Nicht-visuelle Interaktion bei Smartphones mit Touchscreen

Josef Hofbauer

DIPLOMARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Juli 2011

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Hagenberg, am 18. Juni 2011

Josef Hofbauer

Inhaltsverzeichnis

Kurzfassung

Diese Arbeit untersucht Techniken für nicht-visuelle Interaktion bei Smartphones mit Touchscreen. Es wird ein kurzer Überblick über die Interaktionsmöglichkeiten mit Smartphones und Touchscreens gegeben. Neben der Bedeutung von crossmodalem Feedback wird auch auf sehbehinderte Menschen und deren Schwierigkeiten im Umgang mit digitalen Geräten im Alltag eingegangen. Relevante existierende Ansätze für Interaktion auf nicht-visueller Basis werden aufgeführt. Diese Methoden werden miteinander verglichen und evaluiert, woraus sich mehrere Richtlinien und Vorschläge für das Design eines nicht-visuellen Interfaces ergeben. Ein auf diesen Grundlagen entwickelter Prototyp in Form einer Smartphone-Applikation (*EyesFree-NotePad*) wird vorgestellt. Dieser Prototyp bietet nicht-visuellen Zugang zu Bildschirmelementen. Durch die Möglichkeit der Exploration des Bildschirminhalts kann sowohl durch eine Anwendung navigiert, als auch Text eingegeben werden, ohne dabei den Blick auf das Display richten zu müssen.

Abstract

The present work describes an interface for eyes-free interaction on touchscreen based smartphones. An overview of capabilities of interacting with smartphones and touchscreens is presented, including relevant insights from the fields of crossmodal feedback and visually impaired people dealing with digital devices in daily life. A critical review of relevant existing approaches is given. These methods are evaluated and contrasted with each other, resulting in a number of conclusions about guidelines and the design of an interface for eyes-free interaction. A prototypical application (*EyesFree-NotePad*) that is based on these principles is presented. *EyesFree-NotePad* provides non-visual access to screen elements. This includes exploring the display content and navigating through a menu as well as entering text without the need to look at the touchscreen.

Kapitel 1

Einleitung

1.1 Motivation und Problemstellung

Bei der Verbreitung von Smartphones kann in den letzten Jahren ein kontinuierlicher Anstieg verzeichnet werden. Spätestens seit dem erfolgreichen Einsatz von berührungsempfindlichen Displays – bestes Beispiel ist das iPhone von Apple – erfreut sich das Smartphone zunehmender Beliebtheit. Die Touchscreens bringen im Vergleich zu herkömmlichen Eingabesystemen basierend auf Tasten auch mehrere Vorteile mit sich. So kann der benötigte Platz einer physischen Tastatur eingespart und für ein größeres Display verwendet werden. Je nach Anwendungsfall kann dadurch ein individuelles User-Interface dargestellt werden, wie zum Beispiel eine virtuelle Tastatur. Zudem erleichtert ein berührungsempfindliches Display durch direkte Auswahlmöglichkeit von angezeigten Elementen vor allem unerfahrenen Benutzern die Bedienung von technischen Geräten im Alltag. Es handelt sich dementsprechend um eine neue Art der Interaktion, die flexiblere Formen der Navigation und Texteingabe ermöglicht [8].

Doch gerade der Aspekt, dass der Touchscreen sowohl als Ausgabe- als auch Eingabemedium herangezogen wird, bringt auch Nachteile mit sich. Information wird hauptsächlich visuell dargestellt und bedingt bei einer Interaktion den Blick auf das Display. Die Bedienbarkeit ist somit in Situationen, welche erhöhte Aufmerksamkeit für die Umgebung erfordern, sehr eingeschränkt. Analog zu dieser situationsbezogenen Einschränkung [15] ergeben sich auch durch körperliche Einschränkungen in Form von Sehbehinderungen Schwierigkeiten bei der Interaktion mit Touchscreens.

Personen mit dieser Art Handicap haben bei Verwendung von auf Touchscreen basierenden Geräten weitreichende Probleme zu bewältigen. Gerade die Besonderheit, dass herkömmliche Tasten auch „blind“ ertastbar sind, geht durch den Einsatz von Touchscreens verloren. Interaktionselemente am Display sind im Regelfall nur optisch wahrzunehmen. Das zwingt sehbehinderte Menschen sich entweder den Geräten anzupassen und diese eventuell selbst-

ständig zu adaptieren [14], oder zu versuchen, ihnen aus dem Weg zu gehen. Zweiteres ist jedoch kaum mehr möglich, da schon jetzt viele Geräte (Bankomat, Fahrkartenautomat oder auch einfache Infostände) im Alltag nur mehr per Touchscreen bedient werden können [21]. Für Menschen mit beeinträchtigter Sehkraft stellt ein berührungsempfindliches Display so gesehen keine Erleichterung dar. Um eine Bedienung von Geräten mit Touchscreens auch auf nicht-visueller Basis zu ermöglichen, bedarf es somit anderer bzw. zusätzlicher Sinnesreize und neuer Wege der Interaktion.

1.2 Zielsetzung

In dieser Arbeit sollen Wege gefunden werden um Interaktionen mit Touchscreens auf Smartphones in und für Situationen zu ermöglichen, in denen nicht oder nur eingeschränkt visueller Kontakt mit dem Mobilgerät hergestellt werden kann. Im Hintergrund sind hierfür besonders folgende zwei Aspekte ausschlaggebend:

- Für Menschen mit Beeinträchtigung der Sehkraft kann die Bedienung von stark auf visuell ausgerichtete Interfaces erleichtert werden.
- In Situationen, welche erhöhte Aufmerksamkeit erfordern, muss der Blick nicht notwendigerweise von der Umgebung abgewendet und auf das Gerät gerichtet werden.

Um dies zu erreichen, soll mit dem begleitenden Thesis-Project ein Prototyp als Umsetzungsvariante für „blinde“ Bedienbarkeit von Applikationen auf Touchscreens zur Evaluierung und Gegenüberstellung zu anderen nichtvisuellen Interaktionstechniken entwickelt werden.

Diverse Interaktionstechniken sollen anhand verschiedener Kriterien untersucht und miteinander verglichen werden. Da ein Smartphone verschiedene Interaktionskanäle wie Kamera, Display, Lautsprecher, Mikrofon, Kompass, Beschleunigungssensor etc. bietet, ergeben sich zahlreiche Möglichkeiten für eine (blinde) Interaktion. Jede dieser Optionen bringt individuelle Eigenschaften und somit auch Vor- und Nachteile mit sich.

Letztendlich soll erreicht werden, das Interface dahingehend zu erweitern, dass erforderliche Blicke auf das Display deutlich reduziert und im besten Falle hinfällig werden. Zu unterscheiden ist hierbei das Navigieren durch ein Menü und das Eingeben von Text. Beides sollte möglich sein um eine Applikation auf nicht-visueller Basis bedienen zu können.

Die Eingabegeschwindigkeit kann hierbei eine niedrigere Priorität erfahren als der weitestgehend geringe Blickkontakt mit dem Display. Trotzdem sollte die Bedienung zügig und intuitiv erfolgen können.

1.3 Gliederung der Arbeit

Den zentralen Gegenstand dieser Arbeit bildet die Entwicklung eines Prototypen in Form einer Applikation für ein Smartphone mit Touchscreen. Aufbauend darauf werden nach der Einführung in Kapitel 1 die Grundlagen bezüglich Smartphones und deren Möglichkeiten der Interaktion mit allgemeinen Informationen in Kapitel 2 behandelt. Kapitel 3 beschäftigt sich mit verschiedenen Interaktiontechniken und Umsetzungen für nicht-visuelle Interaktion und geht kurz auf dahingehend relevante Varianten ein. Die Basis für eine Evaluation wird in Kapitel 4 anhand mehrerer relevanter Kriterien gebildet, wie sie auch weiterführend für die Arbeit wichtig ist.

In Kapitel 5 und 6 wird schließlich der Hauptteil der Arbeit, die Entwicklung des Prototypen für nicht-visuelle Interaktion, vorgestellt. Im Anschluss daran beschäftigt sich Kapitel 7 mit der Diskussion bezüglich der Interaktionsmöglichkeiten mit dem Prototypen. Abgerundet wird die Arbeit im letzten Kapitel mit Schlussbemerkungen und kurzem Ausblick.

Kapitel 2

Eigenschaften von Smartphones

2.1 Das Smartphone

Der Begriff *Smartphone* hat sich in den letzten Jahren durchgesetzt, um eine bestimmte Gruppe von Handys zu bezeichnen. Das *Smart* in diesem Wort hat dabei nichts mit einem „intelligenten“ Mobilgerät zu tun, sondern soll nur auf den erweiterten Funktionsumfang von diesen Geräten gegenüber herkömmlichen Mobiltelefonen verweisen [7]. Am einfachsten lässt sich das Smartphone als eine Mischung von klassischem Mobiltelefon und einem PDA[1] beschreiben, das sich durch folgende Merkmale auszeichnet:[2]

- Durch vergleichsweise große und hochauflösende Bildschirme, alphanumerische Tastaturen oder Touchscreens sind Smartphones für eine komfortable Bedienung verschiedenster Anwendungen konstruiert.
- Der Einsatz von Betriebssystemen mit offener API ermöglicht das nachträgliche Installieren weiterer Applikationen. Der Benutzer hat so die Möglichkeit, den Funktionsumfang persönlich zu individualisieren.
- Oft verfügen Smartphones über eine große Anzahl sich ergänzender Aktoren und Sensoren. Daraus bietet sich sowohl eine breitere Palette nützlicher Anwendungen als auch unterschiedliche Kanäle um mit dem Gerät zu interagieren.

Eine weitere nennenswerte Eigenschaft von Smartphones ist die Möglichkeit eine Verbindung zum Internet herzustellen. Dies kann sowohl über den Mobilfunkbetreiber, oder häufig auch über ein WLAN-Netzwerk erfolgen. Dadurch können mit dieser Art Mobiltelefon neben konventionellem Telefonieren und Senden von SMS zum Beispiel auch Emails abgerufen werden.[2]

[1] Abkürzung für *Personal Digital Assistant*, ein kleiner tragbarer Computer.

[2] http://de.wikipedia.org/w/index.php?title=Smartphone&oldid=89322860, Kopie auf CD-ROM unter /quellen/W2_sphonewiki/.

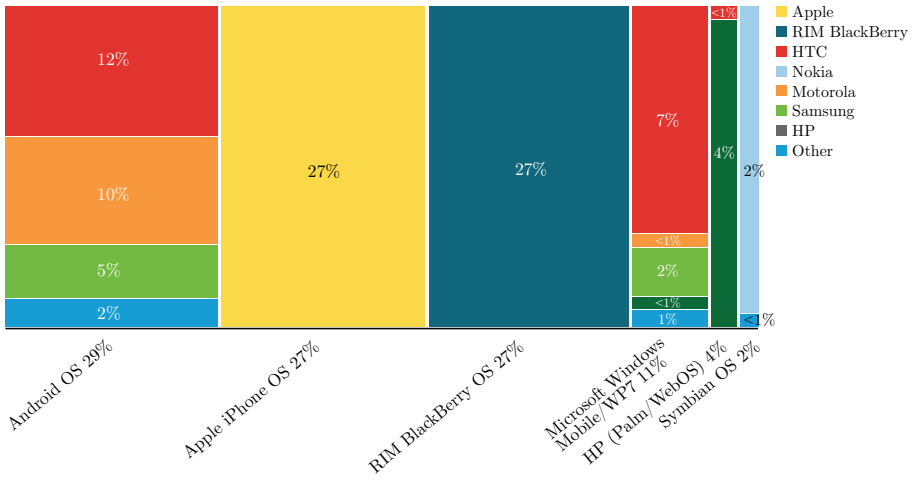


Abbildung 2.1: Marktanteile von Smartphone-Betriebssystemen mit Aufteilung nach Gerätehersteller im März 2011.

Eine wesentliche Grundlage für die Nutzung der genannten Funktionen von Smartphones bildet dabei das auf dem Gerät installierte Betriebssystem. Besonders für die zusätzlich installierbaren Applikationen ist dieses ausschlaggebend. Als die populärsten Betriebssysteme sind hier *iOS* von Apple, *Android* von Google, und *BlackBerry* von RIM (Research in Motion) zu nennen. Wie eine aktuelle Verbreitung dieser Betriebssysteme auf Smartphones verschiedener Hersteller im März 2011 aussieht, zeigt die Statistik in Abb. 2.1 nach *The Nielson Company* für den Markt der USA.[3]

2.1.1 Ausführungen

Auch wenn sich die unterschiedlichen Smartphone-Modelle ihre zuvor genannten Funktionen teilen, können die Geräte grob untereinander abgegrenzt werden. Eine Einteilung ist hier beispielsweise in Tasten basierte Mobiltelefone möglich. Diese verfügen meist über eine QWERTZ- bzw. QWERTY-Tastatur und werden auch als Q-Smartphones bezeichnet.[2] Abb. 2.2 (a) zeigt diese Bauform mit einer QWERTY-Tastatur auf der Frontseite. Bei anderen Ausführungen dieser Bauform, wie in Abb. 2.2 (b), kann die Tastatur entweder durch einen Klapp- oder Schiebemechanismus verborgen werden. Ergänzend zu diesen Tastaturen kann ein Smartphone auch mit Touchscreen ausgestattet sein, per Bedienung mit dem Finger bzw. zusätzlichem Eingabestift, oder es beschränkt sich ausschließlich auf den Einsatz eines Touchscreens ohne ergänzende Tastatur. Letztere werden bei Fingerbedienung auch

[3] http://blog.nielsen.com/nielsenwire/online_mobile/who-is-winning-the-u-s-smartphone-battle/, Kopie auf CD-ROM unter /quellen/W22_nielson/.

(a) (b) (c)

Abbildung 2.2: Das *Blackberry* von RIM mit QWERTY-Tastatur auf der Gerätefront (a). *HTC Desire Z* mit Touchscreen und Schiebetastatur (b). Touchphone *iPhone 4* von Apple (c).

als *Touchphones* bezeichnet, vgl. Abb. 2.2 (c). Gegebenenfalls sind Smartphones mit Touchscreen ergänzend auch mit Softkeys[4] versehen.

2.1.2 Interaktionsfähigkeiten und -kanäle

Da moderne Smartphones mittlerweile über eine große Zahl an Aktoren und Sensoren verfügen, kann für das Interfacedesign aus einem großen Pool von Interaktionsmöglichkeiten geschöpft werden. Je nach Modell können diese variieren und sind in den jeweiligen technischen Spezifikationen nachzulesen. Als Zu- oder Ersatz für herkömmliche Tasten kann mit dem Smartphone dadurch auf flexible und innovative Weise interagiert werden.

Sehen

Durch die standardmäßig integrierte Kamera erlangt das Gerät die Fähigkeit zu „sehen“. Insbesondere für Anwendungen basierend auf *Augmented Reality* ist die visuelle Wahrnehmung der Umgebung unverzichtbar. Manche Smartphones werden nicht nur mit *einer* Kamera ausgestattet, sondern erhalten eine an der Rückseite des Gerätes und eine zweite, meist mit niedriger auflösendem Sensorchip auf der Front.

[4]Bezeichnung für eine Taste, die eine der Bildschirmanzeige entsprechende Funktion ausführt, http://de.wikipedia.org/w/index.php?title=Softkey&oldid=55475490, Kopie auf CD-ROM unter /quellen/W3_softkey/.

Hören und Sprechen

Die Befähigung „hören“ und „sprechen“ zu können ist schon bei einem herkömmlichen Telefon gegeben. Durch eingebaute Mikrofone und Lautsprecher kann ein Smartphone also auch akustische Signale wahrnehmen, verarbeiten und wiedergeben. Das bildet die Grundlage der Spracherkennung und -steuerung eines Smartphones durch verschiedene Anwendungen und des Wiedergebens von Text durch Sprachsynthese.

Berührungsempflindlichkeit

In diesen Bereich fällt einerseits das berührungsempfindliche Display, als auch die Fertigkeit den Tastsinn des Nutzers mit dem integrierten Vibrationsmotor zu „reizen“. Dieser Motor spielt, durch die Fähigkeit spürbare Vibrationen zu erzeugen, im weiteren Verlauf der Arbeit noch eine wichtige Rolle für nicht-visuelle Interaktion. Die nähere Beschreibung zu Touchscreens wird in einem der folgenden Abschnitte behandelt.

Bewegungswahrnehmung

Mit Hilfe von Bewegungssensoren (Beschleunigungssensoren) sind Smartphones schon in der Lage die eigenen Bewegungen im Raum zu erfahren. Bekannt geworden sind diese Art Sensoren insbesondere durch deren Einsatz in Wii-Fernbedienungen, was das Spielerlebnis revolutioniert hat.[5] Auch beim Smartphone ist dies bereits ein essentieller Bestandteil, da dadurch zum Beispiel die Bedienungsoberfläche je nach Ausrichtung des Handys in den *Landscape-* oder *Portrait-Modus* wechseln kann.

Positions- und Richtungsbestimmung

Neben der relativen Bewegungswahrnemung im Raum, kann ein Smartphone mittels GPS oder der Mobilfunkmasten auch dessen absolute Position feststellen. In Kombination mit einem digitalen Kompass ist dann zusätzlich auch die Ausrichtung des Mobilgerätes erfassbar. In Navigationsgeräten ist GPS zur Positionsbestimmung unverzichtbar, und da auch Smartphones mittels eigener Anwendungen zum Navigieren benutzt werden können, spielt dieser Sensor auch hier eine bedeutende Rolle.

Umgebungslicht- und Näherungssensor

Die Sensoren zur Bestimmung des Umgebungslichtes und des Abstands von Objekten zum Handy sind weitere Beispiele zur Erleichterung der Interaktion mit dem Gerät. So können beispielsweise die Eingabefunktion ebenso wie

[5] http://www.eurogamer.net/articles/news160905revolutioncontroller, Kopie auf CD-ROM unter /quellen/W4_revcontr/.

die Bildschirmbeleuchtung ausgeschaltet werden, wenn das Smartphone an das Ohr gehalten wird. Weiters kann durch den Helligkeitssensor die Bildschirmhelligkeit an die Lichtverhältnisse der Umgebung angepasst werden.[6]

2.2 Der Touchscreen

Ein Touchscreen ist ein intuitives Eingabemedium, welches das Potential mit sich bringt, dem Benutzer digitale Geräte ohne aufwendige Einarbeitungsphase dienstbar zu machen. Das Funktionsprinzip der Touchscreens ist jedoch nicht einheitlich beschreibbar. Die Entwicklung von Touchscreens hat bis heute verschiedene technische Umsetzungen hervorgebracht, die alle spezifische Vor- und Nachteile haben und für bestimmte Einsatzgebiete mehr oder weniger gut geeignet sind. Unter anderem bestimmt die gewählte Technik schließlich auch die Eingabe über Finger oder Stift und auch die Fähigkeit mehrere gleichzeitige Berührungen (Multitouch) verarbeiten zu können.

2.2.1 Touchscreen-Technik

Ein kurzer Einblick über verschiedene Umsetzungsvarianten und die Technik dahinter kann nach [16, 29] wie im Folgenden gegeben werden. Das für Mobilgeräte nennenswerteste Funktionsprinzip ist die kapazitive Variante, gefolgt von resistiver Technik. Andere Techniken kommen entsprechend ihrer Eigenschaften vorwiegend in anderen Gebieten zum Einsatz.

Kapazitiv

Wie in Abb. 2.3 dargestellt, arbeitet ein kapazitiver Touchscreen mit einem über eine spezielle Beschichtung möglichst gleichmäßig verteilten elektrischen Feld. In den vier Ecken des Bildschirms sind Elektroden angebracht, welche dieses schwache Feld erzeugen. Wird die Oberfläche nun berührt, so verändert sich das generierte Feld. Diese Änderung wird von einem Controller ausgewertet, wodurch sich schließlich die Position der Berührung bestimmen lässt.

Um das elektrische Feld mit einer Berührung zu beeinflussen, bedarf es entweder einem bloßen Finger oder leitfähigen Eingabestiften. Ein kapazitiver Touchscreen kann also nicht mit herkömmlichen Stiften oder Handschuhen bedient werden. Das zieht den Effekt nach sich, dass Menschen mit Handprothesen diese Art Touchscreen nicht nutzen können.

Trotz des hohen Preises für einen Touchscreen dieser Bauform finden sie sich im Großteil der aktuell vertriebenen Mobilgeräte [29]. Apple, HTC, Samsung und auch weitere namhafte Hersteller setzen auf diese Technik.

[6] http://de.wikipedia.org/w/index.php?title=Apple_iPhone&oldid=89399503, Kopie auf CD-ROM unter /quellen/W6_iphone/.

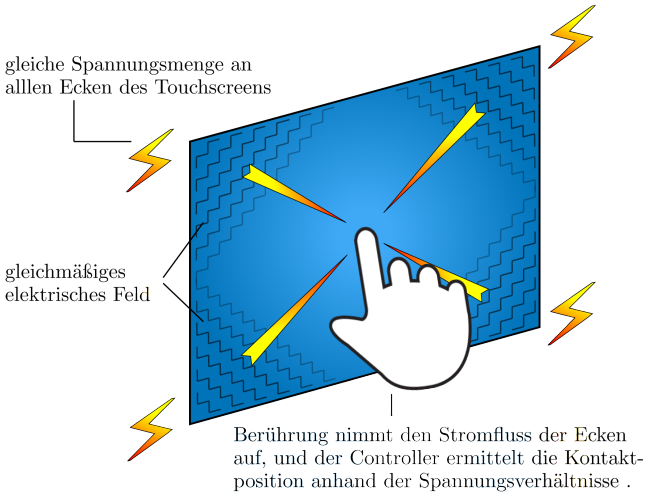


Abbildung 2.3: Funktionsprinzip des kapazitiven Touchscreens nach [29].

Das lässt sich dadurch erklären, dass die Vorteile gegenüber anderen Umsetzungen überwiegen. Es handelt sich um eine Technik, welche unempfindlich gegen äußere Einflüsse wie Schmutz und Chemikalien ist, eine gute Auflösung bietet und den Kontaktpunkt sehr präzise errechnen lässt. Auch die hohe Lebensdauer und die gute optische Transparenz sprechen für dieses Funktionsprinzip.

Resistiv

Beim resistiven Funktionsprinzip reagiert der Touchscreen auf Druck. Der Bildschirm besteht dabei prinzipiell aus zwei Schichten. Auf der unteren Schicht aus Glas oder Kunststoff liegt eine dünne Folie. Zwischen diesen Schichten befindet sich ein Abstandshalter um Glas und Kunststoff voneinander getrennt zu halten. Erst bei Druck auf die äußere Folie durch einen Finger oder beliebigen Gegenstand berühren sich die beiden Schichten, wodurch sich die Kontaktposition bestimmen lässt. Der Kontakt der Folien führt dazu, dass, durch eine angelegte Spannung am Rand einer Folie, schwacher Strom von einer Schicht zur anderen fließt. Ein Controller errechnet dann ähnlich zur kapazitiven Variante die Position der Berührung mittels der entstehenden Spannungsänderungen.

