Universität Konstanz
FB Informatik und Informationswissenschaft
Bachelor-Studiengang Information Engineering

Bachelorarbeit

Sprachsteuerung als Basis multimodaler Interaktion auf grossen, hochauflösenden Displays

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

Studienfach: Information Engineering
Schwerpunkt: Human Computer Interaction
Themengebiet: Informationswissenschaft

von
Simon Fäh
(Matr. Nr. 01/600006)

Erstgutachter: Prof. Dr. Harald Reiterer
Zweitgutachter: Prof. Dr. Daniel A. Keim
Betreuer: Hans-Joachim Bieg (M.Sc)
Einreichung: 30.09.2008

Zusammenfassung

Diese Bachelorarbeit befasst sich mit multimodaler Interaktion für grosse, hochauflösende Displays. Dabei wird vor allem auf die Natürlichkeit der Interaktionskonzepte geachtet. Die Vision einer völlig freien, natürlichen Interaktion, ist unter anderem ausschlaggebend für die Wahl der Eingabegeräte. Das System soll mit den natürlichen Mitteln des Menschen, der Sprache und den Händen, bedient werden können. Das im Verlaufe dieser Arbeit entworfene System, das Nipper-System, kommt dieser Vision, dank der Verwendung von Spracherkennung, Handgestenerkennung und Laserpointer, schon sehr nahe. Die Wahl dieser Eingabegeräte wird jedoch nicht nur von dieser Vision motiviert, sondern wird auch sachlich belegt und die Vor- und Nachteile dieser Eingabegeräte werden aufgezeigt. Bei Nipper wird versucht, die Probleme dieser Interaktionskonzepte, mit Kombinationen bestehender und neuer Konzepte, zu lösen. Dazu werden zwei neue Konzepte vorgestellt, welche den Anforderungen der einzelnen Eingabegeräte, wie auch denjenigen von grossen, hochauflösenden Displays, gerecht werden sollen. Das WYSIWYS-Konzept (What you see is what you say), bietet nützliche Funktionen zur Verwendung der Spracherkennung und ermöglicht, mit einem speziell dafür entworfenen Menü, die multimodale Interaktion. Mit dem Input-Everywhere-Konzept wird die Verwendung der Zeigegeräte massiv vereinfacht, indem die ganze Fläche des Displays zur Interaktion benutzt werden kann. In einer abschliessenden Evaluation werden diese Konzepte getestet und Vorschläge zur Verbesserung und Erweiterung werden eingebracht.

Abstract

This bachelor thesis deals with large, high-resolution displays. It follows the vision, that interaction with such displays should be as loose and as natural as possible. Natural interaction without the need of any sensors or physical input devices is the key concept of this vision. To make a step towards this vision, the system built during this work, uses voice and hands as the main input devices. This system, called Nipper, is a multimodal system that uses speech recognition, freehand pointing and Laserpointer interaction to control its functionality. With these input devices, Nipper is not that far away from this vision. But the choice of those devices is not just based on this vision. Also other, more reliable reasons, to deal with this kind of input devices will be discussed. Nipper tries to improve the single input devices with smart combination of existing and new concepts presented in this thesis. Two new concepts to increase the efficiency of the system will be proposed during this thesis. The first concept, the "what you see is what you say-Concept" or "WYSIWYS-Concept" supports the multimodal interaction with a special regard to speech input. The second concept, called "Input-Everywhere-Concept", is a concept which supports the use of the pointing devices, which can be difficult to handle precisely on large, high-resolution displays. In combination with the speech recognition, it allows the use of the entire display surface for the interaction. There is no need for precise pointing on any interaction elements. This concept allows much faster input and reduces frustration. To complete this work, an evaluation was made, to prove the ideas and concepts introduced by Nipper. Problems and solution to solve those problems are also being presented in the last part of this thesis.

Inhaltsverzeichnis

1 Einleitung

Während die Preise für Displays immer weiter sinken, nehmen die Anzahl der Pixel und die Auflösung stetig zu. Daher werden zukünftig immer mehr Menschen Zugang zu grossen, hochauflösenden Displays haben (Vogel & Balakrishman 2005). Bei der Interaktion mit solchen Displays kann der Mensch sein gutes, aber dennoch limitiertes Sehvermögen optimal ausnutzen. Grössere Entfernung zum Display bietet einen Überblick über das gesamte Display, wobei die einzelnen Pixel nicht mehr unterschieden werden können. Interessante Details können durch physische Bewegung im Raum, in voller Auflösung und Schärfe, betrachtet werden. Diese Art der Interaktion erfordert Eingabegeräte und Konzepte, welche die benötigte Bewegungsfreiheit vor dem Display ermöglichen. Bei der Suche nach anderen Interaktionsmöglichkeiten wurde festgestellt, dass Benutzer häufig eine Vorliebe für multimodale Interaktion haben (Oviatt 2002). Konzepte zur multimodalen Interaktion legen besonderen Wert darauf, die menschliche Kommunikation in die Mensch-Computer-Interaktion zu übertragen. Die Hauptmerkmale der menschlichen Kommunikation sind Sprache, Gesten, Bewegungen und Blicke. Die Sprache als Hauptkommunikationsmittel des Menschen, steht als solche auch im Fokus dieser Arbeit. Allerdings wird die Sprache nicht als separates Eingabemittel sondern in Kombination mit anderen Eingabegeräten wie Laserpointer oder Handgesten betrachtet. Die Verwendung möglichst natürlicher Eingabegeräten soll der Vision einer Interaktion, frei von jeglichen Sensoren oder Eingabegeräten am Köper, näher kommen. Gerade für grosse Displays ist diese freie Art der Interaktion besonders anstrebenswert. Keine Sensoren, keine Mikrofone und auch keine anderen physischen Eingabegeräte die am Körper befestigt werden müssen, sollen dafür benötigt werden.

