

# Multimodales Zoomable User Interface im Automotive - Bereich

## Diplomarbeit

am

Lehrstuhl für Mensch-Maschine-Kommunikation  
Univ.-Prof. Dr.-Ing. habil Gerhard Rigoll  
Fakultät für Elektrotechnik und Informationstechnik  
Technische Universität München

von

**Florian Laquai**  
Matrikel-Nr.: 2369961

Betreuer: Dipl. Ing. Markus Ablaßmeier  
Beginn: 12.12.2006  
Ende: 12.7.2007  
Abgabedatum: 12.7.2007



# Vorwort

Diese Arbeit entstand in Anlehnung an das InfoMAN-Konzept, welches im Rahmen des TUMMIC-Projektes erarbeitet wurde. InfoMAN beinhaltet diverse Ansätze, wie die Fahrerinformationssysteme der Zukunft aufgebaut sein könnten um die stetig wachsenden Anforderungen an diese Systeme zu erfüllen. Hier wurde der Fokus auf die Darstellung und Organisation der steigenden Informationsfülle von verschiedenen Systemen im Automobil gelegt. So sollte das System in der Lage sein die Daten von PDAs, Mobiltelefonen, dem Bordcomputer und eventuell dem heimischen Desktop-Computer zusammenzuführen, zu verwalten und adäquat anzuzeigen.

Bei dem hier vorgestellten System liegt der Hauptaugenmerk auf der Darstellung der Daten und den benötigten Eingabegeräten. Für die Datenanzeige wird das Prinzip des „Eintauchen in Menüs“ verfolgt, das in der Literatur oft als „Zoomable User Interfaces - ZUI“ bezeichnet wird. Es steht die Verringerung der Menütiefe im Vordergrund, wobei gleichzeitig die Übersichtlichkeit gewahrt werden soll. Die Idee hinter solchen Interfaces ist, dass alle Menüobjekte permanent vorhanden sind, sie jedoch erst ab einer gewissen Eintauchtiefe bzw. Zoomstufe angezeigt werden. Durch diese topographische Menüstruktur soll der Nutzer sich besser orientieren und gleichzeitig schneller durch das System navigieren können.

Als Eingabegerät soll ein Touch Screen mit Projected Capacitive-Technologie dienen. Diese Sensoren sind in der Lage, den Benutzerfinger schon vor der Berührung des Sensors zu erfassen. So kann dieses System als dreidimensionales Eingabegerät benutzt werden. Die Tiefeninformation kann einerseits für Zoomeffekte und andererseits zur Menünavigation benutzt werden. Vor allem die Steuerung von zoombaren Menüs bietet sich dabei an.

Da die Texteingabe bei Touch-Eingabe oft problematisch ist aufgrund von begrenztem Platz und geringer Treffsicherheit wurden zwei Eingabemethoden entwickelt. Beide verwenden gezielt die Annäherungsinformation zur Vereinfachung der Bedienung.

Das letzte eingebrachte Konzept ist die Einbindung eines Spracherkenners um eine multimodale Interaktion zu ermöglichen. Momentan wird die Sprache hauptsächlich zur Menünavigation oder als alternative Texteingabe herangezogen. Neben dieser ebenfalls berücksichtigten Funktionalität wurde eine wesentlich engere Verknüpfung der beiden Modalitäten Sprache und Berührungsangestreb. Dafür wurde unter anderem das „Point And Talk“-Konzept erarbeitet, bei dem der Nutzer auf ein Objekt zeigt und die gewünschte Aktion mit einem Sprachbefehl auslöst.

Die oben beschriebenen Konzepte werden in einem Interface integriert, das in Flash implementiert wird. Ein externer Spracherkennung wird ebenso wie der Touch Screen und ein Datenmanagementsystem an das ZUI angebunden. Anschließend wird dieser gesamte Aufbau in ein Versuchsfahrzeug integriert, um eine Evaluation unter möglichst realistischen Bedingungen durchführen zu können.

## Danksagung

Zunächst möchte ich mich bei meinem Betreuer Markus Ablaßmeier sehr bedanken, der mir bei der Themenauswahl stark entgegenkam und für neue Ideen immer ein mehr als offenes Ohr hatte. In den zahlreichen Gesprächen wurden die meisten der implementierten Funktionen entworfen, ebenso wie ein Großteil der im Ausblick beschriebenen Ideen. Gleichfalls danke ich meinen Kollegen Stephan Schneider und Johannes Wust, die mit konstruktiven Vorschlägen und unterhaltsamen Gesprächen die Arbeit sehr angenehm machten. Auch die Kollegen aus dem Arbeitsraum und insbesondere Tony Poitschke als Mitarbeiter trugen in erheblichem Maße zum Gelingen der Arbeit bei.

Abschließend möchte ich Univ.-Prof. Dr.-Ing. habil Gerhard Rigoll danken, dass er die Rahmenbedingungen für diese Arbeit geschaffen hat.

# Inhaltsverzeichnis

<b>1</b>	<b>Stand der Technik</b>	<b>1</b>
1.1	Bestehende Menüsysteme . . . . .	1
1.1.1	Personal Digital Assistants und Navigationssysteme . . . . .	2
1.1.2	Bordcomputer im Automobil . . . . .	3
1.2	Vorhandene Technologie . . . . .	6
1.2.1	Controller . . . . .	6
1.2.2	Touch Screens . . . . .	7
1.2.3	Spracheingabe . . . . .	13
<b>2</b>	<b>Theoretische Grundlagen</b>	<b>15</b>
2.1	Richtlinien für den Menüaufbau . . . . .	15
2.2	Zoomable Interface . . . . .	20
2.3	Multimodalität . . . . .	23
2.4	Menübedienung . . . . .	25
2.5	Texteingabemethoden . . . . .	26
2.5.1	MatrixKey . . . . .	28
2.5.2	FishKey . . . . .	28
2.6	Listennavigation . . . . .	30
2.7	Datenhaltung . . . . .	34
2.7.1	Adaptivität . . . . .	34
2.7.2	Suchfunktion und TAGs . . . . .	36
2.7.3	Datenmanagementsystem . . . . .	38

<b>3 Implementierung</b>	<b>41</b>
3.1 Hardware . . . . .	41
3.2 Softwarearchitektur . . . . .	41
3.2.1 Flash . . . . .	44
3.2.2 Fingerdetektion . . . . .	46
3.2.3 Pointersteuerung . . . . .	51
3.2.4 Spracherkennner . . . . .	52
<b>4 Ausblick</b>	<b>53</b>
4.1 Sprachfeedback . . . . .	53
4.2 Einbeziehung des HUD . . . . .	54
4.3 Gestenerkennung . . . . .	56
4.4 Adaptivität . . . . .	57
4.5 Ausbau des Menüssystems . . . . .	60
<b>5 Evaluierung</b>	<b>61</b>
5.1 Versuchsplanung und Durchführung . . . . .	61
5.2 Ergebnisse . . . . .	65
<b>6 Zusammenfassung</b>	<b>77</b>
<b>A Fragebogen Evaluierung</b>	<b>79</b>
<b>B Quellcode Sensorauswertung</b>	<b>81</b>
<b>Literaturverzeichnis</b>	<b>87</b>
<b>Erklärung</b>	<b>91</b>

# 1. Stand der Technik

Zunächst soll ein Überblick zu vorhandenen Mensch-Maschine-Schnittstellen (im Folgenden MMI<sup>1</sup> genannt) gegeben werden. Es werden hier nur Systeme für mobile Anwendungen behandelt, da die Anforderungen z.B. an ein Desktop-System stark abweichend sind.

Bei stationären Applikationen muss meist eine hohe Funktionsanzahl untergebracht werden. Ein einzelnes Gerät wird dann für viele oft sehr unterschiedliche Aufgaben verwendet. Als klassisches Beispiel kann hier der PC genannt werden, der sowohl Arbeits- als auch Unterhaltungs- und Kommunikationsgerät benutzt wird.

Die Informationsdichte kann bei stationären Anwendungen sehr groß sein, da die Umgebung weniger Ablenkung hervorruft und gleichzeitig mehr Zeit zum Suchen und Erfassen von Informationen zur Verfügung steht. Dabei sorgen immer größer werdende Displays für mehr Übersicht und es können große Eingabegeräte wie Maus und Tastatur verwendet werden.

Eine gegensätzliche Entwicklung ist bei den mobilen Endgeräten zu beobachten. Hier hat jedes System einen relativ begrenzten Funktionsumfang, als Ausnahme sind nur die PDAs<sup>2</sup> zu nennen. Diese bieten mit gewissen Einschränkungen die gleiche Bedienoberfläche wie ein PC. Die Informationsdichte muss allerdings wesentlich niedriger gehalten werden, da im mobilen Einsatz oft nur eine kurze Blickverweildauer möglich ist. Dabei müssen wichtige Daten sofort fehlerfrei erkannt werden können. Darüber hinaus muss zu dem eben beschriebenen „Ease Of Use“ noch ein gewisser „Joy Of Use“ vorhanden sein, der sowohl ein Kaufkriterium darstellen als auch Interface-Schwächen ausgleichen kann. Elemente wie z.B. Animationen oder grafisches Design zählen zu dieser Kategorie, die immer mehr an Bedeutung gewinnt. Im Idealfall sollten daher beide Eigenschaften vorhanden sein.

Die folgende Aufzählung und Beschreibung vorhandener Technologie kann keinen Anspruch auf Vollständigkeit erheben, da es von vielen Systemen mehrere leicht differierende Varianten gibt und oft aus patentrechtlichen Gründen keine genauen Angaben zur Funktionsweise gegeben werden.

## 1.1 Bestehende Menüsysteme

Im Folgenden werden verschiedene Menü- und Interaktionsmodelle bei mobilen Systemen vorgestellt. Hierbei werden die PDAs und die tragbaren Navigationssysteme zusammengefasst, da diese ähnliche Anforderungen haben und daher auch auf den gleichen Pool von Eingabemethoden zurückgreifen.

---

<sup>1</sup>Man Machine Interface

<sup>2</sup>Personal Digital Assistant

Im Automobil sind die Verhältnisse vor allem bezüglich der Eingabemethoden unterschiedlich, da beispielsweise ein bei PDAs üblicher Stift nicht akzeptabel wäre. Ebenso muss die Einhand-Bedienung und in manchen Situationen eine Blindbedienung möglich sein.

### 1.1.1 Personal Digital Assistants und Navigationssysteme

Bei PDA-Systemen handelt es sich meist um Geräte mit der Größe einer Handfläche. Dadurch wird die verfügbare Displayfläche stark eingegrenzt, genau wie der Platz für Eingabegeräte wie Buttons und dergleichen. Ein typisches Beispiel für diese Gerätelasse ist in Abbildung 1.1 zu sehen. Die häufigsten Eingabegeräte sind hierbei:

- Berührung
- Stifteingabe (Mausersatz)
- Taster und Wippen
- Spracheingabe

Die Eingabe mit einem Stift oder dem Finger ist vor allem für Nutzer mit Vorkenntnissen bei Desktop-Computersystemen sehr intuitiv, da hier die GUI<sup>3</sup>-Elemente in vertrauter Weise bedient werden können. Beide Methoden werden dabei praktisch als Mausersatz verwendet. Eine Spezialität der Stifteingabe ist die natürliche Verwendung um Notizen handschriftlich (grafisch) festzuhalten bzw. als Texteingabe mit Schrifterkennung (OCR<sup>4</sup>).

Zunehmend werden bei aktuelleren Geräten die Hardwaredaten durch Softkeys ersetzt. Beispiele hierfür sind das Apple iPhone (Abbildung 1.2) oder das LG Prada Phone. Dadurch ergeben sich folgende Vorteile:

- geringe Abzahl beweglicher Teile
- große Anzeigefläche
- direkte Manipulierbarkeit
- konsistentes Eingabekonzept
- situationsoptimierte Eingabe

Obige Punkte bieten Vorteile sowohl für den Nutzer als auch den Hersteller. Die Anordnung eines berührungsensitiven und transparenten Sensors über der vollständigen Anzeigefläche erlaubt auch neue Ansätze z.B. in der Listennavigation. Muss bei bekannten GUIs ein Scrollbar betätigt werden, so kann bei dem iPhone eine Liste „gepackt“ und „angestoßen“ werden, um sie dann später mit einer weiteren Berührung wieder anzuhalten. So wird schnelle Navigation ermöglicht und ein physikalisches und damit realistisches Verhalten erzeugt. Auch Gesten können einfach

---

<sup>3</sup>Graphical User Interface

<sup>4</sup>Optical Character Recognition



Abbildung 1.1: HP Pocket PC mit Windows Mobile, aus [19]

angewandt werden, um beispielsweise eine Tastensperre aufzuheben. In Abbildung 1.2 ist zu erkennen, wie ein virtueller Slider verschoben werden muss, um das Telefon zu entsperren.

Teile des obigen Konzepts werden zunehmend auch in anderen tragbaren Geräten angewandt, vor allem die vollständig mechanikfreie Eingabe. Als Messtechnik wird dabei meist eine kapazitive Messung durchgeführt. Damit werden z.B. berührungs-sensitive Streifen („Slider“) oder Tasten implementiert.

### 1.1.2 Bordcomputer im Automobil

In einem Fahrzeug herrschen etwas andere Anforderungen als bei tragbaren Geräten. Die Platzanforderungen sind zwar weniger eingeschränkt, doch muss meist eine wesentlich heterogener Nutzergruppe berücksichtigt werden. So sind Autofahrer nicht unbedingt vertraut mit den Benutzungsprinzipien moderner Betriebssysteme und Eingabemethoden, was es schwieriger macht eben diese Techniken in Bordcomputern anzuwenden. Um eine sichere Eingabe sicherzustellen wird immer noch eine große Zahl mechanischer Taster und Drehknöpfe verbaut. Diese bieten ein definiertes haptisches Feedback und haben meist eine eindeutige Funktionszuordnung. Als Beispiele sind hier der Lautstärkeregler der Musikanlage oder Temperatursteller anzuführen.

Folgende Rahmenbedingungen lassen sich im Automobil angeben:

- gute Erreichbarkeit der Bedienelemente für den Fahrer
- geringe Ablenkung von der primären Fahraufgabe



Abbildung 1.2: Apple iPhone, aus [20]



Abbildung 1.3: BMW iDrive, aus [21]

- wenn möglich Blindbedienung
- eindeutiges Feedback (haptisch, optisch, akustisch)

Alle oben genannten Vorgaben haben zum Ziel, eine schnelle und sichere Bedienung auch mit geringer Aufmerksamkeit für das FIS<sup>5</sup> zu gewährleisten. Dabei werden bei den verschiedenen Implementationen unterschiedliche Kompromisse eingegangen. Die Bedienung mit einem Touch Screen erlaubt direkte Manipulierbarkeit und damit eine hohe Intuitivität. Auch um das Display angeordnete Shortcut- und Funktions-tasten können einfach zugeordnet werden. Der Nachteil dieser Anordnung ist aber eine längere Abwendung des Blicks von der Straße und eine größere Gefahr der Fehlbedienung durch das eingeschränkte haptische Feedback.

Bei einer Trennung von Anzeige- und Bedienelementen wie bei BMW oder Audi kann der Controller blind erreicht und bedient werden. Dies erfordert vom Fahrer weniger Konzentration als das Treffen eines Punktes auf einem Touch Screen, allerdings muss der Nutzer stets wissen, welchen Menüfunktionen die verschiedenen Controllerelemente aktuell zugewiesen sind. Zum Teil wird also der kognitive Aufwand lediglich verschoben und ist stark vom Benutzer abhängig.

Die Fahrzeugherrsteller verfolgen vor allem bei den Eingabegeräten sehr unterschiedliche Konzepte. Während BMW, Renault und Audi auf Flachbildschirme in der Mittelkonsole setzen mit Controllern zwischen Fahrer- und Beifahrersitz, verwendet Lexus einen Touch Screen mit daneben angeordneten Funktionstasten. Zunehmend wird neben Hardware-Controllern auch die Spracheingabe benutzt um Menübefehle oder Telefonnummern einzugeben. Es sind also sehr unterschiedliche Bedienkonzepte vorhanden, vor allem bezüglich der räumlichen Anordnung von Ein- und Ausgabe-geräten.

Zusätzlich zu den gängigen Displays im Kombiinstrument (hinter dem Lenkrad) und dem CID<sup>6</sup> werden immer selbverständlicher HUDs<sup>7</sup> benutzt, um die wichtigsten Informationen des Navigationssystems oder Fahrzeugwarnungen anzuzeigen. Da diese Anzeige im direkten Blickfeld des Fahrers platziert wird, kann die Ablenkung durch einen Blickrichtungswechsel minimal gehalten werden. Die dargestellte Informationsmenge wird meist sehr gering gehalten, um nicht den negativen Einfluss einer Ablenkung zu erzeugen. Durch die Überlagerung des HUDs mit der Umgebung ergibt sich die Möglichkeit eines kontaktanalogen Displays. Diese Technik wird beispielsweise bei Lexus verwendet, um nachts Bilder einer Infrarotkamera anzuzeigen und so auf der Straße befindliches Wild deutlicher erkennbar zu machen. Zur Menünavigation werden diese Anzeigen derzeit noch nicht verwendet.

Vor allem die Shortcut-Tasten (siehe Abbildung 1.4) und die Spracheingabe stellen eine Neuerung bei den Bedienkonzepten dar. Sie erlauben Sprünge über mehrere Menüebenen und zwischen verschiedenen Ästen des Menübaumes. Manche dieser Sprungziele sind vom Hersteller vordefiniert, es können jedoch meist noch eigene festgelegt werden. Dies kann z.B. ein oft frequentiertes Navigationsziel, eine Radiostation oder ein Suchbegriff sein. Auf diese Weise kann eine gewisse Adaption des Systems an das Nutzerverhalten realisiert werden, was eine zügigere Bedienung und eine stärkere Identifizierung des Users mit dem Gerät erlaubt.

---

<sup>5</sup>FahrerInformationsSystem

<sup>6</sup>Central Information Display

<sup>7</sup>Head Up Display



Abbildung 1.4: VW Navigationssystem mit Funktionstasten, aus [22]



Abbildung 1.5: Links: BMW iDrive Controller aus [21], rechts: Audi MMI Controller mit Funktionstasten, aus [23]

## 1.2 Vorhandene Technologie

In diesem Abschnitt wird die Technik vorgestellt, die bestehende Fahrerinformationssysteme ermöglicht. Es werden zunächst die gängigen Hardware Controller und ihre Verwendung vorgestellt. Nachfolgend wird ein Überblick zu bestehenden Touch Screen Technologien gegeben und abschließend die Spracheingabe untersucht.

### 1.2.1 Controller

Als von der Anzeige getrennte Eingabegeräte haben sich Drehsteller etabliert, die von einer unterschiedlichen Anzahl Funktionstasten umgeben sind. Diese Controller können meist parallel zur Drehachse gedrückt oder senkrecht dazu verschoben werden. So besitzen sie bis zu sechs Freiheitsgrade, die je nach Anwendungskontext unterschiedlich belegt sein können. Die Spezialität dieser Eingabegeräte ist, dass sie sowohl rotatorische als auch translatorische Bewegungen umsetzen können.

Gewisse Modelle dieser Controller sind in der Lage Force Feedback<sup>8</sup> zu erzeugen. Dies wird durch Bremsmechanismen, Motoren oder Mischungen aus beidem erzeugt. So kann je nach aktueller Anwendungssituation die Haptik des Eingabegeräts angepasst werden. Es ist also möglich unterschiedliche Rastungen oder einen Anschlag bei der Listennavigation herzustellen. Dieses Feedback stellt dem Nutzer zusätzliche Informationen bereit, ohne jedoch auf Blickkontakt angewiesen zu sein.

---

<sup>8</sup>Kraftrückkopplung

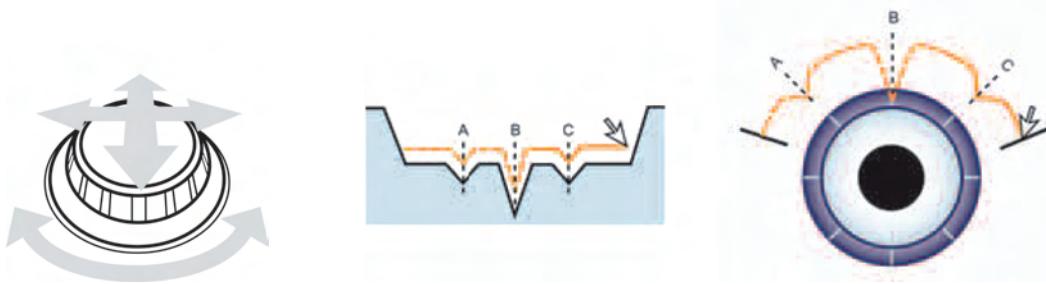


Abbildung 1.6: Drehsteller als Eingabegerät mit Force Feedback, rechtes Bild aus [24]

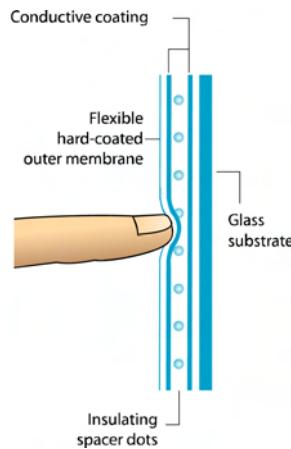


Abbildung 1.7: Aufbau eines resistiven Touch Screens, aus [25]

### 1.2.2 Touch Screens

Berührungssensitive Bildschirme existieren seit einigen Jahren. Seitdem wurden zahlreiche Technologien dafür entwickelt, die großen Einfluss auf das Einsatzgebiet und die Art der Eingabe haben. Generell kann man unterscheiden zwischen Messmethoden, die eine Berührung zur Detektion benötigen und solchen, die eine bloße Annäherung erfassen können.

#### Resistive Messtechnik

Die wohl älteste Touch Technologie ist die von resistiven Touch Screens. Aufgrund des einfachen Aufbaus und der guten Handhabbarkeit wird dieser Typ weiterhin angewandt. Das Grundprinzip besteht darin, dass eine kontinuierlich leitfähige Schicht durch eine Berührung auf eine darunter liegende Platte gedrückt wird, an deren Ecken Elektroden angebracht sind. Zwischen der Schicht und der Platte besteht im Grundzustand keine elektrisch leitende Verbindung, da sie durch dünne Abstandshalter voneinander getrennt werden. Da die oberste Schicht ein erhöhtes Spannungspotential hat, wird durch den bei einer Berührung hergestellten Kontakt das Potential der Elektroden angehoben. Durch die unterschiedlichen Abstände des Berührpunkts zu den Elektroden und damit deren unterschiedliche Potentiale kann die Fingerposi-

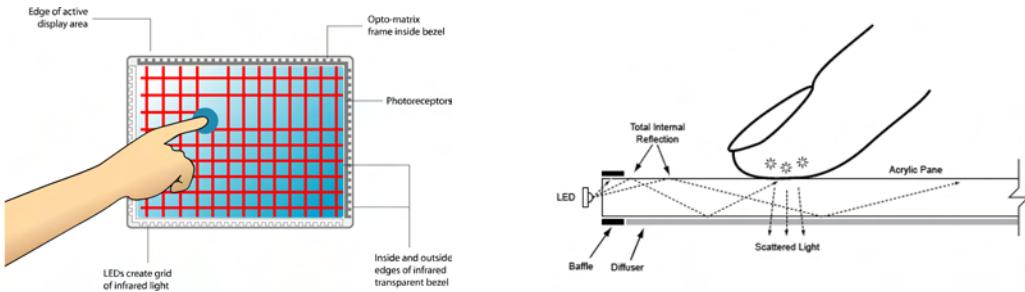


Abbildung 1.8: Links: Aufbau eines Infrarot Touch Screens, aus [25], rechts: Funktionsweise von FTIR, aus [32]

tion errechnet werden. Aufgrund dieses Prinzips reagiert diese Technologie auf Stift- und Handeingabe. Vorteile dieses Verfahrens sind die hohe Robustheit gegenüber elektrischen Störeinflüssen und dadurch einfacher Installation, gute Bedienbarkeit und die Langzeiterfahrung damit. Nachteilig sind die verminderte Lichtdurchlässigkeit, leichte Drift und die Abnutzung oft benutzter Bereiche. Mit diesem Sensor ist keine Detektion mehrerer Finger gleichzeitig möglich (Multi Touch).

## Optische Messtechnik

Bei der optischen Variante gibt es mehrere Herangehensweisen. Die einfachste ist die Anordnung je eines Paares Sende- und Empfangsdioden für x- und y-Achse. Da die benutzte Wellenlänge im infraroten Bereich liegt, entstehen keine Störungen des Monitorbildes. Bei einer Berührung werden Lichtstrahlen bei beiden Achsen unterbrochen und lassen so einen Rückschluss auf die Position des Eingabegeräts zu. Die größten Vorteile dieser Sensoren ist eine absolute Drift- und Verschleißfreiheit und dass keine optische Beeinträchtigung erfolgt. Es kann jedoch kein Anpressdruck gemessen werden und die Ortsauflösung beträgt typisch ca. 2mm mit einem Fehler von  $\pm 1\text{mm}$ . Daher eignet sich diese Technik lediglich für schwierigste Umgebungen, bei denen Bedienpräzision niedrigere Priorität besitzt. Multi Touch ist theoretisch realisierbar, wird jedoch nicht so intensiv genutzt in den typischen Einsatzgebieten dieser Sensoren.

Auf der rechten Seite in Abbildung 1.8 ist die Funktionsweise eines FTIR<sup>9</sup>-Sensors zu sehen. Es wird an den seitlichen Kanten der Sensorscheibe paralleles Licht in einem Winkel eingestreut, so dass innerhalb der Scheibe lediglich Totalreflexion auftritt. Berührt ein Benutzer die Scheibe, so ändert er das Brechungsverhältnis und der Finger wird beleuchtet. Dieser Leuchtpunkt kann dann von einem unter der Scheibe liegenden Sensor bzw. einer Kamera erfasst und damit in Koordinaten umgerechnet werden. Das Bild wird bei dieser Konstruktion von hinten auf die Scheibe projiziert. So benötigt diese Anordnung eine gewisse Mindestgröße, lässt aber eine fast beliebige Skalierung nach oben hin zu. Der Anpressdruck lässt sich indirekt über die Fläche des reflektierten Fingers ermitteln. Auch ist die Detektion von theoretisch unbegrenzt vielen Fingern gleichzeitig möglich, was eine Vielzahl neuer Eingabemethoden erlaubt.

<sup>9</sup>Frustrated Total Internal Reflection

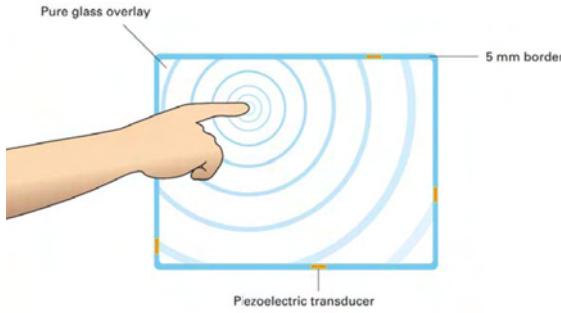


Abbildung 1.9: Funktionsweise von Acoustic Pulse Recognition, aus [25]

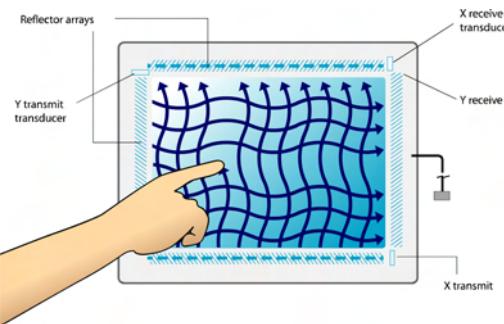


Abbildung 1.10: Funktionsweise der Surface Acoustic Wave, aus [25]

## Akustische Messtechnik

Beide vorgestellten Verfahren benötigen lediglich eine einfache Glasscheibe als Berührfläche, was die Herstellung vereinfacht und die Bildqualität minimal beeinflusst. Die erste hier beschriebene Technik „Acoustic Pulse Recognition“ - APR basiert auf der Tatsache, dass die Berührung einer Glasscheibe an einer beliebigen Stelle ein eindeutig identifizierbares Schallereignis auslöst. Dieses wird dann von piezoelektrischen Wählern aufgenommen und an einen AD<sup>10</sup>-Wandler übertragen. Anschließend wird das gemessene Muster mit einem Look Up Table (LUT) von Referenzmustern verglichen, was in diesem Falle schnellere und robustere Ergebnisse liefert als eine analytische Auswertung. Eine Multi Touch Erkennung ist hier nicht möglich, jedoch lässt sich eine aufliegende Hand beispielsweise bei einer handschriftlichen Eingabe kompensieren.

Ein weiterer Sensor benutzt die „Surface Acoustic Wave“ - SAW zur Berührungsdetektion. Es wird an einer Ecke des Bildschirms ein Ultraschallsignal erzeugt, das entlang der Scheibenkante gerichtet ist. Auf der gesamten Kantenlänge sind teil-durchlässige Reflektoren angebracht, die einen Teil des Schalls durch die Scheibe ablenken. An der gegenüberliegenden Seite existiert eine korrespondierende Reflektorenanordnung, die die ankommenden Wellen in Richtung eines einzelnen Empfängers lenken. Eine entsprechende Anordnung gibt es für x- und y-Achse. Eine Berührung der Oberfläche absorbiert einen Teil des von einer Kante zur anderen laufenden Schalls, was in dem empfangenen Muster erkannt werden kann. Der Anpressdruck kann hierbei ermittelt werden, was eine feinfühliger Bedienung erlaubt.

<sup>10</sup>Analog / Digital

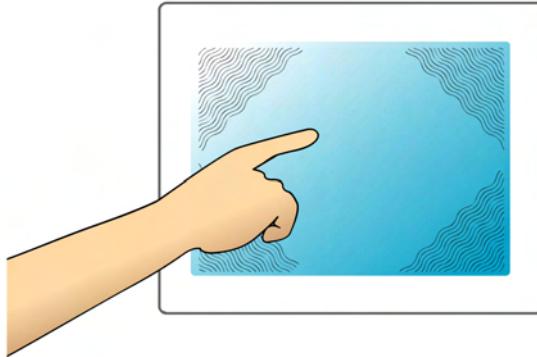


Abbildung 1.11: Funktionsweise von Surface Capacitance Sensing, aus [25]

## Kapazitive Messtechnik

Kapazitive Sensoren benutzen die Eigenschaft des menschlichen Körpers, Ladung speichern zu können. Es sind also Eigenschaften eines Kondensators vorhanden. Diese Kapazität kann andere Kondensatoren beeinflussen, die als Sensorelektroden ausgeführt sind. Form und Material dieser Elektroden hängen stark von der Anwendung und vor allem der Messtechnik ab.

Die einfachste Technik dieser Art ist die, die eine Berührung der Oberfläche benötigt. Der Sensor wird hier aus einer homogenen transparenten Widerstandsschicht gebildet, die beispielsweise aus ITO<sup>11</sup> bestehen kann. Dieses Material bietet mit aktuellen Herstellungsverfahren eine durchschnittliche Transparenz von ca. 90%, die wellenlängenabhängig ist. ITO-Schichten können in beliebigen Formen hergestellt werden, wodurch sie sowohl als Leiterbahnen, Sensorelemente oder Antennen verwendbar macht. Wie in Abbildung 1.11 zu sehen ist, werden an der ITO-Schicht in jeder Ecke Elektroden angebracht. Sie werden dann auf ein gemeinsames gegenüber der Umgebung erhöhtes elektrisches Potential gebracht. Findet eine Berührung statt, so fließt ein geringer Strom von den Elektroden durch die Sensorschicht auf den Körper des Benutzers, der sich dabei auflädt. Die Stromstärke durch jede Elektrode ist proportional zur Fingerposition, da eine unterschiedlich lange Strecke (= Widerstand) zum Finger zurückgelegt wird. Die zugrundeliegenden Gleichungen lauten:

$$I = C_{\text{mensch}} \cdot \frac{dU}{dt} \quad (1.1)$$

$$\tau = R \cdot C_{\text{mensch}} \quad (1.2)$$

Dabei ist  $I$  der bei einer Berührung fließende Strom,  $C_{\text{mensch}}$  die Körperkapazität des Benutzers und  $U$  die Messspannung. Der Ladevorgang hat eine Zeitkonstante  $\tau$ , die von der Körperkapazität und dem Ersatzwiderstand  $R$  der Sensorfolie bestimmt wird. Ein Ersatzschaltbild ist in Abbildung 1.12 zu sehen. Der geschlossene Schalter entspricht einer Berührung, der offene dem Zustand keiner Berührung. Das Ersatzschaltbild ergibt sich folgendermaßen: Die Widerstände  $R_1$  bis  $R_4$  entsprechen den Flächenwiderständen der Folie von der Berührposition zu den vier Ecken. Der Kondensator  $C_{\text{mensch}}$  ist die Körperkapazität und  $R_{\text{entlade}}$  ergibt sich aus der Leitfähigkeit des Körpers und der Verbindung zur Erde. Bei einer Berührung lädt sich der Kondensator über die vier parallelen Widerstände auf. Dabei wird der Strom durch jeden

<sup>11</sup>Indium Tin Oxide

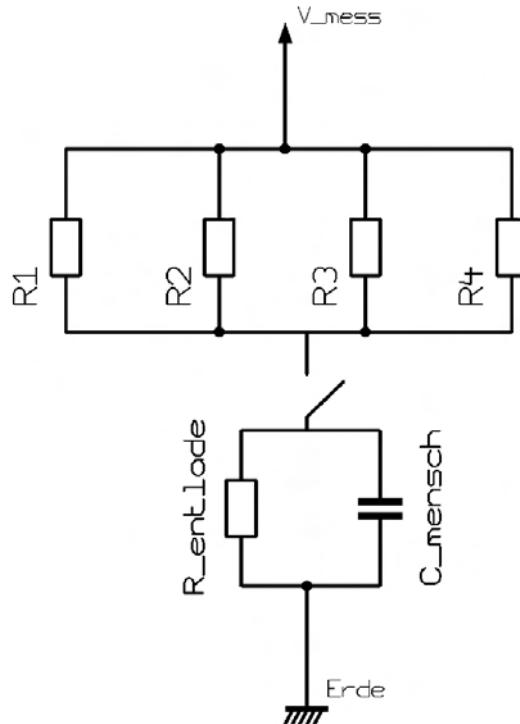


Abbildung 1.12: Ersatzschaltbild bei Surface Capacitance Sensing

einzelnen gemessen, welcher von dem Widerstandswert und daher vom Abstand der Fingerposition zur jeweiligen Elektrode abhängt. Aus den Verhältnissen der Ströme kann auf die Position rückgeschlossen werden.