Auch diese Technik findet sich bei Mobilgeräten, jedoch eher bei Navigationsgeräten als Handys.[7] Es handelt sich auch um eine gegen Schmutz unempfindliche Technik, welche ebenfalls hohe Auflösung und Präzision bietet und sogar preislich attraktiver und durch verschiedenste Gegenstände

[7] http://de.wikipedia.org/w/index.php?title=Touchscreen&oldid=89291801, Kopie auf CD-ROM unter /quellen/W8_touchwiki/.

bedienbar ist. Sie ist aber auch empfindlicher gegenüber mechanischen Beschädigungen, beispielsweise Kratzer. Außerdem ist die Lichtdurchlässigkeit etwas schlechter im Vergleich zu kapazitiven Touchscreens.

Induktiv

Diese Touchlösung hat gegenüber den beiden zuvor genannten Umsetzungsvarianten den Nachteil, ausschließlich über einen speziellen Eingabestift bedienbar zu sein. Der Touchscreen sendet stetig elektromagnetische Wellen aus, auf welche der Stift reagieren kann [30]. Dieser wird dadurch mit Strom versorgt und kann somit zusätzliche Informationen wie zum Beispiel von Drucksensoren an den Touchscreen zurücksenden. Durch die so entstehende Kommunikation zwischen Stift und Touchscreen lassen sich dann Position, Druckstärke und weitere Informationen bestimmen. Vor allem in Grafiktablets, wie sie *Wacom* herstellt, finden diese Touchscreens ihren Einsatz. Je nach Modell des Tablets können auch der Neigungswinkel des Stiftes bestimmt und eigene Tasten am Stift genutzt werden.

Akustisch

Die akustische Variante von Touchscreens ist auch unter dem Namen SAW (Surface Acoustic Wave) bekannt [29]. Diese besteht lediglich aus einer klaren Glasscheibe oder Folie und verfügt über Sender, piezoelektrische Empfänger und Reflektoren an den Rändern des Bildschirms. Ausgesendete Ultraschallwellen erzeugen durch die Reflektionen an den Rändern eine Art digitales Netz auf der Fläche. Durch Berührung des Rasters mit einem Finger wird ein Teil der Oberflächenwellen absorbiert. Die so entstehenden Änderungen können schließlich wieder elektronisch ausgewertet und die Berührungsposition bestimmt werden. Diese Technik ist zwar eine der teuersten, ist dafür aber auch für größere Bildschirme geeignet und mit Sicherheitsglas ausstattbar. Das macht SAW vor allem für den Einsatz im öffentlichen Bereich attraktiv, wo höhere Sicherheit gegenüber Vandalismus erforderlich ist.

Optisch

Die optische Umsetzung von Touchscreens arbeitet mit Licht in Form von Infrarotstrahlen. Ähnlich wie bei der akustischen Touchlösung, wird durch eine Reihe von Lichtgebern und lichtempfindlichen Sensoren ein Gitter aus Lichtschranken erzeugt. Unterbricht ein Finger das Lichtgitter an einer Stelle, so kann der Kontaktpunkt wieder elektronisch bestimmt werden. Die Genauigkeit dieses Systems ist jedoch eingeschränkter als bei resistiven oder kapazitiven Systemen. Zudem handelt es sich um eine teure Technik und sie ist unter anderem durch Infrarotfernbedienungen und Schmutz fehleranfällig.

2.2.2 Eingabe über Finger oder Stift

Die Frage nach der Verwendung von Fingern oder Eingabestift für die Bedienung von Touchscreens wird nicht allein durch die Technik dahinter beeinflusst. Es sind auch grundsätzliche Überlegungen zum jeweiligen Einsatzgebiet zu berücksichtigen.[8]

Eine Bedienung mit dem Finger ist zwar leichter und intuitiver und man hat nicht das Problem, den Stift zu vergessen oder zu verlieren, aber sie bringt auch Nachteile.[9] Der Druckpunkt ist aufgrund der Größe des Fingers ungenauer als bei der kleinen Spitze eines Stiftes. Kleine Objekte wie Links auf Internetseiten sind damit nur schwer und manchmal gar nicht zu treffen. Des weiteren reagiert das Handy nicht auf Eingaben, wenn man Handschuhe trägt, und es werden Fingerabdrücke am Display hinterlassen. Eine Tatsache, die jedoch wieder für fingerbasierte Eingabe spricht, ist die Möglichkeit der einhändigen Bedienung. Bei Verwendung eines Stiftes muss das Gerät mit einer Hand gehalten werden um mit der anderen den Stift über das Display bewegen zu können.

2.2.3 Gesten und Multitouch

Bei Touchscreens erfolgt die direkte Interaktion über Gesten. Das Gegenstück zu einem *Click* mit einer Maus auf einem Notebook oder PC wird bei der berührungsemfindlichen Oberfläche *Tap* genannt. Je nach Touchscreen-Technologie erkennt ein Bildschirm auch Berührungen mehrerer Finger zum selben Zeitpunkt.[10] Dementsprechend können auch mehr Bedienmethoden zum Einsatz kommen und intuitive Gesten ermöglicht werden. Mittlerweile bekannte Anwendungsbeispiele sind das Vergrößern und Drehen von Bildern, indem man zwei Finger voneinander wegbewegt bzw. sie zueinander rotiert. [28] hat die wesentlichen Gesten für Touchscreens unter der Bezeichnung *Core Gestures* zusammengefasst, vgl. Abb. 2.4. Folgende Touchbefehle sind in dieser Übersicht abgebildet:

- **Tap:** Kurze Berührung der Oberfläche mit der Fingerspitze.
- **DoubleTap:** Zweimaliges Antippen des Bldschirms mit der Fingerspitze schnell hintereinander.
- **Drag:** Finger über die Oberfläche bewegen ohne diesen abzuheben.
- **Flick:** Schnelles Wischen über den Touchscreen.
- **Press:** Berühren der Oberfläche für einen ausgedehnteren Zeitraum.

[8] http://www.sapdesignguild.org/resources/tsdesigngl/FingerStylus.htm, Kopie auf CD-ROM unter /quellen/W12_procon/.

[9] http://www.pcwelt.de/news/Fingerstift-HTC-meldet-Stylus-fuer-kapazitive-Displays-zum-Patent-an-420914.html, Kopie auf CD-ROM unter /quellen/W11_stiftfinger/.

[10] http://de.wikipedia.org/w/index.php?title=Multi-Touch-Screen&oldid=88809876, Kopie auf CD-ROM unter /quellen/W13_mtouchwiki/.

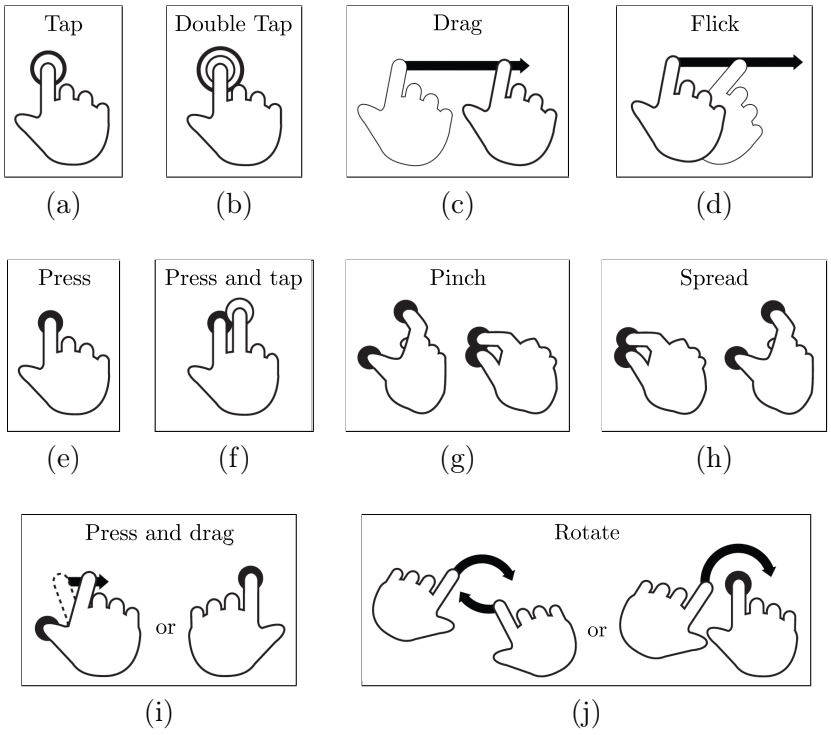


Abbildung 2.4: *Core Gestures* für Touchbefehle nach [28] (a - j).

- **Press and tap:** Einen Finger auf den Touchscreen gedrückt halten und mit zweitem Finger kurz drauftippen.
- **Pinch:** Fläche mit zwei Finger berühren und aufeinander zubewegen.
- **Spread:** Zwei Finger auf der Bildschirmoberfläche voneinander wegbewegen.
- **Press and drag:** Einen Finger auf den Touchscreen gedrückt halten und den zweiten Finger über den Bildschirm bewegen ohne dabei die Fingerspitze abzuheben.
- **Rotate:** Touchscreen mit zwei Fingern berühren und im Uhr- oder Gegenuhrzeigersinn zueinander bewegen.

2.3 Schwierigkeiten der Touchscreen-Bedienung

Trotz oder gerade aufgrund des technischen Fortschritts in Form von Touchscreens sind mobile Geräte für Menschen mit Sehbehinderung unzugänglich [13]. Da Touchscreens standardmäßig kein haptisches oder akustisches Feedback geben, ist es für blinde Menschen nicht möglich, Interaktionselemente zu erkennen. Um mit Elementen am Display interagieren zu können, müssen diese sowohl lokalisiert, als auch identifiziert werden können. Auf visuellem

Wege passiert dies allein durch den Blick auf ein bestimmtes Symbol, eine bestimmte Darstellung eines UI-Elements. Infolge dessen sind gewisse Menschen je nach Grad ihrer Sehbehinderung darauf angewiesen sich von anderen Personen helfen zu lassen, auf Alternativen zurückzugreifen und zu verwenden, oder sich damit abzufinden dieses oder jenes Gerät nicht bedienen zu können.

2.3.1 Sehbehinderung

Sehbehindert zu sein heißt nicht zwangsläufig, völlig blind zu sein. Meist handelt es sich um eine dauerhafte massive Einschränkung der visuellen Reizverarbeitung.[11] Als sehbehindert oder -beeinträchtigt gelten Menschen, die trotz Korrektur von Sehfehlern wie Kurzsichtigkeit, nur über ein eingeschränktes Sehvermögen verfügen. Sehbehinderungen werden in unterschiedliche Stärken und Arten eingeteilt. Die ausgeprägteste Form nennt sich *Amaurose* und bedeutet die vollständige Form der Blindheit ohne jegliche optische Wahrnehmungsfähigkeit.

[17] beschreibt was Menschen je nach Art der Behinderung noch sehen oder erkennen können. Die dort gezeigten Beispielbilder wie in Abb. 2.5 sollen Sehenden ermöglichen, einen Eindruck davon zu erhalten, was es bedeutet, sehbehindert zu sein, und wie die Auswirkung dieser Beeinträchtigungen auf das Sehfeld aussehen kann.

2.3.2 Adaption digitaler Geräte

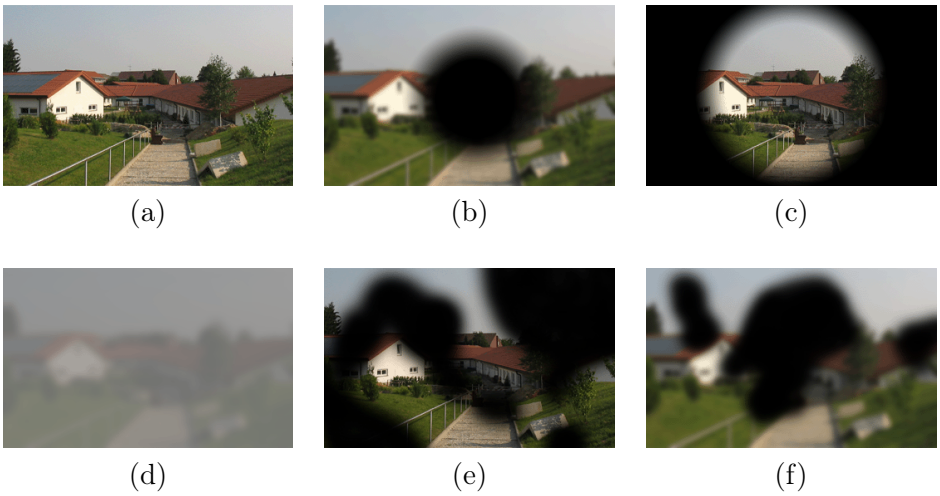
Trotz der genannten Einschränkungen des Sehvermögens meistern auch sehbehinderte Menschen den Umgang mit touchbasierten Geräten, so gut es ihnen gelingt. In [14] werden unterschiedliche Strategien geschildert, welche sehbehinderte Menschen anwenden, um mit den digitalen Geräten im Alltag zurecht zu kommen.

Modifizierung

Soweit es möglich ist, passen manche Sehbehinderte die Einstellungen der Geräte ihren Bedürfnissen entsprechend an. Das Erhöhen der Schriftgröße ist eine Möglichkeit, das Installieren zusätzlicher Software wie z.B. einen *Screenreader* eine andere. Eine weitere Idee ist das Festlegen unterschiedlicher Klingeltöne für verschiedene Personen, wodurch nicht am Bildschirm abgelesen werden muss, um welche Person es sich beim Anrufer handelt. Für Geräte zuhause wie einem Mikrowellenherd können selbstklebende haptische Hilfen wie beispielweise ein Braille-Label[12] genutzt werden.

[11] http://de.wikipedia.org/w/index.php?title=Sehbehinderung&oldid=84199364, Kopie auf CD-ROM unter /quellen/W14_visimpwiki/.

[12] Eine auf Punktmuster basierende Blindenschrift, http://de.wikipedia.org/w/index.php?title=Brailleschrift&oldid=89411333, Kopie auf CD-ROM unter /quellen/W18_braillewiki/.



(a) (b) (c)

(d) (e) (f)

Abbildung 2.5: Normales Sehfeld (a), so nimmt ein normal sehender Mensch seine Umwelt wahr. *Makuladegeneration* (b), Ausfälle im zentralen Gesichtsfeld und Verlust der Sehschärfe. *Retinopathia pigmentosa* (c), wegen dieser Gesichtsfeldausfälle auch als Tunnelblick bezeichnet. *Katarakt* (d), auch unter *Grauer Star* bekanntes trübes Sehen. *Glaukom* (e), auch als *Grüner Star* bekannte Ausfälle des Gesichtsfeldes. *Diabetische Retinopathie*, durch Diabetes verursachte diverse Einschränkungen des Sehvermögens.

Nutzen mehrere Geräte

Unzugängliche Funktionen an einem Gerät können durch das Verwenden eines anderen Gerätes kompensiert werden. Das schwächt weiters auch die Folgen eines Geräteausfalls und der damit verbundenen Besorgnis bezüglich Verlässlichkeit.

Eingeprägte Bedienmuster

Wenn der Text auf Geräten zu klein ist, um diesen ohne Vergößerungsglas lesen zu können, hilft eine weitere Strategie, nämlich das Einprägen von Interaktionsabläufen für eine bestimmte Aufgabe. Zuhause kann in Ruhe mit einer Lupe das Gerät studiert werden, um im öffentlichen Alltag in der Lage zu sein, es einfacher bedienen zu können.

Kapitel 3

Nicht-visuelle Interaktion

Um der Notwendigkeit einer nachträglichen Adaption entgegenzusteuern, sollte das User-Interface bereits von Beginn an so ausgelegt werden, dass der Bildschirmanzeige während Navigation oder Texteingabe weniger Beachtung zugewendet werden muss. Ein entsprechender Schritt in diese Richtung wird beispielsweise in [15] verfolgt. Dieser Ansatz sieht für Situationen, in denen die Aufmerksamkeit bevorzugterweise auf die Umgebung gerichtet sein sollte, größere Elemente am Bildschirm vor. Dieses sogenannte *Walking User Interface* setzt sich aus zwei Ansichten zusammen. Die Standardansicht wird dabei um eine größere Darstellung der Elemente ergänzt und beispielsweise während eines Spaziergangs angezeigt, vgl. Abb. 3.1. Diese Variante setzt dabei aber immer noch den Blick auf das Display voraus. Es behandelt nur den Faktor *Größe* und vernachlässigt alternative Interaktionsmöglichkeiten und zusätzliches nicht-visuelles Feedback. Doch gerade erweitertes Feedback ist für blickfreie Interaktionen sehr wichtig, so kann schon alleine zu ertastendes Feedback den Nutzer dabei unterstützen, virtuelle Tasten schneller und präziser zu drücken [2].

3.1 Feedback

Entsprechende Taktiken für Feedback im Kontext nicht-visueller Interaktion sind die Stimulation des Hör- und Tastsinns des Nutzers. Eve Hoggan und Stephen Brewster et al. sind in diesem Bereich tätig und haben die Effektivität von haptischem und akustischem Feedback auf mobilen Geräten untersucht [8,9]. Durch das zusätzliche Feedback konnten sie, im Vergleich zu rein visueller Basis, einigen Fehlerquellen entgegensteuern und den Durchsatz richtiger Eingaben steigern. Besonders auf Fehler wie die Eingabe falscher Buchstaben, das unbemerkte Abheben des Fingers außerhalb eines virtuellen Elements, oder auch auf doppelte *Taps* und somit unabsichtlich doppelter Eingabe konnte so aufmerksam gemacht werden. Die erhaltene Rückmeldung bei einer Berührung des Displays trug dazu bei, Fehler zu erkennen,

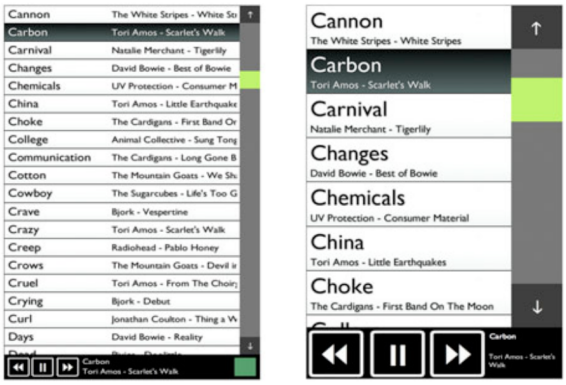


Abbildung 3.1: Das *Walking User Interface* in Form eines Mediaplayers aus [15].

auszubessern oder völlig zu vermeiden [2, 8]. In den folgenden Abschnitten wird ausführlicher darauf eingegangen, wie sich solches auf akustischer und haptischer Basis erzeugte Feedback zusammensetzt.

3.1.1 Akustische und haptische Rückmeldung

In [11] werden hör- und tastbares Feedback gegenübergestellt um diese auf einem einander übergreifenden Weg für Mobilgeräte zu nutzen. Dazu ist es jedoch nötig beide Modalitäten vorher zu differenzieren und zu verstehen.

Zum **akustischen Feedback** können drei sich unterscheidende Varianten gezählt werden [11]. Zum einen ist hier die Sprachsynthese und das damit verbundene computergenerierte Vorlesen von Wörtern oder Texten zu nennen, Näheres dazu siehe Abschnitt 3.3. Zum anderen können Informationen über *Auditory Icons* oder *Earcons* kodiert werden. Ersteres deckt natürliche, alltägliche Geräusche ab, welche auch sehr verbreitet für Aktionen am Computer eingesetzt werden. Sie beinhalten eine zugeordnete Bedeutung für bestimmte Aktionen, beispielsweise das Leeren des virtuellen Papierkorbs. *Earcons* hingegen sind künstlich erzeugte abstrakte Töne, die nicht automatisch einer Aktion zugeordnet werden können, sondern zuvor erlernt werden müssen.

Für **haptisches Feedback** ist mechanische Stimulation durch den Vibrationsmotor in einem Gerät zuständig. Bezogen auf die allgemeine haptische Wahrnehmungsfähigkeit des Menschen (vgl. Abb. 3.2), ist dies die einzige Modalität, die aktuelle Mobilgeräte als Resonanz geben können. In Form von konkretem Feedback werden die erzeugten Vibrationsmuster auch als *Tactons* bezeichnet [11].

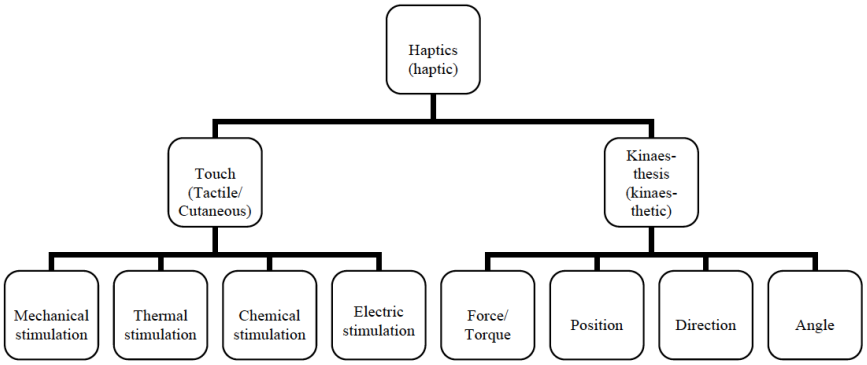


Abbildung 3.2: Terminologie haptischer Wahrnehmung aus [11].

Earcons

Ein *Earcon* ist das akustische Äquivalent zu einem visuellen Icon. Während ein visuelles Icon durch ein Bild, ein Symbol bestimmt ist, werden *Earcons* aus kurzen, profilierten Tönen gebildet. Ein Beispiel dafür wäre ein Klang für eine eingehende Nachricht bei einem Chat-Dienst oder ein Alarmton bei auftretenden Fehlern in einem Betriebssystem. Um unverkennbare *Earcons* zu erzeugen, kann man verschiedene Parameter variieren [11]:

- **Tonhöhe:** Je nach gewählter Schwingungsfrequenz bei der Tonerzeugung, erhält man hohe oder tiefe Töne. Je größer die Frequenz, desto höher ist der Ton. Um unterschiedliche Tonhöhen eindeutig voneinander differenzieren zu können, bedarf es großer Unterschiede in der gewählten Frequenz [11].
- **Intensität:** Die Lautstärke ist laut [11] eine Größe, die nicht zur Parametrisierung von *Earcons* herangezogen werden sollte. Nutzer empfinden laute Töne eher als belästigend und wollen die Lautstärke außerdem selbst kontrollieren können.
- **Klangfarbe:** Die Klangfarbe ist durch die Signalform bestimmt. Zwei Instrumente, die einen Ton mit gleicher Tonhöhe und Intensität erzeugen, rufen durch ihre überlagernden, individuellen Teilschwingungen verschiedene Klänge hervor.
- **Tempo:** Die Abspielgeschwindigkeit bietet eine weitere Adaptionsmöglichkeit für eine Differenzierung von *Earcons*.
- **Dauer:** Entsprechend einer Variation der Länge eines abgespielten Tones können unter Beibehaltung aller anderen Parameter bereits unterschiedliche *Earcons* definiert werden.
- **Räumliche Wahrnehmung:** Die Fähigkeit den Ursprungsort eines Tones einer Richtung zuordnen zu können, ist beim *Earcon*-Design nebensächlich. Dieser Parameter enthält keine nutzbare Information,

da der Ton prinzipiell aus der Richtung des erzeugenden Mobilgerätes kommt. Lediglich die Verwendung von stereofähigen Kopfhörern würde wieder eine Unterscheidung auf Links-/Rechts-Basis bieten.

