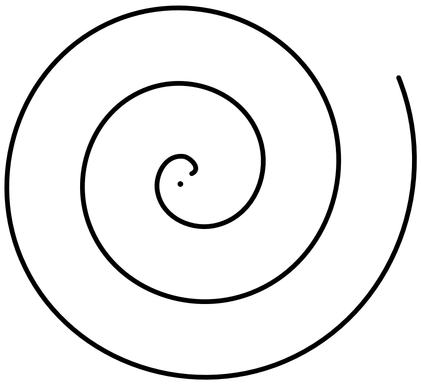
Datum:

Bitte von außen nach innen
einen Strich zwischen die Linien setzen

Vor der Sitzung

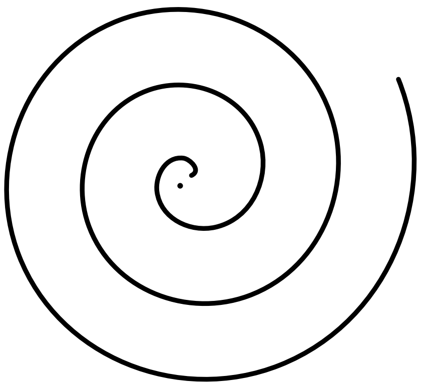


Linke Hand

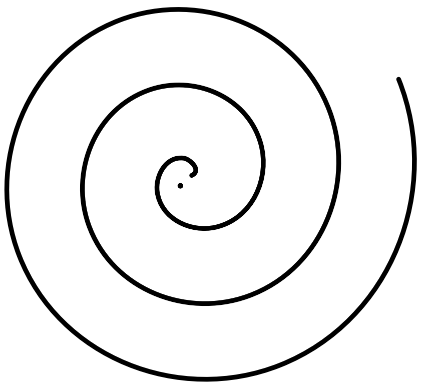


Rechte Hand

Nach der Sitzung



Linke Hand



Rechte Hand

Bibliography

Alexander Clauss. http://www.icab.de/.

James M Dabbs, E-Lee Chang, Rebecca A Strong, and Rhonda Milun. Spatial ability, navigation strategy, and geographic knowledge among men and women. *Evolution and Human Behavior*, 19(2):89–98, 03 1998. URL `http://linkinghub.elsevier.com/retrieve/pii/S1090513897001074?showall=true`.

Leah Findlater, Alex Jansen, Kristen Shinohara, Morgan Dixon, Peter Kamb, Joshua Rakita, and Jacob O. Wobbrock. Enhanced area cursors: reducing fine pointing demands for people with motor impairments. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 153–162, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0271-5. doi: 10.1145/1866029.1866055. URL `http://doi.acm.org/10.1145/1866029.1866055`.

Paul M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6):381–391, 1954.

Eric J. Frett and Kenneth E. Barner. Accuracy and frequency analysis of multitouch interfaces for individuals with parkinsonian and essential hand tremor. In *Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility*, Assets '05, pages 60–67, New York, NY, USA, 2005. ACM. ISBN 1-59593-159-7. doi: 10.1145/1090785.1090799. URL `http://doi.acm.org/10.1145/1090785.1090799`.

Simon Grätzer. http://git.graetzer.org/foxbrowser/.

M. Grimm and Kristian Kroschel. Evaluation of natural emotions using self assessment manikins. In *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on*, pages 381–385, 2005. doi: 10.1109/ASRU.2005.1566530.

Tovi Grossman and Ravin Balakrishnan. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 281–290, New York, NY, USA, 2005. ACM. ISBN 1-58113-998-5. doi: 10.1145/1054972.1055012. URL `http://doi.acm.org/10.1145/1054972.1055012`.

Niels Huck. Touchscreen text input for users with hand tremor. Diploma thesis, RWTH Aachen University, January 2012.

Jan Hurtmanns. Swabbing: touchscreen input for users with hand tremor. Bachelor thesis, RWTH Aachen University, May 2011.