Im Gegensatz zu bisherigen Technologien, die eine wirkliche Berührung der Sensorfläche voraussetzen, existieren mehrere Messtechniken, die eine Annäherungsdetektion erlauben. Die Körperkapazität wird dabei meist parallel zu den Sensorkapazitäten geschalten, wie in Abbildung 1.13 zu sehen ist. In der Rechten Schaltung ist ein Relaxationsoszillatator zu sehen, der eine Möglichkeit zur Kapazitätsmessung darstellt. Diese Technik wird beispielsweise von Cypress Semiconductor angewandt. Die Funktionsweise des Oszillators ist folgendermaßen: Der Kondensator  $C_P$  steht für die Kapazität der Sensorpads inklusive parasitärer Leitungskapazitäten. Er wird als anfangs entladen angenommen und der Schalter ist geöffnet. Durch die Stromquelle wird der Kondensator geladen wodurch die Spannung am Komparatoreingang ansteigt. Sobald diese den Wert von  $V_{TH}$  erreicht, schaltet der Komparatorausgang augenlogisch „High“, der Schalter wird geschlossen und  $C_P$  entladen. Der Oszillationsvorgang beginnt nun von Neuem. Bei Annäherung wird  $C_P$  erhöht, was eine Erhöhung der Periodendauer des Oszillators und damit des Zählers bewirkt. Es wird also keine Berührung benötigt, um die Anwesenheit des Fingers zu detektieren. Der Zähler wird nach jeder Periode zurückgesetzt, also ist der Zählerstand ein Index für die Kapazität und kann daher zur Annäherungsmessung herangezogen werden.

Der Hersteller Quantum benutzt unter anderem die „Charge Transfer“-Methode. Dabei wird Ladung zwischen nebeneinander liegenden Sensorelektroden bewegt. Nähert sich z.B. ein Finger, so wird ein Teil dieser Ladung auf den menschlichen Körper übertragen, wodurch sich die an die eigentliche Empfangselektrode transferierte La-

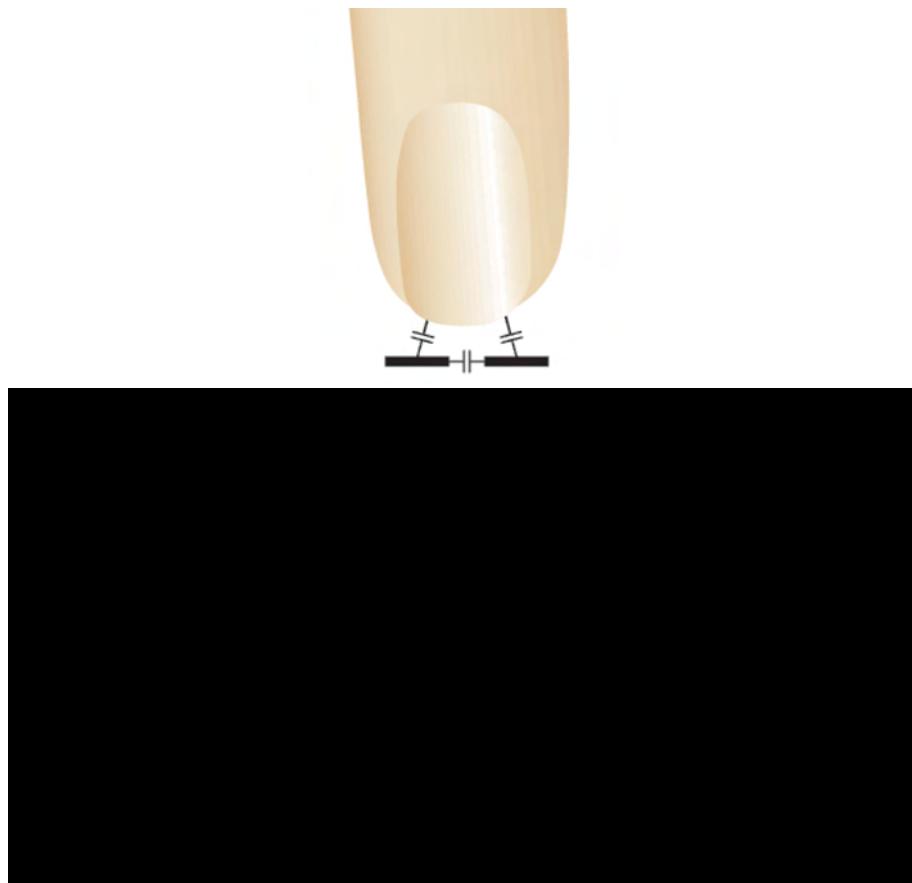


Abbildung 1.13: Ersatzschaltbild Projected Capacitance Sensing, aus [25] und Relaxationsoszillator, aus [27]

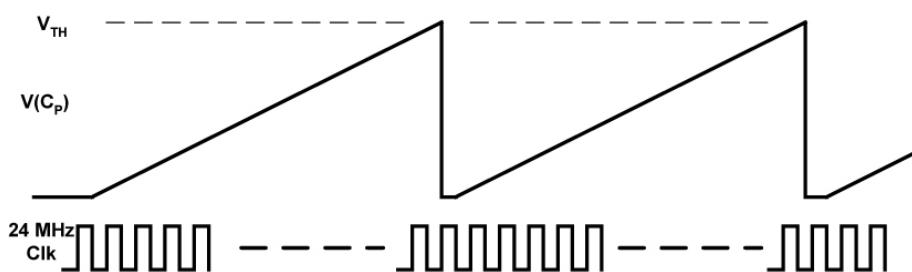


Abbildung 1.14: Spannungsverlauf am Komparatoreingang und Zählpulse, aus [27]

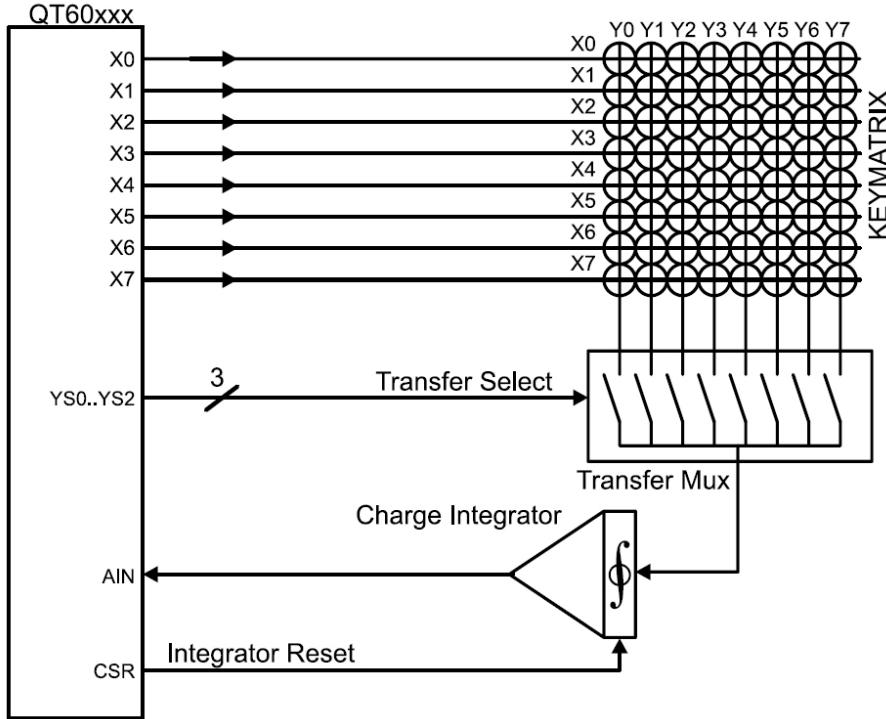


Abbildung 1.15: Prinzipschaltbild einer Charge Transfer Schaltung für Touch Pads, aus [26]

dung verringert. Es gibt hier also Sende- und Empfangselektroden. Die Verschaltung der Sensorpads zu einer Matrix um ein Touch Pad zu implementieren ist in Abbildung

### 1.2.3 Spracheingabe

Mit Spracheingabe werden bei mobilen Systemen sehr unterschiedliche Funktionalitäten abgedeckt. Eine der einfachsten Varianten wurde schon vor Jahren bei Mobiltelefonen eingesetzt. Dabei musste der Benutzer den Befehl um beispielsweise eine bestimmte Nummer zu wählen mehrere Male vorsprechen. Wollte man diesen Befehl erkennen und ausführen lassen, musste eine PTT<sup>12</sup>-Taste gedrückt werden, um die Aufnahme zu starten. Diese Aufnahme wurde dann mit den zuvor gespeicherten verglichen und diejenige mit der höchsten Übereinstimmung ausgewählt. Bei dieser Vorgehensweise wird also lediglich eine begrenzte Menge an Befehlen durch den Benutzer vorgegeben, die meist nicht miteinander kombiniert werden können. Komplexe Eingaben können damit also nicht realisiert werden.

Leistungsfähigere Systeme wie das in dieser Arbeit benutzte „Scansoft VoCon 3200“ arbeiten mit Phonemen, akustischen Sprachmodellen verschiedener Sprachen, Wörterbüchern und statistischen Methoden um vor allem eine sprecherunabhängige und flexible Erkennung zu ermöglichen. Dadurch sind diese Systeme in der Lage, als „Speech To Text“-Anwendung benutzt zu werden um z.B. einen Brief zu diktieren. Die Leistungsmerkmale sind wie folgt:

<sup>12</sup>Push To Talk

- Erkennungsgeschwindigkeit = Sprechgeschwindigkeit (ca. 120 Wörter pro Minute)
- Erkennungsrate  $\approx 95\%$
- natürlichstes Eingabemedium
- sehr intuitive Bedienung, gute Grammatikprogrammierung vorausgesetzt

Probleme der automatischen Spracherkennung sind unter anderem Hintergrundgeräusche, Differenzierung von mehreren gleichzeitigen Sprechern und die Entscheidung, ob im Sprachfluss erkannter Text für das System relevant ist oder nicht. Die genannten Schwierigkeiten werden im Fahrzeug momentan so gemindert, dass die Erkennung durch einen „Push To Talk“-Knopf aktiviert werden muss. Es entscheidet also der Bediener, wann eine Sprachäußerung für das FIS bestimmt ist.

## 2. Theoretische Grundlagen

Im folgenden Kapitel werden die Grundlagen erläutert, aus denen gewisse Grundlagen für die Hard- und Softwarearchitektur, den Menüaufbau und die graphische Aufbereitung erarbeitet worden sind. Grundsätzlich wurden immer bestehende Systeme analysiert und deren Vor- und Nachteile bewertet. Speziell wurde die Eignung verschiedener Teilespekte für zukünftige Anwendungen bewertet. So kann die Funktionsvielfalt künftiger Bordcomputer nicht mehr mit den momentanen Strukturen abgedeckt werden bzw. nur unter starken Einbußen der Bedienbarkeit. Dies liegt an den wesentlich restriktiveren Vorgaben im Fahrzeug verglichen mit einem Desktop-Computersystem. Im Automobil steht nur eine begrenzte und gleichzeitig verteilte Anzeigefläche zur Verfügung. Die Eingabegeräte müssen wesentlich fehlertoleranter und trotzdem vielseitig sein, um nicht von der Fahraufgabe abzulenken und gleichzeitig komplexe Operationen zu ermöglichen. Mit die größte Herausforderung stellt die eingeschränkte Aufmerksamkeit für die FIS dar, was die Anforderung eine hohen Intuitivität einer hohen Funktionsvielfalt gegenüberstellt.

### 2.1 Richtlinien für den Menüaufbau

Im Zuge der Rechnerentwicklung wurden mehrere Evolutionsschritte bezüglich der Benutzerschnittstellen vollzogen. So waren die ersten für den Massengebrauch erhältlichen Anwendungen noch rein textbasiert und es wurde nur eine Applikation gleichzeitig ausgeführt. Vor allem das WYSIWYG<sup>1</sup>-Prinzip konnte aufgrund mangelnder Rechenkapazitäten nicht angewandt werden. Auch die ersten Fahrzeug-Bordcomputer hatten rein textbasierte Benutzerschnittstellen, die mit wenigen Funktionstasten Informationen zum aktuellen Kraftstoffverbrauch oder die Aussentemperatur anzeigen. Die Dateneingabe beschränkte sich meist auf das Einstellen der Uhr.

Die nächste Stufe bei Computerbetriebssystemen war die Einführung von fensterbasierten Oberflächen, die mit Maus und Tastatur gesteuert werden. Hier wurden viele graphische Objekte und Interaktionsmethoden zum ersten Mal angewandt, die noch heute Standard sind. Die bekanntesten Vertreter dieser Systeme sind Microsoft Windows, MacOS und die Fenstermanager für Linux, KDE und Gnome. Seit ihrer Einführung hat sich am grundsätzlichen Bedienkonzept graphischer Oberflächen nicht viel verändert. Allen gemeinsam ist das WIMP<sup>2</sup>-Konzept, das in Abbildung 2.2 zu sehen ist.

Fast zeitgleich hat sich bei allen großen Betriebssystemen eine Reihe dynamischer Widgets wie Icon-Zooming oder Taskumschaltung mit Fenstervorschau etabliert. Die

---

<sup>1</sup>What You See Is What You Get

<sup>2</sup>Windows, Icon, Menus, Pointer



Abbildung 2.1: Textverarbeitung Microsoft Word 5.5

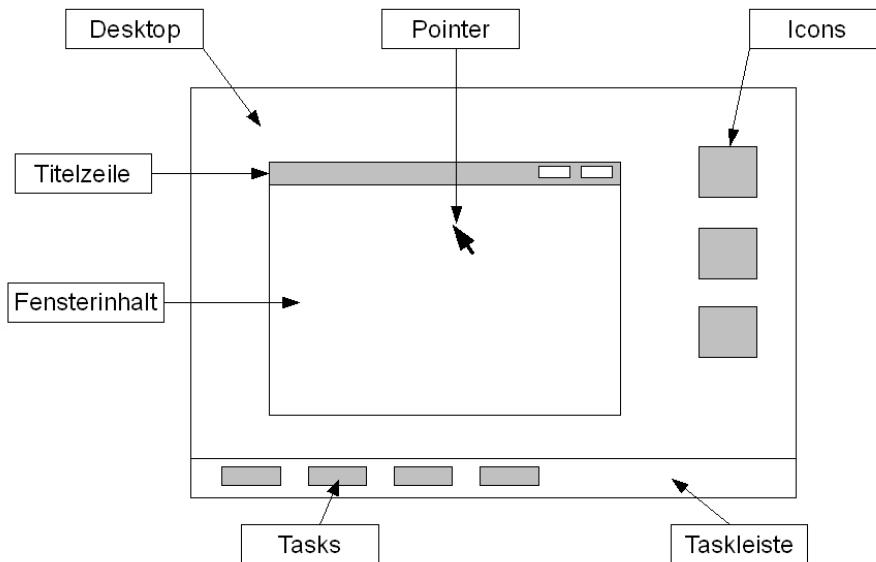


Abbildung 2.2: Das WIMP-Konzept



Abbildung 2.3: 3D-Desktop unter Linux mit XGL und Beryl-Erweiterung

größte Neuerung ist allerdings die Erweiterung des Desktops in die dritte Dimension. So können mehrere Desktops auf oder in einem Würfel angeordnet werden, was neben dem graphischen Effekt auch eine bessere Orientierung erlaubt. Ebenso können übereinander liegende Fenster in einer dreidimensionalen Anordnung gezeigt werden, was das Umschalten zwischen Applikationen erleichtert. Des weiteren können zur Anwendungsauswahl sämtliche offene Fenster verkleinert und mit Transparenzeffekten versehen nebeneinander angezeigt werden, was ebenfalls die Auswahl erleichtert. Bei Abbildung 2.3 ist eine Implementierung der 3D-Desktop Anordnung zu erkennen. Eine weitere Neuerung bei fast allen Betriebssystemen ist die Dateivorschau mit dem aktuellen Inhalt schon in der Dateiliste. Die Datei kann also nicht nur anhand des Namens und des Dateityps, sondern auch mit Hilfe der graphischen Vorschau identifiziert werden.

Bei Bordcomputern wird momentan ein mehr screenbasierter Ansatz verfolgt. Das heißt es werden in der Regel keine beweglichen Fenster angewandt, was zum Großteil auch an den dafür nicht geeigneten Eingabegeräten liegt. Andere graphische Elemente wie scrollende Listen oder Splash Screens werden jedoch eingesetzt. Die Art und das Einsatzgebiet dieser Elemente werden stark von den benutzten Eingabegeräten bestimmt. So wäre eine Bildschirmtastatur mit QWERTZ-Layout in Kombination mit einem iDrive Controller nicht sinnvoll, die im iDrive Menü eingesetzte lineare Anordnung jedoch schon. Das rein screenbasierte Layout stößt vor allem bei den zukünftigen noch vielfältigeren Anforderungen an seine Grenzen, da ab einer gewissen Anzahl an Screens die Übersicht stark leidet und vor allem der Zusammenhang zwischen ihnen leicht verloren gehen kann. In Abbildung 2.4 ist ein beispielhafter Menübaum eines aktuellen FIS skizziert. Mit schwarz und grau sind die Menüs und deren logische Verknüpfungen zu erkennen, die man standardmäßig durchlaufen kann. In magenta sind exemplarisch Sprünge markiert, die durch Drücken der „Home“-Taste zum Ausgangspunkt von beliebigen Menüästen durchgeführt werden können. Zusätzlich ist ein Sprung in blau markiert, der von jedem Menüpunkt aus bei Drücken der Navigationstaste ausgeführt wird. Dies scheint zunächst ein Bruch in der Bedienlogik zu sein, doch entspricht dies mehr der Denkweise und den Anfor-

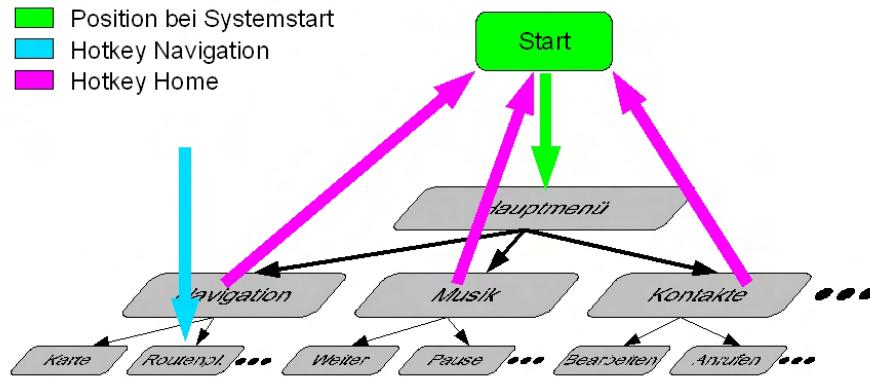


Abbildung 2.4: Menüaufbau aktueller FIS

derungen des Nutzers. Gleichzeitig wird eine Hilfestellung gegeben, falls keine Orientierung mehr vorhanden sein sollte und der Benutzer auf diese Weise an definierte Stellen im System springen kann. Dieser Menüaufbau kann für künftige FIS nicht beibehalten werden, wenn zusätzliche Funktionalitäten wie Email, Terminverwaltung oder Multimediadatenbanken hinzukommen. Menüsprünge werden jedoch weiterhin möglich müssen und auch sinnvoll sein, doch sollten sie beispielsweise durch animierte Transitionen dem Benutzer verdeutlicht werden. Das optische Feedback wird dadurch wesentlich verbessert und kann zusätzlich den Joy Of Use erhöhen. Auch wird dadurch die Bildung einer mentalen Karte unterstützt, wie in [4] untersucht wird. Gleichzeitig können nicht die von Desktop Systemen bekannten Vorgehensweisen angewandt werden aufgrund der stark unterschiedlichen Rahmenbedingungen. Es müssen also neue Interaktionsmodelle gefunden werden, die den bekannten in nichts nachstehen.

Die optische Umsetzung und der darunter liegende Menübaum sind die für den Benutzer offensichtlichsten Eigenschaften einer graphischen Oberfläche. Je selbstverständlicher und intuitiver die vielen Details dieser Schnittstelle sind, desto stressfreier und unterhaltsamer ist die Arbeit damit. Der Nutzer kann diese Eigenschaften oft nicht eindeutig identifizieren, doch sind sie essentieller Bestandteil des gesamten Bedienkonzepts. Bei Personal Computer-Schnittstellen gilt es meist, möglichst viel Information und viele Funktionen um diese zu bearbeiten zur Verfügung zu stellen. Da Zoom-Techniken hier bisher noch nicht verwendet werden, fallen die Schaltflächen, Symbole und Text oft sehr klein aus. So sind zwar viele Informationen gleichzeitig zu sehen, jedoch sinkt dabei auch die Übersichtlichkeit. Erfahrene Benutzer für die jeweilige Applikation haben so nach einer gewissen Einarbeitungszeit Vorteile, ein Anfänger ist jedoch die meiste Zeit mit der Funktionssuche beschäftigt. Werden viele Funktionalitäten allerdings in mehreren Menüebenen verschachtelt, kann wieder das Problem des schnellen Auffindens eintreten. Es muss also ein der Anwendungssituation angepasster Kompromiss gefunden werden. Bei Desktop-Systemen verschiebt sich dieser in Richtung vieler gleichzeitig erreichbarer Funktionen und großer Informationsmenge. So sind bei einer durchschnittlichen Textverarbeitung ca. 50 Buttons und ca. 10 Pull Down Menüs zu sehen, wobei die Zahl der Elemente durch zusätzliche Werkzeugleisten leicht erhöht werden kann. Microsoft Windows XP lässt standardmäßig die Definition von ca. 20 Systemfarben zu, die nahezu den

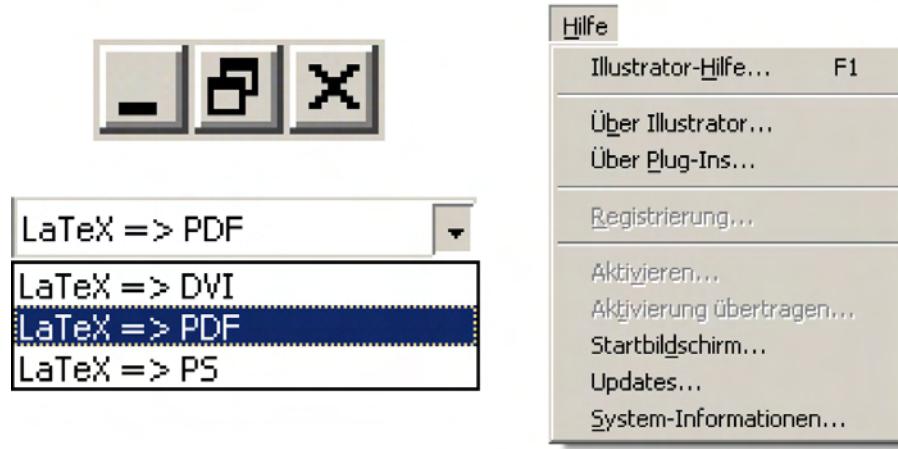


Abbildung 2.5: links oben: Fensterbuttons, links unten: Drop Down Liste, rechts: Pull Down Menü

gesamten optischen Eindruck festlegen. Im Automobil wächst die Funktionsvielfalt rapide an, wobei zur Funktionssuche, Bedienung und Eingabe wesentlich weniger Zeit und Aufmerksamkeit vorhanden ist. Der oben beschriebene Kompromiss wird also mehr in Richtung Übersichtlichkeit und robuste Eingabe verschoben. Es werden also weniger verschiedene Farben und Interface Widgets herangezogen. Mit dem Begriff „Interface Widgets“ sind Standardelemente wie Pull Down Menüs, Drop Down Listen, Fenstersteuerungsbuttons oder Popup Menüs zusammengefasst. Grundsätzlich sind weniger graphische Elemente vorhanden, einerseits um eine schnelle Erfassung durch den Benutzer zu ermöglichen und andererseits um die Elemente in einer sinnvollen Größe darstellen zu können.

Die Anforderungen, die aus den obigen Überlegungen abgeleitet werden können setzen sich dann wie folgt zusammen:

- geringe Menütiefe
- wenige effiziente Bedienelemente
- geringe Farbenanzahl
- animierte Menütransitions
- Nutzung mehrerer Modalitäten

Da bei der hier implementierten Oberfläche nur immer ein Menü maximiert sein kann und Menüs auch nicht verschoben werden können, werden einige Schaltflächen aus bekannten WIMP-Systemen nicht benötigt. Es wird keine Funktion „Minimieren“ geben, lediglich ein „Maximieren“- oder ein „Schließen“-Button wird eingesetzt. Um das System aus jedem Zustand in den Ausgangszustand zu versetzen, wird in der permanent vorhandenen Taskleiste eine „Home“-Schaltfläche plaziert.

Ist einer der Hauptmenüpunkte maximiert, befinden sich alle anderen minimiert in der Taskleiste, so dass sie jederzeit angewählt werden können. Hauptmenüpunkte können sich so nie überlappen, wodurch auch keine Anforderung für verschiebbare

Fenster besteht. Zusätzlich zu den zu diesen Fenstern können Popup-Fenster erzeugt werden, die wiederum Hauptmenüfenster verdecken können. Sie werden meist als reine Informationsfenster oder zur Texteingabe herangezogen. Vor allem bei während der Texteingabe ist es wichtig, dass diese nicht versehentlich durch Aktivierung eines anderen Hauptmenüs unterbrochen wird und daher bewusst bestätigt und geschlossen werden muss. Popups werden in dem vorliegenden Menü bildschirmfüllend angelegt, die Taskleiste bleibt jedoch immer frei und zugänglich.

Zur optischen Bestätigung einer Aktion werden zum Beispiel beim Speichern eines Kontakts oder der Routenberechnung sogenannte „Splash-Screens“ benutzt. Sie werden auf oberster Ebene über alle graphischen Elemente zentriert platziert, nehmen im Gegensatz zu Popups nur einen Bruchteil der Gesamtfläche ein. Diese Screens haben reinen Informationscharakter, bieten also keine Eingabemöglichkeit. Sie werden automatisch nach wenigen Sekunden ausgeblendet.

## 2.2 Zoomable Interface

Zur Erfüllung der ersten und teilweise der zweiten Anforderung wurde ein neues Menükonzept erarbeitet. Es basiert auf der Idee eines „Zoomable Interface“, wie es bereits seit längerem am Human Computer Interaction Lab<sup>3</sup> an der University of Maryland erforscht wird. Die Grundidee besteht darin, dass Interfaceobjekte vergrößert oder an sie herangezoomt werden kann. So können innerhalb dieses Objekts detailliertere Informationen dargestellt werden und weitere Interaktionsmöglichkeiten geboten werden. Es wird also die Anzahl von Objekten auf dem Bildschirm überschaubar gehalten während der benötigte Detaillierungsgrad erreicht werden kann. Dies kann die zur Durchführung einer Aktion benötigte Zeit um bis zu 30% verkürzen, wie in [1] erarbeitet wurde. Die Abbildung 2.6 zeigt eine Präsentationssoftware die als Demonstrator für Zoomeffekte am HCIL erstellt wurde. Befindet sich der Mauszeiger im oberen Teil des Bildschirms, so wird bei einem Klick die nächste Folie angezeigt. Die geschieht allerdings nicht abrupt, sondern mit einer Animation bei der die alte Folie verkleinert und an den unteren Bildschirmrand verschoben wird während die neue aus ihrer Ursprungsposition aus der Liste am unteren Rand hereingeschoben wird. Befindet sich der Mauszeiger im unteren Viertel des Bildschirms, so wird die horizontale Liste mit Miniaturansichten der Folien und einer Art FishEye Effekt eingeblendet und die Hauptfolie entsprechend verkleinert.

Die soeben beschriebenen Animationen und Zoomfunktionen wurden bei dieser Arbeit kombiniert mit Menüstrukturen, wie sie bei Abbildung 2.7 beispielhaft dargestellt sind. Für FishEye- und Zoomeffekte wurden folgende Richtlinien festgelegt:

- Vergrößerung um maximal 100%
- die kleinsten Elemente müssen entweder lesbar sein oder ausgeblendet sein
- kleinere Elemente werden verschoben, um größeren Platz zu machen
- ab einer bestimmten Annäherung werden die Elemente festgehalten

Diese Richtlinien gewährleisten eine minimale Bedienbarkeit und effektive Darstellung.

---

<sup>3</sup>HCIL

## Piccolo.NET: What is it?

- Toolkit that supports:
  - structured canvas of graphical objects
  - hierarchical scenegraph model
  - creation of "controls" for use within Windows Forms
- Supports 2D object-oriented graphics
  - hierarchies (transformation, transparency, etc.)
  - animation, event handling
  - cameras, layers, views
  - efficiency mechanisms
- Missing structure that relies on underlying renderer
  - GDI+, Direct3D, OpenGL, Java2D, etc.

=> Open, Extensible and Efficient

Where Does It Run?

- Almost everywhere
  - Piccolo.NET
    - WinForms
    - WPF
    - Silverlight
    - Zoom
  - Compact Framework for Pocket PC
  - Piccolo Java
    - Works for everywhere else

5

Abbildung 2.6: Präsentationssoftware mit Zoomfunktion, aus [28]

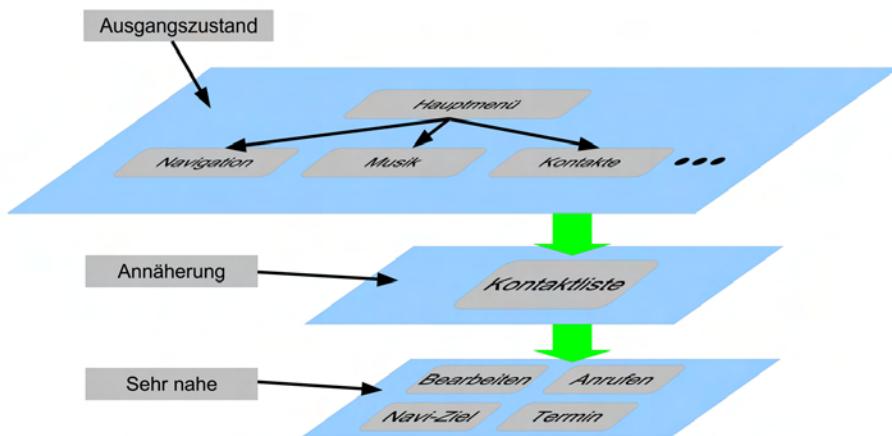


Abbildung 2.7: Beispiel eines Zoomable Interfaces



Abbildung 2.8: Implementation des Hauptmenüs für das PreTouch-System

In Abbildung 2.7 werden drei Zoomstufen für einen Teil eines Kontaktmenüs skizziert. Bei der niedrigsten Stufe ist lediglich die Übersicht des Hauptmenüs angezeigt, wobei das Kontaktmenü eines von mehreren Menüobjekten darstellt. Bewegt sich der Nutzer in Richtung des Kontaktobjekts, wird es vergrößert und zusätzliche Informationen und Funktionstasten werden eingeblendet. In dem zu dieser Arbeit erstellten Demonstrator wird eine Kontaktliste angezeigt und eine Schaltfläche zur bildschirmfüllenden Vergrößerung des Menüs. Wird diese gedrückt, wird das Menü zentriert und maximiert und mit allen verfügbaren Funktionen angezeigt. Um trotz dieser neuen Bildschirmaufteilung noch einen schnellen Zugriff auf die anderen Menüs zu gewährleisten, werden diese in einer miniaturisierten Variante in einer Art Taskleiste am unteren Bildschirmrand angeordnet.