- **Rhythmus:** In Relation zur Zeit können unter Verwendung der genannten Parameter unterschiedliche Rhythmus-Muster geschaffen werden.

Tactons

Tactons stellen das vibrotaktile Gegenstück zu visuellen und akustischen Icons dar. Erzeugt wird dieses haptisch spürbare Feedback mittels Vibrationsmotoren. Informationen können damit über verschiedene Dimensionsparameter an den Nutzer weitergegeben werden:

- **Frequenz:** Die Frequenz mit welcher die Unwucht eines Vibrationsmotors um die Rotationsachse bewegt wird, generiert Vibrationen durch wahrnehmbare grobe oder feine Schwingungen.
- **Intensität:** Wie bei der akustischen Intensität ist auch hier die Amplitude des Signals für die Stärke der Vibration verantwortlich.
- **Signalform:** Die Signalform (Sinus-, Rechteck-, Dreieckssignal etc.) kann nach [11] mit der „Textur“ einer taktilen Stimulation verglichen werden.
- **Dauer:** Unter einer Variation der Vibrationslänge sind unter Beibehaltung anderer Parameter unterschiedliche *Tactons* bestimmbar.
- **Örtliche Zuordnung:** Die örtliche Zuordnung eines haptischen Feedbacks kann auf zwei Wegen erfolgen. Zum Einen kann dieses an verschiedenen Stellen am Körper stattfinden, zum Anderen kann die Vibration auf unterschiedlichen Positionen auf einem Gerät erzeugt werden.
- **Rhythmus:** Wie beim Design von *Earcons* stellt auch hier der Rhythmus einen wichtigen Parameter dar. Verschiedene Rhythmen sind durch Aneinanderreihung von unterschiedlich langen Vibrationsimpulsen gestaltbar.

Smartphones sind standardmäßig mit einfachen Vibrationsmotoren ausgestattet. Diese sind meist nur in der Lage eine vom Entwickler angegebene Vibrationsdauer zu verarbeiten. Um mehr Parameter dynamisch bestimmen zu können, müssten die Ansteuerung der Motoren oder die Motoren selbst adaptiert werden. In weiterer Folge können die Geräte auch um zusätzliche Vibrationsmotoren erweitert werden. In [31] wird ein Smartphone auf insgesamt fünf Motoren ausgebaut, um so auch ein räumliches Gefühl der Vibration zu erhalten. Im Gegensatz zu üblichen Geräten kann ein *Tacton* auf diese Weise auch den Parameter der örtlichen Zuordnung erfahren. Es vibriert nicht einfach nur das Gerät an sich, sondern jene Motoren entsprechend der Position des UI-Elementes.

3.1.2 Multi- und crossmodales Feedback

Die Verwendung von mehr als nur einem Feedbackkanal bei Interaktionen wird als *multimodal* bezeichnet [10]. Die jeweiligen Rückmeldungen haben hierbei keinen Bezug zueinander. Werden mehrere Sinne gereizt um dieselbe Information zu übermitteln, so nennt man dies *crossmodales* Feedback [12]. Crossmodale Icons sind infolgedessen abstrakte Icons welche sowohl als *Earcons* als auch *Tactons* instantiert und intuitiv miteinander äquivalent verglichen werden können. Bei akustischem und haptischem Feedback funktioniert dieses gemeinsame Übertragen der selben Information durch die zueinander ähnlichen Parameter, die bei der Erzeugung bestimmt werden (Dauer, Rhythmus, etc.).

Dieser crossmodale Einsatz der Feedbackmodalitäten hat den Vorteil, dass geeignete Rückmeldung in unterschiedlichen Kontexten gegeben ist. Dies ist bedeutend, da abhängig von der Situation durch Umgebungslautstärke und- erschütterung das akustische oder haptische Feedback unzureichend sein kann [9]. Wenn ein Feedbackkanal ausfällt, ist immerhin noch ein weiterer vorhanden, der durch Crossmodalität in gleicher Weise verstanden wird. Für den Nutzer sollte es dabei möglich sein, zwischen den Modalitäten manuell zu wechseln. Da die Lautstärke und der Erschütterungsgrad der Umgebung durch eingebaute Sensoren wie Mikrophon und Bewegungssensoren bestimmbar ist, kann dies auch automatisch geschehen.

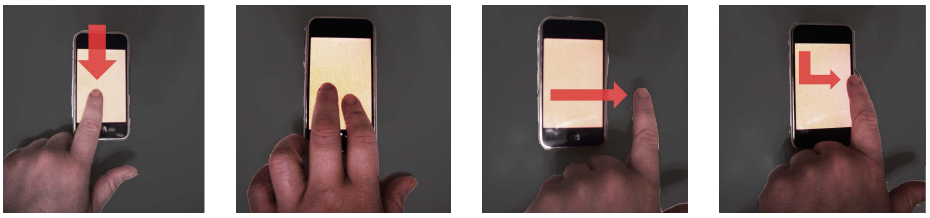
3.2 Praktische Anwendung blinder Interaktion

3.2.1 Navigation

Das Navigieren durch das Betriebssystem oder ein Menü ist neben der Texteingabe ein wesentlicher Teil der Bedienung eines Smartphones. Folgende Ansätze versuchen auf unterschiedliche Weise eine nicht-visuelle Interaktion zu ermöglichen und greifen dafür auf verschiedene Interaktionsmethoden zurück.

Slide rule

In [13] wird dabei insbesondere auf die Steuerung mithilfe von **Gesten** gesetzt. Entwickelt wurde diese Variante unter dem Namen *Slide Rule*. Für diese ist keine zusätzliche Hardware nötig, lediglich ein Multi-Touch fähiges Smartphone mit Audioausgabe. Interaktionen können bei dieser Variante auf Basis eines Sets von vier grundlegenden Gesten erfolgen, siehe Abb. 3.3. Das Feedback beruht zur Gänze auf gesprochenem Text, *Slide Rule* bietet keine visuelle Darstellung von Informationen. Die Ergebnisse einer Studie zeigen, dass bestimmte Aufgaben bei *Slide Rule* im Vergleich zu PDAs schneller ausgeführt werden konnten. Die Bedienung war jedoch auch etwas fehleranfälliger, da Gesten unbeabsichtigt oder falsch ausgeführt wurden.



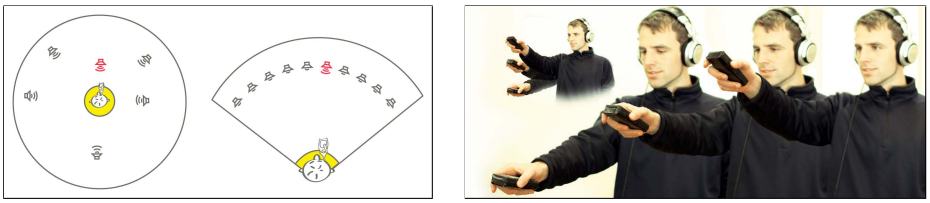
(a) (b) (c) (d)

Abbildung 3.3: Die vier Basis-Gesten von *Slide Rule* [13]. Scrollen durch eine Liste von Elementen (a). Elementauswahl (b). Eine *flick*-Geste für sekundäre Aktionen wie das Wechseln der Seitenansicht (c). Weitere Variante für das Selektieren von Elementen zum Durchsuchen hierarchisch angeordneter Information (d).

Abbildung 3.4: *The Moose* [22] als Eingabegerät zur haptischen Erweiterung von grafischen Benutzeroberflächen.

The Moose

The Moose [22] ist ein haptisches User-Interface für blinde Personen welches für die Nutzung auf PCs ausgelegt ist, nicht auf Smartphones. Dieser Ansatz erweitert die grafische Benutzeroberfläche mit haptischem Feedback, wodurch Elemente wie Programmfenster, Menüs, Buttons etc. als virtuelle Objekte erfahrbar sind. Diese Objekte können infolgedessen auf nichtvisueller Basis lokalisiert, identifiziert, und schließlich betätigt oder manipuliert werden. Die Navigation erfolgt über ein Gerät, das Eingaben ähnlich einer üblichen Maus ermöglicht, mit dem Unterschied, dass dieses das haptische Feedback für den Nutzer erzeugt. Der in Abb. 3.4 abgebildete, weiße, in einer Ebene bewegliche Quader ist das Herzstück des Gerätes. Verbunden mit zwei Linearmotoren wird beispielsweise beim Überfahren der Kante eines Programmfensters ein haptisches Feedback für eine Art Einkerbung simuliert. Neben dieser Rückmeldung wird auch Feedback auf Audio-Basis genutzt. Dieses beschränkt sich aber auf den Einsatz von Sprachsynthese (näheres zu Sprachsynthese wird in Abschnitt 3.3 beschrieben) bei Dateinamen oder Labels.



(a) (b)

Abbildung 3.5: Das *Foogue*-Interface [6] zur Lokalisierung von räumlich angeordneten Ordnern und Dateien (a) und Auswählen dieser mit einer Bewegungsgeste (b).

Foogue

Eine weitere Methode für nicht-visuelle Interaktion bietet *Foogue* [6]. *Foogue* ist ein 3D-Audio Interface und ermöglicht das blinde Auswählen und Abspielen von Musikdateien. Ordner und Dateien sind dabei virtuell im Raum verteilt und werden über Kopfhörer vom Nutzer wahrgenommen, vgl. Abb. 3.5. Die Lokalisierung der Elemente im Raum funktioniert über den eingebauten Kompass im Gerät. Respektive der Ausrichtung des Gerätes können Elemente lokalisiert und mit einer darauffolgenden Aufwärtsbewegung des Gerätes ausgewählt werden. Für die letztere Aktion wird ein integrierter Bewegungssensor benötigt. Dieser ist in Kombination mit dem digitalen Kompass für verschiedene ausführbare 3D-Aktionen verantwortlich.

3.2.2 Texteingabe

UniGest

Alternative Eingabemethoden wie *UniGest* [3] arbeiten mit Beschleunigungssensoren für das Eingeben von Text. In diesem Ansatz wird auf die eingebauten Bewegungssensoren in einer Nintendo Wii Fernbedienung zurückgegriffen. Der Nutzer kann Eingaben tätigen, indem eine dem jeweiligen Zeichen entsprechende Geste ausgeführt wird. Jeder Buchstabe besteht dabei aus einem Set von primitiven Bewegungsabläufen gemäß der Pfeile in Abb. 3.6. Da es auch Smartphones mit Bewegungssensoren gibt, wäre es denkbar, diese Methode auch dort einzusetzen.

8Pen

8Pen [1] ist eine Texteingabemethode welche in erster Linie für schnellere Eingabe auf Touchscreens entwickelt wurde. Der Gedanke der nicht-visuellen Eingabemöglichkeit ist dem Ansatz jedoch nicht fern. Das zugrundeliegende Konzept baut auf Gesten am berührungsempfindlichen Bildschirm auf. Der im Zentrum positionierte Punkt, vgl. Abb. 3.7, stellt den Ausgangs-

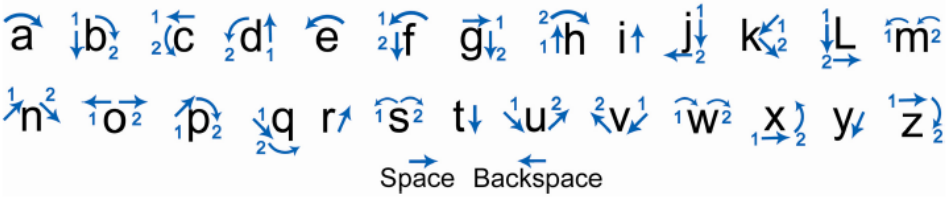
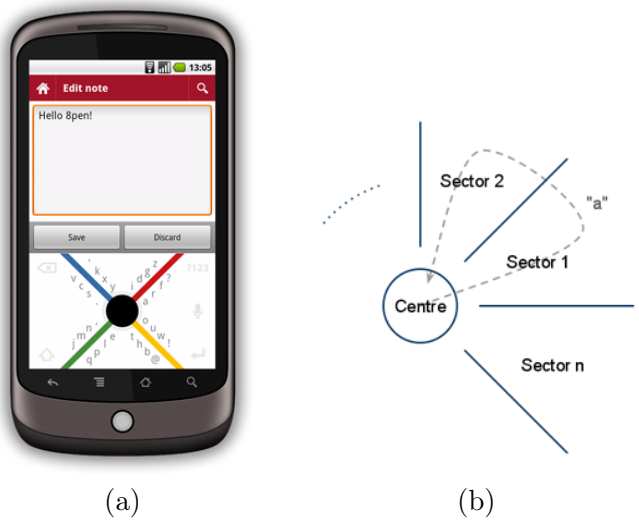


Abbildung 3.6: Das *UniGest*-Alphabet [3] für Texteingabe auf Basis von Gesten durch systematische Bewegungsabläufe.



(a) (b)

Abbildung 3.7: Texteingabesystem speziell für Touchscreens namens *8Pen*. Screenshot der Texteingabemethode auf dem Smartphone Betriebssystem *Android* (a). Auszug aus dem Patentantrag [1] für die gegebene Eingabetechnik (b).

punkt einer Interaktion dar. Die Geste bildet sich dann, indem die Berührung auf einem der Quadranten bzw. Sektoren weiter verläuft. Je nachdem, ob der Finger dann im oder gegen den Uhrzeigersinn geführt wird, ergibt sich eine Vorauswahl von Zeichen. Die Bogenlänge bestimmt weitergehend, welches Zeichen in dieser Vorauswahl beim Erreichen der Ursprungsposition letztendlich ausgewählt und eingegeben wird. Bis auf die Tatsache, dass der Startpunkt visuell gefunden werden muss, ist die Eingabe auch blind durchführbar.

BlindType

Mit *BlindType* wurde eine Möglichkeit geschaffen, Text in gewohnter Weise in Form eines QWERTY-Layouts einzugeben. Die Besonderheit dabei ist,

Abbildung 3.8: Anwendungsbeispiel von *BlindType*. Die Tastatur richtet sich nach der getätigten Tippfolge und Wahrscheinlichkeit einer bestimmten Worteingabe.

Abbildung 3.9: Das *Graffiti Alphabet* [27] für gestenbasierte Texteingabe.

dass das System anhand typischer Muster zu erkennen versucht, was der Anwender tippt. Die Tastatur passt sich der Tippfolge an und dient nur noch als Hilfestellung am Display, vgl. Abb. 3.8. Zum aktuellen Zeitpunkt sind lediglich Demovideos dieser Technik verfügbar.[1]

Graffiti Alphabet

Das *Graffiti Alphabet* ist der grundlegende Bestandteil der Eingabemethode von [27]. Abb. 3.9 zeigt dieses Alphabet mit der Startposition der Geste dargestellt als Punkt. Das Prinzip ist hier ähnlich jenem von *UniGest*, mit dem Unterschied, dass die Geste anstatt durch das Bewegen des Gerätes, per Finger am Touchscreen ausgeführt wird. Der Bewegungssensor wird aber dennoch eingesetzt, und zwar, um durch ein Schütteln des Gerätes das Ende der Texteingabe zu signalisieren.

[1] http://www.ijailbreak.com/applications/blindtype-keyboard-accuracy-not-required/, Kopie auf CD-ROM unter /quellen/EF8_blindtype/.

Abbildung 3.10: Die aus [24] grundlegende Idee der relativen Anordnung von einzugebenden Zeichen zum initialen Berührungspunkt.

Eyes-Free Android

T. V. Raman ist ein Entwickler bei *Google* und selbst sehbehindert (Glaukom).[2] In [24] präsentiert er neben *TextToSpeech* für Android (siehe Abschnitt 3.3) auch einen Ansatz für das blinde Wählen von Nummern oder Auswählen von Personen in einem Adressbuch auf einem Touchphone. Die elementare Idee ist dabei die Positionierung der erscheinenden Tastatur. Diese ist nicht durch eine fixe, absolute Position am Display vorgegeben, sondern richtet sich nach der ersten Berührung am Display. Relativ zu diesem Berührungspunkt ordnen sich dann die Ziffern einer T9-Tastatur an, vgl. Abb. 3.10. Die Ziffer, über welcher der Finger danach abhebt, wird schließlich als Eingabe erfasst.

3.3 Sprachinteraktion

Eine Kommunikation basierend auf Sprache stellt eine besondere Art der Interaktion dar. So kann theoretisch sowohl Sprache vom Computer verstanden, als auch generiert werden. Was jedoch in der Theorie funktioniert, liefert zumindest bei Spracherkennung noch keine akzeptablen Trefferquoten im Vergleich zu Fehlerraten [5].

3.3.1 Sprachsynthese

Unter Sprachsynthese ist die künstliche Erzeugung der menschlichen Sprechstimme zu verstehen und dies wird fälschlicherweise oft als TTS (Text to

[2] The New York Times: *For the Blind, Technology Does What a Guide Dog Can't*, http://www.nytimes.com/2009/01/04/business/04blind.html, Kopie auf CD-ROM unter /quellen/A1_tvraman/.

Speech) bezeichnet.[3] Text-to-Speech beschreibt genau genommen die Umwandlung von geschriebenem in gesprochenen Text. Mit Softwarelösungen für TTS wie einem *Screenreader* können infolgedessen Bedienelemente und Textinhalte von einem Computer sprachlich wiedergegeben werden.

3.3.2 Sprachsteuerung

Unter dem Begriff der Sprachsteuerung ist die Übermittlung von sprachlichen Befehlen an technische Geräte zu verstehen.[4] Dafür ist es nötig, mittels Spracherkennung sprachliche Äußerungen aufnehmen und interpretieren zu können. Genau dies ist jedoch der Kern, an dem heutige Sprachsteuerungssysteme noch scheitern. Laut [5] ist eine Spracherkennungssoftware, die eine überwiegende Zahl der Nutzer ohne Training erkennt und dabei eine Trefferquote von mehr als 90 Prozent erreicht, erst im Jahre 2016 denkbar. Trotz der noch sehr unzuverlässigen Methode der nicht-visuellen Interaktion, ist sie aber beispielsweise auf *Mac OS X* oder auch *Android* bereits verfügbar.

Android Voice Actions

Ab *Android 2.2 Froyo* besteht die Möglichkeit einige Aktionen am Smartphone auch mittels Sprachbefehl auszuführen.[5] Ein Auszug diverser Sprachbefehle ist in Tabelle 3.1 dargestellt.

[3] http://de.wikipedia.org/w/index.php?title=Sprachsynthese&oldid=87867685, Kopie auf CD-ROM unter /quellen/W17_sprsyntwiki/.

[4] http://de.wikipedia.org/w/index.php?title=Sprachsteuerung&oldid=71991010, Kopie auf CD-ROM unter /quellen/W16_sprstwiki/.

[5] http://www.google.com/mobile/voice-actions/index.html, Kopie auf CD-ROM unter /quellen/W9_voiceactions/.

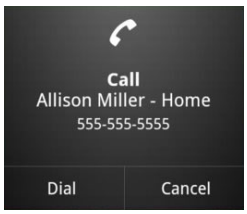
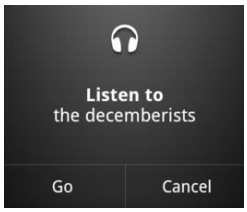
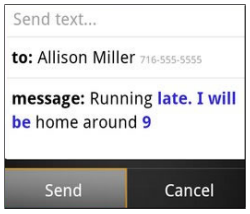
Screenshot	Sprachbefehl	Beispiel
	call [contact name] [phone type]	*call Allison Miller home*
	listen to [artist/song/album]	*listen to the decemberists*
	go to [website]	*go to Wikipedia*
	send text to [recipient] [message]	*send text to Allison Miller Running late. I will be home around 9*

Tabelle 3.1: Auszug aus den in *Android* verfügbaren *Voice Actions*.

Kapitel 4

Evaluierung nicht-visueller Interaktionstechniken

4.1 Evaluationskriterien

Um konzipierte Systeme für nicht-visuelle Interaktion bewerten und untereinander vergleichen zu können, bedarf es einiger Kriterien anhand derer sich eine Beurteilung abgeben lässt. Diese Evaluationskriterien setzen sich einerseits aus messbaren Größen zusammen, und andererseits aus bestimmten nicht-visuellen Eigenschaften, die ein System erfüllt. Zu Ersterem zählt zum Beispiel die Geschwindigkeit, mit welcher Zeichen eingegeben werden können, Letzteres beinhaltet verschiedene Umstände wie beispielsweise die Gegebenheit einer Multi-Touch-Unterstützung. Zu unterscheiden ist hier weiters zwischen Beurteilungskriterien für Texteingabe oder das Erledigen einer bestimmten Aufgabenstellung (Navigation). Bestimmte Kriterien sind entweder nur für eine einzelne Betrachtungsrichtung verwendbar, oder für beide geeignet, wenn auch teilweise nur in entsprechend abgewandelter Form.

Die zwei gebräuchlichsten Kriterien zur Evaluation beziehen sich auf Geschwindigkeit und Genauigkeit von Interaktionen [19]. Im Kontext der Texteingabe sind dies nach [18, 20] im Allgemeinen *Words per Minute*, *Word Error Rate* (%) und *Keystrokes per Character*. Während die Texteingabegeschwindigkeit für nicht-visuelle Interaktion noch zweitrangig ist, sind vor allem allgemeine Eigenschaften „blinder“ Bedienbarkeit und fehlerfreie Eingaben entscheidend.

4.1.1 Geschwindigkeit

Das bedeutende Maßkriterium Geschwindigkeit wird in Form von Eingaberaten und Eingabezeiten sowohl für das Evaluieren von Texteingabe, als auch für Navigation eingesetzt. Es tritt dabei unter anderem in Bezugsformen wie *Words per Minute*, *Keystrokes per Second* oder *Task Completion Time* auf.