Nipper, das System welches im Rahmen dieser Arbeit entwickelt wurde, legt daher grossen Wert auf möglichst natürliche Eingabegeräte. Dieses multimodale System für grosse, hochauflösende Displays (LHRDs), verwendet daher Spracherkennung, Handgestenerkennung sowie einen Laserpointer zur Interaktion. Für einige Funktionen kann zusätzlich ein Eye-Tracker verwendet werden.

Die Anwendung, die im Nipper-System integriert ist heisst „NipMap“ und kann einfache grafische Objekte generieren und manipulieren. Nipper versucht Probleme von LHRDs und multimodaler Interaktion zu lösen und die Vorteile solcher Systeme aufzuzeigen. Dazu wird ein spezielles GUI-Konzept, das „What you see is what you say“-Konzept kurz „WYSIWYS“ vorgestellt, welches die Interaktion mit Sprachbefehlen und vor allem die Kombination von Sprache mit anderen Eingabegeräten unterstützen soll. Dieses Konzept beinhaltet ein adapti-

ves Menü, welches multimodale Eingabe ermöglicht und zugleich anzeigt, welche Optionen beim aktuellen Vorgang ausgeführt werden können. Das Menü ist so konzipiert, dass der Arbeitsfluss so wenig wie möglich unterbrochen wird. Ein weiteres Eingabekonzept, das „Input-Everywhere-Konzept“, erleichtert die Parametereingabe, indem das gesamte Display als Interaktionsfläche benutzt werden kann. Parametereingaben können wahlweise multimodal oder unimodal ausgeführt werden.

Im ersten Teil dieser Arbeit soll motiviert werden, warum diese grossen, hochauflösenden Displays überhaupt Sinn machen, welche Vor- und Nachteile die Spracherkennung hat und wie multimodale Interaktion am sinnvollsten eingesetzt werden kann.

Der zweite Teil befasst sich mit multimodalen Systemen, deren Konzepte und Eingabegeräte. Dabei werden bestehende Konzepte analysiert und daraufhin untersucht, wie einzelne Ideen und Konzepte miteinander kombiniert werden können.

Im dritten Teil werden die Konzepte und Funktionen von „Nipper“ vorgestellt. Dabei sollen die guten Eigenschaften bestehender Systeme vereint und eine robuste multimodale Interaktion für grosse, hochauflösende Displays ermöglicht werden (Siehe Abbildung 1).

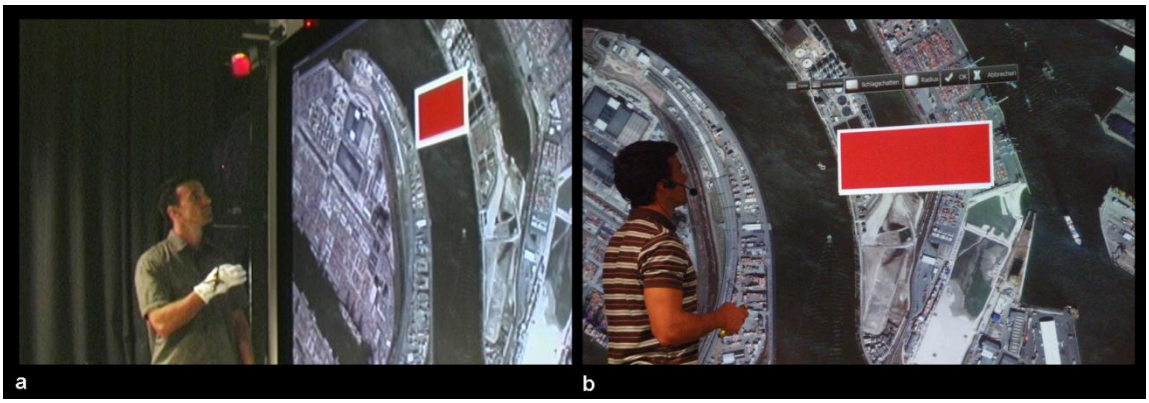

Abbildung 1: Ein Benutzer interagiert mit NipMap a.) mit Handgestenerkennung und b.) mit Laserpointer