Es wurde also kurzzeitig die Informationsdichte an einer vom Benutzer ausgewählten Stelle erhöht, was grundsätzlich auch der menschlichen Wahrnehmung entspricht. Nun kann aus der Liste ein Kontakt ausgewählt werden, welcher dann ebenfalls vergrößert dargestellt wird. Erhöht man den Zoomfaktor noch weiter, so werden Funktionstasten zu dem aktuell fokussierten Kontakt angezeigt um ihn z.B. anzurufen oder zu bearbeiten. Es ist also mit einer einzigen Bewegung möglich, gezielt einen Kontakt anzurufen. Diese Aktion würde bei den meisten aktuellen FIS oder auch Mobiltelefonen mehrere Menüsprünge erfordern. Wird nun der Zoomfaktor wieder reduziert, so sinkt der Detailgrad wieder und das Kontaktmenü ist in der ursprünglichen Skalierung an der Ausgangsposition.

Eine Steuerung der Zoomtiefe ist mit herkömmlichen Eingabegeräten nur unzureichend möglich. Des Weiteren ist für eine Anwendung im Fahrzeug kein Gerät wie eine Maus oder ein markerbasiertes Trackingsystem nicht akzeptabel. Die Anforderung ist also ein Eingabegerät, das die kontinuierliche Positionierung eines Cursors auf der Bildschirmfläche erlaubt und eine Tiefeninformation zur Verfügung stellen

kann. In dieser Arbeit wird dafür ein kapazitiver Touch Screen benutzt, der mit Projected Capacitive Messtechnik arbeitet. Diese Messmethode benötigt zur Positionsmessung keine Berührung und kann daher dreidimensional die Fingerposition erfassen. Des weiteren ist für diese Sensoren kein Marker bzw. Target erforderlich, da wie zuvor beschrieben die Kapazität des menschlichen Körpers ausgenutzt wird. Wie in Abbildung 2.8 zu sehen ist, wurden lediglich die Farben „schwarz“ für den Hintergrund, „weiß“ für Text und „orange“ für Menüänder, Schaltflächen und Piktogramme benutzt. Dadurch wird nicht nur ein simples und klares Design erreicht, sondern vor allem die schnelle Orientierung und gute Lesbarkeit bei verschiedenen Lichtverhältnissen gewährleistet. Des weiteren wird eine technische und effiziente Optik erreicht, die sich gut in vorhandene Bedienelemente im Automobil einfügt. Grundsätzlich werden keine Farbflächen oder Farbveläufe benutzt um keine unnötigen graphischen Elemente einzubringen.

## 2.3 Multimodalität

Unter Multimodalität versteht man, dass ein MMI mit mehreren Modalitäten, also Kommunikationswegen, bedient werden kann. In dieser Arbeit werden die Eingaben mit Berührung bzw. Annäherung und Sprache kombiniert. Bei vielen multimodalen Benutzerschnittstellen werden gewisse Eingaben immer einer Modalität zugeordnet, beispielsweise die Menünavigation der Toucheingabe und die Texteingabe der Spracheingabe. Im Gegensatz dazu wird in dieser Arbeit eine Mischung vorgenommen, das heißt gewisse Aktionen können von mehreren Modalitäten ausgelöst werden. Es handelt sich dabei immer um aktive Eingabemodi, das heißt der Benutzer muss immer bewusst eine Eingabe initiieren. Nach [5] ist die Interaktion „zeitlich kaskadiert“, wobei Eingaben der einen Modalität Einschränkungen für die andere Modalität bedeuten können.

Aktive Eingabemodi sind unter anderem:

- Berührung
- Gesten (Hand, Kopf, Augen, etc.)
- Sprache

Passive Modi sind im Gegensatz dazu unwillkürliche Aktionen des Nutzers, die als Informationsquelle für ein Interface dienen können:

- Mimik
- Lidschlag, Blickrichtung (bedingt)
- Körpertemperatur, Puls, Hautleitfähigkeit

Ein Beispiel bei dem die beiden Modalitäten Sprache und Berührung vollkommen parallel und unabhängig voneinander einsetzbar sind, ist der in dem Demonstrator implementierte Lautstärkeregler in Abbildung 2.9. Die Modalitäten stehen in direkter Konkurrenz, wobei keine eine höhere Priorität besitzt. Die Wahrscheinlichkeit eines von beiden absolut gleichzeitig ausgelösten Manipulationsereignisses ist vernachlässigbar klein. Sollte es doch zu einer Kollision kommen, könnte ein von der

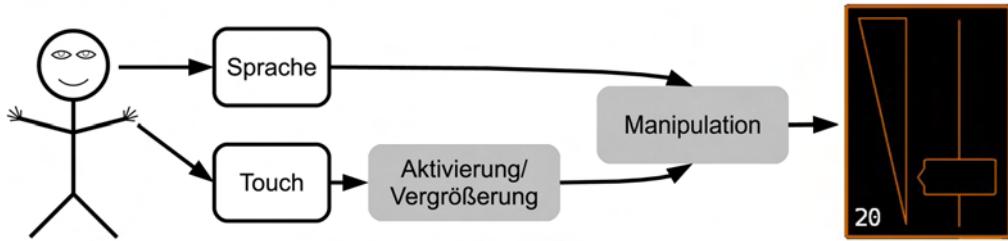


Abbildung 2.9: Flussdiagramm für multimodale Slidermanipulation

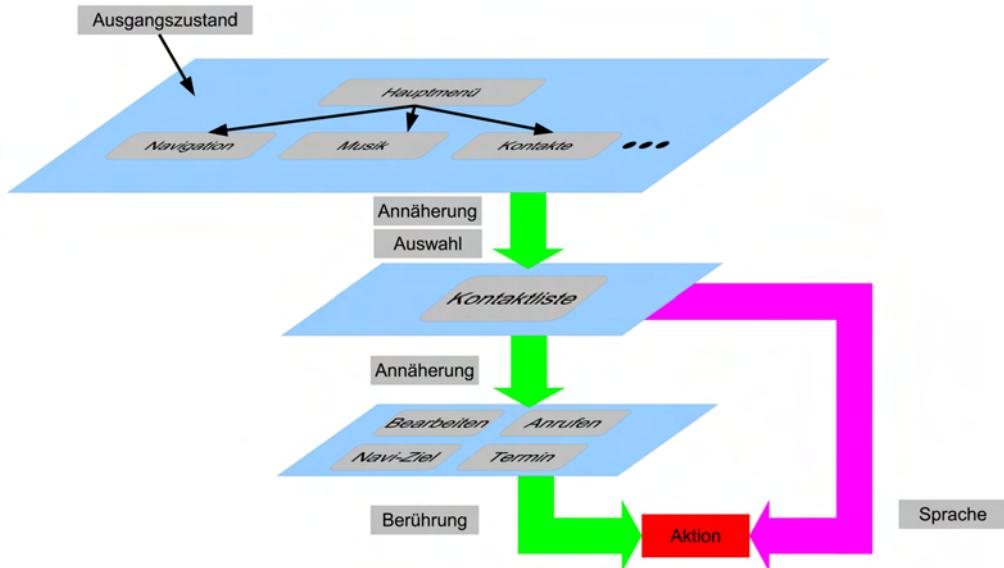


Abbildung 2.10: Ablaufdiagramm für einen „Point And Talk“-Vorgang

Benutzerintention abweichender Wert eingestellt werden, der jedoch keine Fehlfunktion des Interfaces hervorrufen würde.

Trotz der starken Parallelität bestehen Unterschiede bei den zur Auslösung einer bestimmten Aktion benötigten Schritte. Wie in Abbildung zu sehen ist, muss zur Manipulation per Touch Screen der Slider erst mit einer Handannäherung aktiviert werden und kann dann mit einer Berührung verschoben werden. Bei Spracheingabe, z.B. „Musik lauter“, wird die Sliderposition auch ohne vorherige Aktivierung verändert. Gleichzeitig kann auch bei aktiviertem Slider eine Manipulation per Spracheingabe erfolgen. Eine weitere Möglichkeit zur Verwendung multimodaler Eingaben bietet die Menünavigation und die Listenauswahl. Bei klassischen WIMP-Interfaces wird die Eingabe nach dem „Point And Click“-Vorfahren durchgeführt. Dies ist auch mit dem Touch Screen problemlos möglich, doch durch die Sprache wird zusätzlich die Option des „Point And Talk“ angeboten. Hierbei zeigt der Nutzer im Erkennungsbereich des kapazitiven Sensors auf ein Interfaceobjekt und initiiert durch ein Sprachkommando eine damit verknüpfte Aktion. Im Unterschied zur vorherigen reinen Parallelität der Eingaben sind diese Aktionen zwingend abhängig voneinander, was in Abbildung 2.10 dargestellt wird. So wäre also die Toucheingabe alleine ohne weitere Konsequenz, es würde lediglich ein Interfaceobjekt markiert bzw. vergrößert werden. Auch eine Spracheingabe wie „Kontakt anrufen“ würde keine weitere Aktion

auslösen, da kein Kontakt markiert ist oder gar kein Kontakt angezeigt wird. Ein weiteres Beispiel für einen „Point And Talk“-Vorgang wäre das Maximieren von Hauptmenüpunkten. Der Benutzer kann beispielsweise auf das Menü Navigation zeigen und den Sprachbefehl „Menü maximieren“ geben. Dieses wird dann bildschirmfüllend vergrößert und die anderen Menüs verkleinert in die Taskleiste verschoben. Der Befehl „Menü Start“ bringt das gesamte Interface wieder in den Ausgangszustand zurück. Im Gegensatz zu vorher ist keine zusätzliche Eingabe vom Touch Screen nötig, da das Sprachkommando in jedem Systemzustand die gleiche Funktion hat und nicht auf ein spezielles Objekt wirkt.

Der in dieser Arbeit verwendete Spracherkennner ScanSoft VoCon3200 ist durchgehend aktiv, es muss also nicht wie bei vielen bestehenden FIS ein „Push To Talk“-Knopf gedrückt werden um ein Kommando zu initiieren. Um ein für das FIS bestimmtes Kommando von zufällig erkannten Segmenten aus dem natürlichen Sprachfluss zu unterscheiden, müssen die Befehle einer vorgegebenen Grammatik folgen. Dabei haben sämtliche Befehle den Aufbau „Subjekt + Verb“, die zugrundeliegende BNF<sup>4</sup> lautet wie folgt:

```
<volume> ::= "lauter" | "leiser";  
  
<menuaktion> ::= "maximieren" | "start" | "navigation" | "musik" | "kontakte";  
  
<musik> ::= "musik" <volume>;  
  
<menue> ::= "menu" <menuaktion>;
```

Obige Grammatik umfasst nur die vorher aufgeführten Beispiele. Um eine umfassendere Spracheingabe zu ermöglichen werden noch zusätzliche Befehle benötigt um Musiktitel abzuspielen, Kontakte anzurufen oder Routenziele zu buchstabieren. Letzteres könnte alternativ zur Texteingabe per Bildschirmtastatur erfolgen.

## 2.4 Menübedienung

Wie oben beschrieben soll die Bedienung multimodal erfolgen, wobei die beiden Modalitäten „Berührung“ und „Sprache“ benutzt werden können.

Zuerst soll die Eingabe per Touch Screen beschrieben werden. Der kapazitive Sensor liefert grundsätzlich kontinuierliche Werte für x-, y- und z-Achse, wobei „x“ und „y“ den 2D-Bildschirmkoordinaten entsprechen und „z“ die Entfernung zur Bildschirmoberfläche darstellt. Der Ursprung des Koordinatensystems liegt also in der linken oberen Ecke des Touch Screens. Bei manchen Interfaceobjekten ist es sinnvoll, den durch die z-Achse aufgespannten Bereich zu unterteilen, da das System zwar grundsätzlich „zoomable“ ist, viele graphische Elemente aber nur gewisse diskrete Zustände einnehmen können. Es wird also eine Zuordnung zwischen dem Zustand des aktuell markierten Interfaceobjekts und dem Fingerabstand gemacht. Dabei wird für die meisten Objekte zwischen zwei Abstandsbereichen unterschieden, zwischen denen ein hysteresebehafteter Übergang stattfindet. Die Hysterese verhindert dabei den Zustand, dass ein Objekt zwischen beiden Zuständen mit großer Frequenz hin- und herspringt wenn der Finger sich an der Grenze zwischen den Bereichen befindet und

---

<sup>4</sup>Backus-Naur Form

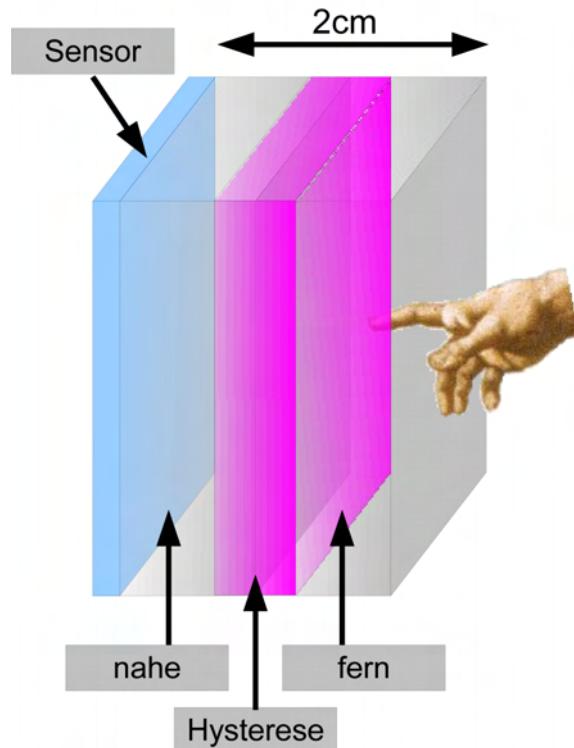


Abbildung 2.11: Aufteilung des Sensorraums

Sensorrauschen vorhanden ist. Zusätzlich wird die Eingabe dadurch vom Benutzer als robuster empfunden, da ungenaue Eingaben etwas kompensiert werden bzw. der letzte Zustand länger beibehalten wird. Grundsätzlich können sämtliche Vorgänge mit dem Touch Screen durchgeführt werden. Sowohl Menünavigation als auch Texteingabe sind primär für diese Modalität bestimmt. Die Spracheingabe kann alternativ zu Toucheingaben benutzt werden, diese ergänzen und teilweise andere Menüsprünge initiieren. Alle Interfaceaktionen können im Normalfall auch sinnvoll durch Sprachbefehle gesteuert werden, lediglich die Texteingabe würde mit einer reinen Spracherkennung praktikabler sein als die Steuerung einer Bildschirmtastatur damit.

Obwohl die Annäherungsinformation den Zustand von Interfaceobjekten beeinflusst, werden jedoch keine Daten oder Menüobjekte dadurch manipuliert. Dies erfordert immer eine echte Berührung des Bildschirms, um zusätzlich ein haptisches Feedback zu gewährleisten. Ebenso wird dadurch erreicht, dass keine Daten durch ein zufälliges Vorbeistreichen mit der Hand am Bildschirm ungewollt verändert werden.

## 2.5 Texteingabemethoden

Neben der Menünavigation wurden verschiedene Texteingabemethoden untersucht, die durch einen Touch Screen und vor allem unter Einbeziehung der Annäherungsinformation bedient werden können. Gängige Bildschirmtastaturen wie EyesBoard in Abbildung 2.12 haben das gleiche Tastaturlayout wie eine physikalische Standardtastatur. Um ein möglichst natürliches Schreiben mit 10 Fingern zu ermöglichen, wird zwingend ein Sensor benötigt der auf Fingerberührungen reagiert. Bei Sensoren für Stifteingabe muss jede Taste einzeln gesucht und gedrückt werden, was einen we-



Abbildung 2.12: Links: Bildschirmtastatur EyesBoard, aus [29], rechts: POBox, aus [7]

sentlich höheren Zeitaufwand bedeutet. Eine weitere Einschränkung kann die Größe der zur Verfügung stehenden Bildschirmfläche sein. Bei Handheld-Systemen ist die Fläche meist so klein, dass ohnehin nur eine sichere Eingabe per Stift möglich ist. Zusätzlich zu einer reinen Buchstabeneingabe bieten manche Systeme wie in Abbildung 2.12 rechts zu sehen ist verschiedene Vorschläge aus einem Wörterbuch an. Befindet sich das gewünschte darunter, so kann dieses ausgewählt werden und der Rest des Wortes muss nicht mehr eingegeben werden. Dieses System der Vervollständigung durch Wortvorschläge findet sowohl bei Bildschirmtastaturen als auch bei der Nutzung von Nummerntasten bei Mobiltelefonen (T9) Verwendung.

Eine weitere Verringerung der nötigen Eingaben ergibt sich, wenn alle Möglichkeiten aus einem endlichen Repertoire von Wörtern besteht. So kann die Anzahl der angezeigten Optionen verringert werden, sobald Teile eines Worts bekannt sind und dadurch gewisse Buchstabenkombinationen nicht mehr auftreten können. Dieses System wird beispielsweise bei der Suche in Adresslisten bei iDrive von BMW angewandt. Eine solche Unterstützung kann sich aber auch nachteilig auswirken, z.B. wenn ein dem System unbekanntes Wort eingegeben werden soll und dies durch die reduzierte Buchstabenanzahl nicht mehr möglich ist.

Das Ziel viel Text mit möglichst wenigen Aktionen einzugeben wird also momentan auf verschiedenste Weise gelöst. Dabei muss oft ein Kompromiss geschlossen werden zwischen Flexibilität bei der Wortwahl und Verkürzung der Eingabedauer. Bei Bildschirmtastaturen besteht vor allem das Problem der Treffsicherheit und der Übersichtlichkeit. Dem wurde in den unten beschriebenen Texteingabemethoden Rechnung getragen.

Ein weiterer Unterschied bei den Bildschirmtastaturen besteht darin, dass die Tastatur entweder den ganzen Bildschirm ausfüllt und dadurch den Tippkomfort erhöht, oder dass nur ein Teil der Anzeigefläche zu Lasten des Komforts verwendet wird. Letzteres erlaubt aber eine bessere Übersicht und Orientierung, wo der Text momentan eingefügt wird. Die Verwendung von Zoom- und FishEye-Effekten erlaubt eine Minimierung der benötigten Fläche bei höherer Treffsicherheit gegenüber einer normalen Bildschirmtastatur.

Die Texteingaben werden durch Berührung einer Textzeile eingeblendet und zeigen mit einem pfeilförmigen Rahmen die momentan gewählte Zeile an. Wird eine andere Zeile berührt, so bewegt sich die Eingabe an diese Stelle. Dadurch bleibt eine gute Orientierung für den Nutzer über den Systemzustand erhalten.

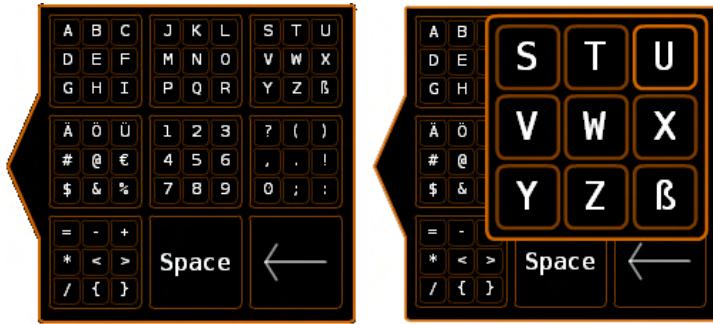


Abbildung 2.13: Links: MatrixKey normal, rechts: MatrixKey beim Maximieren

### 2.5.1 MatrixKey

Die im folgenden beschriebene Eingabemethode basiert auf einer Entwicklung der „Linköpings Universitet“ und „IBM Almaden Research Center“ wie sie in [6] erklärt wird. Der ursprüngliche Verwendungsbereich ist als Texteingabe für Geräte mit einer Fernbedienung als Eingabemedium. Speziell wird hier lediglich der normalerweise quadratisch angeordnete Ziffernblock der Zahlen „1“ bis „9“ benötigt. Durch geschickte Kombination dieser Tasten lässt sich das gesamte Alphabet, Ziffern, Satz- und Sonderzeichen abbilden. Wie in Abbildung 2.13 links zu sehen ist, wird die Fläche der Texteingabe in neun quadratische Felder eingeteilt, die teilweise wiederum neun Felder mit den Zeichen enthalten. Eine Sonderposition nehmen bei dieser Implementation die Leertaste „Space“ und die Löschtaste „<-“ ein, da sie ein ganzes der Hauptfelder belegen. Dies erleichtert die Erreichbarkeit dieser oft benutzten Funktionen. Insgesamt stehen also 65 Tasten zur Verfügung, wobei noch die Möglichkeit besteht, die Belegung mit einer Hochstelltaste zu ändern. Somit ist dann auch Groß- und Kleinschreibung sowie noch weitere Sonderzeichen möglich.

Bei der in [6] vorgestellten Variante wird mit dem Ziffernblock zuerst das gewünschte Hauptfeld gewählt, welches dann farblich markiert wird. Mit einem zweiten Tastendruck kann dann das gewünschte Zeichen ausgewählt, das dann an das eingegebene Wort angehängt wird. Da bei dem hier implementierten System jedoch ein Touch Screen zur Eingabe benutzt wird, muss die Feldmarkierung und Auswahl auf andere Weise erfolgen. Durch Annäherung an eines der Hauptfelder wird dieses zuerst farblich hervorgehoben und bei weiterer Annäherung vergrößert. Bei Berührung wird das jeweilige Zeichen an die bisherige Eingabe angehängt. In Abbildung 2.14 ist die oben beschriebene Zuordnung von momentaner Annäherung und dem dadurch erzeugten Zustand der Texteingabe dargestellt. Es ist damit also möglich, mit einer einzigen Bewegung ein Zeichen zu markieren und auszuwählen. Zusätzlich wird durch die Vergrößerung die Treffsicherheit erhöht, was ein entspannteres Schreiben ermöglicht. Von der ursprünglichen Idee zu dieser Texteingabe wurden also lediglich das grundsätzliche Layout und die Aufteilung des Eingabevorgangs in mehrere Phasen übernommen.

### 2.5.2 FishKey

Bei dieser Texteingabe wird das bekannte Layout vorhandener Bildschirmtastaturen übernommen. Um jedoch die Größe zu reduzieren, wurden die einzelnen Tasten verkleinert, so dass die Eingabe lediglich ca. 1/6 der Bildschirmfläche benötigt. Da

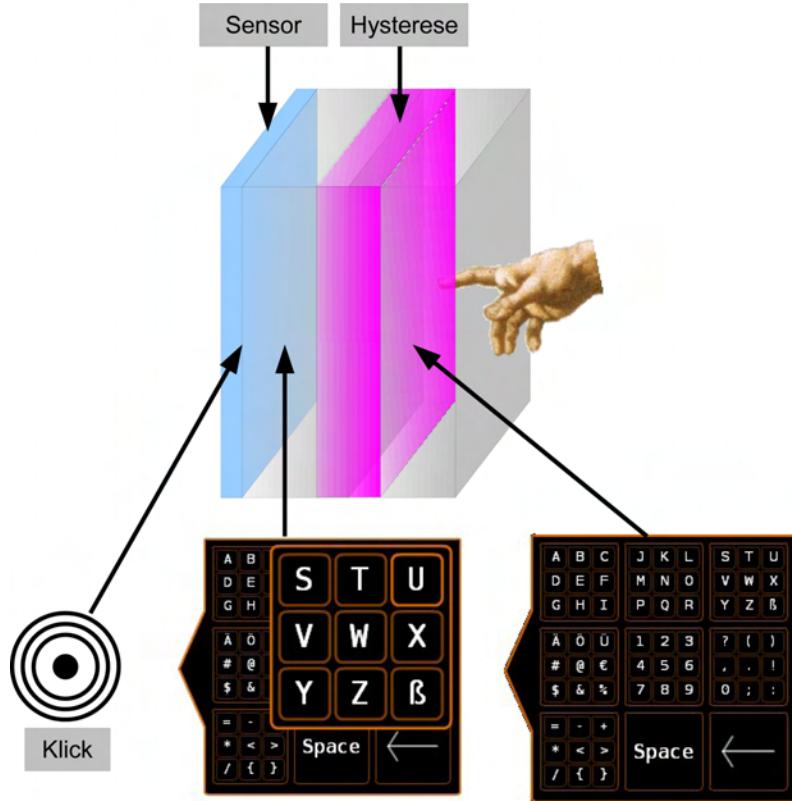


Abbildung 2.14: Zuordnung der Annäherungsregionen auf den Interfacezustand

diese Tastengröße für einen Touch Screen wesentlich zu klein ist, wird der unter dem Finger liegende Bereich mit einer Art FishEye-Technik vergrößert. Hierfür werden die Tasten einerseits skaliert und gleichzeitig vom Finger weg verschoben, um für die größeren Tasten Platz zu schaffen.

Zur Realisierung dieser Funktionalität wird von jeder Taste die Ausgangsposition permanent gespeichert. Bei jeder Fingerbewegung wird dann der Abstand und indirekt auch die Position des Fingers relativ zu jeder Taste berechnet. Aus diesem Abstand werden daraufhin die Richtung und Größe der Translation und die Größe der Skalierung berechnet. Das Resultat sind dann eine Verschiebung  $\Delta x$ ,  $\Delta y$  und die Skalierung  $m$  in Prozent. Als Zwischengröße wird hierfür der Wert  $diff$  errechnet, der wie eine Verschiebungsskalierung zu verstehen ist. Die Fingerdistanz zum Bildschirm wird mit  $z$  bezeichnet.

Die zugrundeliegenden Gleichungen lauten wie folgt, eine Übersicht der verwendeten Größen ist in Abbildung 2.15 zu sehen:

$$r = \sqrt{dx^2 + dy^2} \quad (2.1)$$

$$diff = r \cdot \frac{30}{10 + (z + 10) \cdot e^{[0.03 \cdot r]}} \quad (2.2)$$

$$\Delta x = dx \cdot \frac{diff}{r - 10} \quad (2.3)$$

$$\Delta x = 0 \text{ für } r < 15 \quad (2.4)$$

$$x = x_{alt} + \Delta x \quad (2.5)$$

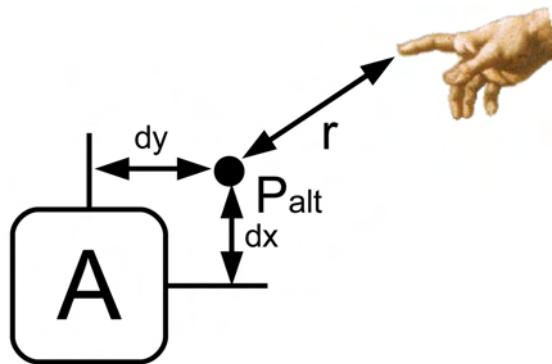


Abbildung 2.15: Verwendete Größen für den FishEye-Effekt

$$m = 100 + \frac{3000}{z \cdot e^{0.001 \cdot r^2}} \quad (2.6)$$

$$m = 250 \text{ für } m > 250 \quad (2.7)$$

Die zugehörigen Graphen für Translation und Skalierung sind in Abbildung 2.16 dargestellt. In beiden sind bei kleinen Distanzen  $r$  Sättigungseffekte zu erkennen. Dies bewirkt vor allem bei der Translation einen „Einschnappeffekt“, der bei der Bedienung unterstützend wirkt. Gleichzeitig wird bei der Skalierung ein Maximum festgelegt, um eine sinnvolle Begrenzung zu gewährleisten. Im Gegensatz zur oben beschriebenen MatrixKey-Eingabe wird hier die Annäherungsinformation kontinuierlich verarbeitet. Dies hat einen positiven Einfluss auf den Joy Of Use, kann jedoch auf manche Benutzer zu unruhig wirken. Die Implementierung der oben beschriebenen Texteingabe ist in Abbildung 2.17 dargestellt. Ebenfalls zu erkennen ist dabei der stilisierte Pfeil an der Kante des Außenrahmens und das Kreuzsymbol zum Schließen der Eingabe. Deutlich zu sehen ist die Skalierung des unter dem Benutzerfinger liegenden Zeichens und die Verschiebung der umliegenden Tasten. Aus obigen Formeln ist zu erschließen, dass die maximale Skalierung für jeweils x- und y-Richtung 2,5 beträgt, also die Fläche um bis zu einem Faktor von 6,25 erhöht wird. Damit wird sowohl die Lesbarkeit als auch die Treffsicherheit wesentlich verbessert. Vorteilhaft ist auch das gewohnte QWERTZ-Tastenlayout, wodurch die Suchdauer für einzelne Zeichen verkürzt wird.

## 2.6 Listennavigation

Neben der Texteingabe wird auch ein Konzept für die Listenanzeige und -navigation unter Einbeziehung der Annäherungsinformation erarbeitet. In den meisten gängigen graphischen Oberflächen besteht eine Liste aus einer Box mit einer festen angezeigten Zeilenanzahl, bei der im Falle von zu vielen Einträgen ein Scrollbar und Pfeilbuttons angefügt werden.

Eine solche Liste ist jedoch zur Bedienung mit einem Touch Screen im Automobil nicht sehr gut geeignet. Durch die begrenzte Bildschirmfläche können nicht sehr viele Einträge in ausreichender Größe angezeigt werden. Auch wird für die Navigation in den Listen ein feinfühliges Eingabegerät wie z.B. eine Maus benötigt, was ebenfalls nicht zur Verfügung steht. Die Bewegung in Listen geschieht an einem Desktop-System normalerweise mit der Maus indem entweder auf die Scrolltasten

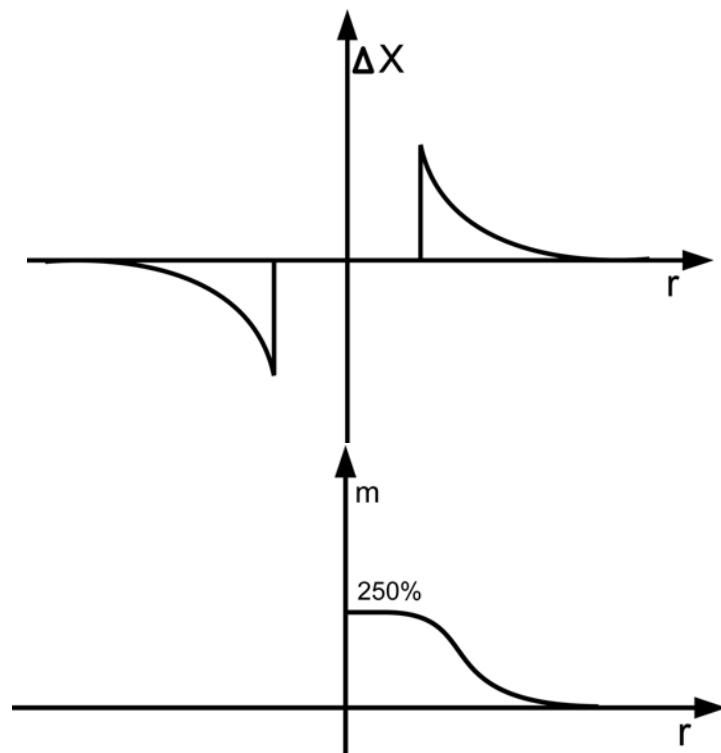


Abbildung 2.16: Oben: Translation, unten: Skalierung

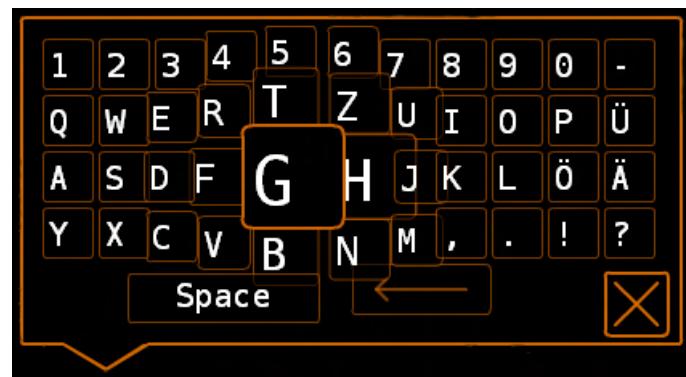


Abbildung 2.17: Bildschirmtastatur „FishKey“

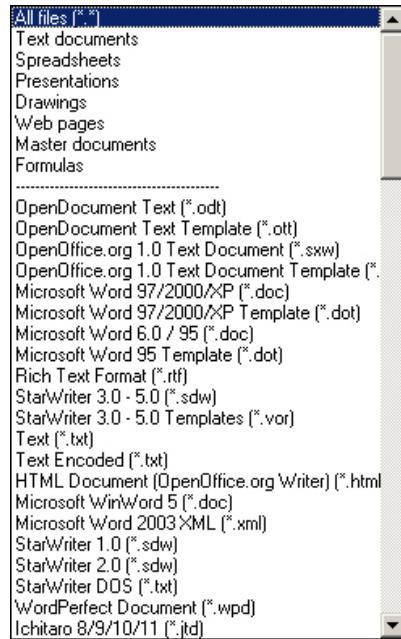


Abbildung 2.18: Gängige Liste mit Scrollbar

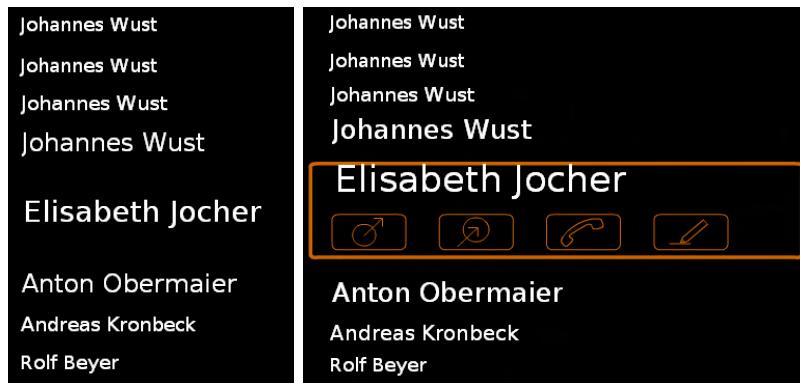


Abbildung 2.19: Links: FishEye-Liste, rechts: FishEye-Liste mit Funktionsbuttons

an den Listenenden geklickt wird oder der Scrollbalken „gezogen“ wird. Alternativ kann nachdem die Liste markiert wurde mit den Pfeiltasten der Tastatur die Liste durchsucht werden.