Texteingabe

- **Words per Minute**: Abgekürzt als *WPM* werden hier die eingegebenen Wörter pro Minute angegeben. Ein *Wort* besteht dabei im Durchschnitt aus fünf Zeichen. Diese Messgröße findet sich in sämtlichen Arbeiten mit Evaluation von Texteingabesystemen und bietet somit einen guten Vergleich zwischen diesen Methoden. Als Zeichen werden hier alle im Endergebnis sichtbaren Zeichen inklusive Leerzeichen, Punktation etc. gewertet. Entsprechend

$$W_m = \frac{L_{text} - 1}{t} \cdot \frac{60}{5} \tag{4.1}$$

  errechnet sich WPM aus der Zeichenanzahl L_{text} im Verhältnis zur Zeit t in Sekunden. Das (-1) muss berücksichtigt werden, da das erste Zeichen als Startpunkt der Zeiterfassung betrachtet wird. $(\frac{60}{5})$ steht für die Umrechnung in Minuten und die durchschnittliche Wortlänge von fünf Zeichen.
- **Characters per Second**: Als Alternative zu *Words per Minute* kann die Texteingabegeschwindigkeit auch in *CPS* als Verhältnis eingegebener Zeichen pro Sekunde angegeben werden [26].
- **Keystrokes per Second**: Als *KSPS* (K_{sec}) werden die getätigten Eingaben pro Sekunde bezeichnet [20]. Das inkludiert sämtliche Eingaben wie ein Leerzeichen, oder auch einen Rückschritt. Nachdem *Keystroke* genau genommen auf Deutsch *Tastenanschlag* bedeutet, kann für gestenbasierende Eingaben auf *Gestures per Second* (GPS) ausgewichen werden. Die Gleichung

$$K_{sec} = \frac{N_{input} - 1}{t} \tag{4.2}$$

  ist hier ähnlich aufgebaut wie Gl. 4.1, mit dem Unterschied, dass es sich hier bei N_{input} um alle getätigten Eingaben inklusive Löschaktionen handelt, und nicht nur um die schlussendlich angezeigten Zeichen.

Navigation

- **Task Time**: Bei [15] als auch [13] wird *Task Time* als Messgröße für die aufgewendete Zeit einer bestimmten Aufgabe verwendet. Die einzelnen Aufgaben unterscheiden sich in [15] durch den Parameter der Größe von Interaktionselementen am Display und der Interaktion mit dem Gerät während sich der Nutzer in unterschiedlichen Situationen befindet. In [13] wird konkret von *Task Completion Time* gesprochen, und meint damit die durchschnittlich benötigte Zeit, welche für das Tätigen eines Anrufs, oder Auswählen einer Musikdatei bei nicht-visuellem Ausgangspunkt benötigt worden ist.

Originaltext:	***	Elvis	sagt	wir	fahren	nach	Memphis
Eingabe:	ja	Elvis	sagt	vier	fahren	***	Memphis
Fehlerklasse:	*ins*	-	-	*sub*	-	*del*	-
Wortfehlerrate:	$E_{words} = \frac{N_{sub}+N_{del}+N_{ins}}{N} \cdot 100 = \frac{1+1+1}{6} \cdot 100 = 50\%$						

Tabelle 4.1: Berechnung der Wortfehlerrate aus [4] unter Berücksichtigung von drei Fehlerklassen: Wörter gelöscht (DEL), ersetzt (SUB) und eingefügt (INS).

4.1.2 Genauigkeit

Unter den Begriff Genauigkeit lassen sich verschiedene Messwerte im Bereich von Fehlerraten und dazugehörigen Leistungswerten wie *Keystrokes per Character* fassen. Es handelt sich dabei um Angaben, welche die Eingabegenauigkeit des Nutzers wiedergeben. Die Anzahl falsch getätigter Eingaben kann so Schwachstellen und die Fehleranfälligkeit des Systems aufzeigen.

Texteingabe

- **Word Error Rate**: Als einfaches Mittel zur Bestimmung der Genauigkeit wird die *Word Error Rate* (WER) häufig bei der Bewertung von Spracherkennungsystemen eingesetzt [4, 26]. Die Wortfehlerrate setzt sich aus dem Verhältnis aller Falscheingaben zur Gesamtanzahl der Wörter im Originaltext zusammen. Bei den Falscheingaben sind die in Tabelle. 4.1 ersichtlichen Fehlerklassen zu unterscheiden.
- **Minimum String Distance**: Basierend auf der *Minimum String Distance* (MSD) errechnet sich eine Fehlerrate, die im Gegensatz zur *WER* nicht Wörter, sondern einzelne Zeichen als Fehler berücksichtigt [26]. MSD (d_{min}) bezeichnet dabei die Anzahl benötigter Eingaben, um eine Zeichenkette mittels *Löschen*, *Ersetzen* und *Hinzufügen* in eine andere Zeichenkette zu transformieren. Das Verhältnis der MSD vom Originaltext (a) zum transkribierten Text (b) zur maximalen Länge der Zeichenketten ergibt letztendlich die Fehlerrate, vgl. Tabelle 4.2 und
$$E_{msd} = \frac{\mathrm{d}_{\min}(a,b)}{\max(|a|,|b|)} \cdot 100. \tag{4.3}$$
- **Keystrokes per Character**: *KSPC* berücksichtigt ähnlich wie *KSPS* mit Bezug auf eingegebene Zeichen auch die getätigten Korrekturen und andere Eingaben [26]. Der Wert der *KSPC* (K_{char}) für einen Text mit Zeichenlänge 20 und *einer* Falscheingabe samt Korrektur durch die Rücktaste und Neueingabe des Zeichens ergäbe somit $K_{char} = 22/20 = 1{,}1$. Als Eingaben sind hier unter anderem auch das *Umschalten* auf Großbuchstaben und Ähnliche zu zählen.

Original-/Eingabetext	Zeichenanzahl	MSD	Fehlerrate
the quick brown fox	19	-	-
thiquick brown fox	18	2	10,5%
the quicj beown fix	19	3	15,8%
the quick brown foxxx	21	2	9,5%

Tabelle 4.2: Beispiele für Fehlerratenbestimmung basierend auf der *Minimum String Distance* aus [26].

Navigation

- **Task Errors**: Ähnlich zur *Task Time* wird in [15], als auch in [13], eine Messgröße für aufgetretene Falscheingaben in einer Aufgabenstellung eingebracht. Als Falscheingaben wurden hier alle fälschlicherweise aktivierten Elemente oder unabsichtlich getätigten Gesten aufgezeichnet. Die Fehlerrate ergibt sich bei beiden Arbeiten aus dem Durchschnitt der Falscheingaben pro Testlauf.

4.1.3 Allgemeine Eigenschaften

Unter den allgemeinen Eigenschaften sind abseits von Geschwindigkeit und Genauigkeit von Eingaben verschiedene Modalitäten zu erachten. Diese werden im Folgenden grob in drei Bereiche unterteilt und stehen entweder im Bezug zum gesamten System, sind speziell bei Eingaben bedeutsam oder sind vor allem im Kontext der Umgebung relevant.

Systemmodalitäten

- **Blicke**: Eine eindeutig messbare Größe für die Beurteilung eines Systems auf nicht-visueller Interaktionsbasis ist die Erfassung der auf das Display gerichteten Blicke während einer Aufgabenstellung. In [15] werden bei den Tests direkt die *Number of Glances* registriert und als Durchschnitt der Blicke pro Testszenario ausgewertet. Je weniger Blicke nötig sind, desto „blindenfreundlicher“ ist die Bedienung. Im Idealfall kann die Interaktion restlos „blind“ erfolgen.
- **Exploration**: Wenn man nicht auf das Display schauen kann, darf oder will, ist es erforderlich, die Interaktionselemente auch blind lokalisieren zu können. In [13] wird dabei die Wichtigkeit einer risikofreien Exploration hervorgehoben. Es sollte möglich sein, das Display mit dem Finger nach Interaktionselementen zu scannen und zu identifizieren ohne diese dabei zu aktivieren.
- **Intuitivität**: Unter den Begriff Intuitivität fallen besondere Merkmale. Ein zu bedienendes unbekanntes System muss im Grunde erlernt

werden. Dies könnte je nach Interface eventuell ohne jegliche Vorkenntnisse erfolgen, man ist aber meist auf vorher übermitteltes Wissen angewiesen. Dabei ist zu berücksichtigen, ob ein System schon auswendig bedient werden muss, oder noch auf nicht-visueller Basis erfahrbar ist. Gesten oder Sprachbefehle müssen beispielsweise grundsätzlich vorher erlernt werden.

- **Tätigkeit**: Schließlich ist auch der Kontext zu betrachten, für den das System konzipiert wurde. Also ob es rein zur Navigation durch ein Menü oder Ähnliches gedacht ist, oder ob der Zweck des Systems schlichtweg das Eingeben von Text ist.

Eingabemodalitäten

- **Handhabung**: Die Handhabung eines Gerätes während der Eingabe kann sehr unterschiedlich ausfallen. Grundlegend ist die Unabhängigkeit einer Bedienung für Rechts- oder Linkshänder. Weiters kann eine Unterstützung für beidhändige Bedienung die Eingabegeschwindigkeit steigern. Letzteres setzt gegebenenfalls auch die Multi-Touch-Fähigkeit des Gerätes voraus.
- **Abhängig vom Wortschatz**: Je nach System kann das Eingeben von Text dahingehend eingeschränkt sein, dass nur bestimmtes Vokabular, bestimmte Sprachen oder Wortschätze verarbeitet werden können.
- **Elementgröße**: [25] differenziert bei der Bedienung von Interaktionselementen zwischen Fingernagel, Fingerspitze und letztendlich Fingerkuppe. Je kleiner die Elemente dargestellt werden, desto schwieriger fällt es, diese präzise auszuwählen. Für die Interaktion mit einem Touchscreen empfehlen [23, 25] keinen Durchmesser unter 10 mm zu wählen. Im Detail beträgt die empfohlene Mindestgröße für die Fingerspitze 8 bis 10 mm und für die Fingerkuppe 10 bis 14 mm.

Umgebungsmodalitäten

- **Umgebungsverhältnisse**: Respektive verschiedener Störfaktoren seitens der Umgebung wie Lärm und Erschütterung ist festzuhalten, unter welchen Bedingungen ein System in nicht-visueller Interaktion beeinträchtigt wird. Ausschlaggebend ist hier die Wahl des Feedbacks, siehe Abschnitt 3.1.
- **Diskrete Bedienung**: Eine diskrete Interaktion setzt voraus, dass während der Gerätebedienung keine Aufmerksamkeit in der Umgebung erregt wird. Auffällige Bewegungen mit dem Gerät oder Sprachbefehle bringen einschränkende Nebenwirkungen für die Wahrung der Privatsphäre mit sich [14]. Ob ein System diskret zu bedienen ist, ist maßgeblich durch die zu Grunde liegende Interaktionstechnik bestimmt.

4.2 Gegenüberstellung nicht-visueller Interaktionsmethoden

Die in Abschnitt 3.2 vorgestellten Ansätze zur Optimierung und Erleichterung der Touchscreen-Bedienung auf Smartphones werden in Tabelle 4.3 gelistet. Entsprechend der Evaluationskriterien werden diese Interaktionsmethoden zusammenfassend beurteilt und gegenübergestellt. Da jede Methode grundlegend anders ist und nicht zwingend gleiche Ziele verfolgt werden, ist der Vergleich nur eingeschränkt gegeben. Tabelle 4.3 gibt dennoch einen einfachen Überblick über Stärken und Schwächen bedeutender Ansätze auf dem Weg zu nicht-visueller Interaktion.

Im direkten Vergleich ist vor allen Dingen zu erkennen, dass außer der Sprachinteraktion, hier aufgeführt mit den *Voice Actions* von Android, keine Komplettlösung für nicht-visuelle Interaktion gegeben ist. Jede Methode ist entweder auf das Navigieren und Steuern durch ein Menü spezialisiert, oder wendet sich rein der Eingabe von Zeichen zu. Das hat auch zur Folge, dass nicht alle Evaluationskriterien bei allen Methoden anwendbar sind. Die Abhängigkeit des Systems von einem bestimmten Wortschatz ist beispielsweise in erster Linie bei Texteingabesystemen wesentlich. *BlindType* ist hier ein gutes Beispiel, da es die Eingaben mit einem systeminternen Wörterbuch vergleichen muss. Das Einfügen von Wörtern funktioniert durch das Abschätzen des Eingabemusters und das Übernehmen des dazu wahrscheinlichsten Wortes, das im besten Falle natürlich mit der vom Nutzer gewollten Eingabe übereinstimmt.

Blick- Eingabe- und Fehlerraten

Nennenswert ist die Tatsache, dass fast alle Systeme gänzlich ohne Blicke auf das Display zu bedienen sind. Dabei ist aber auch zu erwähnen, dass bei den meisten Systemen bestimmte Vorkenntnisse notwendig sind. Nur *The Moose* arbeitet mit altbekannter Anordnung von Elementen bzw. einem Layout am Display wie man sie von der Verwendung eines PCs kennt. Das Gegenstück für Texteingabe ist hier *BlindType*, auch hier wird auf Altbekanntes Wert gelegt und das Tastenlayout der normalen QWERTY-Tastatur vorausgesetzt.

Bezüglich Eingabe- und Fehlerrate, sowie der Eingabezeit stehen für die erwähnten Methoden nur teilweise real getestete Werte zur Verfügung. Die Eingabegeschwindigkeit von *UniGest* ist dabei nur eine Hochrechnung aus anderen Testwerten, und somit nicht direkt mit den Werten aus *Graffiti Alphabet* vergleichbar.

Umgebungsverhalten

Die Einsatzfähigkeit der Systeme in erschütterungs- oder lärmbelasteten Umgebungen ist bei keiner gelisteten Methode uneingeschränkt gegeben. Man-

	Slide Rule	**The Moose**	**Foogue**	**Unigest**	**8Pen**	**BlindType**	**EyesFree Android**	**Graffiti Alphabet**	**Voice Actions**
Tätigkeits-feld	Navigation	Navigation	Navigation	Texteingabe	Texteingabe	Texteingabe	Vorwiegend Texteingabe	Texteingabe	beides
Eingaberate bzw. -zeit	11,69 *sec*	k. A.	k. A.	27,9 *wpm*	k. A.	k. A.	k. A.	7,6 *wpm*	k. A.
Fehlerrate	0,2 *errors/trial*	k. A.	k. A.	k. A.	k. A.	k. A.	k. A.	$E_{msd} = 0{,}4\%$ $K_{char} = 1{,}3$	k. A.
Blind bedienbar	ja	ja	ja	ja	bedingt	ja	ja	ja	ja
Risikofreie Exploration	ja	ja	ja	n. a.	nein	nein	nein	n. a.	n. a.
Intuitiv	nein	ja	nein	nein	nein	ja	bedingt	nein	bedingt
Wortschatz-unabhängig	n. a.	n. a.	n. a.	ja	ja	nein	ja	ja	nein
Elementgröße	empfehl-ungsgerecht	n. a.	n. a.	n. a.	bedingt	n. a.	empfehlungs-gerecht	n. a.	n. a.
Umgebungs-unabhängig	bedingt	bedingt	nein	bedingt	bedingt	bedingt	bedingt	bedingt	nein
Diskrete Bedienung	ja	ja	nein	nein	ja	ja	ja	ja	nein

Tabelle 4.3: Vergleichstabelle der Interaktionstechniken aus Abschnitt 3.2 unter Berücksichtigung der beschriebenen Kriterien. (*k. A.* = keine Angabe, *n. a.* = Beurteilungsmerkmal nicht anwendbar)

che bieten zwar akustisches oder haptisches Feedback, welches teilweise auch multimodal eingesetzt wird, keines der Systeme setzt dabei aber auf crossmodales Feedback. Es gibt somit keine ungehinderte Rückmeldung bei unterschiedlichen Umgebungsverhältnissen. Bei den *Voice Actions* kommt der Umstand hinzu, dass Sprachbefehle bei Umgebungslärm schwieriger bis gar nicht vom System zu erkennen sind.

Eine diskrete Bedienung ohne in der Öffentlichkeit aufzufallen ist, bis auf drei Systeme, vollständig gegeben. Bei *Foogue* und *UniGest* ist dies aufgrund der großen nötigen Handbewegungen, bei den *Voice Actions* wegen der akustisch deutlichen Aussprache der Befehle nicht möglich.

4.3 Design und Richtlinien

Aus den zuvor behandelten Evaluationskriterien und dem Vergleich der Interaktionsmethoden in Tabelle 4.3 lassen sich mehrere Richtlinien und Vorschläge für ein nicht-visuelles Interface herleiten. Die folgende Auflistung als Nummerierung soll keinen Stellenwert in Form von Wichtigkeit repräsentieren, sondern dient lediglich zur Bezugnahme und Referenzierung in späteren Kapiteln.

1. Für den Nutzer sollte klar bzw. einfach erfahrbar sein, in welchem Zustand, welchem Status sich die Anwendung zum jeweiligen Zeitpunkt befindet. Je weniger Blicke dafür von Nöten sind, desto besser. Idealerweise muss der Blick dafür überhaupt nicht auf das Display gerichtet werden.
2. Je weniger Vorkenntnisse für die Interaktion mit nicht-visuellen Systemen notwendig sind, desto intuitiver fällt die Bedienung aus. Sehbehinderte Menschen greifen dadurch auch gern auf bereits familiäre Layouts zurück, um nicht auf umständlichem Wege neue Interaktionsmuster lernen zu müssen [14]. Wenn für neue Techniken Vorkenntnisse unumgänglich sind, so ist darauf zu achten, diese möglichst gering zu halten. Es muss dabei nicht das ganze Layout bekannt sein, sondern es kann sich auch auf eine einfache Explorationsmöglichkeit beschränken, siehe Punkt 3.
3. Nach [24] sind für eine Interaktion mit Elementen am Display drei *Atomic Tasks* gegeben. Das sind *Lokalisieren*, *Identifizieren* und schließlich *Aktivieren*. Dabei ist eine risikofreie Möglichkeit zur Exploration der Interaktionslemente am Display bereitzustellen. Risikofrei meint in diesem Sinne, dass Buttons, Textfelder, etc. lokalisiert und identifiziert werden können, ohne sie dabei zu aktivieren.
4. Eine Bedienung sollte sowohl für Rechts- als auch Linkshänder uneingeschränkt möglich sein. Gegebenenfalls kann auch eine beidhändige Eingabe und Multi-Touch-Unterstützung für schnelleres Tippen von Vorteil sein.

5. Um vielseitig einsetzbar zu sein, ist es von Nutzen, wenn ein System nicht auf die Anwendung bestimmter Sprachen oder eines bestimmten Wortschatzes angewiesen ist. Sowohl bei der Exploration eines Menüs, als auch beim Eingeben von Text stellt die Einschränkung auf eine bestimmte Sprache oder vorgegebenes Vokabular ein zusätzliches Hindernis dar.
6. Die Größe der Interaktionselemente, sofern ein System auf solchen aufgebaut ist, sind so zu wählen, dass sie nach [23, 25] noch präzise auswählbar sind. D. h. grob einen Mindestdurchmesser von 10 mm einzuhalten, vgl. Abschnitt 4.1.3 Punkt *Elementgröße*.
7. Die Wahl des Feedbacks erfolgt im idealen Falle in Form eines crossmodalen Feedbacks, damit auch in unterschiedlichen Umgebungsverhältnissen und bei verschiedenen Störfaktoren ein geeigneter Kanal für eine Rückmeldung gegeben ist.
8. Schließlich sollte eine Systembedienung vorzugsweise auf diskreter Interaktion basieren, um keine Unannehmlichkeiten betreffend dem Auffallen in der Öffentlichkeit und Bedenken bezüglich Privatsphäre hervorzurufen.

Kapitel 5

Ideen und Konzept

5.1 Ausgangssituation

Wie in den vorangegangenen Kapiteln bereits erläutert wurde, sind aktuell vertriebene Handys mit Hauptbedienung per Touchdisplay, im Gegensatz zu Handys mit physischen Tasten, ohne genauere Blicke auf den Bildschirm nicht mehr flüssig bedienbar. Eine Interaktion ist somit bei bestimmten Sehbehinderungen und Situationen mit erhöhter Aufmerksamkeit für die Umgebung nur eingeschränkt oder gar nicht möglich. Wie in Kapitel 3 beschrieben, existieren zwar bereits vereinzelt Ansätze für Teillösungen dieses Problems, jedoch noch keine verbreitete Umsetzung, die eine durchgehend nicht-visuelle Interaktion ohne Einschränkungen ermöglicht.

5.2 Ziel und Anforderungen

Der Ausgangssituation zufolge sollte also ein Prototyp konzipiert werden, welcher die Interaktion mit einem *Touchphone* gewährleistet, ohne den Blick dabei notwendigerweise auf das Display richten zu müssen. In Form einer Smartphone-Applikation (App) muss dabei sowohl das Navigieren durch eine App als auch das Verfassen eines Textes abgedeckt werden. Dafür wird der Prototyp als simple Applikation zum Erstellen, Lesen, Editieren und Löschen von Notizen verwirklicht (*CRUD*[1]). Diese Art *Notepad* hat dabei sämtliche resultierenden Richtlinien und Vorschläge aus Abschnitt 4.3 zu erfüllen. Als Smartphone-Betriebssystem wird hierbei, ausgehend von der in Abschnitt 2.1 gezeigten Verbreitung, das *Android OS* festgelegt.

[1] CRUD bezeichnet die grundlegenden Datenbankoperationen *Create*, *Read*, *Update* und *Delete*, siehe http://de.wikipedia.org/w/index.php?title=CRUD&oldid=87972443, Kopie auf CD-ROM unter /quellen/W19_crudwiki/.

5.3 Erste Entwürfe und Lösungsansätze

Die Konzeption des Prototyps erfolgte hauptsächlich durch die Aufteilung in *Navigation* und *Texteingabe*. Nach mehreren Lösungsversuchen, wurde schließlich ein finales Konzept erstellt, das beide Teile der Interaktionen umfasst.

5.3.1 Navigation

Als erste und besonders einfache Variante für das Navigieren durch die Anwendung sollten die virtuellen Tasten am Rand des Display angeordnet werden. Unter der Annahme, dass die Anordnung und die Funktion der Tasten erlernt und somit wieder ohne Blick auf den Bildschirm gefunden bzw. aktiviert werden können, wäre ohne größeren Aufwand bereits eine nicht-visuelle Interaktion gegeben. In Abb. 5.1 sind gedachte Anordnungen dargestellt. Die Tasten sollten am unteren Rand positioniert werden und zusätzlich am linken bzw. rechten Bildschirmrand erweitert werden können (a). Die Entscheidung zwischen einer erweiterten Anordnung auf der linken oder rechten Seite wäre hier von der Handhabung als Links- oder Rechtshänder abhängig. Möglichkeit (b) sollte eine genauere Unterscheidung zwischen den Tasten durch Positionierung auf allen vier Displaykanten bringen, wodurch aber auch nicht *mehr* als vier Tasten gesetzt werden könnten, und das Element an der Oberkante mit dem Finger schlechter erreichbar ist. Darum und weil nicht sicherstellen werden kann, dass man ein exaktes Gefühl für die Fingerposition an der Displaymitte oder -ecke hat, wurde dieser Ansatz nicht wie gezeigt verwendet. Es wurde letztendlich auf eine bestimmte Anordnung der virtuellen Tasten verzichtet, und eine Technik verwendet, welche nicht voraussetzt, die Position und Funktion einer Taste zu wissen, sondern diese während der Bedienung interaktiv erfahrbar macht. Das inkludiert auch die Auswahl geeigneter Rückmeldungen vom System für alle Interaktionen, siehe Abschnitt 5.4.