Abschliessend wird Nipper in einer Evaluation mit sechs Versuchspersonen getestet. Dabei soll herausgefunden werden, ob die verwendeten Konzepte erstens gut und robust implementiert wurden und zweitens, ob diese auch intuitiv vom Benutzer angewendet werden können. Da an der Universität Konstanz bislang noch nicht mit Spracherkennung gearbeitet wurde, wird auch die verwendete Spracherkennung getestet. Bei diesem Test geht es nicht darum eine Genauigkeit der Spracherkennung angeben zu können, sondern zu sehen wie sich die Genauigkeit verhält, wenn unterschiedliche Benutzer die Spracherkennung verwenden. Im Zusammenhang mit der Evaluation werden Verbesserungs- und Erweiterungsvorschläge gemacht, die das System robuster und effizienter gestalten sollen.

2 Motivation

Der Computer hat sich seit 1960 erstaunlich schnell entwickelt. Bis heute scheint sich die Entwicklung integrierter Schaltkreise ans das Mooresche Gesetzt (Moore 1965) zu halten. Moore sprach von einer „Verdoppelung der Anzahl der Schaltkreiskomponenten auf einem Computerchip alle zwei Jahre“. Er wird damit wohl noch über das heutige Datum hinaus recht behalten. Moore selbst geht davon aus, dass seine Maxime noch ungefähr zehn Jahre bestehen wird (Martell 2007). Ähnliche Fortschritte sind auch in anderen Bereichen der Computertechnologie zu verzeichnen. Displays mit einer Ausgabegeschwindigkeit von zehn Zeichen pro Sekunde wurden durch schnelle Megapixel Displays abgelöst. Die 8 Zoll-Diskette mit 80 Kilobyte von 1971 wird heute durch BlueRay Discs mit bis zu 50 Gigabyte ersetzt[1]. Im Gegensatz dazu wurde diese Arbeit auf einer QWERTY-Tastatur geschrieben, deren Layout dem der 1870er Jahren entspricht.[2] Abgesehen von der Digitalisierung, einigen Funktionstasten und einem wesentlich leichteren Anschlag hat sich an der Tastatur nicht viel geändert. Nicht ganz so alt, aber ebenfalls in die Jahre gekommen und genauso alltäglich wie die Tastatur ist die Computer-Maus[3]. Seit der ersten Maus im Jahre 1963, hat sich auch daran im Wesentlichen nichts geändert. Die Form wurde ergonomischer und anstelle einer staubfangenden Kugel übernimmt heute eine Laserdiode die Positionsbestimmung. Das Konzept hat sich nicht verändert. Es ist durchaus erstaunlich, dass die Maus und die Tastatur nach wie vor die gängigsten Eingabegeräte sind. Zahlreiche moderne Computersysteme, mit unterschiedlichsten Ausgabegeräten, verlangen jedoch nach neuen Interaktionskonzepten und Eingabegeräten.

2.1 Grosse, hochauflösende Displays

Die Produktion von Displays wird immer günstiger und die Displays werden immer grösser. Schon bald „werden wir komplette Wände mit hochauflösendem visuellen Output“ haben (Vogel & Balakrishman 2005). Ursprünglich wurden solche Displays aus mehreren einzelnen Bildschirmen zusammengestellt. Heute werden, sofern möglich, Rückprojektionsdisplays verwendet. Verbesserung in Ausrichtung und Kalibration sorgen dafür, dass solche Displays kaum noch sichtbare Übergänge haben.

[1] Eine Übersicht über Datenträger ist im Internet verfügbar (Wikipedia, 2008).
[2] Patent: 207,559 (US) 27. August 1878 von Christopher Latham Sholes
[3] Dough Engelbart hat 1963 die erste Maus gebaut und 1970 ein Patent dafür angemeldet (Mifflin, 2000).

Grosse, hochauflösende Displays finden überall dort Verwendung, wo Überblick und Detail über komplexe Informationsräume gefragt sind. Teams von Wissenschaftlern und Entwicklern können sich mehrere Anwendungen oder Visualisierungen auf einem Bildschirm anzeigen lassen und vergleichen. Dabei ermöglicht die hohe Auflösung eine optimale Darstellung von Details und die grosse Fläche des Displays bietet dem Benutzer einen Überblick. Diese Art von Display bietet sich zur Analyse und Exploration von komplexen Informationsräumen an. Ball et al. (2005) vergleichen Such- und Navigationsaufgaben bei einer kartenbasierten Anwendung auf einem einzelnen Monitor mit derer auf einem 9-Monitor-Display. Dabei stellten sie fest, dass die Navigationsaufgaben bis zu doppelt so schnell ausgeführt wurden und 70% weniger Mausklicks benötigen. Ein 9-Monitor-Display, mit 17“ Monitoren aufgebaut, kann mit einer Breite von ca. 120cm und einer Höhe von ca. 95cm von einem Benutzer stationär überblickt werden. Bei Displays mit grösseren Abmessungen kann der Benutzer aufgrund seines limitierten Sehvermögens nicht mehr alles überblicken. Der Benutzer muss sich vor dem Display bewegen, um alle Details optisch erfassen zu können. Überblick und Detail können mit physischer Navigation selber kontrolliert werden. Je höher die Auflösung und je grösser das Display desto deutlicher wird dieser Unterschied. Nach Ni et al. (2006) bewältigen Benutzer „Navigations-, Such-, und Vergleichsaufgaben am effektivsten mit grossen, hochauflösenden Displays“. Shupp et al. (2006) sehen die Vorteile von LHRDs darin, dass „besseres Ausnützen des menschlichen Sehvermögens und natürliche, physische Navigation potentielle Frustrationen bei virtueller Navigation reduzieren können“.