Für die Verwendung im Fahrzeug wird daher ein alternatives System benutzt. Um die Lesbarkeit zu erhöhen und gleichzeitig die Übersichtlichkeit zu bewahren wird ein FishEye Effekt ähnlich dem der oben beschriebenen Texteingabe angewandt. Die dabei durchgeführte Skalierung und Verschiebung arbeitet nach dem gleichen Prinzip wie bei der FishKey-Tastatur, nur dass sich die Effekte auf eine Dimension beschränken. Ein Beispiel dafür ist in Abbildung 2.19 auf der linken Seite zu sehen. Auch hier wird ein „Einschnappen“ angewandt, um die Bedienung robuster zu gestalten. Bei gängigen graphischen Oberflächen werden Listen meist lediglich zur Objektauswahl benutzt, wobei die Funktionselemente zur Objektmanipulation oft von der Liste getrennt sind. Alternativ wird bei Mausbedienung beispielsweise mit einem Rechtsklick ein Kontextmenü geöffnet, dass auf den Objekttyp zugeschnittene Aktionen zulässt. Da Eingaben dieser Art mit den hier verwendeten Schnittstelle



Abbildung 2.20: Eintauchen durch Annäherung

nicht möglich sind, wird die Annäherungsinformation dafür benutzt. Wird eine gewisse Distanz unterschritten, so wird um das unter dem Finger liegende Objekt ein Rahmen angezeigt, in dem Funktionstasten angefügt sind. Zusätzlich zeigt ein Symbol den Objekttypen an, was bei Bildobjekten durch eine Vorschau ersetzt werden kann.

Sollten mehr Elemente in der Liste sein als angezeigt werden können, so können diese erreicht werden indem der Finger über das gewünschte Listenende gehalten wird. Die verdeckten Elemente werden dann hereingeschoben bzw. die Liste vom Finger weg verschoben. Dies geschieht zunächst langsam, je näher der Finger jedoch dem Listenende kommt, desto schneller wird diese bewegt. Diese Scrollingzonen erstrecken sich über ca. ein Viertel der Gesamtlistenlänge.

Wie oben im Abschnitt zur Multimodalität erwähnt, kann auch diese Liste teilweise durch Sprachkommandos bedient werden. Das „Point And Talk“-Verfahren wird so eingesetzt, dass das unter dem Benutzerfinger liegende und damit auch größte Element mit einem Sprachbefehl aktiviert bzw. ausgewählt werden kann. Zusätzlich können die mit dem Rahmen angezeigten Aktionen auch schon bei größerer Entfernung (also noch ohne den Rahmen) durch Sprache ausgelöst werden. Dazu müssen allerdings die möglichen Befehle dem Benutzer bekannt sein.

Die Liste verfügt über insgesamt 3 Zustände:

- Ausgangszustand: alle Listenelemente sind gleich groß
- Vergrößert (FishEye): markierte Elemente sind vergrößert
- Vergrößert mit Rahmen: am größten Element sind Rahmen und Buttons angebracht

Obige Zustände werden lediglich durch die Annäherung ausgelöst. Wird eine solche Liste mit einem annäherungssensitiven Menü kombiniert, so kann mit einer einzigen Bewegung sehr tief in die Menühierarchie eingetaucht werden. In einem herkömmlichen Menü würde die Auswahl eines Kontakts und der Anruf aus der Hauptmenüebene ca. drei Menüsprünge bzw. Auswahl- und Bestätigungs vorgänge benötigen. Bei der Annäherungsvariante geschieht dies mit einer einzigen Bewegung und einer Be rührung des Anrufbuttons.

Zusätzlich zum haptischen Feedback der Bildschirmberührung bei der Buttonaktivierung wird bei erfolgreicher Auslösung einer Aktion ein sogenannter „Splash Screen“ angezeigt. Dieser ist sämtlichen Menüobjekten überlagert, wird am Bildschirm zentriert ausgerichtet und verschwindet automatisch nach mehreren Sekunden. Dadurch

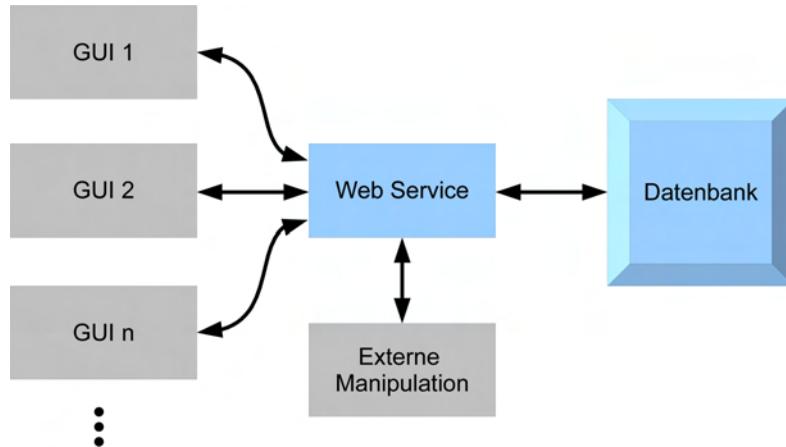


Abbildung 2.21: Datenhaltung für die FIS

wird ein optisches Feedback der erkannten Aktion gegeben. Der Splash Screen gibt in Textform die erkannte Aktion aus.

## 2.7 Datenhaltung

Mit dem Begriff Datenhaltung ist in diesem Zusammenhang die benutzte Datenbasis und deren Aufbau gemeint. Um das Gesamtsystem möglichst modular und flexibel zu gestalten, werden die in der graphischen Oberfläche angezeigten Daten soweit wie möglich in einer zentralen Datenbank gespeichert und verwaltet. Dies ermöglicht die Verwendung von verschiedenen Benutzerschnittstellen mit der gleichen zugrundeliegenden Datenbasis. Durch diese Aufteilung ist es auch möglich, mit einem weiteren Programm während der Laufzeit Daten zu verändern, beispielsweise um während der Fahrt Meldungen des Fahrzeugs zu simulieren oder Verkehrs- und Wetterdaten einzuspeisen. Auch können mehrere FIS gleichzeitig auf das Datensystem zugreifen. Die von der Datenbank zur Verfügung gestellten Funktionalitäten und Informationen haben entscheidenden Einfluss auf die von der GUI bereitgestellten Funktionalität. Als Anforderungen wurden folgende Kriterien erarbeitet, angelehnt an [8]:

- Zugriff und Veränderbarkeit der Daten zur Laufzeit
- Bereitstellung von komplexen Datentypen
- Suchfunktionen
- Auswertungen (meistgenutzte/zuletzt genutzte Objekte)
- Benutzung von Standard-Software und -Schnittstellen

### 2.7.1 Adaptivität

Durch diese Voraussetzungen ist es vor allem auch möglich, verschiedene Aspekte der GUI adaptiv zu gestalten. Dabei kann die Anpassung aufgrund von äußeren Einflüssen wie Tageszeit, Verkehrsdichte etc. oder aufgrund des Nutzerverhaltens erfolgen. Es können folgende Arten von Adaptivität unterschieden werden:

- Funktionsadaptivität: Schneller Zugriff auf häufig genutzte Funktionen
- Datenadaptivität: Filterung / Anzeige von selektierten Daten
- Kontextsensitivität: Anpassung von Menüeigenschaften durch äußere Einflüsse / Menüzustand

Adaptivität soll dem Nutzer helfen, ein System schneller oder mit weniger Aufwand zu bedienen. Das kann beispielsweise durch eine Reduktion der angezeigten Funktionen oder der Daten erfolgen, was einer Verringerung der Auswahlmöglichkeiten entspricht. Dadurch steigt die Übersichtlichkeit und ein Objekt kann schneller gefunden werden. Erfolgt die Adaption jedoch in zu hohem Maße, so kann sie sich auch negativ auswirken weil sich der Nutzer durch die Veränderung schlechter in dem gewohnten System zurechtfindet. Ein Beispiel hierfür sind die von Microsoft eingeführten „angepassten Menüs“, bei denen selten genutzte Funktionen ausgeblendet werden. Benötigt man jedoch eine dieser Funktionen, so müssen diese erst wieder sichtbar gemacht werden und eventuell mehrere Menüs durchsucht werden. Die Adaption hat in diesem Fall also eine negative Auswirkung.

Eine andere Implementierung solcher Adaptivität wäre die Anzeige von bevorzugten Funktionen an exponierter Stelle, im Gegensatz zur bloßen Ausblendung in bekannten Menüs. Durch das Datenbanksystem kann beispielsweise der Navigationsweg eines Nutzers durch das System aufgezeichnet werden. Wird ein bestimmter Weg immer wieder durchlaufen, so kann dieser durch Vorschläge der Art „Was möchten Sie als nächstes tun?“ verkürzt werden. Die Anzahl der Vorschläge sollte dabei allerdings sehr niedrig bleiben, um nicht den gegenteiligen Effekt zu erzeugen. Eine mögliche Implementierung könnte eine annäherungssensitive Schaltfläche sein, die im Ausgangszustand wenig Platz benötigt und bei Annäherung 2-3 Aktionen vorschlägt. Da dies eine zusätzliche Option zu den bekannten Menüs darstellt, wird die Orientierung weiter gewährleistet während für den Nutzer die veränderlichen Optionen klar zu erkennen sind.

Bei der Datenadaptivität können bevorzugt benutzte Daten wie z.B. Musikstücke oder Kontaktinträge bevorzugt angezeigt werden. Ebenso wie bei der oben beschriebenen Funktionsadaptivität sollten die automatisch selektierten Datenobjekte in einem gesonderten Bereich dargestellt werden. Der Benutzer weiß also, dass dies nur ein Auszug aus allen möglichen Daten ist. Ein möglicher Anwendungsfall wäre die Anzeige der drei letztbenutzten Radiosender und MP3-Musikstücke beim Überfahren des Musikmenüs mit dem Finger. Eine weitere Möglichkeit wäre, beim Anlegen eines Termins im Kalender die zuletzt benutzten Kontakte in der Kontaktliste optisch abgetrennt ganz oben anzugeben. Die gleichen Kontakte sollten dabei in der alphabetisch sortierten Liste weiter unten noch einmal vorkommen.

Um die Akzeptanz dieser automatisierten Vorschläge zu optimieren gilt es, dem Nutzerverständnis möglichst ähnliche Gewichtungsfunktionen zu erstellen. Diese müssen dann unter anderem folgenden Größen verbinden:

- Anzahl der Nutzungen
- Zeitspanne seit der letzten Nutzung
- Art des Objekts

Diese Attribute erlauben es, dass ein vor mehreren Wochen sehr oft gespieltes Musikstück in der Liste von dem zuletzt gespielten verdrängt wird oder eine Fahrzeumeldung Vorrang vor einem Verkehrshinweis hat.

Die Kontextsensitivität kann sowohl Einfluss auf globale Systemeigenschaften haben als auch obige Adoptionsfunktionen beeinflussen. Ein Beispiel für die Anpassung globaler Attribute kann die Farbgebung und Helligkeitsanpassung an die Umgebungs-Helligkeit bzw. die Tageszeit sein. Als weitere Größe kann die Fahrzeuggeschwindigkeit herangezogen werden. Es kann dann ab einer bestimmten Geschwindigkeit die Option zur Texteingabe deaktiviert werden oder gewisse Menüpunkte vollständig ausgeblendet werden, die eine zu hohe Konzentration erfordern.

### 2.7.2 Suchfunktion und TAGs

Neben der oben beschriebenen Adaptivität unterstützt der Datenservice auch die effiziente Suche und Filterung von Daten. Dies gewinnt vor allem durch die steigende Funktionsanzahl und die Zunahme sowohl der Datenmenge als auch der unterschiedlichen Datentypen. Bekannte Suchfunktionen arbeiten meist nach dem Prinzip einer Suche durch Objektnamen, wobei der Suchbegriff exakt vorkommen muss um ein Ergebnis zu erhalten. Um auch den Inhalt von Objekten zu durchsuchen, werden oft Erweiterungen oder spezielle Programme benötigt. Solche Einschränkungen sind jedoch im Fahrzeug nicht akzeptabel, weshalb den Such- und Filterfunktionen wesentlich mehr den Nutzer unterstützende Funktionalität gegeben werden muss.

Folgende Funktionen werden eingesetzt:

- Unscharfe (Fuzzy) Suche
- Suchvorschläge, „Search as you type“
- Systemübergreifende Volltextsuche
- Vergabe und Filterung nach Marken (TAGs)

Oben genannte Hilfestellungen greifen zu einem großen Teil ineinander bei einem Suchvorgang. Schon während der Eingabe eines Suchbegriffs wird nach jedem Zeichen eine Datenbankanfrage durchgeführt. Diese liefert dann Begriffe, die mit den bis dahin eingegebenen Zeichen beginnen („Search as you type“). Eine solche Vorgehensweise ist auch bei dem Firefox Internetbrowser bei der Textsuche innerhalb einer Internetseite zu sehen. So muss der Nutzer einerseits im Idealfall nicht den gesamten Begriff eingeben und kann andererseits überprüfen, ob dieser überhaupt in der Datenbank vorkommt. Bei einem Tippfehler liefert diese Suchanfrage keine Ergebnisse, da eine unscharfe Suche sowohl länger dauern und meist zu viele Vorschläge ergeben würde.

Ist der Begriff dann eingegeben, so kann die Suche gestartet werden. Diese wird von Anfang an unscharf durchgeführt, da auf diese Weise die gesuchten Objekte als erste ausgegeben werden und zusätzlich noch weitere Vorschläge. Bei der unscharfen Suche wird im Datenbanksystem die Levenshtein-Distanz [10] berechnet. Die Levenshtein-Distanz ist ein Maß für die Anzahl an benötigten Ersetzungen, Löschungen und Einfügungen um eine Zeichenfolge in eine andere zu überführen. Je kleiner diese Distanz also ist, desto größer ist die Übereinstimmung. Eine mit Tippfehlern behaftete Eingabe kann damit also gut verarbeitet werden.

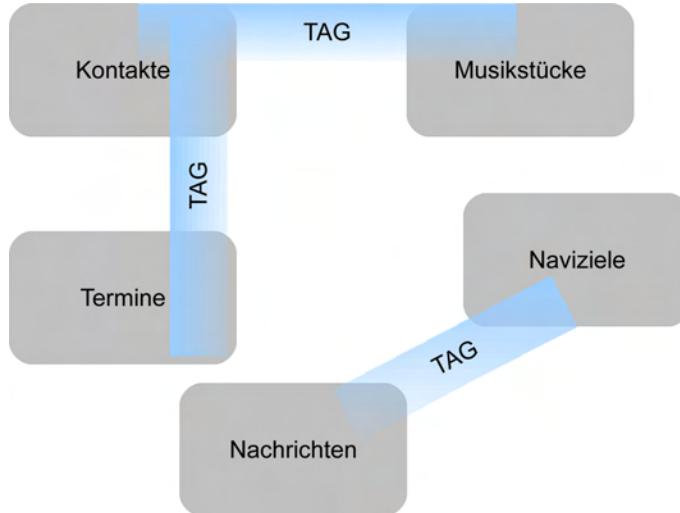


Abbildung 2.22: TAGs verbinden unterschiedliche Datentypen

Das TAG-Konzept hält immer mehr Einzug in Systeme mit heterogenen Datenstrukturen oder sehr unterschiedlichen Inhalten. Ein TAG ist wie eine Marke zu verstehen, die ein Objekt einer gewissen Menge oder Gruppe zuordnet. So ist es möglich, Objekte verschiedenster Typen zusammenzufassen oder die Marke als Filterkriterium zu verwenden (Abbildung 2.22). Ebenso können einem Objekt mehrere TAGs zugewiesen werden. Ein Beispiel für die TAG-Vergabe wäre, sowohl einem Termin als auch einem Kontakt den TAG „MMK“ zu geben, da beide dem Lehrstuhl für Mensch-Maschine-Kommunikation zugeordnet sind. Angenommen der Nutzer ist nicht mehr sicher, wann und ob er einen Termin beim MMK-Lehrstuhl hat, gibt er bei der Suche „MMK“ ein, findet unter anderem ein TAG mit dieser Bezeichnung und führt eine neue Suche über dieses TAG aus. Dabei findet er sowohl den Kontakt als auch den Termin. Durch eine Berührung dieses Terms wird er auf den Kalender weitergeleitet, wo dieser eingetragen ist. Das TAG-Konzept ermöglicht auch Suchvorgänge, die mit den meisten gängigen Methoden nicht durchführbar wären. Als Beispiel soll ein Geschäftskontakt dienen, der bei der Erstellung mit dem TAG „BMW“ versehen wurde. Da der Name allerdings sehr ungewöhnlich und schwierig zu merken war, führen Suchvorgänge selbst mit der unscharfen Suche zu keinem Ergebnis. Jedoch ist noch bekannt, dass der Kontakt zu BMW gehört, deshalb wird eine Suche nach Objekten mit diesem TAG gestartet. Beim Lesen des Kontakts kehrt die Erinnerung zurück und er kann angerufen werden.

Eine Schwachstelle des TAG-Konzepts ist, dass es nur dann sinnvoll genutzt werden kann wenn die TAGs vom Nutzer auch vergeben werden. Ist dies nicht der Fall, so muss zu konventionellen Suchmethoden gegriffen werden. Diese Problematik könnte teilweise umgangen werden indem im GUI die TAGs als elementarer Bestandteil (optisch) eingebettet werden und der Nutzer beim Anlegen eines Objekts explizit nach einer TAG-Vergabe gefragt wird. Wurde schon eine gewisse Menge an TAGs angelegt, so muss der Nutzer bei der Zuweisung lediglich aus einer Liste auswählen, ohne erneut Text eingeben zu müssen.

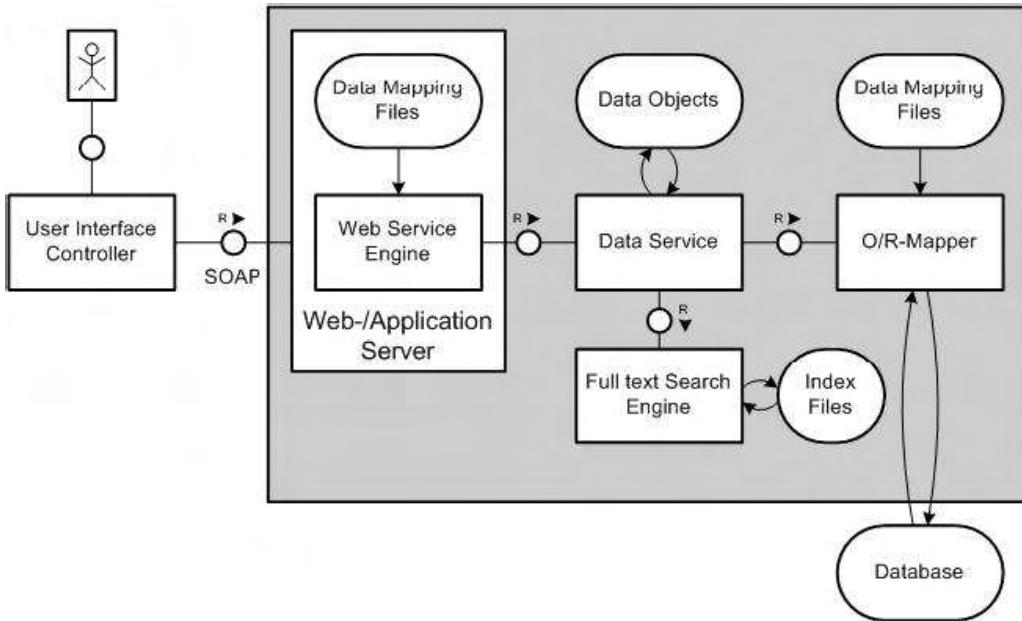


Abbildung 2.23: Innerer Aufbau des Datenmanagementsystems, aus [8]

### 2.7.3 Datenmanagementsystem

Das Datensubsystem wird über einen Web Service[11] angebunden, der mit dem Simple Object Access Protocol<sup>5</sup> arbeitet. Dieser Service wird von einem Apache Webserver mit Axis-Erweiterung bereitgestellt und kann theoretisch von Java, .Net, diversen anderen Programmiersprachen und Flash verwendet werden. So kann eine hohe Kompatibilität und Wiederverwendbarkeit erreicht werden. Als Datenbank dient für das System MySQL. Sämtliche Komponenten sind frei verfügbar und wurden auf mehrere Betriebssysteme portiert. Bei SOAP werden komplexe Datentypen bzw. Objekte mit XML<sup>6</sup> serialisiert. Das heißt, dass ein Objekt, das vom Service an die GUI geschickt wird zuerst in seine Unterobjekte und primitiven Datentypen zerlegt und in einem XML-Datenstrom beschrieben wird. Nach dem Empfang wird dieses Objekt auf der GUI-Seite wieder zusammengesetzt. Die bereitgestellten Datentypen und Funktionen werden der GUI vom Service in einer WSDL<sup>7</sup>-Datei zur Verfügung gestellt. Darin werden sämtliche verfügbaren Funktionen und Datentypen definiert.

Die gesamte Kommunikation zwischen Flash und dem Web Service erfolgt nach dem „Request - Response“-Prinzip. Das heißt, dass der Client (Flash) eine Anfrage sendet und der Service eine Antwort darauf schickt, in die umgekehrte Richtung ist keine Kommunikation vorgesehen. Es können also keine Events vom Datenservice initiiert werden, die GUI muss also zyklisch den Zustand von Objekten überwachen um eine Änderung zu registrieren. Der in Abbildung 2.24 beschriebene Vorgang setzt sich wie folgt zusammen:

- Nutzer gibt Daten eines neuen Kontakts in ein Formular ein

<sup>5</sup>SOAP

<sup>6</sup>eXtended Markup Language

<sup>7</sup>Web Service Description Language

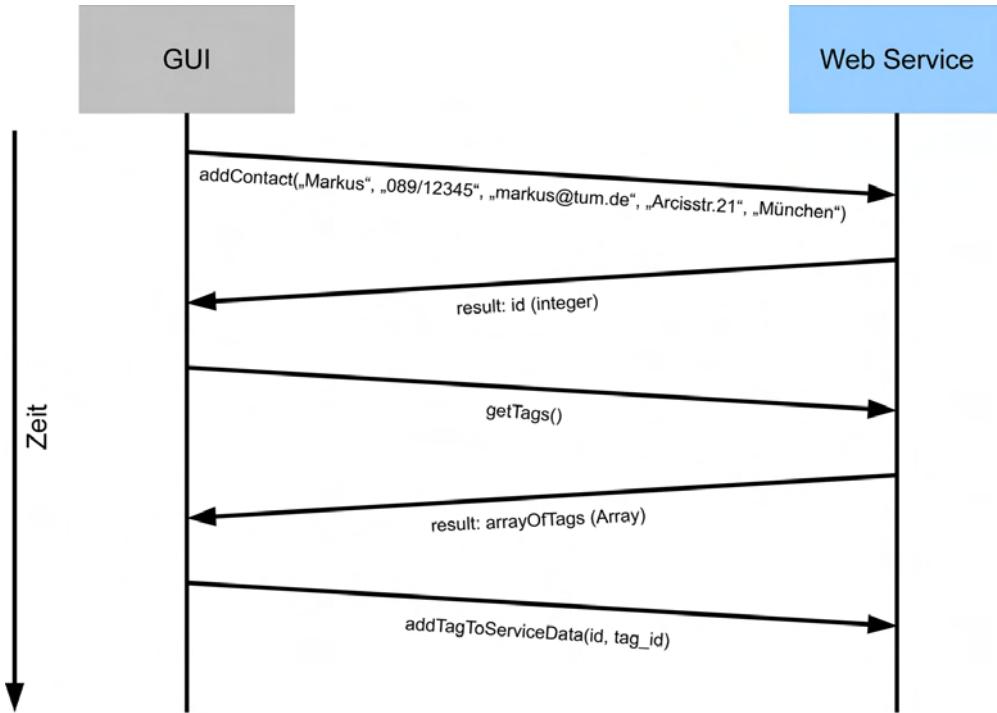


Abbildung 2.24: Erstellung eines Kontakts und Zuweisung eines TAGs

- Anlegen eines Kontakts mit `addContact`, als Parameter dienen die Daten aus dem Formular
- Rückgabe der automatisch zugewiesenen ID
- Anfrage für die Liste der vorhandenen TAGs, um den Nutzer eine Auswahlmöglichkeit zu geben
- Rückgabe eines Arrays von TAGs
- Nutzer wählt ein TAG aus der Liste
- Zuweisung wird mit `addTagToServiceData` dem Datenbanksystem übergeben

Die Kommunikation zwischen GUI und Datenservice verläuft asynchron. Das bedeutet, dass die GUI nicht mit festen Antwortzeiten rechnen kann und daher eventbasiert gearbeitet werden muss. Dabei wird zu jeder Anfrage ein Objekt generiert, dessen „onResult“-Methode bei der zugehörigen Antwort aufgerufen wird. Vor allem bei Funktionen, deren Rückgabe von der GUI zur Anzeige benötigt werden muss dieses unsichere Timingverhalten berücksichtigt werden.

Sämtliche Objekte werden in einer gemeinsamen SQL-Datenbank gespeichert, die prinzipiell keine Datentypen wie bei gängigen Programmiersprachen kennt. Deshalb wird aus Organisationszwecken wie bei obigem Vorgang mit IDs und Typbezeichner in Textform gearbeitet. Jedes Objekt bekommt eine individuelle ID bei der Erstellung zugeordnet, die automatisch vom Datenservice erstellt wird. Ebenso wird der Typbezeichner „Discriminator“ gesetzt, der eine Erkennung des Objekttyps zulässt. Da bei einer Suchanfrage beispielsweise ein Array vom Typ „ServiceData“ geliefert wird, ist der Discriminator die einzige Identifikationsmöglichkeit. In Abbildung 2.25

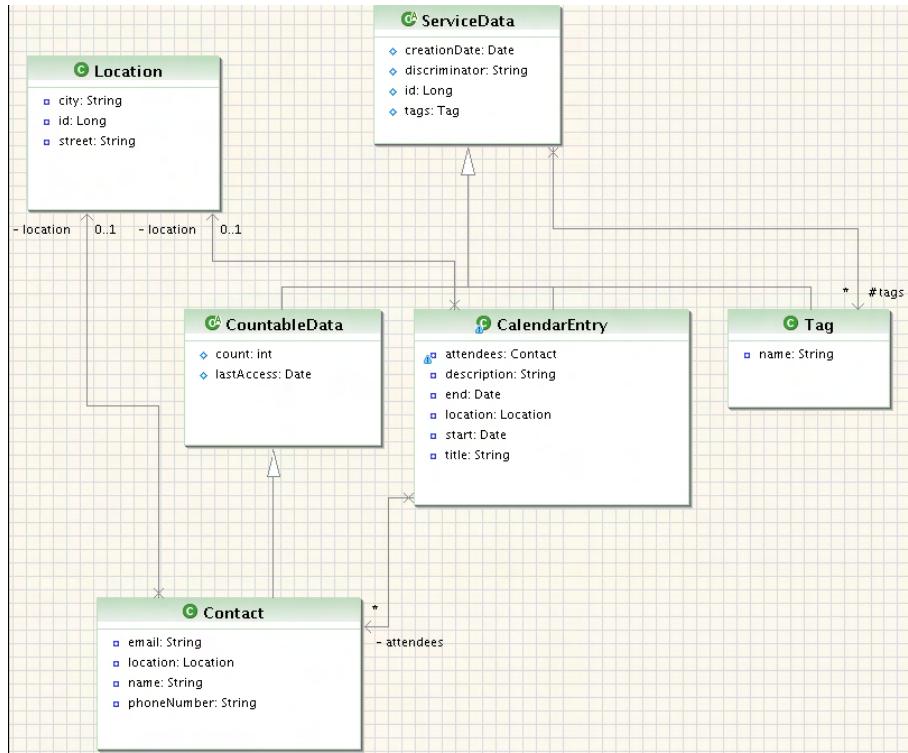


Abbildung 2.25: Datenstruktur des Datenmanagementsystems (Ausschnitt), aus [8]

ist ein Ausschnitt der Verbindungen zwischen den Datentypen zu sehen. Grundsätzlich sind sämtliche verwendeten Typen in einer Hierarchie aufgebaut, einerseits um den Programmieraufwand zu begrenzen und andererseits um in sich konsistente Datenstrukturen zu erstellen. Der Basisdatentyp für die meisten Objekte heißt „ServiceData“ und beinhaltet bereits das Erstellungsdatum, eine ID, den Discriminator und TAGs. Folglich besitzen sämtliche abgeleiteten Datentypen die gleichen Attribute.

# 3. Implementierung

Bei der Umsetzung der obigen theoretischen Vorüberlegungen kann grob in Hard- und Softwarearchitektur unterschieden werden. Dabei soll zunächst der realisierte physikalische Aufbau beschrieben und darauffolgend die Software erläutert werden. Der aus den vorangegangenen Überlegungen erstellte Demonstrator besteht aus folgenden Komponenten:

- PC mit Flachbildschirm und darauf montierten Touch Sensor und Mikrofon
- Flash GUI eingebettet in MDM Zinc
- Spracherkennung angebunden über Socket
- Externe Datenbank angebunden über Web Service

Im Folgenden werden diese Teilsysteme näher erläutert.

## 3.1 Hardware

Die verwendete Hardware besteht aus zwei zeitgemäßen Desktop PCs, die mit einer 100MBit/s Netzwerkverbindung ausgestattet sind, einem Standard 15 Zoll Flachbildschirm und dem darauf montierten Touchsensor von IQ Automation. Der Sensor ist baugleich mit der Projected Capacitive Serie von Elo und verwendet die Mess-ASICs<sup>1</sup> von Zytronic. Die Datenverbindung und Stromversorgung erfolgt über den seriellen Port des einen PCs, auf dem auch die graphische Oberfläche läuft. Der zweite PC dient der Bereitstellung des Web Service. Der Rechner auf dem die Oberfläche ausgeführt wird ist zusätzlich mit einem dynamischen Mikrofon verbunden das als Signalquelle für den Spracherkennung dient.