5.3.2 Texteingabe

Beim Eingeben von Text wurde der Ausgangspunkt durch die Ideen von *Eyes-Free Android* (Abschnitt 3.2.2) geprägt. Das beinhaltet erstens das Anwenden einer relativen, anstatt einer absoluten Positionierung der Tastatur, und zweitens eines Tastaturlayouts, wie z.B in Abb. 5.2 gezeigt. Das Tastaturlayout soll sich demnach mit seinem Zentrum an die Fingerposition am Display anpassen. Die schwarzen Kreise in (a) dienen hier als Tasten für die Ziffern von 1 bis 9 oder für Gruppierungen von Buchstaben mit dem Auswählen einer Art Unterebene (graue Kreise) für konkrete Buchstabeneingabe. In weiterer Folge wurde über eine Anordnung der Tasten in Kreisform und das Hinzufügen zusätzlicher virtueller Tasten in weiteren Reihen nachgedacht,

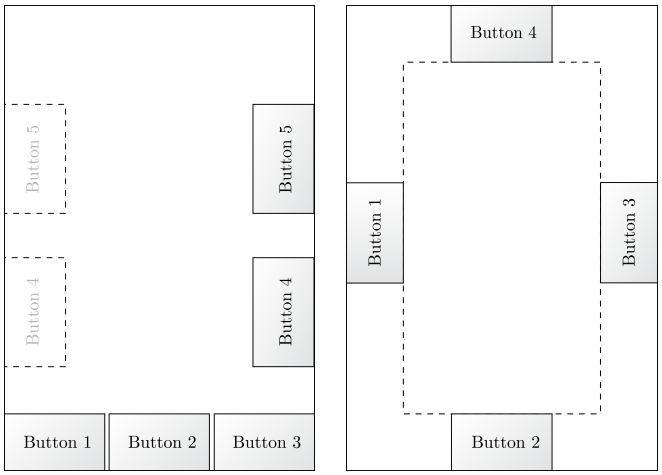


Abbildung 5.1: Mögliche Tastenanordnung am unteren und den seitlichen Bildschirmrand (a). Navigation durch eine Anordnung der Tasten auf allen vier Displaykanten (b).

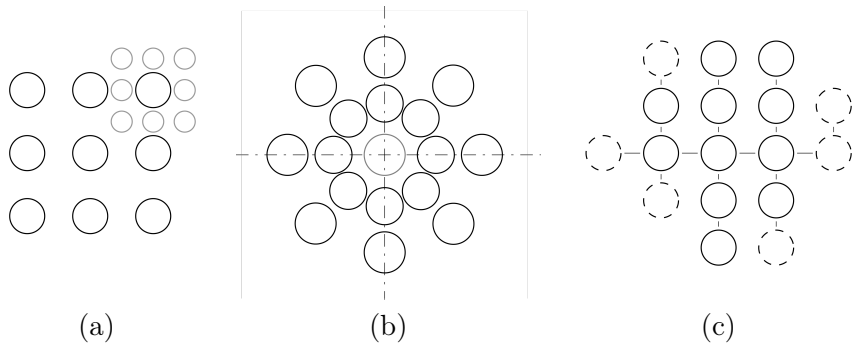


Abbildung 5.2: Schematische Darstellung (a) der Tastenanordnung von *Eyes-Free Android* in Abschnitt 3.2.2. Abgeänderte Form als Anordnung im Kreis und zweiter Tastenreihe (b). Alternative zum Kreis als rechteckige Gitteranordnung (c).

siehe Abb. 5.2 (b). Als Alternative dazu wurde überlegt, diese Tasten auf einer horizontalen Linie und eine mögliche Auswahl untergeordneter Tasten in vertikaler Richtung zu positionieren. Die letztere Variante (c) als rechteckige Gitteranordnung wurde aber nicht näher verfolgt, da die Zugänglichkeit zu den einzelnen Zeichen je nach Zentrumsnähe bzw. -entfernung stark variiert. Vom Zentrum aus sind hier nur vier Elemente direkt benachbart und schnell erreichbar. Beim kreisförmigen Layout sind dies doppelt so viele, was einen entscheidenden Vorteil gegenüber dem anderen Modell bedeutet.

Die Entscheidung für das getroffene Layout bezog letztendlich auch die Bestimmung der Zeichen mit ein, die mit der Tastatur einzugeben sein sollten. Durch die folgende Übersicht konnte dafür im Vorraus grob abgeschätzt werden, ob und wo Zeichengruppierungen bei welchem Layout nötig sind. Für die Variante in Kreisform (Abb. 5.2 (b)) wären somit pro angezeigtem Layout 16 Zeichen möglich.

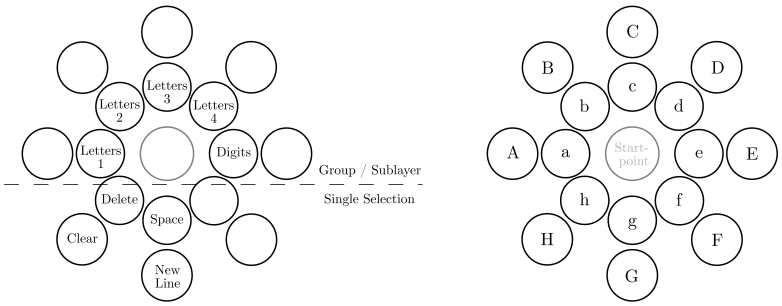
Wesentliche Zeichen:

- **Großbuchstaben:** Alle 26 Buchstaben des Alphabets. Keine Umlaute oder andere sprachspezifische Zeichen.
- **Kleinbuchstaben:** Eingrenzung wie bei den Großbuchstaben.
- **Ziffern** von 0 - 9.
- **Systemtasten:** Leerzeile; Neue Zeile (Eingabetaste); Rückschritttaste; Alle Eingaben löschen.
- **Sonderzeichen:** ? ! & @ - _ . : , ; ( ) ” / < > [ ] $ + * % § # = °
- **Sprachbezogene Zeichen:** Umlaute, *ß*, etc.

Generell wurde überlegt, ein Layout als Hauptanzeige zu bestimmen, und daraus verschiedene Unterebenen erreichbar zu machen. In diesen können dann andere Zeichen zur Auswahl bereitgestellt werden. In Abb. 5.3 wird eine erste Version eines solchen Texteingabelayouts gezeigt. Es wird dabei zwischen einer Gruppierung und einer Direktauswahl unterschieden. Im Hauptlayout ist der obere Teil für Zeichengruppierungen reserviert, im unteren Teil werden Eingaben direkt verarbeitet. Ersteres eignet sich um Buchstaben, Ziffern oder Sonderzeichen zusammenzufassen, Letzteres ist für das Einfügen eines Leerzeichens, Löschen eines Zeichens und ähnliche Funktionen in direkter Weise gedacht, siehe Abb. 5.3 (a). Grafik (b) zeigt eine mögliche Anordnung von Buchstaben in einer gewählten Unterebene. In diesem Beispiel wird die äußere Reihe von Kreisen für die Großbuchstaben verwendet. Eine alternative Variante wäre das Auslagern aller Großbuchstaben in separate Sublayer. Es besteht im Allgemein ein großer Spielraum für unterschiedliche Anordnungen. Die letztendlich gewählte Struktur ist im nächsten Kapitel erläutert.

5.4 Finales Konzept

Beim finalen und letztlich umgesetzten Konzept wird bei der Navigation auf die Möglichkeit gesetzt, Elemente am Bildschirm per Displayberührung lokalisieren und identifizieren zu können, ohne sie dabei zwangsweise auch schon zu aktivieren. Der Bildschirminhalt kann theoretisch also nicht-visuell exploriert werden. Für die Texteingabe wurde ein polares Gitter (*Polar-Input-Grid*) konzipiert, das im Wesentlichen aus einem zweireihigen Ring besteht und gegebenenfalls erweitert werden kann. Sowohl bei der Navigation



(a) (b)

Abbildung 5.3: Gruppierungen von Zeichen am oberen Teil bzw. direkte Auswahl von Eingaben am unteren Teil des Haupt-Layouts der Tastatur (a). Mögliche Anordnung von Buchstaben in einer der Unterebenen, am Beispiel von *Letters 1* (b).

als auch beim Eingeben von Text wird dabei auf entsprechendes Feedback gesetzt, welches von *Text-To-Speech*, *Earcons* und *Tactons* Gebrauch macht.

5.4.1 Lokalisation und Identifikation

Bildschirmelemente

Durch eine Anlehnung an die Funktionalität eines *Screenreaders* und der Interaktionsgegebenheiten bei einem Touchscreen ergibt sich eine Lösung ähnlich der von *The Moose*. Der Bildschirminhalt kann ähnlich der haptischen Umsetzung bei *The Moose* erkannt werden. Demnach kann der Finger über das Display geführt werden, wobei beim Überfahren von Elementkanten ein spezielles Feedback gegeben wird. Die Anordnung der Tasten ist also nebensächlich. Allein durch das gegebene crossmodale Feedback kann erkannt werden, ob man mit der Fingerspitze zum Beispiel eine virtuelle Taste betritt oder diese gerade verlässt. Diese Art des Auffindens von Interaktionselementen wird für alle relevanten Elemente, mit denen interagiert werden kann, eingesetzt. In Abb. 5.4 wird diese Methode vereinfacht dargestellt. Des Weiteren zeigt Abb 5.4 den Schritt der Identifizierung und Aktivierung eines Elements. Für die Betätigung wird wie üblich ein einfacher *Tap* verwendet. Das Identifizieren eines Elementes erfolgt durch „gedrückt halten“ des Fingers am Display. Dieser *Long-Click* bzw. *Press* führt dann zur Ausgabe von akustischem und/oder haptischem Feedback. Im gesonderten Falle erfolgt hier auch eine Sprachausgabe mittels Text-To-Speech. Wie genau dieses zugehörige crossmodale Feedback zum Erkennen von Elementen aussieht, wird in Abschnitt 5.4.3 genauer beschrieben.

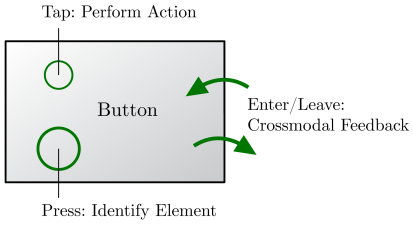


Abbildung 5.4: Blinde Lokalisierung von User-Interface-Elementen durch crossmodales Feedback bei Kantenüberquerung. Identifizierung des Elements durch einen *Long-Click* bzw. *Press*.

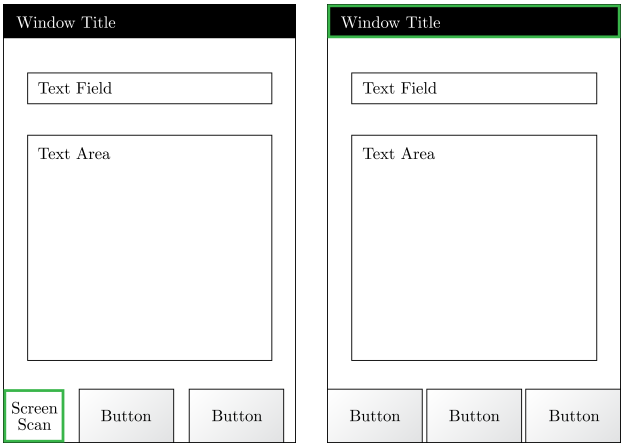
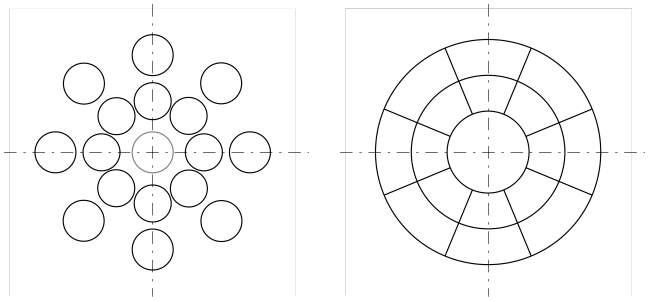


Abbildung 5.5: Identifizierung der Fensteransicht im linken unteren Eck (grüne Umrahmung) bei eigenem Startpunkt für das Abscannen des Displays nach Interaktionselementen mit dem Finger (a). Mehr Platz für virtuelle Tasten durch Identifizierung der Ansicht im grün umrandeten *Header* der Anwendung (b).

Anwendungsfenster

Um sich auch innerhalb einer Anwendung mit verschiedenen Fenstern und Ansichten zurecht zu finden, wird auch eine Möglichkeit zur nicht-visuellen Nachvollziehbarkeit des aktuell angezeigten Fensters gegeben. Zunächst war dafür ein eigener Bereich im linken unteren Eck der Bildschirmanzeige gedacht, der auch als Startpunkt für das Abscannen der Bildschirminhalte dienen sollte. Da dieser Bereich aber Platz für wichtige Elemente besetzt, wurde diese Funktionalität kurzerhand während der Umsetzung in den *Header* des Ansichtsfensters verlagert, vgl. Abb. 5.5. Das Scannen nach Elementen erfolgt infolgedessen auch von jedem beliebigen Punkt der Anwendung aus.



(a) (b)

Abbildung 5.6: Ausgehender Entwurf des umzusetzenden Tastaturlayouts bestehend aus einzelnen Kreiselementen (a). Finale Umsetzung als polares Eingabegitter bestehend aus 16 Kacheln aufgeteilt auf zwei Ringe (b).

5.4.2 Polar-Input-Grid

Von den zuvor abgebildeten Entwürfen für eine Texteingabemethode wurde die Kreisform-Variante zu einem *polaren Eingabegitter* weiterentwickelt, siehe Abb. 5.6. Dadurch ist ein einfacherer Übergang von einem Eingabeelement zum Nächsten gegeben und der verfügbare Platz lückenlos genutzt. Die Eingabefelder werden schließlich ringförmig um den platzierten Finger am Display angeordnet. Nach einer Fingerberührung, und der Positionierung der Ringe um den Berührungspunkt, behält das polare Eingabegitter seine Position während nachfolgender Fingerbewegungen am Display bei. Eine Aktion oder Zeichen wird ausgewählt, indem der Finger bei der entsprechenden Kachel wieder vom Display abgehoben wird.

Einteilung von Zeichen und Funktionen

Es können in einem Ringgitter 16 verschiedene Zeichen und Aktionen untergebracht werden. Um nun alle Zeichen auswählen zu können, welche für eine uneingeschränkte Texteingabe erforderlich sind, wird die Eingabe, wie bereits erwähnt, auf zwei Schritte bzw. Ebenen aufgeteilt. Zuerst wird der gewünschte Zeichensatz im Hauptring (Abb. 5.7) ausgewählt, worauf dann in den neu erscheinenden Ringen (Abb. 5.8 bis Abb. 5.10) das letztendliche Zeichen eingefügt werden kann. Als rein optische Hilfestellung werden die Eingabefelder zur Auswahl von Gruppierungen und Direkteingaben farblich zueinander unterschiedlich dargestellt. Ebenso sind Außen- und Innenring farblich differenziert. Leere Eingabefelder in dem Polargitter werden halbtransparent angezeigt und sind nicht wählbar bzw. geben bei Auswahl ein entsprechendes Feedback für einen Fehler. Der genaue Einsatz des gesamten Feedbacks wird im nachfolgenden Abschnitt erläutert.

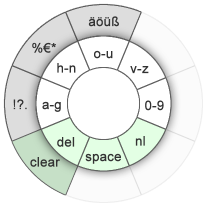


Abbildung 5.7: Gruppierungen für Buchstaben, Ziffern und Sonderzeichen im Hauptgitter-Layout sowie Funktionen zur Direktauswahl (*del* = letzte Eingabe löschen, *clear* = alles Löschen, *space* = Leerzeichen einfügen, *nl* = neue Zeile einfügen).

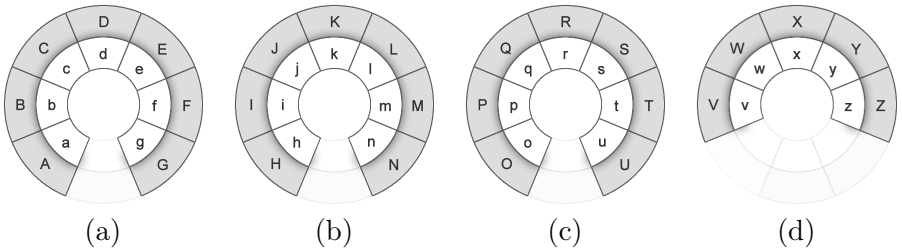


Abbildung 5.8: Die vier Unterkategorien der Buchstaben.

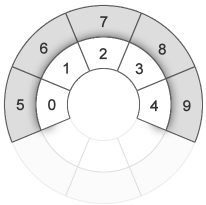


Abbildung 5.9: Anordnung der Ziffern-Unterebene.

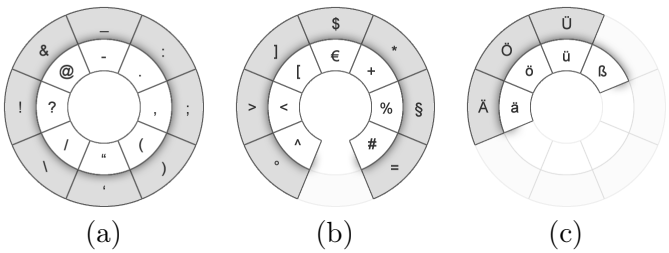
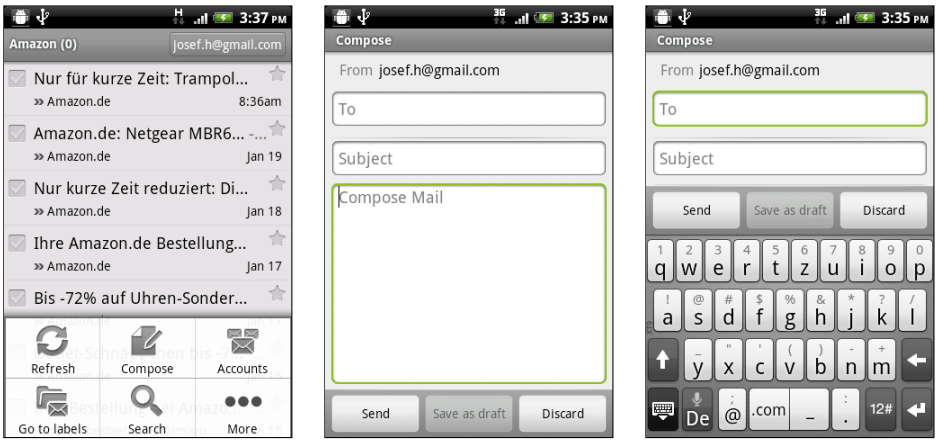


Abbildung 5.10: Unterkategorien für Sonderzeichen (a, b) und Sprachspezifische Zeichen, hier für *Deutsch* (c).



(a) (b) (c)

Abbildung 5.11: Gmail-App auf *Android* mit einer Listenansicht der Emails (a), Formularansicht für das Erstellen einer Nachricht (b) und angezeigter Eingabemethode (c).

5.4.3 Gesamtsystem

Der Prototyp wurde für die Umsetzung als Anwendung mit drei Ansichten konzipiert. In Anlehnung an vergleichbare Smartphone-Applikationen (Email, SMS) beinhaltet dies eine Listenansicht von Einträgen, ein Formularfenster zum Hinzufügen, Anzeigen, Editieren und Löschen von Notizen, und schließlich eine eigene Ansicht beim Eingeben von Text, vgl. Abb. 5.11. Durch alle drei Ansichten zieht sich außerdem ein einheitliches Feedback in Form von TTS und dem crossmodalen Einsatz von *Earcons* und *Tactons*.

Earcons und Tactons

Earcons und *Tactons* umfassen das Feedback für alle in der Anwendung getätigten Interaktionen. Dies beginnt bei der Lokalisation von Elementen durch das Berühren der Elementkanten. Die Wahl der Rückmeldungen für das „Betreten“ und „Verlassen“ eines Elements wird dabei so getroffen, dass diese untereinander differenziert werden können. Weiters wird auch beim Überfahren der Gitterlinien, sowie beim Verlassen des inneren oder äußeren Ringes der konzipierten Texteingabemethode, als Hilfestellung zur Orientierung, entsprechende Rückmeldung gegeben.

Ähnliches gilt bei der Identifikation von Elementen. Um z.B. die abstrakte Funktion eines Buttons nachvollziehbar zu machen, bekommen diese je nach Art der Funktion eigene *Earcons* und *Tactons* zugeordnet. Ein Button, welcher eine „positive“ Funktion aufweist (Bestätigung, OK, Zustimmung, etc.), wird durch andere Earcons und Tactons gekennzeichnet, als ein Element das mit „negativen“ Funktionen verknüpft werden kann (Abbrechen, Löschen,

etc.). Diese Rückmeldungen für Buttons sind zueinander aber trotzdem ähnlich, um sich auch eindeutig von anderem Feedback für Elemente wie z.B. Textfeldern zu unterscheiden. Dazu sind folgende Arten von Earcons für spätere Verwendung umzusetzen:

- Töne mit einfacher Grundfrequenz (z.B. $2kHz$) und weitere Töne mit relativ dazu höherer und niedrigerer Tonhöhe und unterschiedlicher Dauer.
- Earcons als *Sweep*[2] von „niedrigen“ zu „höheren“ Tonhöhen und umgekehrt.
- Earcons für bestimmte Aktionen (Falscheingabe, Zeicheneingabe, Click bzw. Tap).

Um Crossmodalität zu gewährleisten, werden die *Earcons* und *Tactons* durch geeignete Bestimmung ihrer Parameter als zusammengehörig kenntlich gemacht. Das entsprechende Gegenstück zu zwei kurz und schnell aufeinanderfolgenden Tönen wäre beim haptischen Feedback beispielsweise durch zwei kurze und schnell aufeinanderfolgende Vibrationen gegeben.