Da in den unterschiedlichsten wissenschaftlichen und industriellen Bereichen vermehrt LHRDs als Ausgabegeräte verwendet werden, sind die Überlegungen zur Interaktion mit diesen durchaus gerechtfertigt (Yost & North 2006). Sobald grosse Datenmengen visualisiert werden müssen, bieten sich derartige Displays an. Die Universität Konstanz verfügt über ein solches Display. Die sogenannte „Powerwall“ hat eine Darstellungsfläche von 5.20m x 2.15m bei einer Auflösung von 4640 x 1920 Pixel. Abbildung 2 zeigt dieses Rückprojektionsdisplay welches mit acht Hochleistungsprojektoren betrieben wird (König et al. 2007b).

Die physische Navigation ist abhängig von der Bewegungsfreiheit des Benutzers vor dem Display. Diese Bewegungsfreiheit zieht unweigerlich die Frage nach neuen Interaktionskonzepten nach sich. Denn die Verwendung von Maus und Tastatur schränken die Bewegungsfreiheit des Benutzers aufgrund der notwendigen, stationären Auflagefläche massiv ein. Bewegungsfreiheit und flüssige Interaktion mit dem „interaktiven“ Display könnten beispielsweise durch Freihandgesten, Laserpointer, Touch-Display oder Spracherkennung ermöglicht

werden (Föhrenbach et al. 2008, König et al. 2007b, Shneiderman & Plaisant 2005). Die Gewährleistung der Bewegungsfreiheit ist eines der Hauptziele dieser Arbeit.

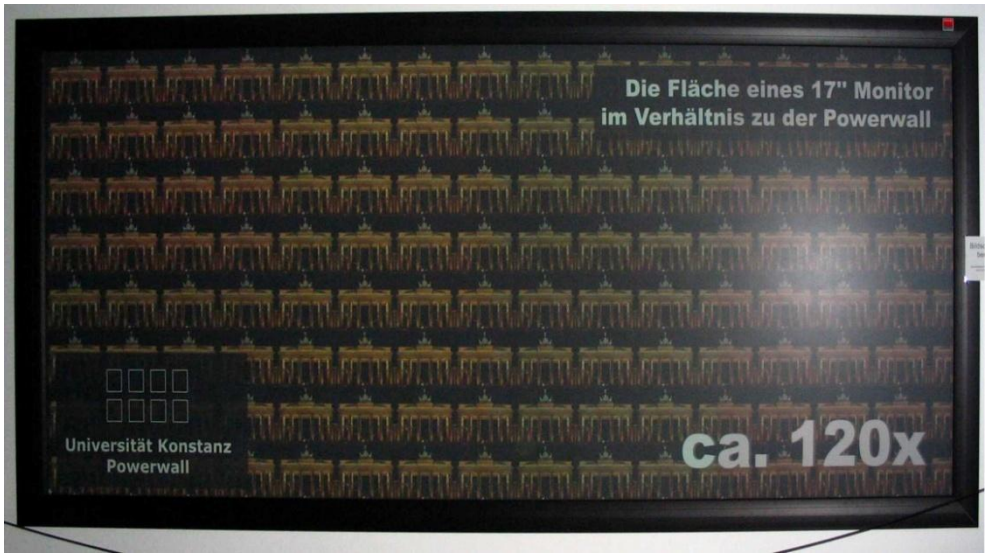

Abbildung 2: Powerwall der Universität Konstanz

Die Entwicklung neuer Eingabegeräte alleine hilft noch nicht im Umgang mit LHRDs. Auch die Konzepte der GUIs müssen an die Gegebenheiten von LHRDs angepasst werden. Mit herkömmlichen GUIs ist eine flüssige Interaktion auf LHRDs teilweise sehr schwierig. Sehr oft wird der Benutzer durch Dialogboxen, Popups oder neue Fenster im Arbeitsfluss unterbrochen (Guimbretière et al. 2001). Zudem werden diese auf LHRDs häufig nicht an optimaler Stelle angezeigt. Guimbretière et al. betonen auch die Wichtigkeit einfacher Menüstrukturen in kollaborativen Umgebungen, da dort der Benutzer nicht primär mit der Interaktion beschäftigt ist. Konversationen und Analysen mit anderen Benutzern sollen dabei im Vordergrund stehen. Popups oder Dialogboxen können daher schnell übersehen werden oder lenken von der eigentlichen Aufgabe ab. Je nach Auflösung des Displays und Entfernung des Benutzers zum Display sind bei herkömmlichen GUIs schlicht die Buttons und die Schrift zu klein um eine flüssige Interaktion zu ermöglichen. Die Entwicklung einer passenden GUI für LHRDs ist ein weiteres Ziel dieser Arbeit. Die Bewegungsfreiheit und GUIs werden daher im Folgenden als zentrale Aspekte der Interaktion mit LHRDs aufgegriffen und diskutiert.