## 3.2 Softwarearchitektur

Die Software wurde in mehrere Module aufgeteilt, einerseits aus technischer Notwendigkeit und andererseits um eine Aufteilung auf mehrere Personen zu ermöglichen. Vor allem der Web Service muss vollkommen eigenständig sein, da er von mehreren anderen GUIs benutzt wird und in einer eigenen Arbeit entwickelt wurde. Zur Erstellung der GUI wird Adobe Flash verwendet, da es gute Werkzeuge für eine zügige Entwicklung optisch ansprechender Oberflächen erlaubt. Es entwickelt sich in

---

<sup>1</sup> Application Specific Integrated Circuit

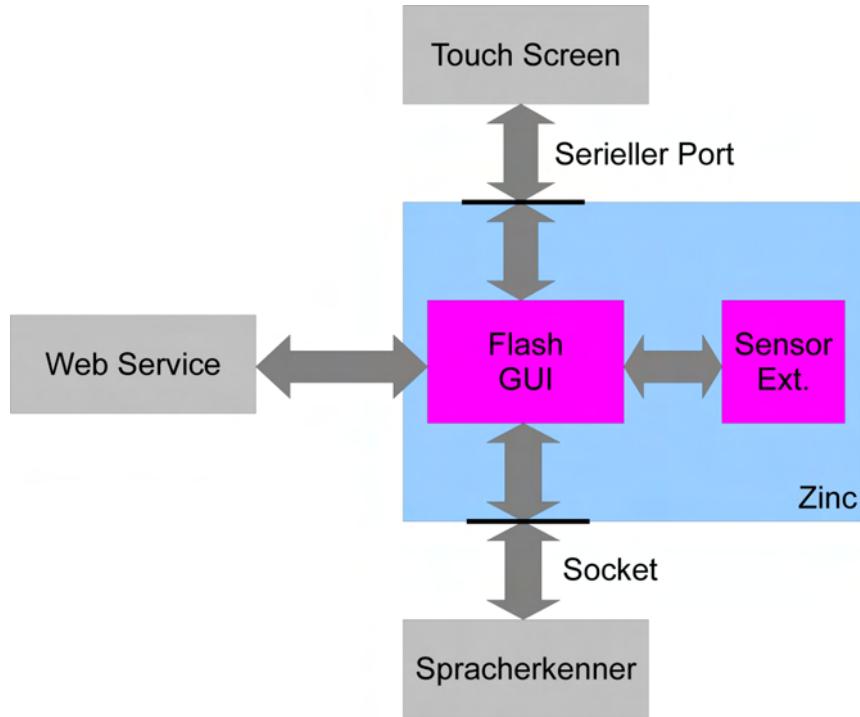


Abbildung 3.1: Architektur des Gesamtsystems

der letzten Zeit zum de-facto-Standard bei „Rich Internet Applications“, bei denen vollständige Programme innerhalb einer Internetseite ausgeführt werden.

Ein gewisses Problem stellt Flash bezüglich verschiedener Datenverbindungen zu Hardware und anderer Software dar. Da es als plattformübergreifend und als reine Web-Applikation konzipiert wurde, stehen praktisch keine standardisierten hardwarenahen Schnittstellen oder Betriebssystem-spezifische APIs<sup>2</sup> zur Verfügung. Grundsätzlich stehen nativ die in 3.1 aufgeführten Schnittstellen zur Datenkommunikation zur Verfügung.

Wie man aus Abbildung 3.1 entnehmen kann, ist die einzige verwendete Schnittstelle die Flash nativ unterstützt der Web Service. Alle anderen Datenverbindungen werden durch MDM Zinc[17] bereitgestellt. Zinc ist ein sogenannter „Shockwave To EXE Converter“, der kompilierte Flash Dateien zusammen mit einem Projektor in einer ausführbaren Datei kombiniert. Durch diese Integration ist es möglich, Flash zusätzliche Funktionen zur Verfügung zu stellen. So bietet Zinc schon in der Grundkonfiguration ca. 600 zusätzliche Befehle an, unter anderem Zugriff auf den seriellen Port und Socket-Kommunikation. Des weiteren können eigene Erweiterungen in Form von Extensions erstellt werden, wovon hier auch Gebrauch gemacht wird. Beim Kompilieren in der Flash-IDE<sup>3</sup> werden die Zinc-Befehle ignoriert, so dass auch eine Verwendung ohne die Erweiterungen nach wie vor möglich ist. Wird die erzeugte SWF-Datei jedoch in Zinc eingefügt, werden die zusätzlichen Anweisungen wie normale Flash-Befehle ausgeführt.

Der Zugriff auf den seriellen Port ist nötig, um mit dem Touch Sensor kommunizieren zu können. Nach einer minimalen Vorverarbeitung werden diese dann an die

<sup>2</sup>Application Programming Interface

<sup>3</sup>Integrated Development Environment

Technik	Vorteile	Nachteile
XML-Dateien	<ul style="list-style-type: none"> <li>• einfache Realisierung</li> <li>• standardisiertes Format</li> </ul>	kein Datenaustausch zur Laufzeit
Socket-Verbindung	<ul style="list-style-type: none"> <li>• schneller bidirektionaler Datenaustausch</li> <li>• standardisiertes Protokoll(XML)</li> </ul>	<ul style="list-style-type: none"> <li>• Komplexe Realisierung in C</li> <li>• keine Remote Procedure Calls möglich</li> </ul>
ActiveX Plugin	<ul style="list-style-type: none"> <li>• schneller bidirektionaler Datenaustausch</li> <li>• Eventerzeugung möglich</li> </ul>	<ul style="list-style-type: none"> <li>• einbettende Software nötig</li> <li>• keine Remote Procedure Calls möglich</li> <li>• umständlicher Zugriff auf Datenbank</li> </ul>
Flash Remoting	<ul style="list-style-type: none"> <li>• hohe Performance</li> <li>• gute Einbindung</li> </ul>	<ul style="list-style-type: none"> <li>• Kommerzielles Produkt</li> <li>• proprietäre Schnittstelle</li> </ul>
Web Service	<ul style="list-style-type: none"> <li>• standardisierte Schnittstelle</li> <li>• Objektaustausch möglich</li> <li>• Remote Procedure Calls möglich</li> </ul>	Overhead durch XML-Serialisierung

Tabelle 3.1: Kommunikationsmöglichkeiten mit Flash, angelehnt an [8]

eigens erstellte Extension übergeben, die daraus die Fingerposition und den Abstand errechnet. Beides wird wieder an die GUI übergeben und sowohl die Mausposition gesetzt als auch die Annäherungsinformation allen grafischen Elementen zur Verfügung gestellt.

Für die Anbindung des Spracherkenners wird eine Socket-Verbindung gewählt. Damit ist es relativ einfach möglich, die vom Spracherkennner erzeugten Zeichenfolgen mit wenig Overhead an die Hauptapplikation zu schicken.

### 3.2.1 Flash

Adobe Flash wurde ursprünglich von Macromedia als Animationsformat für Internetseiten entwickelt. Die Flash-Dateien wurden dabei mit den ersten Versionen der IDE nur mit Timeline-Animationen definiert. Eine solche Darstellung wird generell beim Erstellen und Editieren von Bewegtbildern angewandt. Grundsätzlich werden dabei die Positionen von Grafikelementen in speziellen Key Frames festgelegt und die Entwicklungsumgebung berechnet die nötigen Schritte zwischen zwei Key Frames. Diese Form der Erstellung prädestiniert die erzeugten Inhalte für eine sequentielle Wiedergabe. Es sind jedoch sehr rudimentäre Sprünge und andere nichtlineare Abläufe möglich.

Erst nachträglich wurde die Verwendung von Action Script als Programmiersprache eingeführt. Diese Sprache basiert auf ECMAScript und hat daher starke Ähnlichkeit mit Java Script bzw. JScript. Für diese Arbeit wurde Action Script 2 verwendet, welches objektorientierte Programmierung erlaubt und bereits viele komplexe Datentypen vor allem für Multimediaanwendungen bereitstellt. Aus Gründen der Abwärtskompatibilität und Flexibilität ist seit der Action Script-Einführung sowohl eine Entwicklung mit der Timeline als auch per Script möglich.

Durch die Parallelität von Timeline-Animationen und Action Script-Code ergibt sich eine Programmiertechnik, die nicht mit herkömmlichen Sprachen wie C++, Java oder Basic vergleichen lässt. Es ist unter anderem möglich, Code gewissen Key Frames zuzuweisen, den Filmablauf zu steuern und auch umgekehrt das Script von Film-Events steuern zu lassen. Zu einem Frame zugewiesener Code wird erst bei Erreichen des Frames ausgeführt und dessen Variablen sind erst ab diesem Zeitpunkt existent. Des weiteren besitzt jedes graphische Element seine eigene Timeline, wodurch sich sehr einfach Zustände des jeweiligen Objekts realisieren lassen. Zusätzlich kann jedem graphischen Objekt in der Bibliothek einer .fla-Datei eine Action Script-Klasse zugewiesen werden. Auf diese Weise können effizient Klassen von Widgets erzeugt werden, die alle das gleiche Grundverhalten aufweisen wie beispielsweise Splash Screens, Buttons oder Hauptmenüpunkte.

Die Bibliothek („Library“) enthält alle graphischen Elemente, die dynamisch während der Laufzeit erzeugt werden. Jedes Library-Objekt hat dabei den Charakter eines Prototypen, das heißt es können beliebig viele Instanzen davon erzeugt werden. Dies gilt sowohl für den graphischen Anteil als auch die verknüpfte Action Script-Klasse. Ein Beispiel hierfür sind Listeneinträge oder Tasten einer Texteingabe.

Grundsätzlich ist die GUI so konstruiert, dass möglichst wenige Objekte statisch vorhanden sind. Aus diesem Grund ist die „Stage“, also die Grundfläche des Dokuments, in der IDE leer. Die Taskleiste und Hauptmenüpunkte werden erst durch den der Stage zugewiesenen Code erzeugt was zur Folge hat, dass eine Änderung des Hauptmenüs sowohl bezüglich des Layouts als auch der Anzahl der Menüpunkte absolut

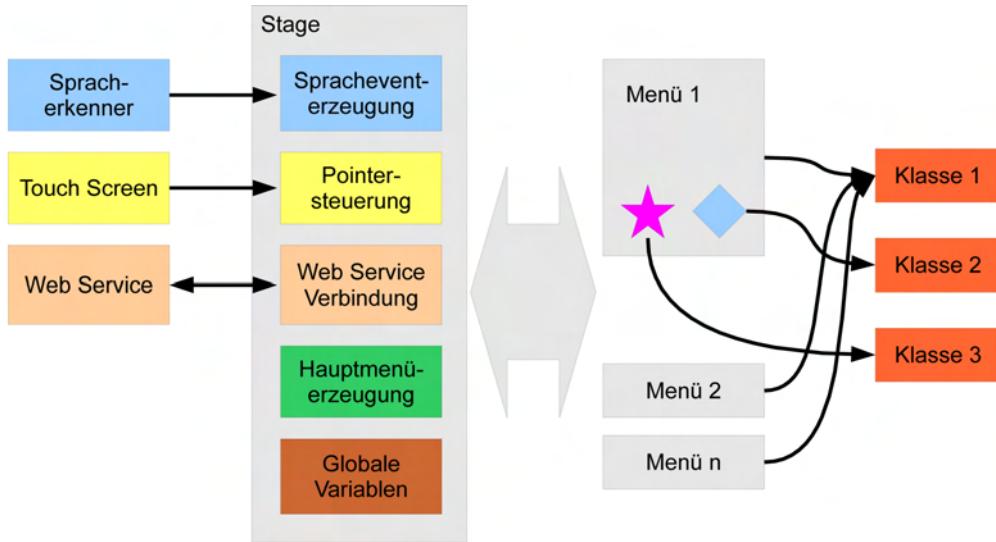


Abbildung 3.2: Architektur der Flash-GUI

flexibel ist. Der auf Stage-Ebene geschriebene Code verwaltet globale Variablen wie z.B. den Fingerabstand, die Zustände mancher Menüs oder die Verbindung zum Web Service, wie aus Abbildung 3.2 zu entnehmen ist. Auch werden an dieser Stelle systemweite Events erzeugt, die vom Spracherkennner oder dem Touch Sensor ausgelöst werden. Flash bietet für verschiedenste Vorgänge bereits vordefinierte Events an, beispielsweise Mouse-Events, Web Service-Events oder Animations-Events. Zusätzlich können noch eigene Events erzeugt werden um eine komfortable applikationsinterne Kommunikation zu erstellen, ein Beispiel hierfür sind Sprach-Events.

Die Spracheventerzeugung funktioniert dabei folgendermaßen: Zunächst wird ein Objekt erzeugt, das als Event Dispatcher fungiert. Dafür muss es zwingend folgende Methoden enthalten, um die Anforderungen des Eventsystems in Flash zu erfüllen:

- function dispatchEvent() ;
- function addEventListener() ;
- function removeEventListener() ;
- public function newEvent(befehl: String) { ... }

Ein Dispatcher ist ein Event-erzeugendes Objekt, bei dem sich mehrere Empfänger (Listener) registrieren können. Wird ein Event vom Dispatcher emittiert, so werden gleichzeitig sämtliche Listener benachrichtigt, wobei der Event noch zusätzliche Daten enthalten kann. Beim Beispiel des Sprach-Events existiert nur ein einziger Eventtyp, wobei der erkannte Text als Datenobjekt beim Event als Argument an die Listener übergeben wird. Konkret werden also sämtliche graphischen Objekte, die auf einen Sprach-Event reagieren sollen bei dem Dispatcher-Objekt registriert. Wird ein Sprach-Event ausgelöst, überprüft jeder Listener, ob er auf diesen Event hört oder nicht und reagiert entsprechend darauf.

Zusätzlich zu den Flash-internen Events können mit Hilfe von Zinc auch betriebssystemweite Events erzeugt werden, in diesem Falle speziell Maus-Events. Diese lassen

sich noch einmal in „Mouse Move“-Events und „Click“-Events unterteilen. Damit lassen sich Events emulieren, die sonst von einer Maus stammen, worauf wiederum Flash-interne Events ausgelöst werden. Die Erzeugung der Pointer-Position geschieht nach folgendem Ablauf:

- Eingehende Daten vom Touch-Sensor werden zu Datensätzen zusammengefasst
- Eventuell Initialisierung der Detektorfunktionen mit diesen Daten
- Aufruf der Calculate-Funktion mit den Sensordaten
- Parsen der zurückgelieferten Werte
- Setzen der Pointer-Position
- Eventuell Erzeugung eines Click-Events

Obiger Ablauf wird für jeden neuen Datensatz durchgeführt. Durch diese Vorgehensweise können zahlreiche bereits in Flash vorhandene Funktionen und Events für GUI-Vorgänge benutzt werden. Der Mauszeiger des Betriebssystems wird zusätzlich unsichtbar gesetzt, da er irritierend wirkt wenn er direkt unter dem Benutzerfinger angezeigt wird.

### 3.2.2 Fingerdetektion

Ein essentieller Bestandteil des Demonstrators ist der hier verwendete Touch Sensor. Er besteht wie im Kapitel „Stand der Technik“ bei den Projected Capacitive Sensoren beschrieben aus mehreren verklebten Glasscheiben, in die ein Netz aus feinsten Kupferadern eingebettet ist. Die Kupferleitungen werden entlang der Außenkante zu den Mess-ASICs weitergeführt, die auf der Scheibenrückseite verklebt sind. Durch ein einziges Kabel wird sowohl die serielle Verbindung als auch die Stromversorgung hergestellt. Die Verbindungsgeschwindigkeit 9600 Baud, worüber die 33 Byte umfassenden Datenpakete versandt werden. Für jeweils x- und y-Achse besitzt der Screen 16 Sensoren, die unabhängig voneinander arbeiten. Jedem dieser Sensoren wird im Protokoll ein Byte zugewiesen, dessen Zahlenwert dem Kapazitätswert des Sensors entspricht. Dadurch werden die ersten 32 Bytes im Datensatz durch Sensordaten belegt, das letzte Byte hat den Wert 0 und dient als Trennzeichen der einzelnen Datensätzen, die folgenden Aufbau haben:

[X0][X1]...[X15][Y0][Y1]...[Y15][NULL]

Obiger Datensatz kann wie in Abbildung 3.3 dargestellt werden. Ein solches Bild ergibt sich typischerweise bei der Annäherung mit einem Finger, mit dieser Messmethode wäre jedoch auch die Detektion mehrerer Finger gleichzeitig möglich. Der hier angewandte Algorithmus zur Berechnung der Fingerposition geht jedoch von einem einzigen Finger aus. Nähern sich mehrere Objekte gleichzeitig, so wird das Objekt mit der höchsten Sensoraktivierung erkannt. Durch Erweiterung des Algorithmus wäre also eine Multi Touch-Eingabe möglich, wie sie beim Apple iPhone angewandt wird um Zoomfunktionen zu steuern, siehe [20]. Bei jedem empfangenen Byte wird

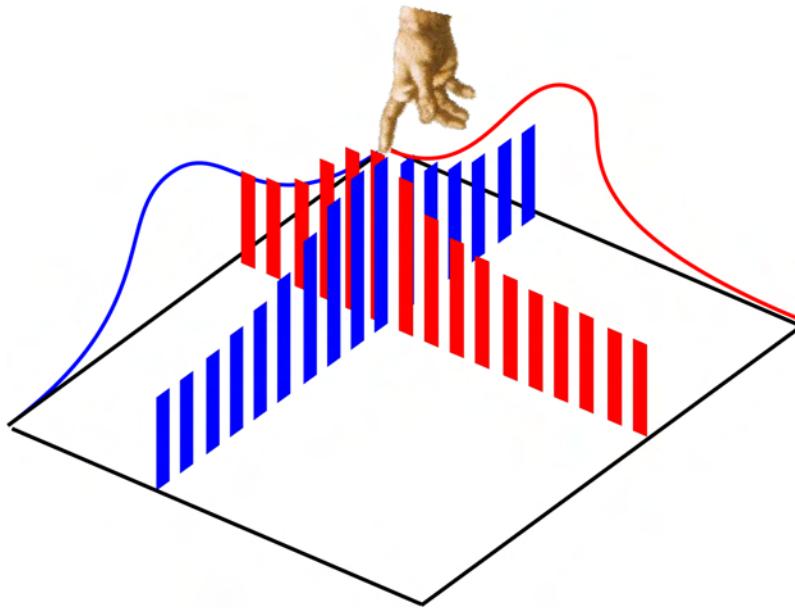


Abbildung 3.3: Visualisierung der Sensordaten

durch Zinc ein Flash-Event ausgelöst, worauf die obige Datenstruktur zusammengesetzt wird und an die *init()*-Funktion bzw. die *calculate()*-Funktion aufgerufen wird. Beide sind in einer Zinc Extension implementiert, die aus einer in C geschriebenen DLL<sup>4</sup>-Datei erzeugt wird. Im Anhang A ist der Quelltext zu dieser DLL zu finden. Die Fingerdetektion erfolgt in mehreren Stufen:

- Erstellung der Referenz-Samples (einmalig)
- Ermittlung des Offsets für den Sensor (einmalig)
- zeitliche Glättung zwischen letztem und aktuellem Datensatz
- örtliche Glättung innerhalb jeder Achse mittels gleitendem Durchschnitt
- Suchen des Maximums und Minimums für x- und y-Achse
- Berechnung des Fingerabstands aus den Maxima und Minima
- Berechnung der Korrelationsindizes
- Suchen der Korrelationsmaxima
- quadratische Interpolation um die Maxima
- Festlegung des interpolierte Maximums als Fingerposition

Mit Hilfe obiger Schritte kann aus den Sensordaten die x-, y- und z-Position ermittelt werden und zum Setzen der Pointerposition und zur Bereitstellung des Fingerabstands an verschiedene Interfaceobjekte benutzt werden. Die Eingabe per Touch Screen kann mit der Leertaste ein- und ausgeschalten werden.

<sup>4</sup>Dynamic Link Library

Beim Aktivieren des Sensors werden zunächst die momentanen Messwerte mit der *init()*-Funktion als Offset gespeichert. Damit wird die Software auf die momentanen Kapazitätswerte geeicht, worauf zahlreiche Umgebungseinflüsse wie der Abstand zum Bildschirm oder metallische Halterungen einen Einfluss haben. Der Offset wird dann bei nachfolgenden Messungen von nachfolgenden Datensätzen abgezogen um eine Anwendung der nachfolgenden Algorithmen erst zu ermöglichen.

Da die Sensorwerte permanentem Messrauschen überlagert sind, werden vor der weiteren Verwendung diverse Glättungen durchgeführt. Zunächst wird eine zeitliche Glättung bzw. Mittelung zwischen dem letzten Datensatz und dem aktuellen. Hierfür wird der alte Wert  $w_{alt}$  mit dem aktuellen  $w$  nach der Formel  $w_{neu,i} = \frac{1}{2} \cdot (w_{alt,i} + w_i)$  in  $w_{neu}$  geschrieben, was für alle 32 Werte geschieht. Diese Glättung hat den Effekt eines Tiefpasses und bewirkt eine leichte Verzögerung der Pointeraktualisierung auf eine Fingerbewegung. Danach folgt eine örtliche Glättung, die allerdings für die beiden Achsen getrennt berechnet wird. Diese Operation wird mit Hilfe des gleitenden Durchschnitts berechnet, der eine Breite von 2 hat. Wird dies nur in eine Richtung durchgeführt (z.B. Index 0 bis Index 15), so würde das Maximum in Richtung der Berechnung verschoben werden. Deshalb wird der gleiche Vorgang in die entgegengesetzte Richtung noch einmal angewandt, was zusätzlich eine weitere Glättung bewirkt und die Verschiebung teilweise ausgleichen kann. Die örtliche Glättung bewirkt keine zeitliche Verzögerung der Pointerreaktion.

Nachfolgend werden für beide Koordinatenachsen separat die Maxima und Minima gesucht. Dabei werden die Werte der jeweiligen Achse mit aufsteigenden Indizes miteinander verglichen. Ist der aktuelle Wert höher als der vorige, so ist wird der aktuelle als neues Maximum gesetzt. Die umgekehrte Vorgehensweise wird zur Minimasche angewandt. Eine solche Maximumssuche würde für eine Pointerdetektion nicht ausreichen, da lediglich eine Auflösung von 16 Werten auf jeder Achse erreicht werden könnte, entsprechend der Anzahl der Sensoren pro Achse. Für eine Berechnung des Fingerabstandes ist eine solche Information allerdings ausreichend, da die Annäherungsinformation wesentlich fehlerbehafteter ist und eine geringere Genauigkeit vom Benutzer nicht so störend wahrgenommen wird wie bei der x- und y-Achse. Als für den Abstand repräsentative Größe wird hier der Unterschied der Sensorwerte der Minima und Maxima gewählt.

Zur Abstandsberechnung wird zuerst der Mittelwert der Extremadifferenzen von beiden Achsen berechnet. Durch Versuche konnte festgestellt werden, dass dieser Mittelwert keinen linearen Zusammenhang mit dem Abstand hat, sondern einen annähernd quadratischen Verlauf hat. Als Ausgleich wird daher mit der Quadratwurzel des Mittelwerts gearbeitet, was in folgender Formel zu erkennen ist:

$$z = 60 - 10 \cdot \sqrt{\frac{1}{2} \cdot (diff_x + diff_y)}$$

mit  $diff_x, diff_y$  : Extremadifferenzen

Als nächster Schritt wird die Position des Fingers über der Bildschirmebene ermittelt. Wie oben angedeutet, genügt aus Gründen der benötigten Auflösung und der Fehlertoleranz nicht eine einfache Maximumssuche. Der maximale Signal-Rausch-Abstand (SNR<sup>5</sup>) beträgt im Maximalfall 35 dB und kann auf 3 dB sinken, was mit der Maximumssuche nicht vorhersehbare Pointersprünge zur Folge hätte. Ein wesentlich robusteres Werkzeug, das auch in der Bilderkennung angewandt wird, ist dafür die

---

<sup>5</sup>Signal to Noise Ratio

Kreuzkorrelation, die zur Klasse der nichtlinearen Filter zählt. Wie in [12] beschrieben wird, beinhalten diese Filter das Muster, das mit ihrer Hilfe detektiert werden soll. Bei der Anwendung des Filters wird dieses Muster über den zu untersuchenden Signalausschnitt bewegt und die Korrelation berechnet. Dabei entsteht für jede Position ein Korrelationsindex, der ein direktes Maß für die Ähnlichkeit des betrachteten Ausschnitts mit dem Muster ist. Bei periodischen Signalen kann dieser Index auf 1 normiert werden, bei aperiodischen wird diese Normierung ausgelassen. In dem hier vorgestellten Fall ist eine Normierung unnötig, da die Indizes nur relativ zueinander verglichen werden und daher keine absoluten eindeutigen Größen benötigt werden. Für den Signalverlauf bei der Fingerdetektion wird hier angenommen, dass der Finger einen gaußförmigen Werteverlauf erzeugt. Die der Korrelation zugrundeliegende Formel lautet:

$$korr\_index_i = \sum_{n=0}^{15} sensor_n \cdot sample_{i,n} \quad (3.1)$$

Hierbei sind:

- $i$  : Index des Vergleichssamples
- $sensor$  : Datenvektor einer Achse
- $sample$  : Datenvektor eines Referenzsamples

Obige zur Korrelationsberechnung verwendete Formel ist gegenüber der allgemein bekannten Version so verändert, dass das Vergleichsmuster nicht verschoben wird, sondern es wird bei Programmstart ein Set an Referenzsamples erzeugt, die für den Vergleich dienen. Wie oben erwähnt, werden die Werteverläufe mit einer Gaußkurve angenähert. Folglich werden die Samples ebenfalls mit dieser Kurvenform erzeugt, wobei die Sampleanzahl und damit deren Abstand vorerst die Auflösung vorgeben. In diesem Fall wurden 300 Samples (Resolution) verwendet, die für beide Achsen herangezogen werden. Die Samples werden mit folgender Formel erzeugt:

$$\begin{aligned} step &= \frac{16}{Resolution} \\ sample_{i,n} &= 30 \cdot e^{-0.2 \cdot (n - i \cdot step)^2} \\ \text{für } i &\in [0, 1, \dots, Resolution] \\ n &\in [0, 1, \dots, 16] \end{aligned} \quad (3.2)$$

In Abbildung 3.4 ist obige Formel noch einmal graphisch dargestellt. Es ist zu erkennen, wie die Glockenkurve erst an den linken Rand gelegt wird und dann für alle 16 Sensoren die Werte berechnet werden. Danach wird das Zentrum der Kurve um  $step$  nach rechts verschoben und der gleiche Vorgang wiederholt, was dann für alle Samples geschieht. Nachdem der Korrelationsindex für jedes Sample mit dem aktuellen Datensatz berechnet ist, wird innerhalb der Indizes das Maximum mit der gleichen Vorgehensweise wie bei den Sensordaten gesucht. Wird nur eine Auflösung von 300 Punkten benötigt, so ist mit diesem Schritt die Fingerdetektion beendet. Als Position kann also der Sampleindex benutzt werden.

Die Sampleanzahl und damit die Auflösung wurden experimentell bestimmt. Bei geringerer Sampleanzahl ist die Auflösung geringer, jedoch unterscheiden sich die Korrelationsergebnisse stärker was zu einer eindeutigeren Maximumszuweisung führt.

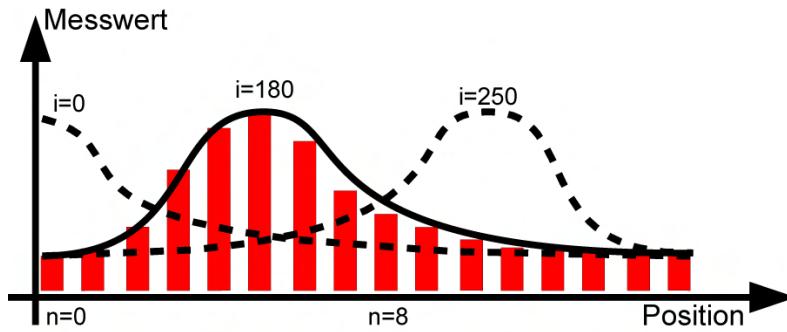


Abbildung 3.4: Visualisierung der Sampleerzeugung

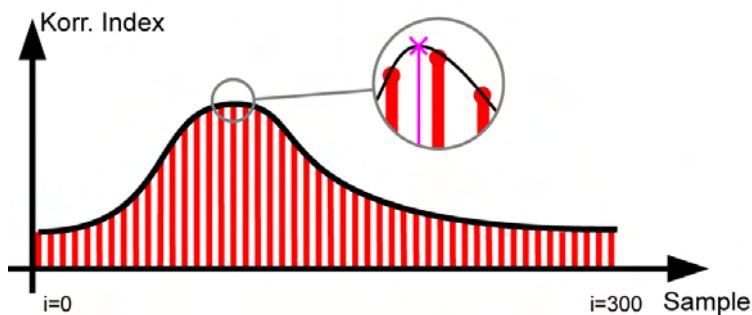


Abbildung 3.5: Interpolation zwischen Korrelationsindizes

Bei größeren Samplezahlen tritt das Gegenteil ein, wobei sich die Korrelationsindizes so wenig unterscheiden können, dass durch Messungenauigkeiten ein permanentes Springen zwischen mehreren Samples auftritt. Dies ist vor allem der Fall, wenn der Finger noch in größerer Entfernung und daher das SNR niedriger ist. Eine gute Positionsdetektion ist mit dem vorhandenen System bis zu 2cm Abstand möglich.

Um die Auflösung noch weiter zu verfeinern, wird in dieser Implementierung noch eine quadratische Interpolation hinzugefügt. Für diese Berechnung werden drei Punkte benötigt, durch die dann eine Parabel gelegt wird, was in Abbildung 3.5 skizziert wird. Das Maximum der Parabel ist dann der neue Positionswert, der auch zwischen den Samples liegen kann. Die drei Punkte werden so gewählt, dass der Maximumsindex  $\max$  der mittlere ( $x_1$ ) und die anderen beiden ( $x_0$  und  $x_2$ ) zwei Samples entfernt sind. Dadurch sind die Punkte ausreichend weit voneinander entfernt, so dass die Parabel nicht zu flach wird und damit das Maximum weniger springt.

$$\begin{aligned}
 x_0 &= \max - 2 & x_1 &= \max & x_2 &= \max + 2 \\
 y_0 &= \text{korr\_index}[x_0] & y_1 &= \text{korr\_index}[x_1] & y_2 &= \text{korr\_index}[x_2] \\
 a &= \frac{y_0 - 2 \cdot y_1 + y_2}{8} \\
 b &= \frac{y_2 - y_0}{2} - 2 \cdot x_1 \cdot a \\
 x &= \frac{-b}{2 \cdot a}
 \end{aligned}$$

Das Ergebnis  $x$  ist also der neue interpolierte Positionswert. Obige Berechnung wird für beide Sensorachsen gleich durchgeführt. Sämtliche Koordinaten werden auf eine

feste Anzahl an Dezimalstellen getrimmt und in einen String geschrieben, der wieder zurück an Flash übergeben wird. Auf der Flash-Seite wird der String dann umgekehrt wieder in einzelne Zahlenvariablen verteilt.

### 3.2.3 Pointersteuerung

Bevor damit die Mausposition gesetzt wird, müssen noch Maßnahmen zur Erhöhung der Robustheit angewandt werden. Sie lassen sich in folgende Schritte einteilen:

- Speicherung alter Pointerpositionen
- abstandsabhängige Glättung
- Berechnung der summierten Positionsunterschiede
- bedingtes Setzen der neuen Position
- Klickerzeugung

Zunächst werden für die spätere Glättung neben der aktuellen Position noch die zwei vorherigen gespeichert. Sie sollen nachfolgend mit  $x_0$ ,  $x_1$  und  $x_2$  bezeichnet werden, wobei  $x_0$  der aktuell berechneten x-Koordinate entspricht. Das Vorgehen ist für die y-Koordinate identisch. Damit der Tatsache Rechnung getragen wird, dass der SNR mit steigendem Fingerabstand abnimmt, wird die Glättungstiefe vom Abstand abhängig gemacht. Je größer der ermittelte Abstand ist, desto mehr alte Positionen werden zur Glättung herangezogen. Konkret wird folgender Algorithmus benutzt:

- $z \in [12, 19]$   
 $\Rightarrow pos = x_0 \cdot 2, 75 - 10$
- $z \in [20, 29]$   
 $\Rightarrow pos = \frac{x_0 + x_1}{2} \cdot 2, 75 - 10$
- $z \in [30, 39]$   
 $\Rightarrow pos = \frac{x_0 + x_1 + x_2}{3} \cdot 2, 75 - 10$

Nachfolgend wird jeweils die Differenz von aktueller und letzter Pointerposition für x- und y-Achse berechnet und addiert. Falls diese Summe größer als 2 und kleiner als 20 ist und der Abstand zwischen 20 und 40 liegt, wird der Pointer an diese Position gesetzt. Der Bereich für z zwischen 12 und 19 wird für die Klickerzeugung benutzt. Bei der Erzeugung eines Klickevents ist der Verlauf der z-Koordinate entscheidend. Dafür wird zunächst eine Grenze bei  $z = 16$  und eine Hysteresebreite von  $b = 1$  festgelegt. Sobald die Grenze abzüglich der Hysterese unterschritten wird, ist die Klickerzeugung „scharfgestellt“. Wird dann die Grenze zuzüglich der Hysterese wieder überschritten, wird ein Klickevent an der aktuellen Pointerposition ausgelöst, wie in Abbildung 3.6 demonstriert wird. Durch die Hysterese wird verhindert, dass viele falsche Events ausgelöst werden, wenn der Finger sich genau an der Grenze befindet und bedingt durch Sensorrauschen einmal über und einmal unter der Grenze liegt.