Text-To-Speech

TTS wird als Ergänzung zum akustischen Teil des crossmodalen Feedbacks eingesetzt. Dies ist vor allem beim Identifizieren von Interface-Elementen der Fall. Sei es nun bei Listeneinträgen, einem Textfeld, einem Button, dem Ansichtsfenster oder den Kacheln im Polargitter für die Texteingabe. Die Sprachausgabe ersetzt dabei die *Earcons*, die für die Elementidentifikation zuständig sind. Für das Lokalisieren von Elementen wird TTS nicht eingesetzt.

[2] Ein periodisches Signal, das in einer definierten Zeit seine Frequenz von einem Startwert zu einem Endwert hin ändert, siehe http://de.wikipedia.org/w/index.php?title=Sweep_(Signalverarbeitung)&oldid=79189577, Kopie auf CD-ROM unter /quellen/W20_sweepwiki/.

Kapitel 6

Der Prototyp

6.1 Systemkontext

6.1.1 Plattform

Der Prototyp wurde für das freie *Google OS Android* umgesetzt. Die zum Umsetzungszeitraum aktuellen Versionen von *Android* wechselten von 2.1 (*Eclair*) über 2.2 (*Froyo*) schließlich zu 2.3 (*Gingerbread*). *Gingerbread* war bis zur Fertigstellung des Projekts aber noch offiziell dem Smartphone *Google Nexus S* vorenthalten.

Smartphone Betriebssystem: Google Android 2.2

6.1.2 Entwicklungsumgebung

Als Entwicklungsumgebung wurde *Eclipse Galileo* verwendet, wie es auch auf *Android-Developers*[1] empfohlen wird. Auf dieser Webseite wird die Installation samt *Starter Package* zur Entwicklung für *Android* beschrieben. Das SDK (*Software Developement Kit*) wechselte während der Umsetzung von Version 2.1 auf 2.2.

Eclipse SDK Version: Galileo 3.5.0

Starter Package Version: Android SDK 2.2

6.1.3 Testgerät

Um auch das haptische Feedback des Vibrationsmotors fassbar zu machen, wurde nicht der von Google bereitgestellte AVD-Manager (*Android Virtual Devices*) für eine virtuelle Geräteoberfläche zum Testen verwendet. Stattdessen wurde die Anwendung direkt auf einem realen Gerät ausgeführt.

Modellbezeichnung: HTC Legend

Geräte-Seriennummer: HT04YNX00462

[1] Offizielle *Android* Developer Webseite http://developer.android.com

6.2 Umsetzung

6.2.1 Projekt Übersicht

Für den Prototypen grundlegend ist das auf *Android-Developers* bereitgestellte Tutorial *NotePad*, welches vor allem die Verwendung der lokalen Datenbank am Gerät aufzeigt. Basierend darauf wurde dann ein neues Projekt namens *EyesFreeNotePad* angelegt, das um Funktionen für nicht-visuelle Interaktion erweitert und abgeändert wurde. Das umfasst zum einen mehrere Klassen für die Erweiterung verschiedener Standardkomponenten auf *Android* wie `Button`, `TextView` und vor allem aber auch die eigens entwickelte Texteingabekomponente. Zum anderen beinhaltet der Projektordner sämtliche Grafiken, Sounds und Layoutbeschreibungen die im Prototyp eingesetzt werden. Abb. 6.1 zeigt den Projektordner, wie er in *Eclipse* zusammengesetzt wird.

6.2.2 Allgemeiner Aufbau und Ablauf der Anwendung

Die Ansichten einer Anwendung auf *Android* werden als *Activities* bezeichnet und dienen jeweils einer bestimmten Tätigkeit. Beim Prototypen sind insgesamt drei *Activities* im Einsatz. Die App startet mit der Haupt-Activity in Form einer *ListActivity*, Abb. 6.2 (a). Eine *ListActivity* ist eine in *Android* bereitgestellte Activity die speziell für Listenansichten eingesetzt werden kann. Diese Hauptansicht wird demnach für das Anzeigen aller Notizen in einer Liste verwendet. Von hier aus kann dann entweder eine bestimmte Notiz im Detail angezeigt, oder eine neue Notiz hinzugefügt werden, vgl. Abb. 6.3. Bei beiden Aktionen wird eine Formularansicht zum Editieren der angezeigten bzw. neu erstellten Notiz in Form einer neuen *Activity* gestartet. Das Formular beinhaltet zwei Textfelder (Android Komponente *TextView*) zum Definieren eines Titels und einer Beschreibung für die Notiz, siehe Abb. 6.2 (b). In diesem Zustand kann einerseits der Notizeintrag gelöscht oder eventuelle Änderungen bestätigt werden und andererseits entweder der Titel oder die Beschreibung durch Aufruf der weiteren *Activity* für Eingabe eines Textes bearbeitet werden, siehe Abb. 6.2 (c). In dieser Eingabe-Activity kommt die entworfene Texteingabekomponente zum Einsatz. Der gegebenenfalls eingegebene oder editierte Text kann dann entweder übernommen und bestätigt oder die Eingabe verworfen bzw. abgebrochen werden. In beiden Fällen wird wieder zur Formular-Activity gewechselt, vgl. Abb. 6.3. Der Aufbau und Ablauf des Prototyps wird also durch die drei *Activities* bzw. Klassen `EyesFreeNotePad`, `EyesFreeNoteEdit` und `EyesFreeInput` gebildet. Zu jeder Klasse wird außerdem eine XML-Datei für das allgemeine Layout und die Positionierung der Bildschirmelemente beschrieben (`main.xml`, `note_edit.xml`, `text_input.xml`).

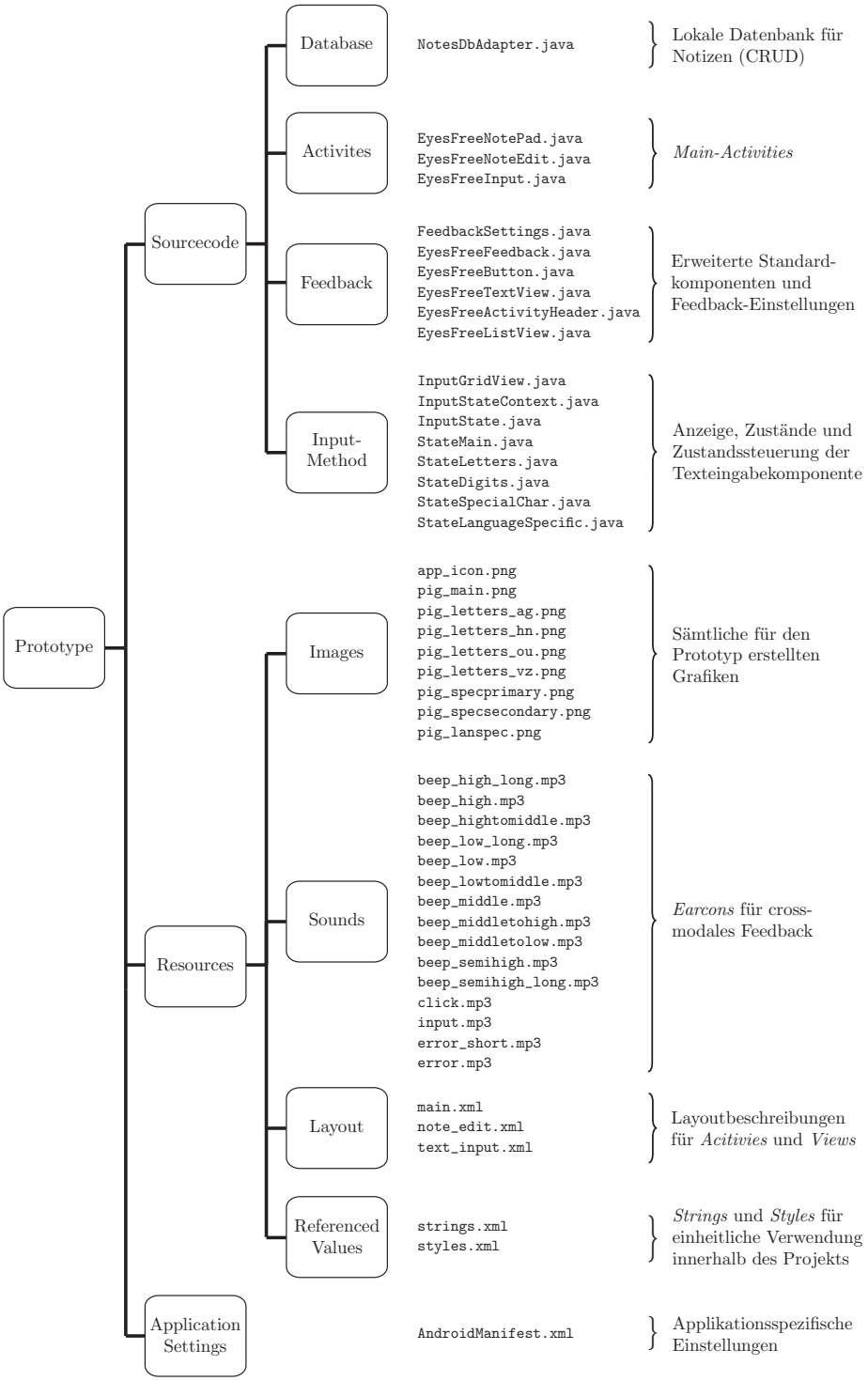
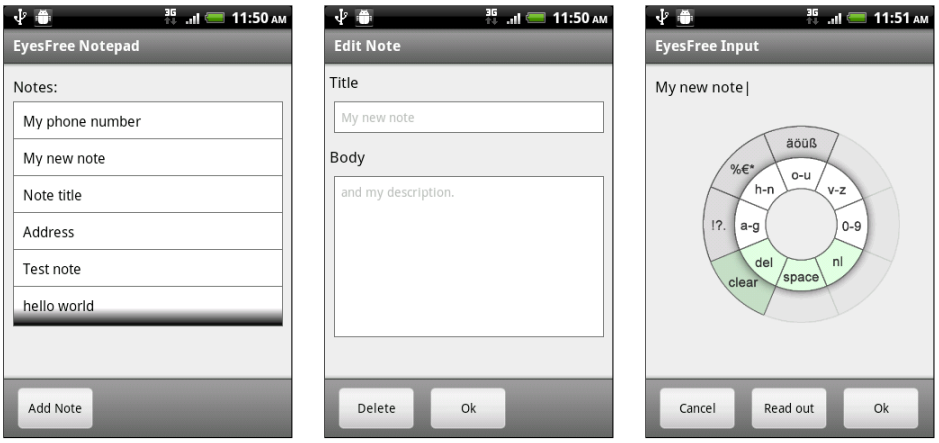


Abbildung 6.1: Schematische Darstellung des *Eclipse* Projektordners mit allen verwendeten Ressourcen.



(a) (b) (c)

Abbildung 6.2: Screenshots des Prototyps mit den drei *Activites* der Hauptansicht (a), Formularanzeige (b) und Texteingabe (b).

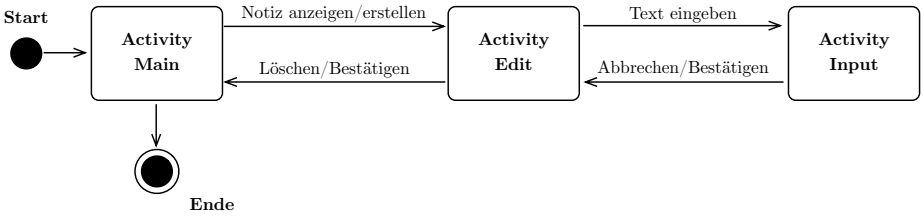


Abbildung 6.3: Anwendungszustände und Abfolge auf Basis der drei *Activities Main* (a), *Edit* (b) und *Input* (c).

Lokale Datenbank

Für das persistente Speichern und Verwalten der Notizen wurde die integrierte *SQLite* Datenbank verwendet. Zugriff und Verwendung der Datenbank wurden vom NotePad-Tutorial[2] übernommen (Klasse `NotesDbAdapter`).

Feedback allgemein

Um einheitliches Feedback für wiederkehrende Interaktionen bereitzustellen, wurde die Klasse `EyesFreeFeedback` erstellt. In dieser Klasse kann erstens mittels Zugriff auf den *SystemContext* von Android der Vibrationsmotor für haptisches Feedback angesprochen werden, und zweitens auch durch Einsatz der *Text-To-Speech Library* von Google eine Sprachausgabe gegeben

[2]Offizielles Android Developer NotePad-Tutorial für das persistente Speichern von Daten: http://developer.android.com/resources/tutorials/notepad/index.html, Kopie auf CD-ROM unter /quellen/W21_tutorial/.

Abbildung 6.4: Screenshot bei ausgeklapptem *Options*-Menü für das Deaktivieren bzw. Aktivieren von TTS, Earcons und Tactons.

werden. Je nachdem, mit welcher Konstante, bezogen auf eine bestimmte Aktion, die `performFeedback`-Methode aufgerufen wird, wird die entsprechende Rückmeldung wiedergegeben. Diese Klasse wird von allen erstellten Klassen für blind bedienbare UI-Elemente (`EyesFreeButton`, etc.) für die Feedbackwiedergabe verwendet. Im *Options*-Menü der Haupt-Activity kann dabei gewählt werden, welche Art von Feedback aktiviert bzw. deaktiviert sein soll, siehe Abb. 6.4.

Text-To-Speech und der Vibrationsmotor

Beim Starten des Prototyps ist es erforderlich zu überprüfen, ob am Smartphone die für die Sprachausgabe benötigte *Text-To-Speech Library* installiert ist. Ist dies nicht der Fall, so wird man vom Prototypen aus zum *Android-Market* und der zu installierenden Bibliothek weitergeleitet, um dies nachzuholen. Mit der dadurch zur Verfügung stehenden Methode `speak` werden dann Text und Elementbeschreibungen bei aktiviertem TTS ausgegeben.

Beim Vibrationsmotor kann bei *Android* sowohl eine bestimmte Dauer für haptisches Feedback, als auch ein beliebiges Muster abwechselnder *Vibrationen* und *Pausen* festgelegt werden. Es ist aber keine Stärke der Vibrationen einstellbar.

Crossmodales Feedback

Für crossmodale Rückmeldung wurden mit dem Programm *Audacity*[3] (Version 1.3.13) verschiedene *Earcons* erstellt (vgl. Tabelle 6.1), um diese dann mit

[3]Freie Open-Source-Software zur Aufnahme und Bearbeitung von Tönen, siehe http://audacity.sourceforge.net

Earcon	Beschreibung	Frequenz	Dauer
`beep_low`	einfacher kurzer Ton	$1kHz$	$80ms$
`beep_low_long`	doppelte Dauer zu `beep_low`	$1kHz$	$160ms$
`beep_middle`	höherer Ton als `beep_low`	$2kHz$	$80ms$
`beep_middle_long`	doppelte Dauer zu `beep_middle`	$2kHz$	$160ms$
`beep_semihigh`	höherer Ton als `beep_middle`	$3kHz$	$80ms$
`beep_semihigh_long`	doppelte Dauer zu `beep_semihigh`	$3kHz$	$160ms$
`beep_high`	höherer Ton als `beep_semihigh`	$4kHz$	$80ms$
`beep_high_long`	doppelte Dauer zu `beep_high`	$4kHz$	$160ms$
`beep_lowToMiddle`	kurzer „steigender“ Sweep	1-$2kHz$	$80ms$
`beep_middleToLow`	kurzer „abfallender“ Sweep	2-$1kHz$	$80ms$
`beep_middleToHigh`	kurzer „steigender“ Sweep	2-$4kHz$	$80ms$
`beep_highToMiddle`	kurzer „abfallender“ Sweep	4-$2kHz$	$80ms$
`click`	einem *Mausklick* zuordenbares *Earcon*	-	$10ms$
`input`	ähnlich einem Tastenanschlag	-	$10ms$
`error`	*Earcon* zur Betonung eines Fehlers	-	$350ms$
`error_short`	kurze Fehlerrückmeldung	-	$150ms$

Tabelle 6.1: Erstellte *Earcons* für sämtliche Interaktionen im Prototyp.

geeigneten, gleich parametrigen *Tactons* zu kombinieren. Bei einem Element wie einem *Button* wird ein Ton beispielsweise zwei mal hintereinander abgespielt, was auch für das *Tacton* eine zweimalige Vibration für eine bestimmte Dauer bedeutet. Ähnlich ist dies, im Vergleich zu bestimmten *Earcons*, auch bei der Änderung der Tonhöhe durch einen Sweep der Fall. Auch dabei ist eine Veränderung innerhalb des Vibrationsmusters im *Tacton* wahrzunehmen. Da die Wahl der Parameter für Vibrationen sehr eingeschränkt ist (nur Vibrationsdauer bestimmbar), werden einige Tactons für mehrere Aktionen genutzt. In Tabelle 6.2 ist genau aufgeschlüsselt zu welcher Aktion welches Feedback gegeben wird. Auch wenn die Kombinationen der *Earcons* und *Tactons* für das crossmodale Feedback bei manchen Aktionen überlappend ausfallen, so sind sie trotzdem im jeweiligen Kontext zueinander unterschiedlich. Es treten also nur kontextübergreifend gleiche akustische und haptische Rückmeldungen auf.

Die Tabelle 6.2 listet außerdem jene Stellen und Interaktionen im Prototypen auf, welche durch eine Sprachausgabe (TTS) unterstützt werden. Das ist bei allen Aktionen zur Identifikation von Elementen und bei der Zeicheneingabe der Fall und ersetzt dabei zwangsweise die Ausgabe der dazugehörigen *Earcons*.

6.2.3 Navigation

Wie im Konzept beschrieben, wird bei der Navigation darauf gesetzt, relevante Bildschirmelemente für blinde und sehbehinderte Menschen erkennbar zu machen. Beim implementierten Prototypen sind dies insgesamt drei verschiedene Komponenten die erweitert wurden.

Kontext	Aktion	TTS	Earcon	Tacton
Allgemein	Click bzw. Tap auf Interaktionselement	nein	`click`	50ms
	Element betreten	nein	`lowToMiddle`	30ms, 100ms
	Element verlassen	nein	`MiddleToLow`	100ms, 30ms
Texteingabe	Leere Kachel gewählt, Fingerposition überschreitet Polargittergrenzen	nein	`error_short`	200ms
	Polargitter abgeschnitten dargestellt	nein	`error`	200ms
	Finger über innerem Ring	nein	`middle`	40ms
	Finger über äußerem Ring	nein	`semihigh`	60ms
	Kachelgrenzen überquert	nein	`middle_long`	50ms
	Zeichen eingeben	ja	`input`	50ms
Button	Identifizieren (positiv)	ja	2 mal `semihigh`	40ms, 40ms
	Identifizieren (neutral)	ja	2 mal `middle`	2 mal 60ms
	Identifizieren (negativ)	ja	2 mal `low`	2 mal 90ms
Textfeld	Identifizieren	ja	3 mal `middle`	3 mal 40ms
Liste	Trennstrich von Listenelement überquert	nein	`middle_long`	50ms
	Zum Ende der Liste gescrollt	nein	`low_long`	2 mal 100ms
Header	Header betreten	nein	`middleToHigh`	50ms, 120ms
	Header verlassen	nein	`highToMiddle`	120ms, 50ms
	Identifizieren	ja	`high`	50ms

Tabelle 6.2: Übersicht des crossmodalen Feedbacks für auftretende Interaktionen im Prototyp inklusive der Option für aktiviertes TTS.

Erweiterte Standardkomponenten

Zu den erweiterten Komponenten sind in erster Linie die *Buttons* zu zählen. Diese werden in jeder *Activity* verwendet. Bei der Haupt-Activity (Abb. 6.2) kommt weiters eine anzupassende `ListView` und in der *Formular-Activity* die `TextView` zum Einsatz. Die *Header* der einzelnen *Activities* wurden durch eine entsprechende `TextView` ersetzt. Insgesamt wurden also drei Komponenten erweitert. Die Klassen dieser Standardkomponenten auf Android wurden demnach abgeleitet und mit Methoden ergänzt, die eine blinde Lokalisierung und Identifikation möglich machen. In der jeweiligen *Activity* wird dabei ein auftretendes *TouchEvent* abgefangen bevor es an die einzelnen Interaktionselemente weitergegeben wird. So werden bei einem *TouchMove*-Event alle Elemente der *Activity* auf ihre *BoundingBox* und der Touchposition überprüft und je nach Fingerbewegung am Display das Feedback für das Betreten oder Verlassen eines Elements gegeben. Bei einem direkten *Tap* oder *Press* auf ein Element, werden die entsprechenden *EventHandler* in den erweiterten Komponentenklassen ausgeführt. Bei einem *Long-Click* bzw. einem *Press* wird so die Wiedergabe des aktivierten Feedbacks zur Identifikation eingeleitet, vgl. Abb. 6.5 (a).

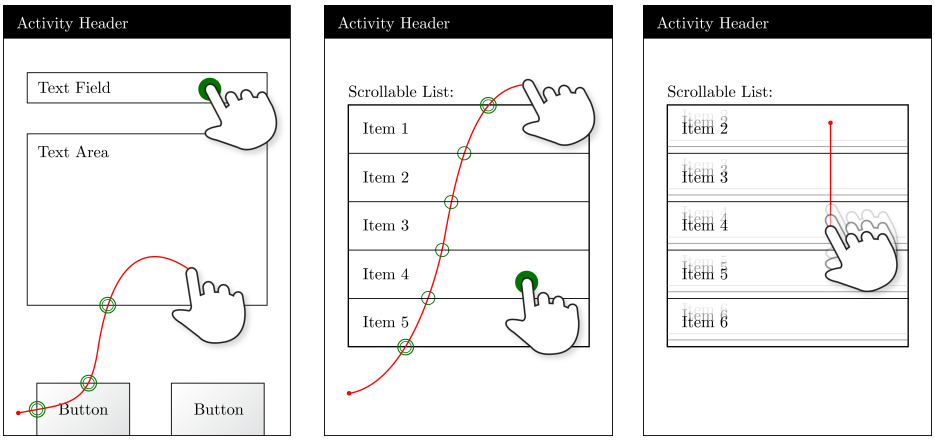


Abbildung 6.5: Identifizieren von Elementen durch ein *Press* (grüner Punkt) und Lokalisieren von Bildschirmelementen durch *Scannen* des Displays mit dem Finger (rote Linie) mit Feedback (grüne Kreise) bei Kantenüberquerung (a). Explorieren von Listenelementen durch initialen Berührungspunkt des Fingers *außerhalb* der Liste mit unterschiedlichem Feedback bei Überquerung des Listenrahmens und der Trennstriche der einzelnen Elemente (b). Scrollen der Liste durch Fingerbewegung mit Startpunkt *innerhalb* der Liste (c).