2.2 Warum Spracherkennung

Die oben erwähnte Bewegungsfreiheit, kann unter anderem durch die Verwendung von Spracherkennung gewährleistet werden. Mit modernen Funkmikrofonen oder anderen Aufnahmetechniken, können Stimmen so aufgezeichnet werden, dass die Bewegungsfreiheit uneingeschränkt bleibt. Diese Bewegungsfreiheit ist ein Teil der Vision der völlig freien Interaktion. Für deren Umsetzung bietet sich die Spracherkennung an, da Sprachbefehle auch über eine gewisse Distanz aufgezeichnet werden können und somit keine Mikrofone am Körper benötigt werden (Faeh 2008). Aus Gründen der Machbarkeit wird für diese Arbeit ein Funkmikrofon eingesetzt. Im Folgenden werden noch andere, belegbare Gründe für den Einsatz der Spracherkennung diskutiert.

Menschen kommunizieren mit anderen Menschen auf unterschiedlichste Art und Weise. Körpersprache, Gesten, Text, Bilder, Zeichnungen und die Sprache gehören zu den gängigsten Kommunikationsmitteln. Die Sprache ist nach Leong (2005) das „Hauptkommunikationsmittel des gesunden Menschen mit anderen Menschen“. Desweiteren ist die Sprache ein sehr effizienter Weg um Ideen und Wünsche auszudrücken. Es ist daher nicht verwunderlich, dass der Mensch schon immer das Bedürfnis hatte, Maschinen mit der Sprache kontrollieren zu können. In der Realität existiert dieses Paradigma schon seit Jahrhunderten. Als anstelle von Maschinen noch Tiere benutzt wurden, waren Sprachbefehle das gängige Mittel den Tieren Befehle zu erteilen (Schafer 1995). Diese Art der Kommunikation mit den Tieren ermöglicht es beispielsweise dem Bauern mit beiden Händen den Pflug zu lenken. Nach Paget (1930) ist die „Erfindung“ der Sprache nicht darauf zurück zu führen, dass Menschen ihre Gefühle ausdrücken wollten, sondern auf „die Schwierigkeit sich mit vollen Händen mittels Körpersprache ausdrücken zu können“. Bei der Interaktion mit einem Computer wird in der Regel nicht gepflügt, aber es kann vorkommen, dass die Hände für etwas anderes benötigt werden. So werden zum Beispiel in modernen Landwirtschaftsmaschinen die Arbeitsvorgänge im Bordcomputer protokoliert, in Autos werden Navigationssysteme bedient und in Flugzeugen gibt es noch weit aus mehr Systeme die bedient werden müssen. Dabei kann es fatale Folgen haben, wenn der Fahrer oder Pilot die Hände für die Eingabe im Computer benötigt. Nach Shneiderman und Plaisant (2005) kann Spracherkennung in folgenden Situationen sinnvoll angewandt werden:

- Der Benutzer ist Sehbehindert
- Der Benutzer benötigt seine Hände für andere Aufgaben
- Der Benutzer muss sich frei bewegen können

- Die Augen des Benutzers sind beschäftigt
- Die Umstände lassen die Verwendung von Maus und Keyboard nicht zu

Für Schafer (1995) beinhalten die „offensichtlichen Vorteile“ der Spracherkennung folgende Punkte:

- Sprache ist die natürliche Art der Kommunikation für Menschen
- Sprachsteuerung ist besonders attraktiv, wenn die Hände oder Augen des Benutzers anderweitig beschäftigt sind
- Zwei-Wege Sprachkommunikation ist ein effektiver Weg um telefonisch mit einer Maschine zu interagieren, welche sprechen, zuhören und verstehen kann.

Bei der Interaktion mit LHRDs treffen in den meisten Fällen nicht alle der oben genannten Punkte zu, dafür wird hier erneut die Bewegungsfreiheit als wichtiger Punkt angesprochen.

Um die „Vorzüge des allgegenwärtigen Computers geniessen zu können“, werden neue, bessere Benutzerschnittstellen und Paradigmen benötigt (Weiser 1993). Sprachbasierte Benutzerschnittstellen sind eine Möglichkeit diese Anforderung zu erfüllen. Denn nach Cohen (1995) sind „Sprachbasierte Benutzerschnittstellen in manchen Situationen zweckdienlicher als grafische Benutzerschnittstellen (GUIs) und werden vermutlich in naher Zukunft im alltäglichen Gebrauch auftreten“. Auch Snider (1997) sieht ein Defizit bei den aktuellen GUIs, da diese „die natürlichen Kommunikationsmittel des Menschen nicht unterstützen“.