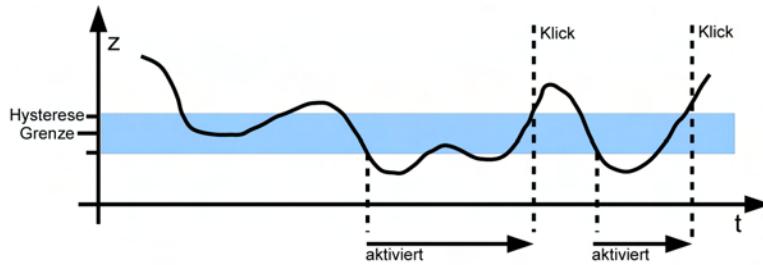


Abbildung 3.6: Erzeugung von Klickevents gesteuert durch z-Bewegung

### 3.2.4 Spracherkenner

Wie weiter oben schon erwähnt basiert der Spracherkenner auf der Engine von ScanSoft VoCon 3200. Um den Entwicklungsaufwand niedrig zu halten, wurde das „Calculator EM“-Beispielprogramm als Grundlage benutzt, da es schon zahlreiche der benötigten Funktionen aufweist. So kann bereits zwischen verschiedenen Sprachen ausgewählt werden und es besteht ein Ausgabefenster für die erkannten Begriffe. Die wichtigste Eigenschaft ist jedoch, dass der Erkennungsvorgang permanent aktiviert ist und damit kein „Push to talk“-Knopf benötigt wird.

Die Verwendung eines solchen Knopfes verbietet sich aus mehreren Gründen. Zunächst sollte kein unnötiger mechanische Knopf benutzt werden und eine Implementierung als Softkey in der GUI ist ebenfalls nicht vorteilhaft. Vor allem die Bedienung des multimodalen „Point And Talk“-Vorgangs würde dadurch wesentlich erschwert werden und eventuell sämtliche Vorteile einbüßen. Auch wird der Bedienvorgang nicht durch eine weitere Knopfbetätigung unterbrochen, was eine noch bessere Integration von Sprach- und Berührungseingabe ermöglicht.

Um die benötigte Grammatik in die Software einzubringen, wird die bereits vorgegebene ersetzt. Sie wird dann dynamisch zur Laufzeit geladen und legt fest, welche Bildschirmausgabe den erkannten Befehlen zugeordnet wird. Durch die permanente Erkennung eingehender Sprache kann es vorkommen, dass fälschlicherweise im Rendefluss Begriffe aus der Grammatik erkannt werden. Um diese falschen Ergebnisse nicht zu werten, wird der „Confidence“-Wert der Engine ausgewertet, der von 1 bis 10000 reicht. Je höher dieser Wert liegt, desto sicherer wurde der richtige Ausdruck erkannt. Versuche haben gezeigt, dass ab einer Grenze von 5000 ein gutes Ergebnis vorliegt.

Hat eine Spracheingabe dann die Confidence-Grenze überschritten, wird der Befehl einerseits auf dem Bildschirm ausgegeben und andererseits über eine Socket-Verbindung an die GUI geschickt. Sobald diese dann einen neuen Sprachbefehl erhält, wird ein Sprachevent ausgelöst und an die registrierten Event-Listener verteilt.

# 4. Ausblick

Im folgenden Abschnitt soll ein Überblick gegeben werden zu möglichen Erweiterungen oder Verbesserungen vorhandener Funktionen. Die Implementierung dieser Optionen hätte den zeitlichen Rahmen der Arbeit bei weitem übertroffen, die Ideen sollen jedoch hier vorgestellt werden.

## 4.1 Sprachfeedback

Für die Bedienbarkeit und damit die Akzeptanz eines FIS ist die Möglichkeit zur Blindbedienung entscheidend. Dabei ist die Spracheingabe ein wichtiges Element, doch um das Konzept zu vervollständigen muss auch Sprachfeedback erzeugt werden. Mittlerweile sind mehrere Systeme erhältlich, mit denen eine Synthese beliebiger Begriffe möglich ist. Diesen Systemen wird also die gewünschte Ausgabe als Zeichenfolge übergeben und diese wird dann abhängig von der gewählten Sprache in eine akustische Ausgabe gewandelt. Es kann also eine Kommunikation mit dem System stattfinden, die für die meisten Aktionen weder Sichtkontakt noch Abwendung von Lenkrad und Schalthebel erfordert. Die einzige Voraussetzung ist dabei, dass der Nutzer sich wenigstens ein grobes mentales Modell des Systems gebildet hat, da eine vollständige Zustandsbeschreibung durch Sprache sehr langsam und aufwendig ist. So sollten also nur die prägnantesten Informationen gegeben werden, die genügen um den Zustand zu erfassen. Ein Beispiel für zu komplexe Ausgabe wäre die Wiedergabe einer 100 Einträge umfassenden Kontaktliste. Grundlegende Informationen, die durch das Sprachsystem ausgegeben werden sollten beinhalten:

- aktuelles Menü beim Menüwechsel bzw. bei Aufforderung durch Spracheingabe „Aktuelles Menü?“ ⇒ „Menü Navigation“
- Zielführung bei der Navigation
- aktuell markierter Listeneintrag, auch bei darüber schwebendem Finger
- eingegebener Text bei Toucheingabe, nur ganze Wörter bzw. aktuelles Zeichen Eingabe: „Stephan Sch.“ ⇒ Feedback: „Stephan“
- Kontexthilfe
- Systemmeldungen: „Eingehender Anruf Stephan Schneider“, „Tank fast leer“

Der erste Punkt dient der Orientierung im System, wobei das Feedback durch Menüaktionen, die durch Touch oder Spracheingabe initiiert wurden, oder durch

eine explizite Frage ausgelöst werden kann. Vor allem die Auslösung der Sprachausgabe durch eine mündlich formulierte Frage wäre sehr interessant, da es eine Orientierung im System z.B. nach einer Bedienpause erlaubt, die nur geringsten Aufwand erfordert, ähnlich einem Beifahrer der helfend zur Seite steht. In Verbindung mit der Annäherungsinformation, durch die beispielsweise Hauptmenüpunkte aktiviert werden, ist auch eine „Quasi-Blindbedienung“ denkbar. Befindet sich der Finger über einem der Menüpunkte, so wird dieser einerseits in der GUI vergrößert und gleichzeitig eine Sprachausgabe mit dem Namen dieses Menüs ausgelöst. So kann der Nutzer den gewünschten Eintrag „suchen“ und dann durch den Befehl „Menü maximieren“ oder „Menü <Menüname>“ die Auswahl treffen.

Bei dem zweiten Punkt handelt es sich um eine bereits zum Standard gewordene Funktion, die hier der Vollständigkeit wegen erwähnt werden soll, da sie im vorliegenden System nicht implementiert wurde. Für die Zielführung bei der Navigation wird in praktisch jedem aktuellen System parallel zur optischen Anzeige eine Sprachausgabe angeboten, da dies die kognitive Belastung wesentlich niedriger hält als bei rein visuellem Feedback.

Auch bei der Listennavigation könnte die Sprachrückmeldung des Systems auf ähnliche Weise erfolgen wie bei den oben beschriebenen Menüaktionen. Dabei kann der Eintrag entweder direkt nach der Auswahl oder noch vorher, wenn der Finger sich noch darüber befindet, ausgegeben werden. Ähnlich wie bei den Menüs kann dann beispielsweise ein Kontakt gesucht und gewählt werden und dann mit dem Befehl „Kontakt anrufen“ ein Anruf ausgelöst werden. Des Weiteren könnte die Länge einer Liste von Suchergebnissen ausgegeben werden, nachdem der Suchbegriff entweder per Texteingabe oder Sprache eingegeben wurde.

Für die Texteingabe wäre ein Sprachfeedback in der Art denkbar, dass jedes vollständige Wort oder das zuletzt gedrückte Zeichen ausgegeben wird. Sollte die Ausgabe jedes einzelnen Zeichens als störend empfunden werden, so kann die Rückmeldung derart reduziert werden, dass bei einem Zeichenanschlag ein Klang abgespielt wird, z.B. das Geräusch einer Schreibmaschine oder ein Klicken. Als zusätzliche Option könnte das Texteingabe-Widget eine Schaltfläche beinhalten, bei der auf Spracherkennung umgeschaltet wird. Hier könnte dann buchstabiert oder ganze Worte erkannt werden.

Zur Integration einer Online-Hilfe bietet sich ebenfalls die Sprachausgabe an. Ist sich der Benutzer an einer Stelle im System unklar über die angebotenen Funktionen, so kann er entweder durch eine Schaltfläche oder einen Sprachbefehl wie „Hilfe“ eine Kontexthilfe anfordern. Dabei bietet sich an, dass die Sprache durch graphische Animationen und Kenntlichmachung des gerade beschriebenen Elements unterstützt wird. Eine reine Texthilfe eignet sich im Fahrzeug nur sehr wenig, da sie eine hohe Konzentration und langen Sichtkontakt erfordert. Die Kombination von Animation und Sprache kann dagegen einen ähnlich hohen Informationsgehalt bei wesentlich geringerer kognitiver Belastung erreichen, siehe [13]. Dabei sollte die Hilfe möglichst nach Aufgaben gegliedert sein und wie sie mit dem System zu bewältigen sind.

## 4.2 Einbeziehung des HUD

Eine zusätzliche Erweiterung des FIS könnte die Einbeziehung des HUDs<sup>1</sup> sein. Dieses Display befindet sich in der Windschutzscheibe direkt vor dem Fahrer, was durch

---

<sup>1</sup>Head Up Display



Abbildung 4.1: Kombination von HUD und CID, aktiviert durch Annäherung

eine Projektion von unten geschieht. Bisher wird es beispielsweise zur Geschwindigkeitsanzeige oder zur Abstandswarnung benutzt, nicht jedoch um als Infotainmentanzeige zu dienen. Da diese Anzeigefläche im direkten Sichtfeld des Fahrers liegt, darf es nicht zu aufdringlich sein und nur eine minimale Informationsdichte aufweisen.

Es wäre nicht sinnvoll, den vollständigen Bildschirminhalt des CID zu duplizieren, da nicht die nötige Fläche zur Verfügung steht und zuviel Information dargestellt wird so dass ein schnelles Ablesen nicht mehr gewährleistet ist. In diesem Fall könnte die Annäherungsmessung eine nützliche Information sein. Sobald der Nutzer seine Hand für einen gewissen Zeitraum über dem CID hält, wird der Menüausschnitt im HUD angezeigt, über dem sich der Zeigefinger momentan befindet. Die Informationsmenge wird also auf den Bereich reduziert, der vom Benutzer aktuell gewünscht wird. Dadurch wird das CID zeitweise zum reinen Eingabegerät und die zugehörige Ausgabe geschieht im HUD. Der angezeigte Bereich sollte dabei nicht kontinuierlich mit dem Finger mitbewegt werden, sondern nur das nächstgelegene Menü vollständig im HUD angezeigt werden.

Mit der Trennung von Ein- und Ausgabemedium kann der Fahrer seine Position beim Lenken des Fahrzeugs weitgehend beibehalten, da der Kopf nicht abgewandt und die Hand nicht aus der Umgebung des Schalthebels wegbewegt wird. Bewegt der Nutzer seine Hand wieder weg vom CID, so wird das HUD wieder umgeschalten auf die üblichen Anzeigen. Falls das System erkennen kann, ob der Fahrer oder der Beifahrer momentan bedient, so darf das HUD nur bei der Fahrerbedienung aktiviert werden. Sonst könnte für den Fahrer eine Schrecksituation entstehen wenn für ihn unvermittelt das HUD verändert wird. Für diesen Eingabemodus ist eine gute anthropometrische Auslegung der CID-Position von noch größerer Bedeutung als bei der Anzeige im CID. So muss ein Kompromiss gefunden werden zwischen einer guten Erreichbarkeit von der Mittelkonsole aus und einer möglichst zentralen Position am Armaturenbrett für einen guten Sichtwinkel. Dabei kann die Konsole zwischen den Vordersitzen unterstützend wirken, indem sie eine Auflage für den Arm bei der



Abbildung 4.2: Mausgeste in einem Web Browser, aus [30]

Bedienung des Touch Screens bildet.

Zusätzlich wäre es denkbar, dass die Verwendung des HUD ab einer gewissen Geschwindigkeit deaktiviert wird. So kann der höheren benötigten Konzentration für die primäre Fahraufgabe Rechnung getragen werden und es verleitet auch nicht zu einer Bedienung bei hohen Geschwindigkeiten.

### 4.3 Gestenerkennung

Durch die Möglichkeit zur dreidimensionalen Messung von Finger- und Handposition ergibt sich die Option zur Erkennung von Gesten. Mit deren Hilfe kann sowohl die Menünavigation als auch beispielsweise die Texteingabe unterstützt oder eine völlig neue Eingabemethode erarbeitet werden.

Die Navigationssteuerung mit Hilfe von Gesten ist bereits ein weit verbreitetes Feature bei Web Browsern. Dazu werden die Gesten indirekt durch die Mausbewegung wiedergegeben und in der Regel als Linie auf dem Bildschirm nachgezogen, siehe Abbildung 4.2. Aus Nutzerkommentaren zu diesen Browsererweiterungen ist zu entnehmen, dass im Durchschnitt 3-5 Gesten regelmäßig benutzt werden. Diese Gesten bestehen meist nur aus ein oder zwei Liniensegmenten, wodurch sie einfach zu merken und schnell durchzuführen sind. Bei den verschiedenen Programmen ist häufig schon ein gewisses Repertoire an Gesten vordefiniert und es können noch weitere trainiert und verschiedenen Funktionen zugewiesen werden.

Als Erweiterung zu diesen zweidimensionalen Eingaben können die Gesten mit dem hier eingesetzten Touch Screen auch dreidimensional durchgeführt werden. Das kann sowohl vor- als auch nachteilig sein. Ein Vorteil ist, dass die Sensoroberfläche nicht mehr erreicht werden muss und damit die Hand nicht so weit in Richtung Bildschirm bewegt werden muss. Gleichzeitig geht das durch die Berührung erzeugte haptische Feedback vollständig verloren und der Bildschirm kann nicht mehr als physikalischer Anhaltspunkt benutzt werden. Die zusätzliche Dimension erlaubt komplexere Gesten bzw. erweitert die Anzahl an möglichen Bewegungen. Neben der Menüsteuerung können Gesten auch zur Texteingabe benutzt werden, sowohl bei der zweidimensionalen als auch bei der dreidimensionalen Eingabe. Die Handschrifterkennung in der Ebene wird bereits seit längerer Zeit benutzt, vor allem bei PDAs oder Tablet PCs wird diese Eingabemethode angewandt. Mittlerweile ist diese Schrifterkennung sehr leistungsfähig, so dass auch unsaubere Handschrift zu einem richtigen Ergebnis führt. Im Gegensatz dazu ist die 3D-Texteingabe nicht so stark verbreitet, hauptsächlich aufgrund fehlender Eingabegeräte. Folglich existieren auch praktisch keine robusten kommerziellen Lösungen.

Alle obigen Eingabemethoden teilen sich die Problematik, dass eine Geste von einer

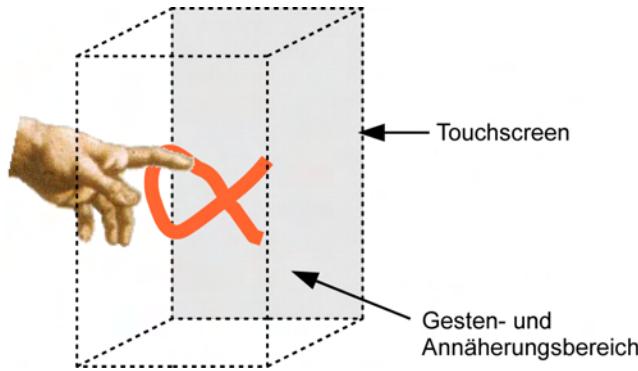


Abbildung 4.3: Dreidimensionale Texteingabe

Pointerbewegung unterschieden werden muss. Bei PDAs wird dies durch dedizierte Eingabebereiche erreicht, bei Betriebssystem- oder Browsererweiterungen wird die Eingabe beispielsweise durch Halten der rechten Maustaste aktiviert. Für die 3D-Texteingabe ist eine solche Segmentierung des Eingabebereichs zwar theoretisch möglich, doch besteht die Schwierigkeit, dass dieser Bereich nicht optisch oder haptisch markiert werden kann. Eine Lösungsoption wäre, die Texteingabe durch eine Schaltfläche ein- und auszuschalten. Dieses Vorgehen eignet sich allerdings nicht für die Menünavigation. Alternativ könnte eine weitere Geste wie z.B. das Zusammenziehen von Daumen und Zeigefinger die Navigationsgesten aktivieren, was auch einer relativ intuitiven Geste entspricht. So kann die Geste für das Kommando „Zurück“ aus dem „Zupacken“ mit den beiden Fingern und einer Handbewegung nach links bestehen.

Eine Grundvoraussetzung für obige Aktionen ist eine leistungsfähige Handerkennung und ein entsprechend empfindlicher Sensor. Mit der momentan eingesetzten Hardware ist dies jedoch nicht möglich, ebenso sind die momentan implementierten Erkennungsalgorithmen nur in der Lage ein einziges Objekt zu detektieren, ebenso kann die Form nicht identifiziert werden.

## 4.4 Adaptivität

Im Datenmanagementsystem sind bereits mehrere Funktionen implementiert, die eine automatische Adaptierung des Systems an das Nutzerverhalten erlauben. Diese Adaptierung lässt sich noch weiter unterteilen auf die Datenebene und die Funktionsebene. Parallel dazu könnte der Nutzer selbst die GUI an seine Bedürfnisse anpassen.

Die Datenbank kann für nahezu alle in ihr gespeicherten Objekte einen Zähler und den letzten Zugriffszeitpunkt verwalten. Dabei ist der Zähler ein Index für die Verwendungshäufigkeit eines Objekts und der Zeitstempel ein Maß für die Aktualität der Zugriffe. Beide werden nur dann aktualisiert, wenn die GUI dies explizit auslöst. So obliegt es dem Programmierer der Benutzeroberfläche, bei welchen Vorgängen er den Nutzungszähler erhöhen will und wann nicht. Basierend auf diesen Aufzeichnungen kann der Datendienst Auswertungen durchführen, dazu gehören unter anderem:

- meistgenutzte Kontakte
- zuletzt genutzte Daten

- meistgenutzte Daten
- zuletzt gespielte Songs
- meistgenutzte Interface-Objekte
- zuletzt genutzte Interface-Objekte

In dieser Liste sind bis auf die letzten beiden Punkte alles Funktionen, die zur Datenadaptivität herangezogen werden, die anderen dienen der Funktionsadaptivität. Wie schon im Kapitel „Implementierung“ erwähnt, können Gewichtungsfunktionen erstellt werden, die die beiden Attribute Nutzungshäufigkeit und Nutzungszeit gewichten und daraus eine Relevanzrangfolge erstellen. So können z.B. im Navigationsmenü die 10 meistgenutzten Kontakte in einer Liste angezeigt werden um sie als Routenziele zu wählen. Allerdings sollte ein Kontakt, der zwar oft benutzt wurde aber die letzte Nutzung ein Jahr zurückliegt nicht in dieser Liste erscheinen da er offenbar momentan nicht relevant ist. Als weitere Anwendung wären die relevantesten Begriffe in der Suchmaschine oder die Top 10 der vorhandenen Songs oder Playlists. Da es im Datenmanagementsystem möglich ist, sowohl GUI-Objekte zu registrieren als auch Transitionen zwischen ihnen, kann eine fast beliebige Funktionsadaptivität hergestellt werden. Die dafür bereitgestellten Funktionen lauten:

- addUIElement
- removeUIElement
- addCountToUIElement
- addUITransition
- removeUITransition
- getMostUsedUIComponent
- getLastAccessedUIComponent
- getMostUsedNextUIComponent
- getLastAccessedNextUIComponent

Mit diesen Funktionen kann die Bewegung des Benutzers durch das System erfasst und zur Adaptierung benutzt werden. Eine denkbare Verwendung für diese Informationen wäre eine Schaltfläche, die Vorschläge für die nächsten Aktionen und damit Abkürzungen „Shortcuts“ anbietet. Diese Art der Adaption verändert also nicht vorhandene Menüs, zu denen der Nutzer ein mentales Modell erarbeitet hat, sondern bietet zusätzliche Shortcuts bzw. Transitionen an, die oft benutzt werden. In der Schaltfläche werden also Vorschläge auf die Frage „Was möchten Sie als nächstes tun?“ aufgeführt. Springt der Nutzer z.B. sehr häufig in das Menü zum Editieren von Playlists, so könnte in der Vorschlagsschaltfläche in der Taskleiste eine Verknüpfung zu diesem Menü erscheinen. Auf diese Weise wurde eine Form von Funktionsadaptivität erzeugt.

In der oben beschriebenen Schaltfläche könnten zusätzlich auch oft oder zuletzt

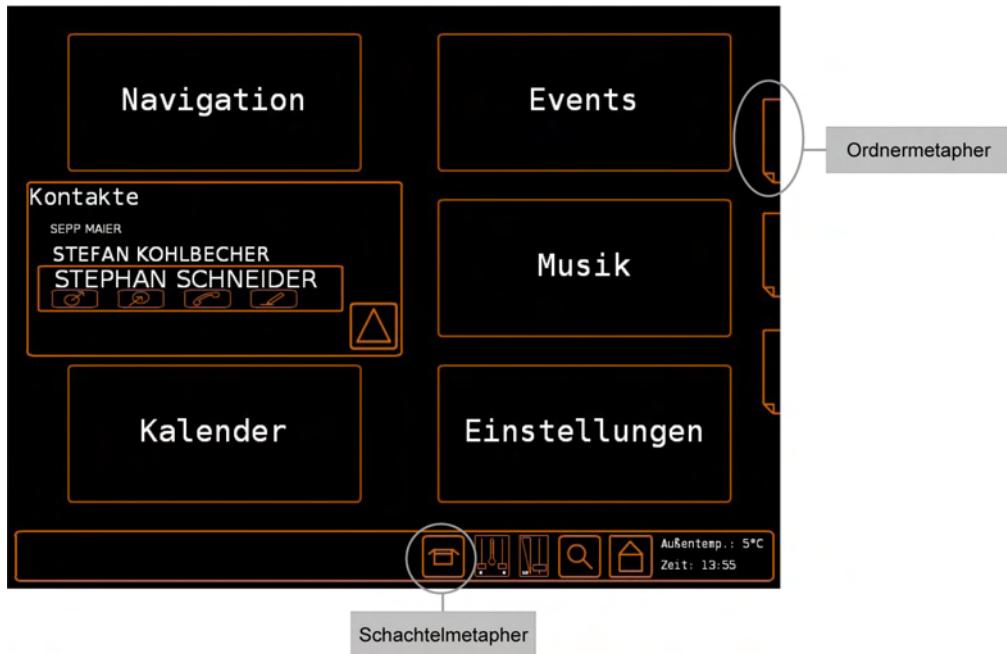


Abbildung 4.4: Ordner- und Schachtelmetapher zur Adaption

gehörte Songs oder Radiostationen aufgeführt sein, um diese schnell abspielen zu können. Ebenso wären zuletzt benutzte Kontakte denkbar, die dann angerufen oder als Navigationsziel benutzt werden können. Damit wäre eine adaptive Datenauswahl realisierbar. Bei geschickter optische Aufbereitung kann so eine Liste geschaffen werden, die theoretisch einen Großteil der häufig anfallenden Aufgaben abdeckt und so eine schnelle und mit geringstem Aufwand verbundene Nutzung des Systems ermöglicht.

Neben der oben beschriebenen systemgetriebenen Adaptivität ist auch eine Individualisierung durch den Nutzer selbst denkbar. Eine mögliche Realisierung einer solchen Funktion wäre die Ausblendung von selten benutzten GUI-Elementen wie z.B. Hauptmenüpunkten. Bei PC-Betriebssystemen hat Microsoft eine solche Adaptierung eingebaut in der Art, dass selten benutzte Menüpunkte ausgeblendet werden. Die Problematik dieser Vorgehensweise liegt darin, dass die Ausblendung automatisch vom System durchgeführt wird und der Nutzer keinen Einfluss darauf hat. Die Akzeptanz ist dementsprechend gering. Wird die Anpassung jedoch vom Benutzer selbst initiiert, so kann er sie in seinem mentalen Modell registrieren. Um die ausgeblendeten Menüs dennoch kenntlich zu machen wäre eine Büroklammer- oder Ordnermetapher denkbar.

In Abbildung 4.4 ist am rechten Rand eine mögliche Implementation der Ordnermetapher angedeutet, wobei für jedes ausgeblendete Menü ein Karteireiter erstellt wird. Eine andere Verwendung für diese Metapher könnten Verknüpfungen zu den jeweiligen Menüs sein. So ergeben sich zwei mögliche Anwendungsszenarien:

- „Verschieben“ in einen Karteireiter
- „Verknüpfen“ mit einem Karteireiter

Bei der ersten Variante wird das Menü von seiner ursprünglichen Position entfernt und in einen Reiter verlagert, in der zweiten Variante bleibt es an seinem alten Ort und es wird nur ein Verweis darauf angelegt.

Des weiteren ist in Abbildung 4.4 am unteren Rand eine Schaltfläche mit einer Schachtel darauf zu sehen. Sie kann als Aufbewahrungsbox gesehen werden, in der nicht benötigte Menüs gesammelt werden. Ähnlich wie beim „Papierkorb“ in aktuellen Betriebssystemen kann der Inhalt der Schachtel eingesehen und bei Bedarf einzelne Elemente daraus wiederhergestellt werden. Ebenso wie bei den Karteireitern kann diese Schachtel als Aufbewahrungsort gesehen werden, nicht nur um ausgeblendete Menüs aufzubewahren, sondern auch für Verknüpfungen zu noch vorhandenen. Die Schachtel kann damit wie eine Art Linkssammlung ähnlich wie in Internet-Browsern verstanden werden.

## 4.5 Ausbau des Menüsystems

Neben den oben erwähnten funktionellen Erweiterungen sind die Grundlagen für zusätzliche Menüs und Funktionalitäten bereits geschaffen. Die Hauptmenüpunkte „Events“, „Kalender“ und „Einstellungen“ sind vorhanden, bieten jedoch keinerlei Funktionalität.

Das Einstellungsmenü kann sowohl Optionen für das FIS selbst bieten als auch für das Fahrzeug. So könnten beispielsweise Farbschemata oder Eigenschaften des Eingabegeräts ebenso wie die Scheinwerfereinstellung in diesem Menü verwaltet werden. Dabei muss vor allem bei den Fahrzeugeinstellungen abgewogen werden, welche davon im FIS und welche mit einem Hardware-Controller bearbeitet werden. Einstellungen, die oft während der Fahrt geändert werden sollten daher einen eigenen Knopf zugewiesen bekommen, der noch näher am Lenkrad oder Schalthebel positioniert ist als das FIS.

Im Menü „Kalender“ könnten sowohl Termine in einer Wochen- oder Monatsübersicht dargestellt werden, ebenso wie eine ToDo-Liste oder Wartungszeitpunkte für das Fahrzeug. Dabei kann eine enge Verknüpfung mit dem Kontakte-Menü realisiert werden. Durch die datengetriebene Menüstruktur könnte ein Sprung aus der Kontaktliste zum Kalender ebenso erfolgen wie von einem Kalendereintrag zum assoziierten Kontakt. Das Datenmanagementsystem bietet die Möglichkeit, mehrere Kontakte zu einem Kalendereintrag hinzuzufügen. In zukünftigen Anwendungen sollte es vor allem möglich sein, die Kalendereinträge von angeschlossenen Mobiltelefonen, PDAs oder Online-Systemen wie Google Calendar<sup>2</sup> mit dem Bordsystem zu synchronisieren. Dazu müssen vor allem noch weitreichende Standards zum Datenaustausch geschaffen werden, wie sie momentan schon teilweise mit den Formaten „iCalendar“ und „CalDAV“ vorhanden sind.

Das „Events“-Menü zeigt aktuelle Meldungen von verschiedenen Systemen an. Es dient als zentrale Anzeigefläche für Meldungen, die mit einer Bewertungsfunktion nach Wichtigkeit und Dringlichkeit sortiert werden. So ist ein Termin in 4 Stunden zwar wichtig, aber nicht so dringlich wie ein Tankstop, der in den nächsten 20km gemacht werden muss. Parallel zu diesem Menü könnten ausgewählte Meldungen auf dem HUD ausgegeben werden, wobei die Priorisierung bei fast zeitgleich auftretenden Ereignissen eine große Rolle spielt, siehe [16].

---

<sup>2</sup>[www.google.com/calendar](http://www.google.com/calendar)

# 5. Evaluierung

In diesem letzten Abschnitt wird die abschließende Evaluierung des oben beschriebenen Systems beschrieben. Parallel zu dieser Arbeit wurden die Diplomarbeit [9] und die Bachelorarbeit von Benedikt Scheller angefertigt, daher fand eine gemeinsame Versuchsdurchführung statt. So konnte auch eine optimale Vergleichbarkeit der Systeme sichergestellt werden.

## 5.1 Versuchsplanung und Durchführung

Die Versuche wurden im Fahrsimulator des MMK-Lehrstuhls durchgeführt. Dabei handelt es sich um ein von BMW bereitgestelltes Fahrzeug, das für die Durchführung von MMI-Versuchen modifiziert ist. Das Chassis wurde hinter den Vordersitzen abgetrennt und die gesamte im Heck befindliche Elektronik im vorderen Teil untergebracht. Des Weiteren wurden sämtliche Anzeigen (HUD, CID, und analoge Instrumente) durch digitale Displays ersetzt, wodurch fast beliebige MMI-Konzepte getestet werden können. Da das hier vorgestellte System jedoch auf einem speziellen Touch-Sensor basiert, der auf 15 Zoll Monitore ausgelegt ist, musste ein externer Bildschirm an der Instrumententafel hinter dem Schalthebel montiert werden. In Abbildung 5.1 ist das Cockpit mit dem Touch Screen in der Mitte zu erkennen. Um die Fahrsimulation realistisch steuern zu können, wurde an das vorhandene Lenkgestänge ein Force-Feedback-Lenkrad angeschlossen. Auf diese Weise kann sowohl die Lenkbewegung des Fahrers erfasst und gleichzeitig eine realistische Kraftrückkopplung erzeugt werden, was den Immersionsgrad erheblich steigert. Ebenso wurde an die vorhandenen Pedale das Pedalmodul des Lenkrads angeschlossen, so dass auch



Abbildung 5.1: Links: Innenansicht des Simulatorfahrzeugs, rechts: Lane Change Task

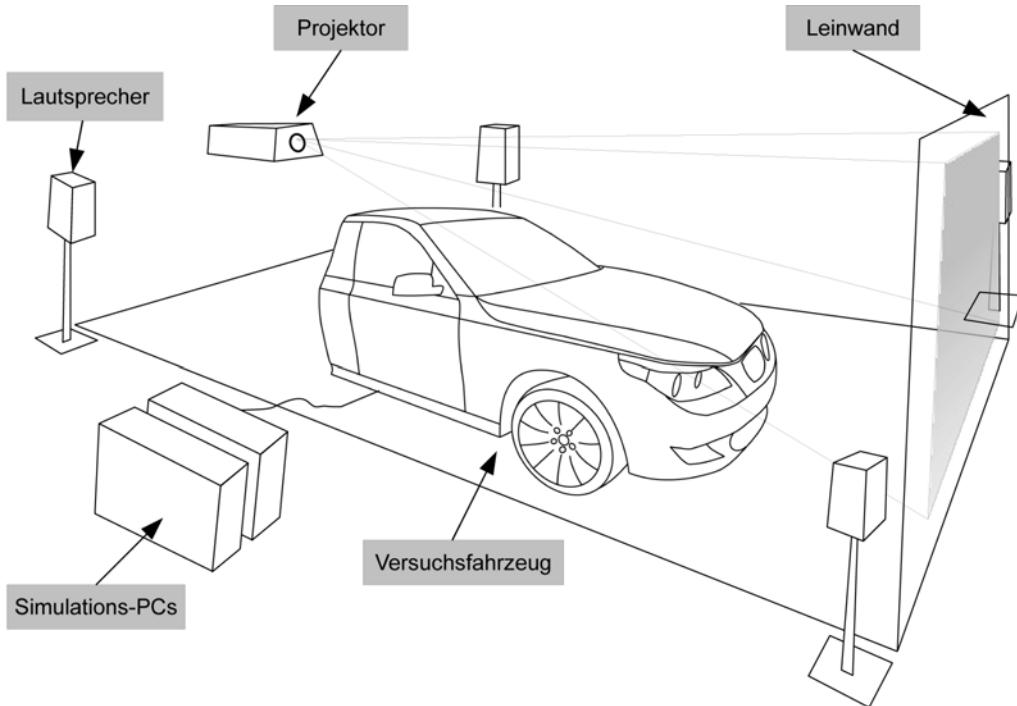


Abbildung 5.2: Schematischer Versuchsaufbau

diese Bedienung absolut intuitiv erfolgen kann.