Bei der `ListView` in der Haupt-Activity benötigten die Methoden zur Erkennung der einzelnen Listenelemente gesonderte Aufmerksamkeit. Erstens war eine Möglichkeit zu schaffen, die einzelnen Elementgrenzen erfahrbar zu machen, und zweitens musste auch die Möglichkeit beibehalten werden, die Elemente in der Liste zu scrollen. Dieses Problem wurde gelöst, indem eine Fingerbewegung in der Liste, die *außerhalb* der Liste startet, zum Erkennen der einzelnen Elementpositionen herangezogen wird, und eine Fingerbewegung, welche *innerhalb* der Liste beginnt, zum hinauf- und hinunterscrollen in der Liste dient, vgl. Abb. 6.5 (b, c). Wenn dabei das Ende bzw. der Anfang der Liste erreicht wird, wird wiederum entsprechende Rückmeldung gegeben. Weiters wird pro verschobenem Notizeintrag in der Liste ein *Earcon* und *Tacton* wiedergegeben.

6.2.4 Texteingabemethode

Für die Texteingabe innerhalb der *Activity* `EyesFreeInput` wurde eine Reihe zusätzlicher Klassen erstellt. `InputGridView` zeigt das polare Gitter an der aktuellen Fingerposition an. Alle weiteren Klassen sind für die Zustandssteuerung des Eingabelayouts zuständig (`InputState`, `InputStateContext`, `StateDigits`, `StateLetters`, etc.).

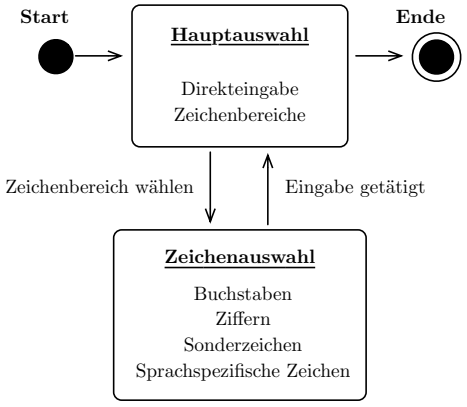


Abbildung 6.6: Zustandsdiagramm der Texteingabekomponente.

Aufbau der Texteingabekomponente

Die Auswahl eines Zeichens wurde auf zwei Schritte bzw. Ebenen aufgeteilt. Demnach gibt es auch verschiedene Zustände der Texteingabekomponente während einer Eingabe. Für diese erforderliche Zustandssteuerung wurde auf das Entwurfsmuster *Zustandsmuster*[4] zurückgegriffen. Beim ersten Aufkommen des Fingers am Display wird das Hauptgitter (`StateMain`) angezeigt. Je nach Ende der Touchbewegung wird der Zustand auf eine der Unterebenen zur Zeichenauswahl (Buchstaben, Ziffern, etc.) geändert. Bei erneutem Aufkommen des Fingers wird dann der neue Zustand angezeigt und wiederum je nach Touchendposition das entsprechende Zeichen ausgewählt. Dieser Ablauf ist vereinfacht in Abb. 6.6 dargestellt. Bei einer fehlerhaften Eingabe wird automatisch zur Hauptauswahl zurückgewechselt. Dies ist bei Wahl einer leeren Kachel, und dem Abheben des Fingers außerhalb der Ringe der Fall. Für Direktauswahl von Aktionen wie *Zeichen löschen* oder *Leerzeichen einfügen* werden keine eigenen Zustände benötigt. Die Zustände beschränken sich also auf die Hauptauswahl und die verschiedenen Unterebenen zur Zeichenauswahl. Jedem Zustand ist dabei ein Set von Grafiken zugeordnet, vgl. Abb. 5.7 bis 5.10.

Bestimmung der Eingabe

Für die Bestimmung der Zeicheneingabe wird die aktuelle Fingerposition bzw. -bewegung am Bildschirm mit der Kachelaufteilung der Texteingabekomponente verglichen. Die Kacheln sind dabei aber keine eigenen virtuellen

[4]Ein Entwurfsmuster (Verhaltensmuster) eingesetzt zur Kapselung unterschiedlicher, zustandsabhängiger Verhaltensweisen eines Objektes, siehe http://de.wikipedia.org/w/index.php?title=Zustand_(Entwurfsmuster)&oldid=87781654, Kopie auf CD-ROM unter /quellen/W23_states/.

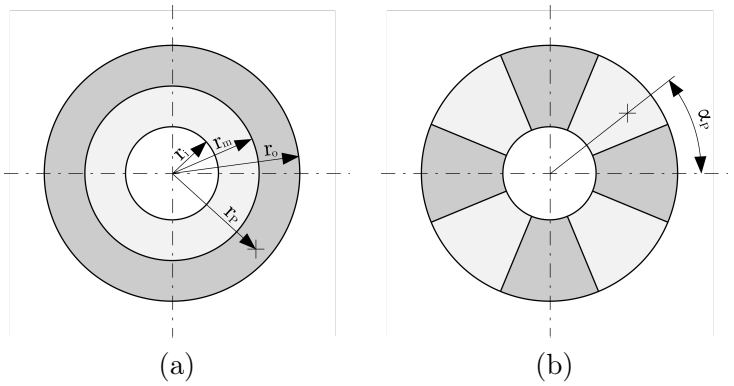
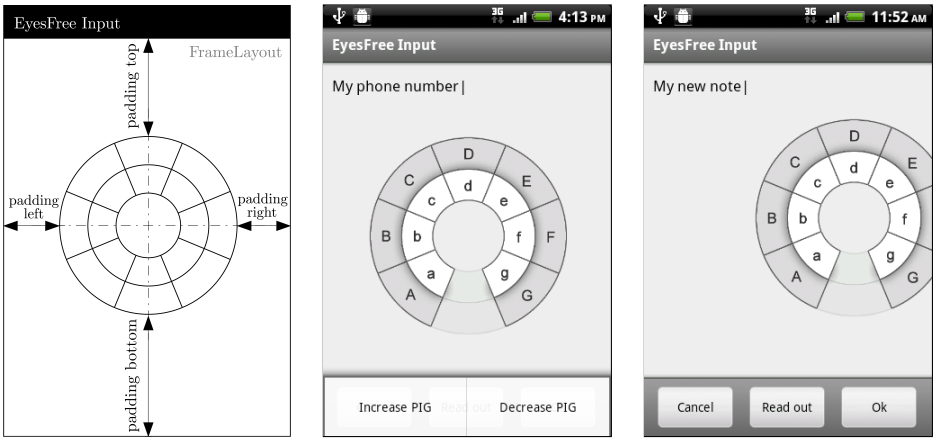


Abbildung 6.7: Abstandsbestimmung (a) des Berührungspunktes r_P zum Zentrum im Vergleich zum *inneren*, *mittleren* und *äußeren* Radius (r_i, r_m, r_o). Berechnung des Winkels α_P vom Punkt der Fingerberührung und Zuordnung zu einem der Sektoren, abgegrenzt durch radiale Unterteilungen (b).

Tasten im Sinne von einem *Button* auf *Android*. Es handelt sich lediglich um eine polare Gitterunterteilung, die durch die zwei Ringe und mehrere radial verteilte Sektoren in diesen Ringen gebildet wird. Um nun die vom Nutzer gewählte Kachel bestimmen zu können, wird zuerst festgestellt in welchem Ring sich der Finger befindet (Abstand zwischen Start- und Endpunkt der Touchbewegung, vgl. Abb. 6.7 (a)). Danach wird der Winkel vom Zentrum des Eingabegitters zum Punkt der Berührung berechnet und einem Sektor zugeordnet (b). Durch Kombination des erhaltenen Rings und Sektors ist somit auch die Kachel vollständig bestimmt und die zugehörige Aktion kann ausgeführt werden.

Anzeige der Texteingabekomponente

Da das vielversprechende Layout `AbsoluteLayout` für die Positionierung von Objekten in einer Ansicht leider in der *Android API 2.2* schon als *deprecated* markiert ist, kann diese Layoutvariante nicht dazu verwendet werden um absolute Positionen für ein Element anzugeben. Deshalb wurde auf das `FrameLayout` ausgewichen und das *Padding* wird links, rechts, ober- und unterhalb der Texteingabekomponente und der Fingerposition gesetzt, vgl. Abb. 6.7 (a). Das *Padding* errechnet sich unter Berücksichtigung der Berührungsposition, dem Außenradius des Polargitters, sowie Breite und Höhe der Anzeige. Die darzustellende Größe der Texteingabekomponente kann dabei stufenweise vergrößert und verkleinert werden, siehe Abb. 6.8 (b). Je kleiner die Darstellung gewählt wird, desto größer ist die Toleranz am Display um das komplette Eingabegitter anzeigen zu können. Dafür sind die Eingabefelder aber auch kleiner und die Bedienung erfordert höhere Präzision. Sollte das Gitter bei entsprechenden Fingerpositionen am Display nicht mehr voll-



(a) (b) (c)

Abbildung 6.8: Positionierung der Texteingabekomponente am Display durch Bestimmung des *Paddings* (a). Options-Menü für das stufenweise Einstellen der Darstellungsgröße des *Polar-Input-Grids* (*PIG*). Abgeschnittene Darstellung des Eingabegitters wegen initialem Berührungspunkt zu nahe am Displayrand (c).

ständig angezeigt werden können, wie beispielsweise in Abb. 6.7 (c) gezeigt, so wird entsprechendes Feedback gegeben. Es kann trotzdem noch eine Eingabe durchgeführt werden, jedoch nur mit jenen Zeichen, die noch sichtbar sind. Das *Error*-Feedback dient also lediglich als Warnung.

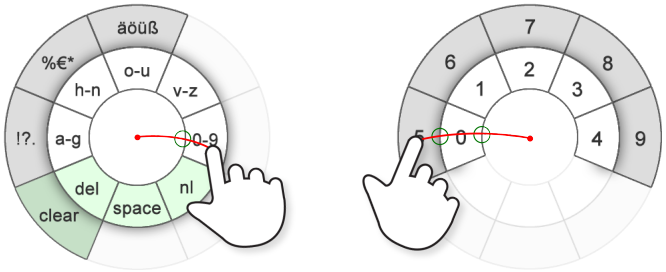
6.3 EyesFree-NotePad Bedienung

6.3.1 Installieren und Starten der Anwendung

Da die App zum aktuellen Zeitpunkt nicht im *Android Market* veröffentlicht ist, kann diese nur manuell installiert werden. Auf http://www.androidpit.de/de/android/wiki/view/APK_manuell_installieren ist das manuelle Installieren von Apps auf Android beschrieben. Hierfür ist es zuerst notwendig, die App bzw. Datei `Prototype.apk` von der CD-ROM unter /applikation/ auf die SD-Karte des Smartphones zu kopieren. Mit einem Dateimanager auf dem Gerät kann die App durch einen Click bzw. *Tap* installiert werden. Zuvor sollte sichergestellt werden, dass unter dem Menüpunkt *Anwendungen* das Häkchen für *Unbekannte Quellen* gesetzt ist.

Zum Starten der App genügt nun ein *Tap* auf das Programmicon, siehe Abb. 6.9. Der Prototyp überprüft dabei ob die TTS-Library auf dem Smartphone installiert ist, und öffnet gegebenenfalls die *Android-Market App* zum Nachinstallieren dieser, für die App notwendige Bibliothek.

Abbildung 6.9: Das Icon des Prototypen *EyesFree-NotePad*.



(a) (b)

Abbildung 6.10: Beispiel für das Eingeben der Ziffer *5* durch Auswahl der zugehörigen Unterkategorie (a) und der entsprechenden Kachel (b).

6.3.2 Anwendungsoberfläche

Da in der internen Datenbank der App beim ersten Starten des Prototypen noch keine Einträge vorhanden sind, wird auch die Listenansicht noch leer angezeigt. Mit dem Button *Add Note* kann eine Notiz hinzugefügt werden. Man gelangt hierbei zur Formularansicht, in der ein Titel und eine Beschreibung für die zu erstellende Notiz angegeben werden kann. Bei einem *Tap* auf die Textfelder wird die Texteingabemethode geöffnet um einen Text einzugeben. Das Eingabefeld wird durch Kontakt des Fingers mit dem Display angezeigt und auf den Kontaktpunkt zentriert. Nach der Fingerberührung, und der Positionierung der Ringe um den Berührungspunkt behält das polare Eingabegitter seine Position während nachfolgender Fingerbewegungen am Display bei. Eine Aktion oder ein Zeichen wird ausgewählt, indem der Finger bei der entsprechenden Kachel wieder vom Display abgehoben wird, vgl. Abb. 6.10. Beim Abheben des Fingers vom Bildschirm wird die Texteingabekomponente zur Hilfestellung noch transparent in der Bildschirmmitte dargestellt. Verweilt der Finger länger als zwei Sekunden auf der selben Kachel, so wird per Sprachausgabe das Zeichen ausgegeben. Die Eingabe kann mit den drei Buttons *Cancel*, *Read Out* und *OK* entweder abgebrochen, überprüft oder bestätigt und übernommen werden, vgl. Abb. 6.2.

Kapitel 7

Diskussion

7.1 Ergebnisse

Mit der Implementation des Prototypen *EyesFree-NotePad* liegt nun eine App vor, die auf einem Touchphone auf gänzlich nicht-visueller Basis bedient werden kann. Mit kleinen Einschränkungen wurden alle aus Kapitel 4 beschriebenen Vorschläge und Richtlinien für ein auf nicht-visuelle Interaktionen basierendes Interface umgesetzt.

Respektive Punkt *1* in Abschnitt 4.3 kann der Nutzer zu jedem Zeitpunkt abfragen, in welchem Zustand bzw. welcher Ansicht sich die Anwendung befindet. Theoretisch muss dafür nicht auf das Display geblickt werden.

Entsprechend Vorschlag *2* bzw. *3* wurde darauf geachtet, dass möglichst keine Vorkenntnisse für die Bedienung und das Interface vorhanden sein müssen. Im Vorfeld muss dem Nutzer jedoch bewusst sein, wie die Technik für die Exploration der Bildschirmoberfläche und das Eingeben von Text mit der entwickelten Texteingabemethode funktioniert. Der Anwender muss dabei aber keine genauen Details der Elemente wissen, da diese eben auch während der Interaktion „blind“ exploriert und erkannt werden können.

Auch dem Punkt *4* wurde bezüglich möglicher Bedienung für Links- als auch Rechtshänder nachgegangen. Die Option für beidhändige Bedienung und Multi-Touch Eingaben sind zwar in der aktuellen Version nicht umgesetzt, es kann aber in diese Richtung erweitert werden, siehe Abschnitt 7.4.

Gemäß der Richtline *5* ist der Prototyp weder bei der Navigation, noch bei der Texteingabe von einer bestimmten Sprache oder bestimmtem Vokabular abhängig. Die App ist zwar in englischer Sprache umgesetzt, würde jedoch in gleicher Weise auch beispielsweise unter Verwendung von Deutsch funktionieren. Es müssten lediglich die Bezeichnungen der Elemente je nach Sprache ausgetauscht, die Sprache der TTS-Komponente umgestellt, und die sprachspezifischen Zeichen bei der Texteingabe angepasst werden.

Bezüglich Punkt *6*, also der Größe der Interaktionselemente, muss eingeräumt werden, dass die empfohlene Mindestgröße von $10mm$ nicht durchge-

hend eingehalten wird. Besonders die Kacheln im inneren Ring der Texteingabekomponente entsprechen dieser Vorgabe als Standardeinstellung nicht. Es besteht jedoch die Möglichkeit, die Darstellungsgröße der Eingabekomponente stufenweise zu vergrößern.

Respektive Vorschlag *7* wurde crossmodales Feedback eingesetzt. Es sollte somit bei unterschiedlichen Umgebungsverhältnissen und verschiedenen Störfaktoren ein zuzuordnendes Feedback wahrgenommen werden können.

Entsprechend dem letzten Punkt *8* kann mit dem Prototypen auch diskret interagiert werden. Es sind weder auffallende Bewegungen mit dem Gerät notwendig, noch wird zwangsweise auf akustischem Wege die Aufmerksamkeit der Umgebung erregt.

7.2 Vergleich mit anderen Methoden

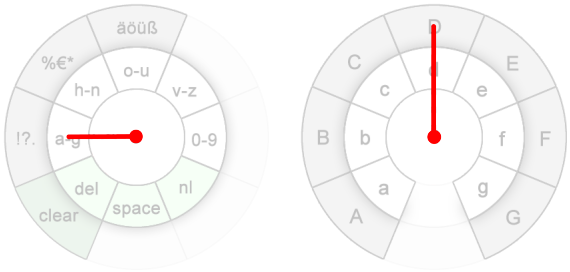
Im Vergleich zu anderen Techniken für nicht-visuelle Interaktion konnten mit *EyesFree-NotePad* die Vorteile bezüglich „blinder“ Interaktion der einzelnen Methoden kombiniert werden. Nachteile wie das Einprägen von bestimmten Sprachbefehlen oder speziellen Gesten treten hier nicht auf.

Ein mögliches Defizit liegt eventuell in der Geschwindigkeit oder auch Genauigkeit der Texteingabe. Da noch keine Studie mit dem Prototypen dazu gemacht, und genaue Daten erhoben wurden, kann derzeit noch nicht nachgewiesen werden, dass die entwickelte Texteingabemethode bezüglich Texteingabegeschwindigkeit und -genauigkeit mit anderen Methoden mithalten oder diese übertreffen kann. Entsprechend den anderen Methoden, welche auf Gesten (z.B. das *Graffiti Alphabet*) basieren, werden auch beim *EyesFree-NotePad* vergleichbare Werte erwartet. Genau genommen kann die umgesetzte Variante als eine Kombination von einer oder zwei aufeinanderfolgenden Gesten, gestützt durch eine visuelle Darstellung, verstanden werden. Wobei es bei der Geste nicht um den Verlauf des Bewegungspfades geht, sondern nur um den Start- und Endpunkt, vgl. Abb. 7.1.

7.3 Verbesserungsmöglichkeiten

7.3.1 Zeichenanordnung

Die Anordnung der Zeichen, Aktionen, sowie Unterebenen der Texteingabekomponente ist in dieser Version als erster Vorschlag zu verstehen. Möglicherweise versprechen andere Aufteilungen im Polargitter bessere Zugänglichkeit und Eingabegeschwindigkeiten. Diese Frage sollte jedoch in einer eigenen Studie untersucht werden. Eine Alternative wäre beispielsweise, die Groß- und Kleinbuchstaben nicht in einer Unterkategorie gemeinsam zu verteilen, sondern separat in getrennten Ebenen zu platzieren.



(a) (b)

Abbildung 7.1: Beispiel für den Vergleich zu gestenbasierter Eingabe bei Zeichenauswahl *D*. Die erste *Geste* wählt die Unterebene (a), die zweite den Großbuchstaben *D* (b).

(a) (b)

Abbildung 7.2: *Text* (a) und *Phone* (b) als zwei Arten einer Eingabemethode auf *Android*.

7.3.2 Android IME

Die Texteingabekomponente im Prototypen ist als fest integrierter Bestandteil der App umgesetzt. Es besteht auf Android jedoch auch die Möglichkeit, eine eigene *InputMethod* (IME) zu implementieren. Es handelt sich bei einer solchen Umsetzung dann um eine zu installierende virtuelle Tastatur. Diese kann dadurch auch von jedem Eingabefeld im Betriebssystem für eine Eingabe aufgerufen werden. Auf *Android Developers* wurde ein eigener Artikel[1] zu dem Thema veröffentlicht. Dieser beschreibt die Umsetzung einer solchen im System integrierten Eingabemethode und unterscheidet dabei auch zwischen verschiedenen Arten der Eingabemethode, die dabei berücksichtigt werden sollten, vgl. Abb 7.2.

[1] http://developer.android.com/resources/articles/creating-input-method.html, Kopie auf CD-ROM unter /quellen/W5_ime/.

7.3.3 Feedback der Liste

Als problematisch erwies sich die Umsetzung des Feedbacks für die Liste der Notizen. Die in der aktuellen Version implementierte Variante weist teilweise unpräzises Rückmeldeverhalten beim Scrollen der Liste auf. Die Strecke der Touchbewegung kann nicht wie umgesetzt auch eins zu eins dafür verwendet werden, um zu bestimmen, wie weit nach unten oder oben gescrollt wurde. Das liegt daran, dass eine kurze Verzögerung stattfindet, bis die Fingerbewegung vom System als Scroll interpretiert wird. Der Fehler liegt also in der Strecke der Touchbewegung, welche während dieser Verzögerung unberücksichtigt bleibt. Auf dieses Verhalten wurde nicht näher eingegangen und somit blieben etwaige Ungenauigkeiten ohne Berücksichtigung.

7.4 Mögliche Erweiterungen

7.4.1 Morsecode

Neben der Möglichkeit sich über die Taste *Read Out* den eingegebenen Text vorlesen zu lassen, könnte auch untersucht werden, ob eine lautlose Variante der Textausgabe sinnvoll umgesetzt werden kann. Womöglich könnte die Zeichenfolge als Morsecode in Form von Vibrationen dem Nutzer kenntlich gemacht werden. Natürlich vorausgesetzt, dass der Anwender diesen Morsecode versteht.

7.4.2 Vordefinierte Strings

Da der verfügbare Platz in dem polaren Eingabegitter noch nicht komplett ausgeschöpft ist, wäre es auch denkbar, die momentan leeren Kacheln für weitere Zeichensammlungen zu nutzen. Eine zusätzliche Unterebene könnte dazu verschiedene vordefinierte Wörter oder Zeichenfolgen beinhalten (Namen, Email-Adressen, Telefonnummern, etc.). Eine andere ergänzende Kategorie könnte für vom Nutzer definierte Zeichenfolgen reserviert werden. Dadurch wären häufige Eingaben einfacher und auch schneller zu tätigen.

7.4.3 Kopieren und Einfügen

Neben weiteren Zeichenbereichen können in gleicher Weise auch noch andere Aktionen als Direktauswahl hinzugefügt werden. Das *Kopieren* und *Einfügen* von Text ist hier sehr naheliegend. Es muss dabei aber auch noch überlegt werden, wie das nicht-visuelle Markieren von Textsequenzen realisiert werden kann.

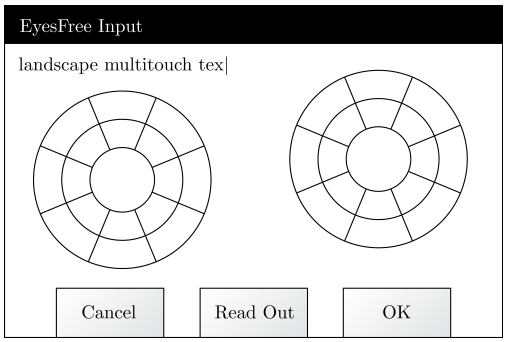


Abbildung 7.3: Mögliches beidhändiges Eingeben von Text per Multitouch im *Landscape*-Modus.