Der Traum des sprechenden Computers ist fast so alt wie der Computer selbst. 1968 setzte Arthur C. Clark mit dem HAL 9000 Computer neue Massstäbe, allerdings nur was die Fantasie anging. Der HAL 9000 stammt aus dem Buch und Film „2001: A Space Odyssey“. Kaum ein Sciencefiction Film kam seither ohne sprachgesteuerten Computer aus. Ganz so einfach wie Clarke’s HAL 9000 ist die Spracherkennung leider nicht. Probleme treten aus Sicht des Computers, aber auch aus Sicht des Menschen auf. „Sprachbefehle beanspruchen das Gedächtnis stärker als Hand- oder Augenkoordination, was die Aufgabe des Benutzers durchaus beeinträchtigen kann. Sprache benötigt ein grösseres Mass an Ressourcen, während Hand- und Augenkoordination parallel zu Denkaufgaben ausgeführt werden können.“ (Shneiderman & Plaisant 2005)

Sprache ist trotz oder gerade wegen ihrer Natürlichkeit nicht unproblematisch. Hintergrundgeräusche oder Personen mit unterschiedlichem Dialekt können nach wie vor Gründe für falsche Erkennungen sein. Auch die Mächtigkeit der Sprache hat Vor- und Nachteile. Zum einen

können unzählige Befehle und Befehlskombinationen mit der Sprache erzeugt, werden zum anderen müssen die Spracherkennungssysteme diese unzähligen Befehle auch erkennen können. Es zeigt sich somit, dass einige der grössten Vorteile der Sprache auch gleichzeitig Probleme mit sich bringen. Trotzdem gilt für White (1976), dass „auch mit den besten High-Tech Hilfsmitteln für die Kommunikation die Sprache konkurrenzlos die schnellste und angenehmste Möglichkeit für Menschen zur interaktiven Kommunikation bleibt". Ähnliche Aussagen trifft auch Cohen (1998) und fügen hinzu, dass Sprache die beste Modalität zur interaktiven Problembehandlungen sei. Mit der Natürlichkeit und der Bewegungsfreiheit erfüllt die Spracherkennung zwei wichtige Punkte für die Interaktion mit LHRDs. Zusätzliche Aspekte der Spracherkennung, die ebenfalls für LHRDs positiv eingesetzt werden können, werden im Zusammenhang der multimodalen Interaktion im folgenden Kapitel aufgezeigt.

2.3 Multimodale Interaktion

Die Spracherkennung ist als alleinstehendes Eingabegerät nicht für die Interaktion mit LHRDs geeignet. Da vor allem Positionsangaben nur schwer via Sprache vorgenommen werden können, sollte diese mit zusätzlichen Eingabegeräten kombiniert werden (Oviatt 1999a). Multimodale Interaktion folgt dem „mehr ist mehr" Prinzip, um das Anwendungserlebnis des Benutzers zu verbessern (Bouchet & Nigay 2004). Bouchet und Nigay versuchen dies zu erreichen, indem sie die Modalitäten um Informationen zu „kontrollieren" und zu „erfahren" durch die Verwendung von zeigen, sehen, berühren und sprechen zu erweitern. Solche multimodale Systeme bieten dem Benutzer mehrere, kombinierte Eingabemöglichkeiten wie beispielsweise mit Stift, Hand, Auge, Körper oder Sprache an. Häufig werden dabei zwei Modalitäten kombiniert und es gibt sogar Systeme die mehrere Modalitäten kombinieren (Cohen et al. 1997). Diese moderne Art der Interaktion repräsentiert „eine neue Richtung in der Mensch-Computer-Interaktion (HCI), welche sich von dem konventionellen WIMP (Windows, Icons, Menus, Pointing) Interface entfernt" (Oviatt 2002).

Es gibt unterschiedliche Klassen von multimodalen Systemen. Es gibt Systeme, die Sprache und manuellen Input verarbeiten und diese zu jeweils unterschiedlichen Interaktionen verwenden. Solche Modalitäten werden auch als „Active Input Modes" bezeichnet. Aktiv, da der Benutzer diese bewusst zur Kontrolle des Systems einsetzt. Dann gibt es Systeme, welche aus Sicht des Benutzers gleiche Modalität über mehrere Kanäle zu erkennen versuchen. So zum Beispiel bei der audio-visuellen Spracherkennung. Da werden die akustischen Informationen der Sprache mit den visuellen Informationen, dazu gehören hauptsächlich die Bewegun-

gen der Lippen, für eine robustere Erkennung kombiniert (Geramisos et al. 2003). Für den Benutzer ergeben sich dadurch aber keine weiteren Interaktionsmöglichkeiten, daher werden solche Modalitäten als „Passiv Input Modes“ bezeichnet. Zu guter Letzt gibt es multimodale Systeme, bei denen auf die gleichen Funktionen mit mehreren Modalitäten zugegriffen werden können. Solche Systeme bieten dem Benutzer an, die Eingabemodalität zu benützen mit der sich die Aufgabe in der aktuellen Situation am effizientesten erledigen lässt.