Das Fahrzeug ist in einem Versuchsraum aufgestellt, in dem eine Leinwand und an der Decke ein Datenbeamer montiert sind. Über diesen Beamer wird die Fahraufgabe projiziert, die von einem eigenständigen Rechner erzeugt wird. Dieser Rechner erzeugt auch Audiosignale wie z.B. Motorgeräusche und gibt sie über die um das Fahrzeug platzierten Lautsprecher aus. Parallel dazu existiert ein weiterer Rechner, auf dem das FIS ausgeführt wird. Der gesamte Versuchsaufbau ist in Abbildung 5.2 schematisch dargestellt. Für den Versuchsablauf wurde folgende Einteilung gewählt:

- Theoretische Einführung in das Versuchsszenario
- Erfassung der generellen Personendaten wie Alter, Geschlecht, Vorwissen etc.
- Erklärung des Fahrzeugs und der grundsätzlichen Funktionsweise des FIS
- Kurze Probefahrt im Simulator
- Erklärung der zu bewältigenden Aufgabe
- Start der Fahrsimulation, Durchführung der Aufgabe mit Zeitnahme
- Stop der Simulation und Ausfüllung des zugehörigen Fragebogens
- Bedienung der Texteingaben
- Bewertung jeder Eingabe mittels dem AttrakDiff-Fragebogen

Der Versuchsteilnehmer wird zunächst in den Hintergrund des Versuchs und des Szenarios eingeführt. So kann er das angestrebte Ziel besser einordnen und kann sich

mit dem System besser identifizieren. Ebenso wird jedem Probanden eine Nummer zugewiesen, was eine Verknüpfung mit dem AttrakDiff-System erlaubt. Anschließend werden die grundlegenden Daten aufgenommen, um später die Zuordnung zu einer noch festzulegenden Kohorte<sup>1</sup> durchführen zu können. Die Abgrenzung der Kohorten wird zum Großteil erst nach den Versuchen durchgeführt, da bei einer angestrebten Zahl von ca. 20 Probanden die Mitgliederzahl jeder Kohorte stark schwanken kann. Es wird grundsätzlich eine möglichst heterogene Zusammensetzung der Probanden angestrebt, um möglichst allgemeingültige Aussagen über die getesteten Systeme zu erhalten.

Mögliche Kohorteneinteilungen sind dabei Geschlecht, Alter, Fahrerfahrung, Erfahrung mit anderen FIS usw. Dabei können sich die einzelnen Attribute bei manchen Probanden auch überschneiden oder sie haben einen kausalen Zusammenhang, z.B. dass ein jüngerer Mensch nicht die gleiche Fahrerfahrung vorweisen kann wie ein älterer.

Danach wird das zu testende System dem Versuchsteilnehmer grundsätzlich erklärt und eine kurze Einführung die die Funktionalität des Simulators gegeben. Dazu wird auch eine Probefahrt mit dem „Lane Change Task“ durchgeführt. Diese sogenannte „Fahraufgabe“ soll eine dem realen Verkehr möglichst ähnliche Beanspruchung erzeugen und eine möglichst definierte Ablenkung bei der Erfüllung der Aufgaben mit dem FIS sicherstellen. Ein Screenshot des Lane Change Task ist Abbildung 5.1 rechts abgebildet. Die Fahrt in der Simulation dauert im Durchschnitt ca. 3min. Während dieser Zeit muss der Proband auf einer dreispurigen Autobahn fahren, während auf Schildern am Fahrbahnrand die zu befahrende Spur angezeigt wird. Wie in der Abbildung zu sehen ist, wird die geforderte Spur mit einem Pfeil markiert. Der Spurwechsel soll so schnell und effizient wie möglich erfolgen, nachdem das anzeigende Schild zu erkennen ist. Bei Fahrtantritt soll der Proband das Gaspedal vollständig durchdrücken, so dass er möglichst schnell die eingestellte Höchstgeschwindigkeit von 60 km/h erreicht. Während der ganzen Fahrt soll das Pedal gedrückt bleiben, da so nur eine Steuerung in der Querrichtung nötig ist und nicht zusätzlich auch in Längsrichtung. Sobald der Proband den Startbereich der Strecke verlässt und die Höchstgeschwindigkeit erreicht hat, wird mit der Bedienung des FIS begonnen. Der Versuchsleiter stoppt dabei die Zeit, die für jede einzelne Aufgabe benötigt wird.

Wurden alle Aufgaben durchgeführt, wird der generelle Bewertungsbogen für das System ausgefüllt. Dabei werden spezielle Eigenschaften des Systems bewertet, weniger der allgemeine Eindruck. Bei dem hier behandelten System wird unter anderem die Bewertung der multimodalen Bedienung und der Touch-Eingabe gefordert.

Im Anschluss daran wird der Attrakdiff-Onlinefragebogen ausgewertet, von dem ein Ausschnitt in Abbildung 5.3 zu sehen ist. In diesem Fragebogen kann wie in [14] beschrieben wird eine Bewertung der insgesamten pragmatischen und hedonischen Qualitäten durchgeführt werden. Ebenso wie fast der gesamte restliche Fragebogen ist er im Stil des semantischen Differentials gehalten. Hierfür werden jeweils zwei gegensätzliche Adjektive gegenübergestellt und eine Skala mit mehreren Zwischenstufen angelegt. Die Versuchsperson kreuzt daraufhin eine Markierung auf der Skala entsprechend ihrer Bewertung an. Bei dieser Art von Fragebogen kann der Proband die Bewertung sehr zügig und dadurch auch sehr intuitiv abgeben, gleichzeitig sind damit aber Aussagen auf relativ hohem Abstraktionsniveau zu erhalten. Darin besteht sowohl der Vor- als auch der Nachteil dieses Systems: Die Bewer-

<sup>1</sup>Bevölkerungsgruppe mit gemeinsamen Eigenschaften

The screenshot shows a portion of the AttrakDiff questionnaire. At the top, there's a logo and a navigation bar with items: Begrüßung, So geht's, Ihre Beurteilung (highlighted in red), Persönliche Angaben, and Freigabe. Below this is a section titled "Beurteilung des Produkts Demo - A". A subtitle says: "Bitte geben Sie mit Hilfe der folgenden Wortpaare Ihren Eindruck zu Demo - A wieder. Bitte klicken Sie in jeder Zeile eine Position an!". Below this is a 10x2 grid of words with radio buttons for rating. The columns are labeled "menschlich" through "direkt" on the left and "technisch" through "unangenehm" on the right. The rows are: menschlich, isolierend, angenehm, originell, einfach, fachmännisch, hässlich, praktisch, sympathisch, umständlich.

menschlich	<input type="radio"/>	technisch							
isolierend	<input type="radio"/>	verbindend							
angenehm	<input type="radio"/>	unangenehm							
originell	<input type="radio"/>	konventionell							
einfach	<input type="radio"/>	kompliziert							
fachmännisch	<input type="radio"/>	laienhaft							
hässlich	<input type="radio"/>	schön							
praktisch	<input type="radio"/>	unpraktisch							
sympathisch	<input type="radio"/>	unsympathisch							
umständlich	<input type="radio"/>	direkt							

1/3      abbrechen      weiter

Abbildung 5.3: Auszug aus dem Attrakdiff-Fragebogen, aus [31]

tung auf hoher Ebene ist gleichzeitig nicht spezifisch genug um eine Aussage über die Gründe der Auswahl zu erfahren. Um gezielte Verbesserungen einbringen zu können, müssen also gezielt Fragen zu Systemeigenschaften gestellt werden, die als relevant für die Usability und den Joy Of Use gehalten werden.

Mit „pragmatischen Qualitäten“ sind hier die Attribute gemeint, die den Ease Of Use beeinflussen. Sie beziehen sich also darauf, ob das System für die geforderten Aufgaben adäquat ist. Für diese Frage können parallel weitere Metriken eingesetzt werden wie z.B. die benötigte Zeit oder die Anzahl der Fehleingaben. Bei den „hedonischen Qualitäten“ werden die Eigenschaften des Joy Of Use und der Repräsentationsfähigkeit nach außen zusammengefasst. Diese Eigenschaften gewinnen immer mehr an Bedeutung, da sie ein kaufentscheidendes Kriterium sein und gleichzeitig Schwächen in der Funktionalität aufwiegen können, siehe [15]. Während ein pragmatisches Interface die Funktionalität möglichst nüchtern darstellt und möglichst geringe Anforderungen an den Nutzer stellt, wird bei hedonisch orientierten Schnittstellen die Funktion auf neue Art und Weise zugänglich gemacht und oft bewusst eine anspruchsvolle aber auch unterhaltsamere Bedienung gewählt.

Die hedonischen und pragmatischen Qualitäten haben oft gegensätzliche Anforderungen. So ist ein pragmatisches Interface meist

- weniger innovativ um auf vorhandenes Wissen aufzubauen
- stark funktionsorientiert
- graphisch schlicht
- mit bekannten Eingabegeräten und -techniken bedienbar

Dagegen ist ein hedonisch orientiertes System häufig

- innovativ in der graphischen Darstellung

- unterhaltsam und verspielt
- repräsentativ
- mit neuartigen Eingabeprinzipien zu bedienen

Wie auch in [14] beschrieben wird, sind die oben umrissenen Herangehensweisen oft gegensätzlich in ihren Anforderungen. In manchen Fällen ist es jedoch möglich, die Funktionalität eines pragmatischen Interfaces mit den unterhaltsamen Komponenten eines hedonischen Interfaces zu kombinieren, was die wünschenswerteste Verbindung ist.

Ein einheitlicher Fragebogen gibt die Möglichkeit zum direkten Vergleich mehrerer Systeme. Aus diesem Grund wird der AttrakDiff-Test mit dem hier vorgestellten PreTouch-System, CarZoom von Stephan Schneider[9] und dem iDrive von BMW durchgeführt. Dabei dient das iDrive-System als kommerziell eingesetztes Produkt praktisch als Referenz in einem einheitlichen Test. Dadurch können Schwächen und damit Verbesserungsmöglichkeiten aufgezeigt werden. Bei der Auswertung der Fragebögen werden den Abstufungen jedes Attributs Punkte zugewiesen, die verschieden gewichtet in die Teilergebnisse eingehen.

Dabei müssen nicht durchwegs hohe Ergebnisse bei allen Attributen als gutes Ergebnis interpretiert werden. Vielmehr bietet sich so eine Möglichkeit, den Gesamteinindruck des Systems an die gewünschte Corporate Identity eines Herstellers anzupassen.

## 5.2 Ergebnisse

Der oben beschriebene Versuch wurde mit insgesamt 20 Probanden durchgeführt. Um wirklich repräsentative Aussagen zu erhalten ist diese Anzahl noch bei weitem zu gering, für die meisten Befragungen werden mindestens 100 Personen herangezogen. Da die Systeme sich aber in einem sehr prototypischen Stadium befinden, ist diese Evaluierung eher als Vorversuchsreihe anzusehen. Selbst bei einer so geringen Probandenzahl konnten jedoch für die meisten Fragestellungen eindeutige Ergebnisse erzielt werden. Die Zusammenstellung der Probanden war dabei wie in Abbildung 5.4 dargestellt, wobei 4 Frauen und 16 Männer teilgenommen haben: Die Einteilung in Kohorten wurde dabei altersmäßig in „unter 20“, „20 - 40“, „40 - 60“ und „über 60“ durchgeführt. Dabei tätigen 75% der Befragten mindestens einmal täglich Texteingaben am Computer oder Handy und 35% haben durchschnittliche Erfahrungen mit FIS in Mittel- oder Oberklassefahrzeugen.

Soweit möglich wurden für die Fragen Thesen aufgestellt, mit welcher Intention gewisse Systemeigenschaften implementiert wurden. Entsprechend dieser Thesen wurden die Fragen formuliert und die erwarteten Ergebnisse abgeschätzt. Oft werden dadurch große Diskrepanzen zwischen dem Vorhaben der Entwickler und der Probandenbewertung festgestellt. Dies sind wertvolle Informationen für nachfolgende Entwicklungszyklen und Neuentwicklungen, da das Nutzermodell des Entwicklers an die tatsächlichen Begebenheiten angepasst werden kann.

Der erste Fragebogenabschnitt behandelt die Visualisierung. Dabei wurden zunächst die Animationen bewertet. Damit sind Menübewegungen, Schaltflächenanimationen und Visualisierungseffekte wie FishEye oder Zoom gemeint. Die Erwartung ist eine

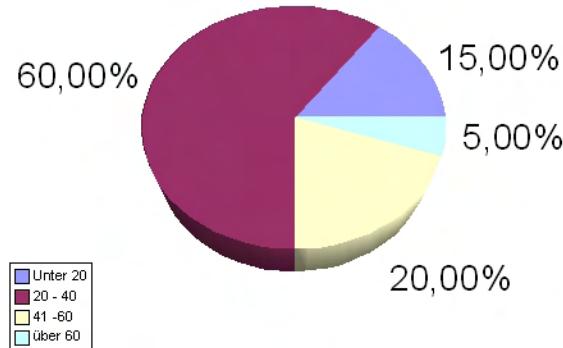


Abbildung 5.4: Zusammenstellung der Probanden

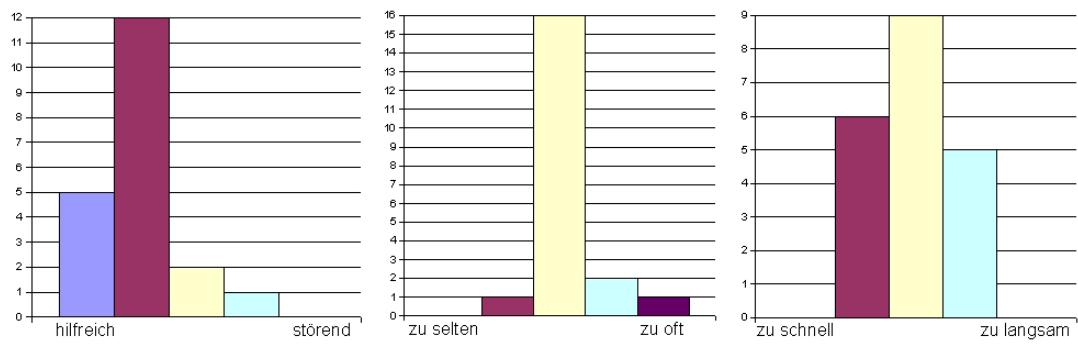


Abbildung 5.5: Ergebnis zu „Wie bewerten Sie die Animationen?“

positive Bewertung, da die Animationen in der Regel dazu beitragen, Zustandsänderungen im Menü für den Nutzer nachvollziehbar zu machen. Das Ergebnis in Abbildung 5.5 zeigt, dass die Animationen als hilfreich eingeschätzt und in angemessenem Maße eingesetzt werden. Bei der Farbgebung wurde auf eine möglichst niedrige Anzahl an verwendeten Farben angestrebt, wobei diese nie flächig eingesetzt werden. Als Richtlinie diente ein schwarzer Hintergrund, orangefarbene Rahmen und weißer Text festgelegt. Diese Farben sollten bei praktisch allen Lichtverhältnissen gut erkennbar sein und lehnt sich gleichzeitig an im Fahrzeug bekannte Farbschemata an. Wie in 5.6 zu entnehmen ist, wurde das Ziel der Schlichtheit und der resultierenden Lesbarkeit auch weitgehend erreicht. Weiter sollten die Probanden eine Einschätzung abgeben, ob Funktionen mehr durch Text oder durch Piktogramme dargestellt werden sollen. Der Vorteil von Piktogrammen ist, dass sie meist einprägsamer sind, jedoch sehr sorgfältig gewählt werden müssen um für jedermann eindeutig zu sein. Im Gegensatz dazu kann mit Text eine präzise Information gegeben werden, die aber manchmal etwas mehr Erkennungszeit beansprucht und oft sprachgebunden ist während ein Bild meist kulturübergreifend die gleiche Bedeutung hat. Bei der Einschätzung für das vorliegende System ist eine eindeutige Tendenz zu einer stärkeren Verwendung von Bildern erkennbar, wie aus Abbildung 5.7 zu entnehmen ist. Eine Alternative zu dieser „Entweder - Oder“-Konfiguration kann eine Kombination von Bild und Text für eine Funktion sein, was in vielen GUIs schon länger verwendet

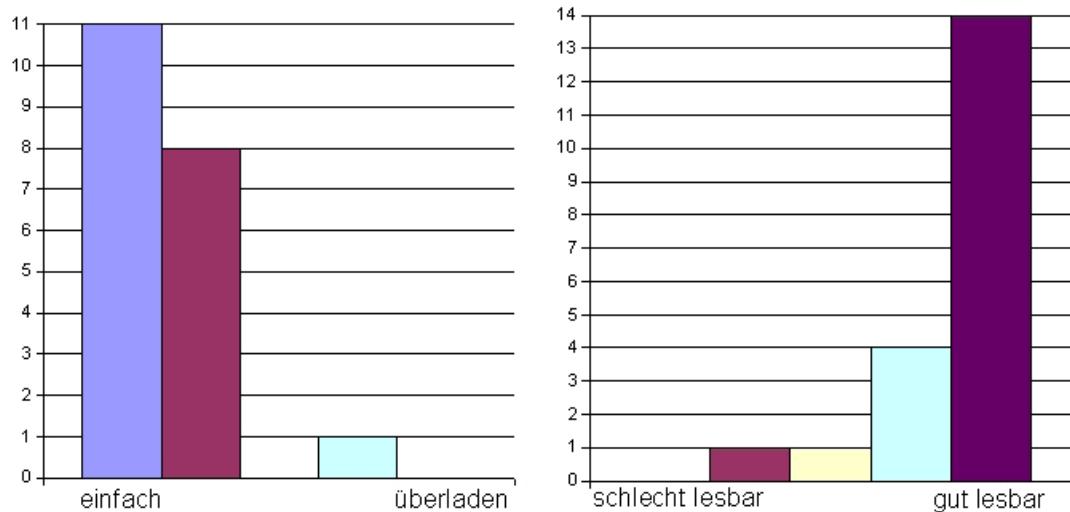


Abbildung 5.6: Ergebnis zu „Wie bewerten Sie die Farbgebung?“

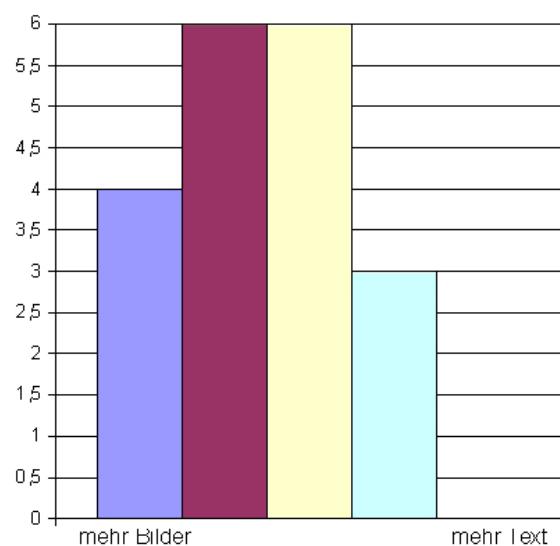


Abbildung 5.7: Ergebnis zu „Sollten mehr Bilder oder mehr Text verwendet werden?“

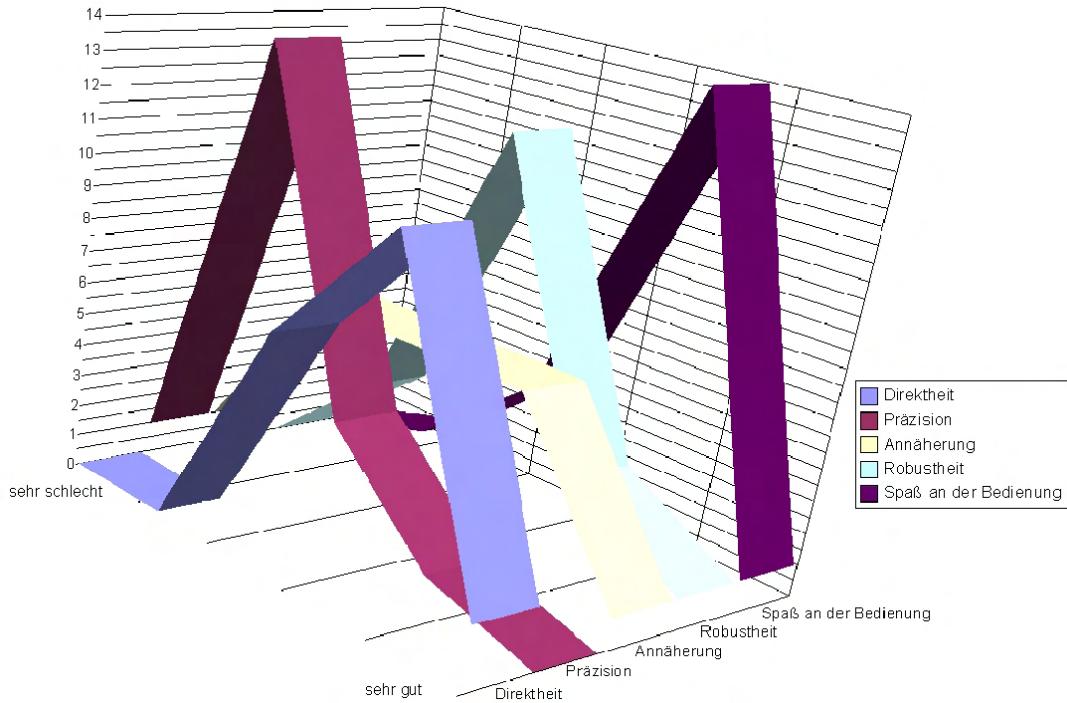


Abbildung 5.8: Ergebnis zu „Wie bewerten Sie folgende Aspekte?“

wird. So kann das Bild zur schnellen Identifikation benutzt werden und bei Unsicherheiten der Text herangezogen werden, welcher beispielsweise permanent, nach einer gewissen Zeit bei Überstreichen („Tooltip“) oder nach Annäherung angezeigt werden kann.

In den folgenden Fragen wurde die Bedienbarkeit bezüglich der multimodalen Eingabe untersucht. Dafür wurde eine Matrix mit der Bewertung „sehr schlecht“ bis „sehr gut“ für verschiedene Eigenschaften der Touch Screen - Bedienung erstellt, deren Ergebnisse in Abbildung 5.8 einzusehen sind. Vor allem die Direktheit (direkte Manipulierbarkeit) und der Spaß an der Bedienung bekamen gute Bewertungen, wo hingegen die Robustheit und Präzision als mittelmäßig bis schlecht eingestuft wurden. Ein relativ undefiniertes Ergebnis bekam die Einbeziehung der Annäherung mit einer mittleren Bewertung und nicht definiertem Maximum.

Mit der Frage „Wie bewerten Sie die Touch-Eingabe?“ sollte der Einfluss der noch zu unruhigen Touch-Bedienung ermittelt werden. Schon bei Vorversuchen wurde das „Zittern“ des virtuellen Zeigers als einer der größten Kritikpunkte genannt, jedoch konnte dies für die abschließende Versuchsreihe nicht vollständig eliminiert werden. Sollte ein Proband die Eingabe als zu unruhig einschätzen, so wird in einer Folgefrage das Maß der negativen Beeinflussung ermittelt. Erwartungsgemäß wurde der Touch-Sensor noch als unruhig angesehen, jedoch zeigt die Folgefrage, dass genau hier noch großes Verbesserungspotential besteht. In mündlichen Kommentaren wurde zusätzlich betont, dass die insgesamte Akzeptanz des Systems stark von der Qualität der Touch-Eingabe abhinge.

Nachfolgend wurde eine Bewertung der multimodalen Eingabe erstellt. Hierfür wurde zuerst die Spracheingabe generell auf ihre Tauglichkeit hin eingestuft, wobei diese als sehr hilfreich angesehen wurde. Obwohl der Spracherkenner teilweise erst bei ei-

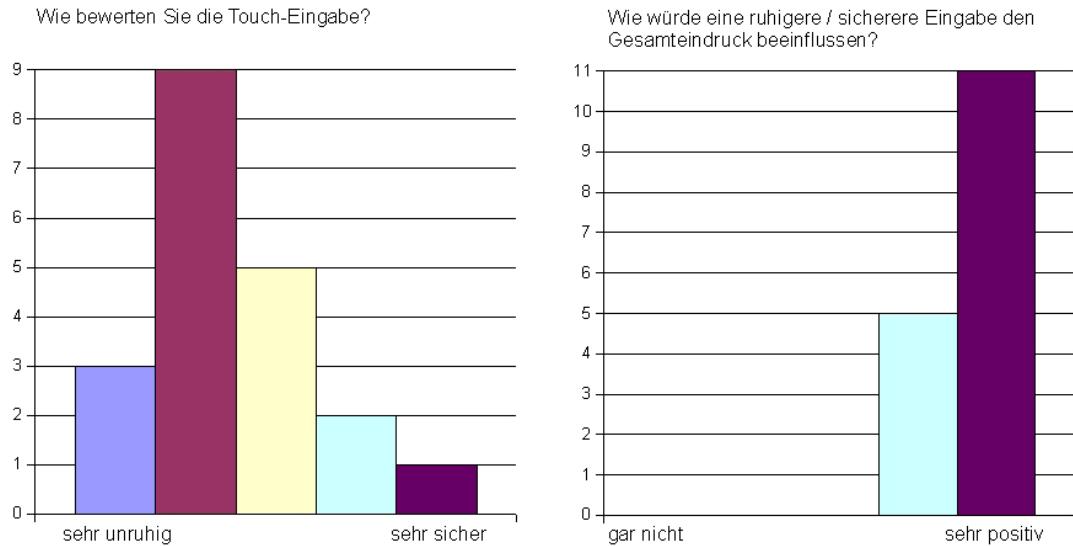


Abbildung 5.9: Ergebnis zur Bewertung der Touch-Eingabe

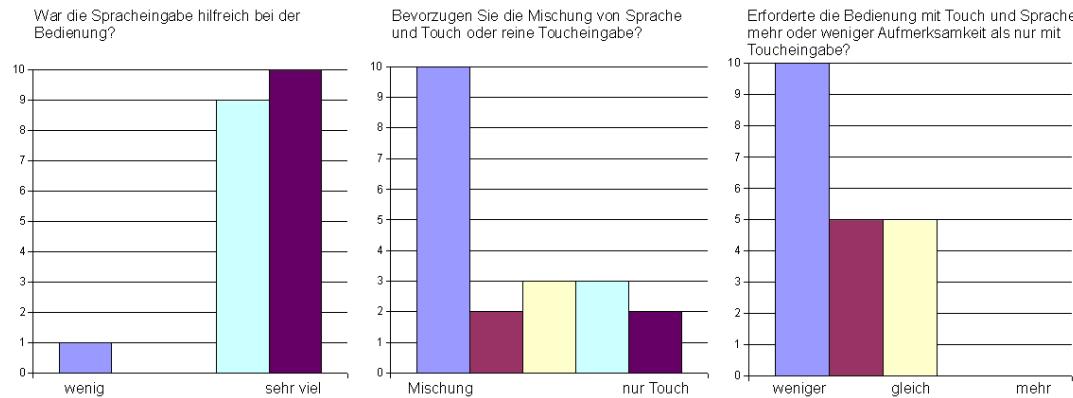


Abbildung 5.10: Ergebnis zur Bewertung der Multimodalität

ner Wiederholung richtig reagierte, scheint dies die Akzeptanz nicht sehr negativ beeinflusst zu haben. Da die Versuche zur Multimodalität während der Fahraufgabe durchgeführt wurden, scheint die Tatsache, den Blick nicht abwenden zu müssen, den Aufwand einer Wiederholung stark zu überwiegen. Des weiteren hat eine Mischung aus Sprach- und Toucheingabe eine wesentlich höhere Akzeptanz als eine reine Toucheingabe, wie in Abbildung 5.10 Mitte erkennbar ist. Dies deckt sich auch mit den Erwartungen, dass die Sprache filigrane Eingaben mit der Hand unterstützen bzw. ersetzen kann und dadurch eine entspanntere Bedienung erlaubt. Bei der Versuchsdurchführung stellte sich vor allem bei der Menünavigation eine Erleichterung heraus, nicht so stark allerdings bei „Point And Talk“-Aktionen. Hierbei entstand die Einschränkung hauptsächlich aufgrund der noch mangelnden Genauigkeit des Touch Screens, da das Konzept grundsätzlich als praktikabel eingestuft wurde. Zusätzlich kommentierten mehrere Probanden, dass eine Rückmeldung des Systems ebenfalls per Sprache eine große Verbesserung darstellen würde, da so ein noch größerer Teil der Bedienung blind erfolgen könnte.

Insgesamt wurde die parallele Eingabe mit Touch und Sprache als weniger belastend

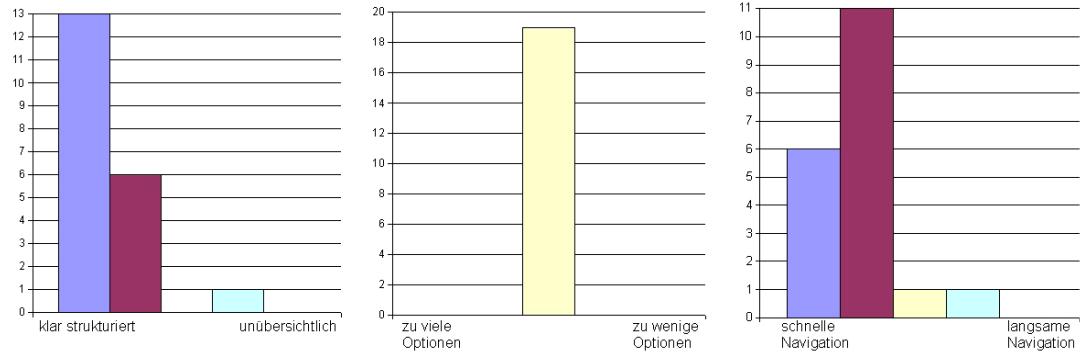


Abbildung 5.11: Ergebnis zur Bewertung des Menüaufbaus

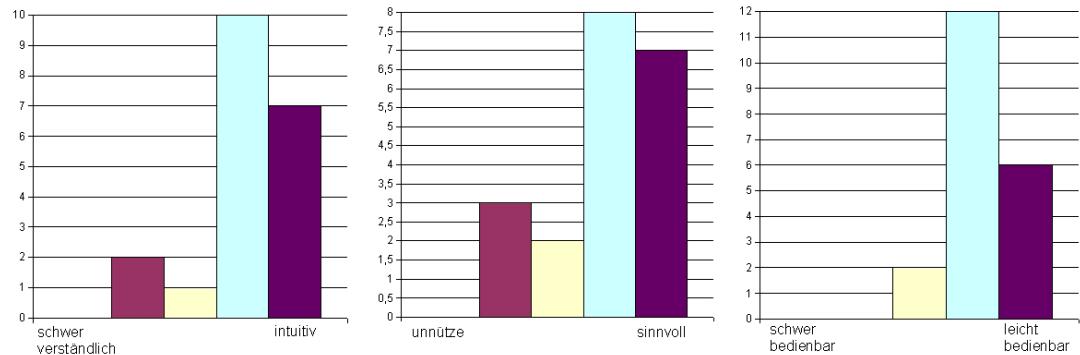


Abbildung 5.12: Ergebnis zur Frage „Wie bewerten Sie das Eintauchen in Menüs?“

empfunden als eine reine Toucheingabe. Die Hinzunahme einer weiteren Modalität verursacht also keine Erhöhung der kognitiven Belastung, sondern verteilt die nötigen Schritte auf mehrere Sinne und stellt damit eine Vereinfachung dar (siehe Abbildung 5.10).

Anschließend wurde der Menüaufbau mit einer Matrix bewertet, die erhaltenen Ergebnisse sind in Abbildung 5.11 ausgewertet. Das Menü wird als klar strukturiert eingestuft und ermöglicht laut den Probanden eine schnelle Navigation. Beide Eigenschaften greifen ineinander, laut der Aussage einer Versuchsperson hat man durch das Zoomable Interface weniger das Gefühl, sich in einem Untermenü zu „verirren“. Bezüglich des Menüaufbaus ist das Ergebnis (Grafik in der Mitte Abbildung 5.11) zur Anzahl der Optionen nicht sehr aussagekräftig, da sich sämtliche Probanden auf eine neutrale Mittelposition festgelegt haben. Dies ist auf eine ungünstige Fragestellung zurückzuführen, da dieser Aspekt wohl mit mehreren Teilfragen behandelt werden müsste.

In den nächsten Fragen wurde eine Bewertung zum Eintauchen in die Menüs und damit für diese Implementierung eines Zoomable Interfaces durchgeführt. Das Konzept wurde als „intuitiv“, „gut bedienbar“ und „sinnvoll“ eingeschätzt, was einen sehr guten Ausgangspunkt für künftige Entwicklungen darstellt. Obwohl dieses Oberflächenkonzept noch nicht so starke Verbreitung hat wie ein listen- oder fensterbasiertes System, war der Umgang damit sehr natürlich und selbstverständlich.