7.4.4 Sprache

Der Prototyp ist in der ersten Version in englischer Sprache umgesetzt. Als Erweiterung könnte in einem der *Option*-Menüs eine Auswahl verschiedener anderer Sprachen gegeben werden. Sämtliche Bezeichnungen und Beschreibungen von Elementen in *EyesFree-NotePad* sowie TTS könnten so auf eine andere Sprache umgestellt werden.

7.4.5 Multitouch und beidhändige Bedienung

Für eine beidhändige Eingabe von Text könnte die Texteingabe bei unterstütztem Multitouch im *Landscape*-Modus erfolgen, vgl. Abb. 7.3. Die Umsetzung kann dann so aussehen, dass mit der im Bild dargestellten Texteingabekomponente auf der linken Seite, zur Eingabe mit der linken Hand (Daumen), ein Zeichen eingegeben werden kann, und dies im gleichen Sinne auch für die rechte Hand auf der rechten Bildschirmhälfte gilt. Alternativ könnte eine Eingabe aber auch durch gleichzeitig getätigte Gesten erfolgen. Die erwähnte zweiteilige Kombination einer Geste für ein bestimmtes Zeichen (Abb. 6.10) könnte aufgeteilt auf das linke und rechte Eingabegitter somit getrennt und simultan erfolgen. Ob dadurch auch die doppelte Eingabegeschwindigkeit erreicht werden kann, wäre durch eine eigene Studie zu klären.

Kapitel 8

Schlussbemerkungen

Mit dieser Arbeit wurden Methoden für nicht-visuelle Interaktionen bei Smartphones mit Touchscreen untersucht. Aus einer Mischung von nicht-visuellen Ansätzen speziell für Mobilgeräte, aber auch angewendeter Techniken im digitalen Alltag eines sehbehinderten Menschen wurden die wichtigsten Aspekte für nicht-visuelle Interaktion ausgewertet. Das bezog sich besonders auf Smartphones, welche ausschließlich auf eine virtuelle Tastatur am Touchdisplay, anstatt physischer Tasten setzen. Schließlich wurde ein Prototyp in Form einer *Android* Applikation als Umsetzungsvariante für nicht-visuelle Interaktion konzipiert und umgesetzt. Dieser Prototyp umfasst sowohl das „blinde“ Navigieren durch eine Anwendung, als auch das Eingeben von Text, ohne dabei den Blick auf das Display richten zu müssen.

Diese Form der Interaktion ist in erster Linie für sehbehinderte Menschen von Bedeutung, da so die Zugänglichkeit zu Geräten, welche stark auf visuelle Interfaces ausgelegt sind, bzw. deren Bedienung erleichtert oder erst möglich gemacht werden kann. In gleicher Weise kann diese Option aber auch für sehende Personen vorteilhaft sein, da in Situationen, welche erhöhte Aufmerksamkeit erfordern, der Blick nicht notwendigerweise von der Umgebung abgewendet und auf das Gerät gerichtet werden muss.

Das Endergebnis des Prototyps ist, in Anlehnung an die erarbeiteten und auch theoretisch erfüllten Kriterien, sehr vielversprechend. Daher wäre in weiterer Folge eine ausführliche Studie naheliegend, um dies auch nachweislich verifizieren zu können oder aufzuzeigen, welche Details nicht bedacht wurden.

8.1 Ausblick

Die erste Version des Prototyps *EyesFree-NotePad* bietet ein paar Stellen zur Verbesserung und Erweiterung. Besonders auf die Umsetzung als *Android Input Method* sollte näher eingegangen werden. Dementsprechend könnte der Teil der Texteingabe von dem Prototypen ausgegliedert und als separat

zu installierende Texteingabemethode zur Verfügung gestellt werden. Als gleichbedeutend ist hier aber auch die Multitouchfähigkeit und beidhändige Bedienung für die Texteingabe zu sehen. Diese könnte sich besonders im *Landscape*-Modus als eine schnellere Option der Texteingabe herausstellen, und sollte deshalb näher verfolgt werden.

Weiters ist auch denkbar, die vorgestellten Varianten für die Navigation auf einer virtuellen Bedienoberfläche, und die Texteingabemethode, auch auf anderen Geräten mit Touchscreens zu nutzen. Die Verwendung muss nicht zwangsweise nur auf Smartphones beschränkt werden. Gegebenenfalls sind diese Techniken besonders auch für digitale Geräte im Unterhaltungsbereich wie beispielsweise bei Fernbedienungen mit Touchdisplays interessant.

Anhang A

Inhalt der CD-ROM

Format: CD-ROM, Single Layer, ISO9660-Format

A.1 Masterarbeit

Pfad: /

im09012.pdf	Masterarbeit

A.2 Applikation

Pfad: /applikation/

Prototype.apk	*Android 2.2* Prototyp

A.3 Quellcode

Pfad: /quellcode/Prototype/

res/drawable/	Erstellte Grafiken
res/layout/	Layoutbeschreibungen der Ansichten (*Activities*)
res/raw/	Verwendete *Earcons*
res/values/	Erstellte *String*- und *Style*-Sammlungen
src/	Erstellte Klassen für den Prototyp
AndroidManifest.xml . .	Applikationsspezifische Einstellungen

A.4 Bilder und Grafiken

Pfad: /bilder/

2_blackberry.png	Produktfoto *Blackberry Bold 9000*

2_capacitivetouch.eps	Funktionsbeschreibung kapazitiver Touchscreens
2_cg_doubletap.png	Geste *Double Tap*
2_cg_drag.png	Geste *Drag*
2_cg_flick.png	Geste *Flick*
2_cg_pinch.png	Geste *Pinch*
2_cg_press.png	Geste *Press*
2_cg_pressndrag.png	Geste *Press and Drag*
2_cg_pressntap.png	Geste *Press and Tap*
2_cg_rotate.png	Geste *Rotate*
2_cg_spread.png	Geste *Spread*
2_cg_tap.png	Geste *Tap*
2_htcdesire.png	Produktfoto *HTC Desire Z*
2_iphone4.png	Produktfoto *iPhone 4*
2_marktanteile.eps	Marktanteile mobiler Gerätehersteller und Betriebssysteme
2_visimp_1.png	Bildbeispiel *Normales Sehfeld*
2_visimp_2.png	Bildbeispiel *Makuladegeneration*
2_visimp_3. png	Bildbeispiel *Retinopathia Pigmentosa*
2_visimp_4. png	Bildbeispiel *Grauer Star*
2_visimp_5. png	Bildbeispiel *Grüner Star*
2_visimp_6. png	Bildbeispiel *Diabethische Retinopathie*
3_8pen1.png	Applikationsansicht *8pen*
3_8pen2.png	Funktionsbeschreibung *8pen*
3_blindtype.png	Anwendungsbeispiel *BlindType*
3_foogue1.png	Interface *Foogue*
3_foogue2.png	Anwendung *Foogue*
3_foogue1.png	Interface *Foogue*
3_graff.jpg	*Graffiti Alphabet*
3_hapterminology.png	Terminologie haptischer Wahrnehmung
3_moose1.jpg	Gerätefoto *The Moose*
3_moose2.jpg	Gerätefoto *The Moose*
3_sr_gesture1.png	Basisgeste *Slide Rule*
3_sr_gesture2.png	Basisgeste *Slide Rule*
3_sr_gesture3.png	Basisgeste *Slide Rule*
3_sr_gesture4.png	Basisgeste *Slide Rule*
3_unigest.jpg	*UniGest*-Alphabet

3_voiceactions1.png . .	Sprachbefehl *Call*
3_voiceactions2.png . .	Sprachbefehl *Listen To*
3_voiceactions3.png . .	Sprachbefehl *Go To*
3_voiceactions4.png . .	Sprachbefehl *Send Text*
3_wui.png	Screenshots *Walking User Interface*
5_activity_id1.png . .	Konzept Identifikation
5_activity_id2.png . .	Konzept *Header* Identifikation
5_borgerbuttons.png .	Tastenanordnung Touchscreen
5_borgerbuttons2.png .	Alternative Tastenanordnung Touchscreen
5_empty_pig.png . . .	leeres polares Eingabegitter
5_gmail_edit.png . . .	Screenshot *Gmail* Formular
5_gmail_input.png . .	Screenshot *Gmail* Texteingabe
5_gmail_list.png	Screenshot *Gmail* Listenansicht
5_nav_locid.png	Element-Identitikation und -Lokalisation
5_pig_buttonstyle.png	Konzept Eingabelayout
5_pig_googlestuff.png	Konzept Eingabelayout
5_pig_gridstyle.png . .	Konzept Eingabelayout
5_pig_letterdetail.png	Eingabelayout in Detailansicht einer Unterebene
5_pig_lettergroups.png	Eingabelayout der Gruppierungen
6_app_icon.png	Icon Prototyp
6_project_folder.eps .	Schematischer Aufbau des Projektordners
6_ime_ring.png	Beschreibung Positionsermittlung
6_ime_sector.png . . .	Beschreibung Positionsermittlung
6_imepos_padding.png	Positionsbestimmung
6_inputex1.png	Beispiel Texteingabe
6_inputex2.png	Beispiel Texteingabe
6_nav_elements.png .	Umsetzung Elementidentifikation
6_nav_listexplore.png .	Umsetzung Listeninteraktion
6_nav_listscroll.png . .	Umsetzung Listeninteraktion
6_ss_edit.png	Screenshot Prototyp Formularansicht
6_ss_input.png	Screenshot Prototyp Texteingabe
6_ss_input_digits.png	Screenshot Prototyp Zifferneingabe
6_ss_input_error.png .	Screenshot Prototyp Anzeigefehler
6_ss_input_menu.png	Screenshot Options-Menü
6_ss_main.png	Screenshot Listenansicht
6_ss_mainmenu.png .	Screenshot Options-Menü

6_stateactivites.eps	Zustandsdiagramm Applikation
6_stateinput.eps	Zustandsdiagramm Eingabe
7_gestencombo1.png	Beispiel Gestenkombination
7_gestencombo2.png	Beispiel Gestenkombination
7_ime_phone.png	Eingabemethode *Phone*
7_ime_text.png	Eingabemethode *Text*
7_landscape_input.png	Eingabekonzept Querformat
pig_digits.png	Zeichenanordnung Ziffern
pig_lanspec.png	Anordnung Sprachspezifischer Zeichen
pig_letters_ag.png	Zeichenanordnung Buchstaben a bis g
pig_letters_hn.png	Zeichenanordnung Buchstaben h bis n
pig_letters_ou.png	Zeichenanordnung Buchstaben o bis u
pig_letters_vz.png	Zeichenanordnung Buchstaben v bis z
pig_main.png	Aktionsanordnung
pig_specprimary.png	Anordnung primärer Sonderzeichen
pig_specsecondary.png	Anordnung sekundärer Sonderzeichen

A.5 Quellen

Jeder angegebene Ordner beinhaltet ein PDF und gegebenenfalls eine Kopie der Webseite samt Link als *Webloc*-Datei.

Pfad: /quellen/

A1_tvraman/	NYTimes Artikel: For the blind, technology does what a guide dog can't
A2_delphi/	Delphi Report: Zukünftige Informations- und Kommunikationstechniken
A3_wacom/	Funktion des kabel- und batterielosen Wacom-Stifts
A4_touchguide/	Touch Gesture Reference Guide
EF1_textentry/	Eyes-free Text Entry on a Touchscreen Phone
EF2_semfeel/	SemFeel: A User Interface with Semantic Tactile Feedback for Mobile Touch-screen Devices
EF3_moose/	The Moose: A Haptic User Interface for Blind Persons
EF4_foogue/	Foogue: Eyes-Free Interaction for Smartphones
EF5_unigest/	Text Entry Using Three Degrees of Motion

EF6_8pen/	8pen: Writing on small devices, in a fast and natural way
EF7_efa/	Google I/O - Looking Beyond the Screen: TTS and Eyes-Free Interaction on Android
EF8_blindtype/	BlindType Keyboard
EV1_evaluation/	Evaluation of Text Entry Techniques
EV2_unipad/	Unipad: Single-Stroke Text Entry With Language-Based Acceleration
EV3_igest/	Designing Gestural Interfaces
EV4_kspc/	KSPC (Keystrokes per Character) as a Characteristic of Text Entry Techniques
EV5_errors/	Measuring Errors in Text Entry Tasks: An Application of the Levenshtein String Distance Statistic
EV6_worderr/	Einführung in die Computerlinguistik
EV7_eval/	Evaluating One Handed Thumb Tapping on Mobile Touchscreen Devices
F1_tactfb/	Investigating the Effectiveness of Tactile Feedback for Mobile Touchscreens
F2_tafbmobev/	Tactile Feedback for Mobile Interactions
F3_whichmodwhen/ . .	Audio or Tactile Feedback: Which Modality When?
F4_diss/	Crossmodal Audio and Tactile Interaction with Mobile Touchscreens
F5_ciwmodev/	Crossmodal Interaction with Mobile Devices
F6_ccon/	Crossmodal Congruence: The Look, Feel and Sound of Touchscreen Widgets
VI1_itacc/	Investigating Touchscreen Accessibility for People with Visual Impairments
VI2_disab/	Freedom to Roam: A Study of Mobile Device Adoption and Accessibility for People with Visual and Motor Disabilities
VI3_wui/	Getting Off the Treadmill: Evaluating Walking User Interfaces for Mobile Devices in Public Spaces
VI4_sliderule/	Slide Rule: Making Mobile Touch Screens Accessible to Blind People Using Multi-Touch Interaction Techniques
W1_smartph/	How Exactly Do Cell Phones Become Smartphones

W2_sphonewiki/	Smartphone Wikipedia
W3_softkey/	Softkey Wikipedia
W4_revcontr/	Revolution Controller: Wii Remote
W5_ime/	Creating an Input Method | Android Developers
W6_iphone/	iPhone Wikipedia
W7_visam/	VISAM Prozesstechnik: Touchscreen Technologien
W8_touchwiki/	Touchscreen Wikipedia
W9_voiceactions/ . . .	Voice Actions for Android
W10_einftouchs/ . . .	Touchscreen Technologies
W11_stiftfinger/	Fingerstift für kapazitive Touchscreens
W12_procon/	Comparison of Finger-Operated vs. Stylus-Operated Touchscreens
W13_mtouchwiki/ . . .	Multi-Touch-Screen Wikipedia
W14_visimpwiki/ . . .	Sehbehinderung Wikipedia
W15_visimpex/	Lippischer Blinden- und Sehbehindertenverein: Sehbehinderungen
W16_sprstwiki/	Sprachsteuerung Wikipedia
W17_sprsyntwiki/ . . .	Sprachsynthese Wikipedia
W18_braillewiki/	Brailleschrift Wikipedia
W19_crudwiki/	CRUD Wikipedia
W20_sweepwiki/	Sweep Wikipedia
W21_tutorial/	NotePad Tutorial
W22_nielson/	Nielson Smartphone Statistik
W23_states/	Zustandsmuster Wikipedia

Literaturverzeichnis

[1] 3qubits: *The 8 Pen – Reinventing the keyboard for touch-enabled devices*, 2011. http://www.the8pen.com, Kopie auf CD-ROM (Dateien /quellen/EF6_8pen/8pen.pdf und /quellen/EF6_8pen/The8pen.mp4 vom 30.05.2010).

[2] Brewster, S., F. Chohan und L. Brown: *Tactile feedback for mobile interactions*. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, S. 159–162, New York, NY, USA, 4/5 2007. ACM.

[3] Castellucci, S. J. und I. S. MacKenzie: *Unigest: text entry using three degrees of motion*. In: *Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, S. 3549–3554, New York, NY, USA, 2008. ACM.

[4] Clematide, S.: *Einführung in die Computerlinguistik*. Techn. Ber., Institut für Computerlinguistik Universität Zürich, Zürich, Schweiz, 2007. http://www.cl.unizh.ch/siclemat/lehre/ws0607/ecl1/script/script.pdf, Kopie auf CD-ROM unter /quellen/EV6_worderr/ vom 29.05.2011.

[5] Cuhls, K. und K. Simone: *Delphi-Report: Zukünftige Informations- und Kommunikationstechniken*, 2008. http://www2.bwcon.de/fileadmin/_fazit-forschung/downloads/FAZIT_Schriftenreihe_Band_10.pdf, Kopie auf CD-ROM unter /quellen/A2_delphi/ vom 14.04.2011.

[6] Dicke, C., K. Wolf und Y. Tal: *Foogue: eyes-free interaction for smartphones*. In: *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services*, Mobile HCI '10, S. 455–458, New York, NY, USA, Sep. 2010. ACM.

[7] Heman, M.: *How exactly do cell phones become smartphones*, Dez. 2010. http://www.cell-phone-plans.net/blog/smartphones/how-exactly-do-cell-phones-become-smartphones/, Kopie auf CD-ROM unter /quellen/W1_smartph/ vom 29.05.2011).

[8] Hoggan, E., S. A. Brewster und J. Johnston: *Investigating the effectiveness of tactile feedback for mobile touchscreens*. In: *Proceeding of the*

Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems, CHI '08, S. 1573–1582, New York, NY, USA, Apr. 2008. ACM.

[9] Hoggan, E., A. Crossan, S. A. Brewster und T. Kaaresoja: *Audio or tactile feedback: which modality when?* In: *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, CHI '09, S. 2253–2256, New York, NY, USA, Apr. 2009. ACM.

[10] Hoggan, E., T. Kaaresoja, P. Laitinen und S. Brewster: *Crossmodal congruence: the look, feel and sound of touchscreen widgets.* In: *Proceedings of the 10th International Conference on Multimodal Interfaces*, ICMI '08, S. 157–164, New York, NY, USA, Okt. 2008. ACM.

[11] Hoggan, E. E.: *Crossmodal audio and tactile interaction with mobile touchscreens.* Dissertation, Department of Computing Science, University of Glasgow, Glasgow, Feb. 2010.

[12] Hoggan, E. E. und S. A. Brewster: *Crossmodal interaction with mobile devices.* In: *Proceedings of the Visual Languages and Human-Centric Computing*, S. 234–235, Washington, DC, USA, Sep. 2006. IEEE Computer Society.

[13] Kane, S. K., J. P. Bigham und J. O. Wobbrock: *Slide rule: making mobile touch screens accessible to blind people using multi-touch interaction techniques.* In: *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility*, Assets '08, S. 73–80, New York, NY, USA, Okt. 2008. ACM.

[14] Kane, S. K., C. Jayant, J. O. Wobbrock und R. E. Ladner: *Freedom to roam: a study of mobile device adoption and accessibility for people with visual and motor disabilities.* In: *Proceedings of the 11th International ACM SIGACCESS Conference on Computers and Accessibility*, Assets '09, S. 115–122, New York, NY, USA, Okt. 2009. ACM.

[15] Kane, S. K., J. O. Wobbrock und I. E. Smith: *Getting off the treadmill: evaluating walking user interfaces for mobile devices in public spaces.* In: *Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services*, Mobile HCI '08, S. 109–118, New York, NY, USA, Sep. 2008. ACM.

[16] Keck, S.: *Touchscreen Technologies*, Apr. 2007. http://www.medien.ifi.lmu.de/lehre/ws0607/mmi1/essays/Susanne-Keck.xhtml, Kopie auf CD-ROM unter /quellen/W10_einftouchs/ vom 29.05.2011.

[17] LBSV Online: *Arten von Sehbehinderungen*, 2002. http://www.lbsv.org/index.php?ID=23, Kopie auf CD-ROM unter /quellen/W15_visimpex/ vom 29.05.2011.

[18] MacKenzie, I. S.: *KSPC (keystrokes per character) as a characteristic of text entry techniques.* In: *Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction*, Mobile HCI '02, S. 195–210, London, UK, 2002. Springer-Verlag.

[19] MacKenzie, I. S.: *Evaluation of text entry techniques.* Text Entry Systems: Mobility, Accessibility, Universality, S. 75–101, 2007.

[20] MacKenzie, I. S. und K. Tanaka-Ishii: *Text Entry Systems: Mobility, Accessibility, Universality (Morgan Kaufmann Series in Interactive Technologies).* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[21] McGookin, D., S. Brewster und W. Jiang: *Investigating touchscreen accessibility for people with visual impairments.* In: *Proceedings of the 5th Nordic Conference on Human-Computer Interaction: Building Bridges*, NordiCHI '08, S. 298–307, New York, NY, USA, Okt. 2008. ACM.

[22] O'Modhrain, M. S. und B. Gillespie: *The Moose: A haptic user interface for blind persons*, 1997. https://ccrma.stanford.edu/~sile/papers/www6-paper.html, Kopie auf CD-ROM unter /quellen/EF3_moose/ vom 30.05.2011.

[23] Perry, K. B. und J. P. Hourcade: *Evaluating one handed thumb tapping on mobile touchscreen devices.* In: *Proceedings of Graphics Interface 2008*, GI '08, S. 57–64, Toronto, Ont., Canada, Canada, Mai 2008. Canadian Information Processing Society.

[24] Raman, T. V. und C. L. Chen: *Looking beyond the screen: Text-to-speech and eyes-free interaction on android.* Online Video, Mai 2009. http://www.google.com/events/io/2009/sessions/LookingBeyondScreenTextSpeechAndroid.html, Kopie auf CD-ROM (Dateien /quellen/EF7_efa/googledeveloperconf.pdf und /quellen/EF7_efa/googledevconf.mp4 vom 30.10.2010).

[25] Saffer, D.: *Designing Gestural Interfaces: Touchscreens and Interactive Devices.* O'Reilly Media, Inc., 2008.

[26] Soukoreff, R. W. und I. S. MacKenzie: *Measuring errors in text entry tasks: an application of the levenshtein string distance statistic.* In: *Extended Abstracts on Human Factors in Computing Systems*, CHI EA '01, S. 319–320, New York, NY, USA, 2001. ACM.

[27] Tinwala, H. und I. S. MacKenzie: *Eyes-free text entry on a touchscreen phone.* In: *Proceedings of the IEEE Toronto International Conference – Science and Technology for Humanity – TIC-STH 2009*, S. 83–88, Sep. 2009.

[28] Villamor, C., D. Willis und W. Luke: *Touch gesture reference guide*, Apr. 2010. http://www.lukew.com/touch/, Kopie auf CD-ROM unter /quellen/A4_touchguide/ vom 10.04.2011.

[29] VISAM GmbH: *Touchscreen Technik*, 2005. http://www.visam.de/04_service/touch.php, Kopie auf CD-ROM unter /quellen/W7_visam/ vom 29.05.2011.

[30] Wacom Europe GmbH: *Funktion des batterielosen Wacom-Stifts*, 2007. http://www.wacom.eu/_bib_user/downloads/tech_bam_de.pdf, Kopie auf CD-ROM unter /quellen/A3_wacom/ vom 10.04.2011.

[31] Yatani, K. und K. N. Truong: *Semfeel: a user interface with semantic tactile feedback for mobile touch-screen devices.* In: *Proceedings of the 22nd annual ACM Symposium on User Interface Software and Technology*, UIST '09, S. 111–120, New York, NY, USA, Okt. 2009. ACM.