Wie von Oviatt (1997) beschrieben, haben Benutzer starke Vorlieben für multimodale Interaktion. Sie zeigt in einem Test, dass bei räumlichen Navigations-, Such- und Vergleichsaufgaben 95% bis 100% der Benutzer die multimodale Interaktion bevorzugen. Im Vergleich dazu schneiden multimodale Interaktionen bei allgemeinen Aufgaben ohne räumlichen Bezug schlecht ab. In 99% der Fälle werden solche Aktionen unimodal durchgeführt. Desweiteren ist zu beachten, das in 20% der Fälle effektiv multimodal interagiert wird (Oviatt et al. 1997). In den restlichen Fällen bedient sich der Benutzer einer einzelnen Modalität, allerdings nicht immer der Gleichen. Diese Tatsachen weisen auf die Unterschiede zwischen simultanen, multimodalen Eingaben und zeitlich unabhängigen, multimodalen Eingaben hin.

Es gibt Systeme, die erwarten simultane Sprach- und Gesteneingabe. Als Beispiele seien hier QuickSet[4] (Cohen et al. 1997) und Media Room (Bolt 1980) genannt. Nach Oviatt treten Sprach und deiktische Gesten jedoch bei weniger als 25% der Benutzer simultan auf (Oviatt et al. 1997). Wobei diese Zahlen ethnische Abhängigkeiten aufweisen. Chinesisch ist als Sprache beispielsweise anders aufgebaut als Englisch oder Deutsch. Daher ist es sehr wahrscheinlich, dass deiktische Gesten im chinesischen Sprachraum stärker gewichtet werden können (McNeill 1992). Zusammenfassend sieht Oviatt „Sprache und Gesten“ als „streng zusammenhängend und synchron in der multimodalen Interaktion, wobei synchron nicht mit simultan gleichzusetzen ist“ (Oviatt 1999b).

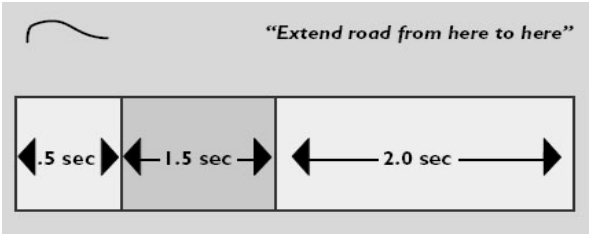

Abbildung 3: Verzögerung zwischen Stift- und Spracheingabe bei multimodaler Interaktion (Oviatt 1999b)

[4] QuickSet ist ein kollaboratives, multimodales System zur Interaktion mit verteilten Anwendungen (Cohen et al. 1997).

Abbildung 3 zeigt diesen Zusammenhang anhand eines Beispiels mit Stift und Sprache. Mit dem Stift soll ein Pfad gezeichnet werden und über die Sprache wird dem System mitgeteilt, eine Strasse entlang des Pfades zu erstellen. Dabei wird ersichtlich, dass die beiden Interaktionen nicht nur nicht simultan sind, es wird sogar eine Pause von 1.5 Sekunden dazwischen angezeigt (Oviatt et al. 1997).

Multimodale Systeme, aber auch sprachgesteuerte Systeme preisen häufig die „Natürlichkeit der Interaktion an". Der Computer ist aber an und für sich nichts Natürliches. Die Natürlichkeit muss sich jedoch nicht direkt auf die Interaktion beziehen, sondern geht häufig von den natürlichen Kommunikationsmitteln des Menschen aus. Nur weil die Sprache ein natürliches Kommunikationsmittel ist, bedeutet dies nicht, dass die Sprachbefehle einem menschlichen Dialog ähnlich sein müssen. Gerade bei multimodaler Interaktion wurde festgestellt, dass die „multimodale Sprache kürzer, syntaktisch einfacher und weniger flüssig ausfällt als bei unimodalen sprachgesteuerten Systemen" (Oviatt 1997). Die folgende Beispielsaufgabe kann unimodal, nur mit Sprache oder multimodal, mit Sprache und Gesten, gelöst werden. Beim unimodalen System könnte der Sprachbefehl wie folgt lauten: „Platziere ein Bootssteg östlich von der Mainau". Beim multimodalen System hingegen zeichnet der Benutzer mit einem Stift oder mit der Hand den Bootssteg östlich von der Mainau ein und gibt den Sprachbefehl „Bootssteg einfügen" oder sogar nur „Bootssteg". Beim multimodalen System wird neben der Position auch die Grösse und Ausrichtung des Steges definiert und zugleich wird die Komplexität der Sprache eliminiert. Dieses Beispiel zeigt auch, dass Benutzer „komplizierte und fehleranfällige Positionsangaben nicht bevorzugt über die Sprache tätigen" (Oviatt 1999b). „Sprachliche Umwege" und „umschreibende Ausdrücke" werden in multimodalen Systemen sehr selten verwendet (Oviatt & Kuhn 1997).