Zhao Xia Jin, Tom Plocher, and Liana Kiff. Touch screen user interfaces for older adults: button size and spacing. In *Proceedings of the 4th international conference on Universal access in human computer interaction: coping with diversity*, UAHCI'07, pages 933–941, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-73278-5. URL `http://dl.acm.org/citation.cfm?id=1766311.1766419`.

Simeon Keates, Faustina Hwang, Patrick Langdon, P. John Clarkson, and Peter Robinson. Cursor measures for motion-impaired computer users. In *Proceedings of the fifth international ACM conference on Assistive technologies*, Assets '02, pages 135–142, New York, NY, USA, 2002. ACM. ISBN 1-58113-464-9. doi: 10.1145/638249.638274. URL `http://doi.acm.org/10.1145/638249.638274`.

Elan D Louis and Joaquim J Ferreira. How common is the most common adult movement disorder? update on the worldwide prevalence of essential tremor. *Mov Disord*, 25 (5):534–541, Apr 2010. ISSN 1531-8257 (Electronic); 0885-3185 (Linking). doi: 10.1002/mds.22838.

I Scott MacKenzie and Kumiko Tanaka-Ishii. *Text entry systems: Mobility, accessibility, universality*. Morgan Kaufmann, 2010.

Anne Collins McLaughlin, Wendy A. Rogers, and Arthur D. Fisk. Using direct and indirect input devices: Attention demands and age-related differences. *ACM Trans. Comput.-Hum. Interact.*, 16(1):2:1–2:15, April 2009. ISSN 1073-0516. doi: 10.1145/1502800.1502802. URL `http://doi.acm.org/10.1145/1502800.1502802`.

Alexander Mertens, Nicole Jochems, Christopher M. Schlick, Daniel Dünnebacke, and Jan Henrik Dornberg. Design pattern trabing: touchscreen-based input technique for people affected by intention tremor. In *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems*, EICS '10, pages 267–272, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0083-4. doi: 10.1145/1822018.1822060. URL `http://doi.acm.org/10.1145/1822018.1822060`.

Tomer Moscovich. Contact area interaction with sliding widgets. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, pages 13–22, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-745-5. doi: 10.1145/1622176.1622181. URL `http://doi.acm.org/10.1145/1622176.1622181`.

Timothy A. Salthouse. When does age-related cognitive decline begin? *Neurobiology of Aging*, 30(4):507 – 514, 2009. ISSN 0197-4580. doi: 10.1016/j.neurobiolaging.2008.09.023. URL `http://www.sciencedirect.com/science/article/pii/S0197458009000219`.

Chat Wacharamanotham, Jan Hurtmanns, Alexander Mertens, Martin Kronenbuerger, Christopher Schlick, and Jan Borchers. Evaluating swabbing: a touchscreen input method for elderly users with tremor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 623–626, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9. doi: 10.1145/1978942.1979031. URL `http://doi.acm.org/10.1145/1978942.1979031`.

Jacob O. Wobbrock and Krzysztof Z. Gajos. A comparison of area pointing and goal crossing for people with and without motor impairments. In *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, Assets '07, pages 3–10, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-573-1. doi: 10.1145/1296843.1296847. URL `http://doi.acm.org/10.1145/1296843.1296847`.

Jacob O. Wobbrock and Brad A. Myers. Analyzing the input stream for character- level errors in unconstrained text entry evaluations. *ACM Trans. Comput.-Hum. Interact.*, 13(4):458–489, December 2006. ISSN 1073-0516. doi: 10.1145/1188816.1188819. URL `http://doi.acm.org/10.1145/1188816.1188819`.

Aileen Worden, Nef Walker, Krishna Bharat, and Scott Hudson. Making computers easier for older adults to use: area cursors and sticky icons. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, CHI '97, pages 266–271, New York, NY, USA, 1997. ACM. ISBN 0-89791-802-9. doi: 10.1145/258549.258724. URL `http://doi.acm.org/10.1145/258549.258724`.

Typeset June 6, 2013