Abbildung 5.13: Ergebnis zur Bewertung verschiedener globaler Eigenschaften

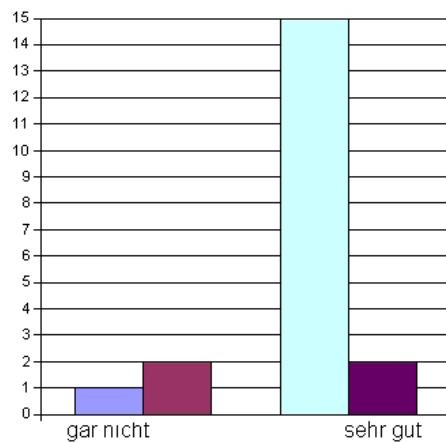


Abbildung 5.14: Ergebnis zur Frage „Können Sie sich dieses System in einem zukünftigen Fahrzeug vorstellen?“

Im letzten Teil des schriftlichen Fragebogens wurden noch einmal die Haupteigenschaften des Systems bewertet. Dabei sollten keine Details abgefragt werden, sondern ein Gesamteindruck abgegeben werden in Bezug auf gewisse Merkmale. Es bestan die Möglichkeit, mehrere Eigenschaften positiv oder negativ zu bewerten. In der rechten Grafik in Abbildung 5.13 werden anteilig die beiden Wertungen aus den linken beiden dargestellt.

Die abschließende Frage ob das System prinzipiell im Fahrzeug vorstellbar wäre wurde eindeutig mit „gut vorstellbar“ eingestuft. Einschränkend wurde jedoch kommentiert, dass dies stark von der Bedienqualität vor allem des Touch Sensors abhängt. Das Ergebnis ist in Abbildung 5.14 visualisiert.

Nach diesem Papier-Fragebogen wurde der AttrakDiff Online-Fragebogen von den Probanden bearbeitet. Wie oben beschrieben handelt es sich hierbei um einen Fragebogen auf Basis des semantischen Differentials. Der Test kann online ausgewertet werden und erzeugt ein Ergebnis bezüglich insgesamter pragmatischer und hedonischer Qualität, eine feinere Auflösung bezüglich dieser Eigenschaften und ein gemitteltes Gesamtprofil bezüglich jedes Wortpaars. In sämtlichen Grafiken der AttrakDiff-Auswertung sind zum Vergleich ebenfalls die Ergebnisse der beiden anderen Systeme „CarZoom“ und „iDrive“ eingetragen. Der einheitliche Test lässt einen direkten Vergleich der „weichen“ Eigenschaften der Systeme zu, was vor allem interessant ist weil sich unter den Systemen das kommerziell eingesetzte „iDrive“ von

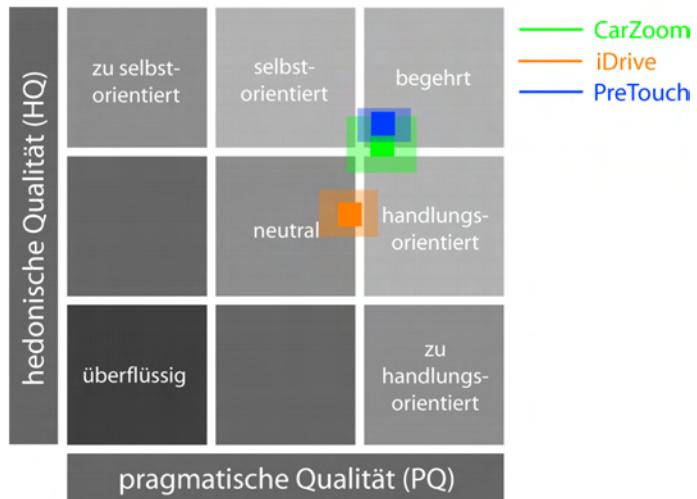


Abbildung 5.15: Ergebnis der AttrakDiff-Auswertung und Konfidenzrechteck

BMW befindet. In Abbildung 5.15 werden die Systeme in ein Koordinatensystem aus pragmatischer und hedonischer Qualität eingetragen. Dabei ist das dunkle Zentrum der Punkt der Mittelwerte und die umgebende hellere Fläche das Konfidenzrechteck. Je weiter ein Produkt in der rechten oberen Ecke positioniert wird, desto positiver wird es insgesamt bewertet. Im Gegensatz dazu ist die linke untere Ecke die niedrigste Bewertung. Das Konfidenzrechteck zeigt an, in welche Richtungen und wie stark die Ergebnisse der einzelnen Probanden vom Mittelwert abweichen. Je größer das Rechteck ist, umso weniger waren sich die Versuchspersonen einig. Ist das Rechteck im Gegensatz dazu sehr klein, so deutet dies auf ein eindeutiges Ergebnis hin, das vor allem auch repräsentativer ist als bei größerer Streuung. Zusammengefasst wird also ein kleines Konfidenzrechteck in der rechten oberen Ecke des Koordinatensystems angestrebt.

Gut zu erkennen ist, dass beide Systeme „PreTouch“ und „CarZoom“ sowohl bei der pragmatischen als auch der hedonischen Qualität höher als das Referenzsystem „iDrive“ eingestuft wurden. Gleichzeitig sind die Konfidenzrechtecke ähnlich groß bzw. kleiner als bei iDrive, was für ein aussagekräftiges Ergebnis spricht. Interessant ist vor allem, dass die pragmatische Qualität selbst bei den nicht vollständig implementierten Systemen höher bewertet wurde als bei dem voll funktionsfähigen iDrive.

In Abbildung 5.16 können obige Aussagen noch einmal detaillierter untersucht werden. Dazu wird die hedonische Qualität „HQ“ aufgeteilt in die hedonische Qualität -Stimulation „HQ-S“ und -Identifikation „HQ-I“. Bei der Stimulation werden Eigenschaften bewertet, die den Nutzer einerseits vor eine gewisse Herausforderung stellen und aber gleichzeitig unterhaltend wirken. Er soll durch neue Interaktionsstile, Funktionen und Darstellungsarten angeregt werden, sich mit dem System zu beschäftigen. Parallel dazu soll das Maß HQ-I anzeigen, wie sehr sich der Bediener mit dem Produkt identifizieren kann. Ebenso wie das Stimulationsmaß ist das Identifikationsmaß stark von der befragten Bevölkerungsgruppe abhängig, wobei angestrebt wird, dass für die größtmögliche Menge an Nutzern ein hoher Wert erreicht wird.

Zusätzlich zu den bisher bewerteten Eigenschaften wird noch das Maß „Attraktivität“ eingeführt, welches eine zusammengefasste Wertung der pragmatischen und hedonischen Qualität darstellt. Die Attraktivität stellt also ein Gesamurteil auf-

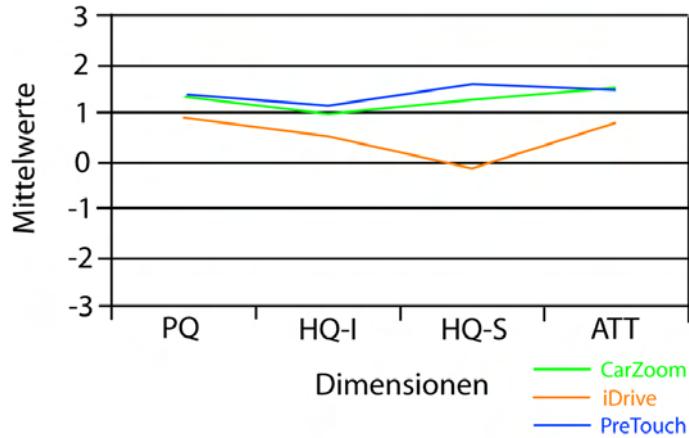


Abbildung 5.16: Mittelwerte der AttrakDiff-Auswertung

grund der oben beschriebenen Maße dar. Auch hier liegen beide Systeme PreTouch und CarZoom höher als das iDrive. Trotz der unterschiedlichen Funktionsausprägungen und Eingabegeräte wurde doch ein praktisch identisches Ergebnis bei den ersten beiden Systemen erzielt.

Die detaillierteste Ergebnisanalyse lässt die Grafik in Abbildung 5.17 zu. Dazu werden die Mittelwerte zu jedem Wortpaar über alle 20 Probanden berechnet und zu einem Graph zusammengefügt. Hier wird auch erkennbar, welche Teile des Fragebogens zu welcher Qualität beitragen, wobei die Gewichtung für den Gesamteindruck nicht gleichmäßig ist und diese immer wieder von der User Interface Design GmbH angepasst wird. Hiermit können gezielt gewisse Eigenschaften analysiert und vor allem optimiert werden.

Abschließend sollen noch die beiden Texteingabesysteme „FishKey“ und „Matrix-Key“ bewertet werden. Beide wurden unter der Prämisse entwickelt, möglichst wenig Bildschirmfläche zu benötigen, die Annäherungsinformation zu verwerten und eine hohe Treffsicherheit zu gewährleisten. Bei der Versuchsdurchführung musste von den Probanden das Wort „Schrobenhausen“ eingegeben werden wobei die dafür benötigte Zeit gemessen wurde. Es wurde bewusst ein längeres Wort gewählt, um Streuungen der Zeiten bei eventuellen Fehleingaben zu minimieren. Neben dem empirischen Indikator „Zeit“ wurde ebenfalls eine subjektive Bewertung mithilfe des semantischen Differentials durchgeführt. Die Ergebnisse dieser Bewertung werden für jede Eingabemethode detailliert aufgestellt und abschließend mit gemittelten Werten verglichen.

Zunächst wird das FishKey-System auf verschiedene Eigenschaften hin untersucht. Das Bedienprinzip war den Probanden praktisch ohne Erklärung sofort einleuchtend. Nach der Eingabe des Testworts wurde mehrmals angemerkt, dass sich die Darstellung zu viel „bewegt“ bzw. zu „unruhig“ ist, was zu einem Großteil von der noch unausgereiften Fingerdetektion verursacht wird. Das Profil in Tabelle 5.1 wurde mit dem Fragebogen ermittelt, wobei die eingetragenen Zahlen die Anzahl der Versuchspersonen repräsentieren, die das jeweilige Feld angekreuzt haben. Es ist zu erkennen, dass die Eignung während der Fahrt und vor allem die Blindbedienung als unzureichend bewertet wurden, die Treffsicherheit und die visuelle Anforderung bekamen ein mittleres Ergebnis. Die kognitive Beanspruchung, der Schwierigkeitsgrad und die Logik der Eingabe wurden als „gut“ oder „sehr gut“ eingestuft.

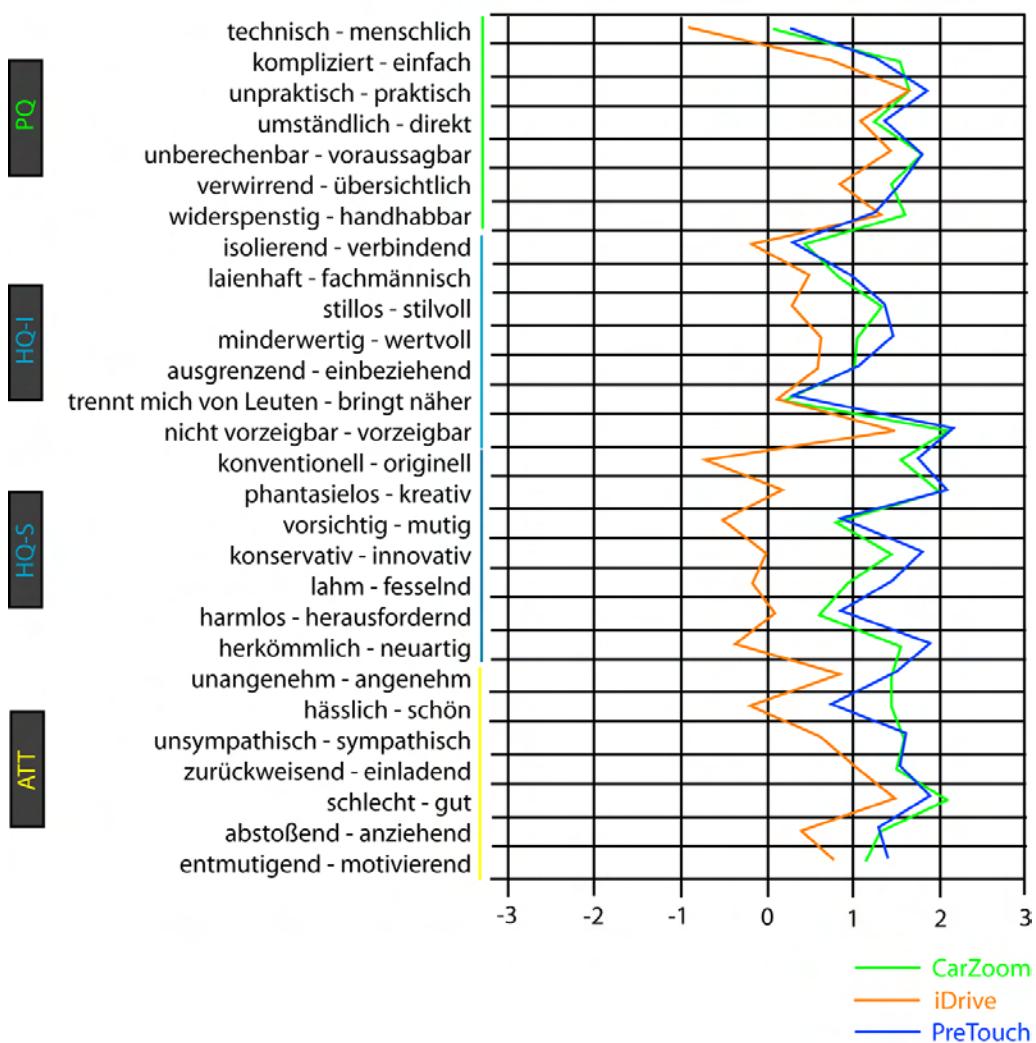


Abbildung 5.17: Gesamtprofil der AttrakDiff-Auswertung

Eigenschaft	1:gut	2	3	4	5:schlecht
Eignung während der Fahrt	1	5	7	5	2
Visuelle Anforderung	1	9	6	3	1
Blindbedienung	0	5	2	3	10
Kognitive Beanspruchung	0	12	7	1	2
Logik der Eingabetechnik	11	8	1	0	0
Schwierigkeitsgrad	2	10	8	0	0
Treffsicherheit	1	5	9	5	0

Tabelle 5.1: Profil FishKey - Texteingabe

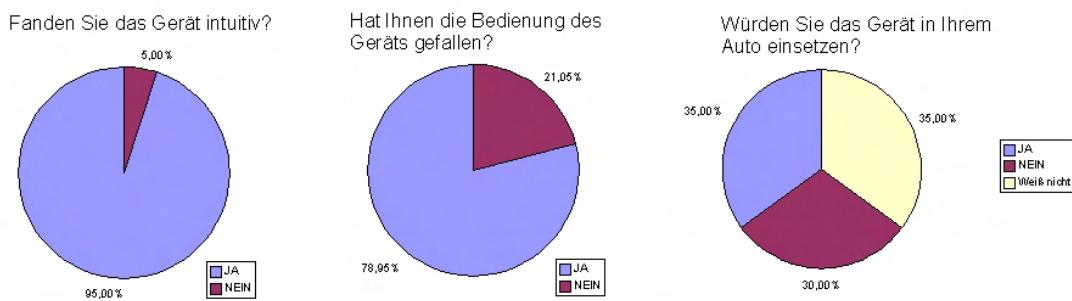


Abbildung 5.18: Allgemeine Fragen zur FishKey-Eingabe

Wie in Abbildung 5.18 erkennbar ist, wird diese Bildschirmtastatur als sehr intuitiv eingestuft, ca. 4/5 der Probanden gefiel die Bedienung und trotz der ungenauen Bedienung würden 35% diese Texteingabe im Fahrzeug nutzen wollen.

Anschließend wurde die Texteingabe mit der MatrixKey-Tastatur getestet. Die Eignung während der Fahrt, die visuelle Anforderung, Blindbedienung und die Treffsicherheit wurden hier durchwegs besser eingestuft als mit der FishKey - Tastatur. Im Gegensatz dazu wurden die Logik der Eingabe verschlechtert und der Schwierigkeitsgrad erhöht. Insgesamt deuteten die Kommentare darauf hin, dass die diskreten Zustände der Eingabe als zuverlässiger und angenehmer zu bedienen empfunden werden als die kontinuierlich angepasste Optik der FishKey - Tastatur. Auch diese Eingabe würde von einer besseren Sensorsauswertung profitieren, jedoch wäre in diesem Fall hauptsächlich eine größere Reichweite für die z-Achse vorteilhaft. Als größtes Problem bei der Bedienung stellte sich die sichere Auswahl der gewünschten Matrixregion heraus, da teilweise beim Überfahren anderer Regionen bei der Annäherung an die gesuchte die „falschen“ ebenso vergrößert wurden. Anschließend musste der Finger wieder entfernt werden und erneut die Eingabe gestartet werden. Da dies jedoch nicht zu falschen Zeicheneingaben führte, wurde dieses Fehlverhalten als nicht so sehr störend empfunden wie beispielsweise eine falsch getroffene Taste bei der FishKey - Tastatur.

Die nachfolgend beschriebenen Texteingaben wurden auf die gleiche Weise evaluiert und sollen Vergleichswerte für die eben vorgestellten liefern. Sie werden in anderen Applikationen eingesetzt und werden daher auch mit anderen Eingabegeräten bedient.

Eigenschaft	1:gut	2	3	4	5:schlecht
Eignung während der Fahrt	1	7	5	7	0
Visuelle Anforderung	2	9	7	2	0
Blindbedienung	1	3	5	5	6
Kognitive Beanspruchung	2	11	9	0	0
Logik der Eingabetechnik	7	10	1	2	0
Schwierigkeitsgrad	2	7	9	2	0
Treffsicherheit	3	5	10	1	1

Tabelle 5.2: Profil MatrixKey - Texteingabe (Touch)

Eigenschaft	1:gut	2	3	4	5:schlecht
Eignung während der Fahrt	4	6	3	5	1
Visuelle Anforderung	2	8	9	0	0
Blindbedienung	5	5	3	3	3
Kognitive Beanspruchung	1	10	4	4	0
Logik der Eingabetechnik	4	12	3	0	0
Schwierigkeitsgrad	0	11	8	0	0
Treffsicherheit	4	9	4	0	1

Tabelle 5.3: Profil MatrixKey - Texteingabe (Tastatur)

Zunächst soll die MatrixKey - Eingabe aus [9] untersucht werden. Sie basiert auf dem gleichen Grundprinzip, dass die gesamte Zeichenmenge auf 9 Matrixregionen verteilt wird, die ihrerseits wiederum in 9 Felder unterteilt sind. Die Bedienung geschieht in diesem Fall mit einer Tastenmatrix von ebenfalls 3 X 3 Feldern, ähnlich dem Nummernblock einer PC - Tastatur. Jede Taste korrespondiert dabei mit einer Region bzw. einem Feld in einer Region. Um also ein „B“ zu schreiben, muss zuerst die Taste links oben gedrückt werden, woraufhin diese Region vergrößert wird. Wird dann die Taste oben in der Mitte gedrückt, so wird ein „B“ eingegeben und die Region wieder verkleinert. Es kann also jedes Zeichen mit genau zwei Tastendrückungen ausgewählt werden. Dieses Vorgehen war den Probanden meist nach wenigen Buchstaben klar und es konnten schnell fehlerfreie Eingaben realisiert werden. Als Kommentar wurde häufig angemerkt, dass dieses System anfangs zwar gewöhnungsbedürftig sei, nach einer Einarbeitungsphase aber durchaus schnell und vor allem blind bedient werden könne. Die erzielten Ergebnisse aus den Versuchen sind in Tabelle 5.3 abzulesen.

## 6. Zusammenfassung



## A. Fragebogen Evaluierung



## B. Quellcode Sensorauswertung

```
#include "stdafx.h"
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <string.h>

//----- globale Variablen
const int resolution = 300;

double temp_x, temp_y, step;
double swp_x, swp_y; // Interpolierte Indizes
int sensor[32]; // Sensor Rohdaten
double sensor_tuned[32]; // Geglättete Sensorwerte ohne Offset
double sensor_tuned_alt[32];
int sensor_offset[32]; // Offset für jeden Sensor
double samples[resolution][16]; // Zweidimensionales Array für
// die Referenz-Samples
double korr_index_x[resolution]; // Korrelationsindizes x-Achse
double korr_index_y[resolution]; // Korrelationsindizes y-Achse
const char* raw_input; // Eingangsstring
char temp[3];
char* rest;
char output[30]; // Ausgangsstring
const char* out;
int sens_x, sens_y; // Interpolierte x- und y-Koordinaten
int i;
double weight;

double temp_z, a, b;
int x0, x1, x2, dx, dy, mx, my, max; // Interpolationsvariablen
double y_0, y_1, y_2;

int max_x = 0; // Indizes der Maxima und Minima der Korr.-Indizes
int max_y = 0;
int min_x = 0;
int min_y = 0;
```

```

double diff_korr_x = 0.0; // Differenzen der Korr.-Indizes
double diff_korr_y = 0.0;

double sensor_x_max = 0.0; // Minima und Maxima der Sensorwerte
double sensor_y_max = 0.0;
double sensor_x_min = 0.0;
double sensor_y_min = 0.0;

double diff_sensor_x = 0.0; // Differenzen der Sensorwerte
double diff_sensor_y = 0.0;

BOOL APIENTRY DllMain( HANDLE hModule,
                      DWORD ul_reason_for_call,
                      LPVOID lpReserved
)
{
switch (ul_reason_for_call)
{
case DLL_PROCESS_ATTACH: // Samples werden erzeugt...

{
step = 16.0 / resolution;

for(int i=0; i<resolution; i++)
{
for(int j=0; j<16; j++) //Glockenkurve
{
samples[i][j] = 30.0 * exp(-0.2 * (j-i*step)*(j-i*step));
}
}
}

case DLL_THREAD_ATTACH:
case DLL_THREAD_DETACH:
case DLL_PROCESS_DETACH:
break;
}
return TRUE;
}

//-----
// Suchen des Fingers
extern "C" __declspec(dllexport)
const char* calculate(const char* input){

// Variablen zurücksetzen
max_x = 0; // Indizes der Maxima und Minima der Korr.-Indizes
max_y = 0;

```

```
min_x = 0;
min_y = 0;

diff_korr_x = 0.0; // Differenzen der Korr.-Indizes
diff_korr_y = 0.0;

sensor_x_max = 0.0;
sensor_y_max = 0.0;
sensor_x_min = 0.0;
sensor_y_min = 0.0;

diff_sensor_x = 0.0; // Differenzen der Sensorwerte
diff_sensor_y = 0.0;

for(i=1; i<33; i++) // Messwerte aus String parsen
{
temp[0] = input[i*3];
temp[1] = input[i*3 + 1];
temp[2] = 0;
sensor[i-1] = (int)strtol(temp, &rest, 16);
}

for(i=0; i<32; i++) // Offset abziehen und alten Wert speichern,
// zeitliche Mittelung
{
sensor_tuned_alt[i] = sensor_tuned[i];

if(sensor[i] >= sensor_offset[i])
sensor_tuned[i]=0.5*(sensor_tuned_alt[i] + sensor[i] - sensor_offset[i]);
else
{
if(sensor[i] >= sensor_offset[i]-64)
sensor_tuned[i] = 0.0;

else
sensor_tuned[i]=0.5*(sensor_tuned_alt[i]+sensor[i]+(255-sensor_offset[i]));
}
}

//Glättung in beide Richtungen, um das Max. nicht zu verschieben
for(i=0; i<15; i++)
{
sensor_tuned[i] = (sensor_tuned[i] + sensor_tuned[i+1]) / 2;
sensor_tuned[i+16] = (sensor_tuned[i+16] + sensor_tuned[i+17]) / 2;
}

//Glättung in beide Richtungen, um das Max. nicht zu verschieben
```

```
for(i=15; i>0; i--)
{
    sensor_tuned[i] = (sensor_tuned[i] + sensor_tuned[i-1]) / 2;
    sensor_tuned[i+16] = (sensor_tuned[i+16] + sensor_tuned[i+15]) / 2;
}

for(i=0; i<16; i++) // Maximum der Messwerte suchen
{
    if(sensor_tuned[i] > sensor_x_max)
    {
        sensor_x_max = sensor_tuned[i];
        sens_x = i;
    }
    if(sensor_tuned[i+16] > sensor_y_max)
    {
        sensor_y_max = sensor_tuned[i+16];
        sens_y = i;
    }
}

sensor_x_min = sensor_x_max; // Minimum der Messwerte suchen
sensor_y_min = sensor_y_max;

for(i=0; i<16; i++)
{
    if(sensor_tuned[i] < sensor_x_min) sensor_x_min = sensor_tuned[i];
    if(sensor_tuned[i+16] < sensor_y_min) sensor_y_min = sensor_tuned[i+16];
}

diff_sensor_x = sensor_x_max - sensor_x_min;
diff_sensor_y = sensor_y_max - sensor_y_min;

// Berechnung des Abstands

temp_z = 60.0 - 10.0* sqrt((diff_sensor_x + diff_sensor_y)/2);
if(temp_z < 0) temp_z = 0;

//----- Kreuzkorrelation

// Korrelationsindizes erst auf 0 setzen
for(i=0; i<resolution; i++)
{
    korr_index_x[i]=0.0;
    korr_index_y[i]=0.0;
}

// Korrelationsindizes für x- und y-Achse berechnen
for(i=0; i<resolution; i++)
{
```

---

```

for(int j=0; j<16; j++)
{
    korr_index_x[i] += samples[i][j] * sensor_tuned[j];
    korr_index_y[i] += samples[i][j] * sensor_tuned[j+16];
}
}

// Maximum der Indizes finden
for(i=0; i< resolution; i++)
{
    if(korr_index_x[i] > korr_index_x[max_x]) max_x = i;
    if(korr_index_y[i] > korr_index_y[max_y]) max_y = i;
}

// Quadratische Interpolation
// für x-Achse
x0 = max_x-2; x1 = max_x; x2 = max_x+2;
if(x0 < 0) x0 = x2;
if(x2 >= resolution) x2 = x0;
y_0 = korr_index_x[x0]; y_1 = korr_index_x[x1]; y_2 = korr_index_x[x2];
a = (y_0 - 2*y_1 + y_2)/8;
b = (y_2 - y_0)/2 - 2*x1*a;
swp_x = -b / (2*a);
// für y-Achse
x0 = max_y-2; x1 = max_y; x2 = max_y+2;
if(x0 < 0) x0 = x2;
if(x2 >= resolution) x2 = x0;
y_0 = korr_index_y[x0]; y_1 = korr_index_y[x1]; y_2 = korr_index_y[x2];
a = (y_0 - 2*y_1 + y_2)/8;
b = (y_2 - y_0)/2 - 2*x1*a;
swp_y = -b / (2*a);

// Edge Gain und Offset dazurechnen
temp_x = 10 + resolution/2 + 1.20*(swp_x - resolution/2);
temp_y = 10 + resolution/2 + 1.20*(swp_y - resolution/2);

if(temp_x < 0) temp_x = 0.0;
if(temp_y < 0) temp_y = 0.0;

if((diff_sensor_x <= 3.0) || (diff_sensor_y <= 3.0)) temp_z = 100.0;

sprintf(output, "%05.1f,%05.1f,%05.1f", temp_x, temp_y, temp_z);

out = output;

return out;
}

//-----
extern "C" __declspec(dllexport)

```

```
void init(const char * input){  
  
    for(int i=1; i<33; i++) // Messwerte aus String parsen  
    {  
        temp[0] = input[i*3];  
        temp[1] = input[i*3 + 1];  
        temp[2] = 0;  
        sensor[i-1] = (int)strtol(temp, &rest, 16);  
        sensor_tuned[i-1] = 0.0;  
    }  
  
    for(int i=0; i<32; i++) // Ermittlung des Offsets  
    {  
        sensor_offset[i] = sensor[i];  
    }  
}
```

# Literatur

- [1] GOOD, L., STEFIK, M., BEDERSON, B.: *A Comparison of Zoomable User Interfaces and Folders For Grouping Visual Objects*. HCIL-2004-33, CS-TR-4641, UMIACS-TR-2004-83, November 2004
- [2] BENJAMIN B. BEDERSON, JON MEYER: *Implementing a Zooming User Interface: Experience Building Pad++*. J.Software: Practice and Experience, 1998
- [3] HAN, J. Y.: *Low-Cost Multi-Touch Sensing through Frustrated Total Internal Reflection*. Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology, 2005
- [4] BEDERSON, BEN, BOLTMAN, ANGELA: *Does Animation Help Users Build Mental Maps of Spatial Information*. Proceedings of InfoViz 1999, IEEE, Los Alamitos, CA, 28-35. HCIL-98-11, CS-TR-3964, UMIACS-TR-98-73
- [5] OVIATT, SHARON: *Handbook of Human-Computer Interaction*. (ed. by J. Jacko & A. Sears), Lawrence Erlbaum: New Jersey, 2002
- [6] MAGNUS INGMARSSON, DAVID DINKA, SHUMIN ZHAI: *TNT - A Numeric Keypad Based Text Input Method*. CHI 04,April 24-29, Vienna, Austria, 2004
- [7] TOSHIYUKI MASUI: *POBox - An Efficient Text Input Method for Pen-based Computers*. Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 1998), 1998, ACM press, pp. 328-335
- [8] JOHANNES WUST: *Datenmanagement in adaptiven Mensch-Maschine-Schnittstellen im Fahrzeug*. Interdisziplinäres Projekt, Lehrstuhl Mensch-Maschine-Kommunikation TU München, 2007
- [9] STEPHAN SCHNEIDER: *Datenorientiertes Zoomable User Interface im Automotive-Bereich*. Diplomarbeit, Lehrstuhl Mensch-Maschine-Kommunikation TU München, 2007
- [10] VLADIMIR I. LEVENSHTEIN: *Binary codes capable of correcting deletions, insertions, and reversals*. Doklady Akademii Nauk SSSR, 163(4) S. 845-848, 1965 (Russisch)
- [11] WOLFGANG DOSTAL, MARIO JECKLE, INGO MELZER, BARBARA ZENGLER: *Service-orientierte Architekturen mit Web Services*. Spektrum Akademischer Verlag, September 2005

- [12] D. A. FORSYTH AND J. PONCE: *Computer Vision: A modern Approach*. Pearson Education, 2003
- [13] A. CAPOBIANCO, N. CARBONELL: *Online help for the general public: specific design issues and recommendations*. Springer Verlag, 2003
- [14] HASSENZAHL, M., BURMESTER, M. UND BEU, A.: *Engineering Joy*. IEEE Software, 1 & 2, p. 70-76, 2001
- [15] MICHAEL BURMESTER, MARC HASSENZAHL UND FRANZ KOLLER: *Usability ist nicht alles - Wege zu attraktiven Produkten*. i-com, p. 32-40, 2002
- [16] HAGEN WOLF, ROLF M. ZÖLLNER, HEINER BUBB: *Ergonomischer Lösungsansatz für die gleichzeitige Rückmeldung mehrerer Fahrerassistenzsysteme an den Fahrer*. Lehrstuhl für Ergonomie, TU München, 2006
- [17] <http://www.multidmedia.com>  
Multidmedia Zinc Homepage
- [18] <http://www.adobe.com/products/flash>  
Adobe Flash Homepage
- [19] <http://www.hewlett-packard.com>  
Hewlett Packard Homepage
- [20] <http://www.apple.com/iphone/technology/>  
Apple iPhone Homepage
- [21] <http://www.bmw.com>  
BMW Homepage
- [22] <http://www.volkswagen.de>  
Volkswagen Homepage
- [23] <http://www.audi.de>  
Audi Homepage
- [24] <http://www.immersion.com>  
Immersion Homepage
- [25] <http://www.elotouch.com/Technologies/>  
Elo Homepage
- [26] <http://www.qprox.com>  
Quantum Research Homepage
- [27] <http://www.cypress.com>  
Cypress Semiconductor Homepage
- [28] <http://www.cs.umd.edu/hcil/piccolo/play/applet/presentation.shtml>  
University of Maryland, HCIL, Piccolo Presentation Tool Homepage
- [29] <http://www.plazalogic.com>  
PlazaLOGIC Homepage

- [30] <http://www.mozilla.org>  
Mozilla Homepage
- [31] <http://www.attrakdiff.de>  
AttrakDiff Homepage
- [32] <http://cs.nyu.edu/~jhan/ftirtouch/>  
New York University, Multi-Touch Interaction Research Homepage



# **Erklärung**

Hiermit erkläre ich, Florian Laquai, diese Diplomarbeit selbstständig angefertigt und keine Hilfsmittel außer denen in dieser Arbeit angegebenen verwendet zu haben.

München, den 8.7.2007

---