Bei der Evaluation von QuickSet (Cohen et al. 2000) konnte gezeigt werden, dass erfahrene Benutzer von QuickSet mit dem multimodalen Interface nach nur kurzer Trainingszeit bis zu sieben mal schneller waren als mit der normalerweise verwendeten Maus-Menu-GUI. Im Durchschnitt wurde die Zeit, die für eine Aufgabe benötigt wurde, um das 3.5 fache reduziert. Fehler wurden 4.3mal schneller korrigiert.

Die Resultate aus Tabelle 1 zeigen, dass multimodale Interaktion (MM) wesentlich schneller sein kann als normale GUIs (GUI). Chapanis et al. (1972) zeigten schon früher, dass unterschiedliche Aufgaben bei der Kommunikation zwischen zwei Menschen via Sprache (Telefon) 2-3mal schneller erledig werden als via Keyboard.

Tabelle 1: Durchschnittliche Zeit in Sekunden für unterschiedliche Aufgaben (Cohen et al. 2000)

Expert Subject	Create Units		Create Control Measurses		Repair Errors	
	MM	GUI	MM	GUI	MM	GUI
S1	8.4	25.6	6.5	27.5	12.9	49.3
S2	6	14.4	5.2	19	7.7	30
S3	6.3	27.2	11	24.3	11.6	56.1
S4	4	18.5	4	17.7	6.3	23
Means	6.2	21.4	6.7	22.1	9.6	39.6

Neben der Effizienzsteigerung gibt es noch weitere nennenswerte Vorteile multimodaler Systeme. Multimodale Interaktion kann Fehler und stockende Sprache um 36-50% reduzieren und dem Benutzer mehr Flexibilität bieten (Oviatt 1997). Erkennungsbasierte Eingabegeräte wie Sprache oder Gesten sind nach wie vor fehleranfällig. Der Vorteil multimodaler Anwendungen ist, dass Fehler durch die Verwendung der zweiten Modalität erkannt und korrigiert werden können. Wie oben erwähnt, wird die Sprache vereinfacht und ist daher weniger fehleranfällig. Treten trotzdem Fehler auf, können diese durch weitere oder parallele Modalitäten korrigiert werden. Nach Oviatt können sich in „gut gestalteten und optimierten multimodalen Anwendungen zwei Eingangssignale gegenseitig erklären“ (Oviatt 1999a). Verwendet ein Benutzer beispielsweise den Sprachbefehl „Schwenken“ und zeichnet einen Pfeil in die Richtung in die er schwenken möchte, so kann es sein, dass weder der Sprachbefehl noch die Geste korrekt erkannt wurden. Sofern das Erkennungssystem über eine Liste mit den „N-besten“ Treffern verfügt, können weitere Ergebnisse verglichen werden. Ergeben die beiden ersten Befehle der Listen der Spracherkennung und der Gestenerkennung keine vom System unterstützte Aktion, werden die folgenden Befehle in den Listen verglichen (Cohen et al. 1997). Findet das System ein Sprachbefehl und eine Geste, die zusammen verwendet werden können, so führt das System den kombinierten Befehl aus.

Die Flexibilität multimodaler Systeme erlaubt es dem Benutzer Modalitäten zu wechseln, wenn eine Situation einen solchen Wechsel erfordert, respektive dieser zur Effizienzsteigerung beitragen kann. Neben der Effizienzsteigerung und der Verwendung natürlicher Eingabemodalitäten ist die Steigerung des Nutzungserlebnisses beim Benutzer ein weiterer positiver Aspekt multimodaler Interaktion.

3 Ein Konzept zur multimodale Interaktion

Es gibt unterschiedliche Konzepte zur multimodalen Interaktion, wovon viele mit Spracherkennung kombiniert werden. Für die Interaktion mit LHRDs scheint sich vor allem das „Speak-and-Point"-Konzept zu eignen. Dabei werden Sprachbefehle mit Zeigegesten kombiniert. Im Folgenden wird zuerst der Media Room von Bolt, das erste System dieser Art, danach werden einzelne Komponenten für ein solches System vorgestellt.

3.1 Media Room

Der Media Room von Bolt (1980) hatte die Grösse eines Büros und war mit einem handelsüblichen Vinyl-Sessel, einem grossen und zwei kleinen Displays ausgestattet. Der Ledersessel wurde an den Armlehnen durch einen drucksensitiven Joystick und ein kleines Touchpad ergänzt. Die beiden kleinen Displays wurden ebenfalls mit einer druckempfindlichen Folie überzogen und konnten somit die mit dem Finger angetippten Koordinaten ermitteln. Das grosse Display war ein ca. 2x2m grosses, farbiges Rückprojektionsdisplay (Vgl. Abb. 4).

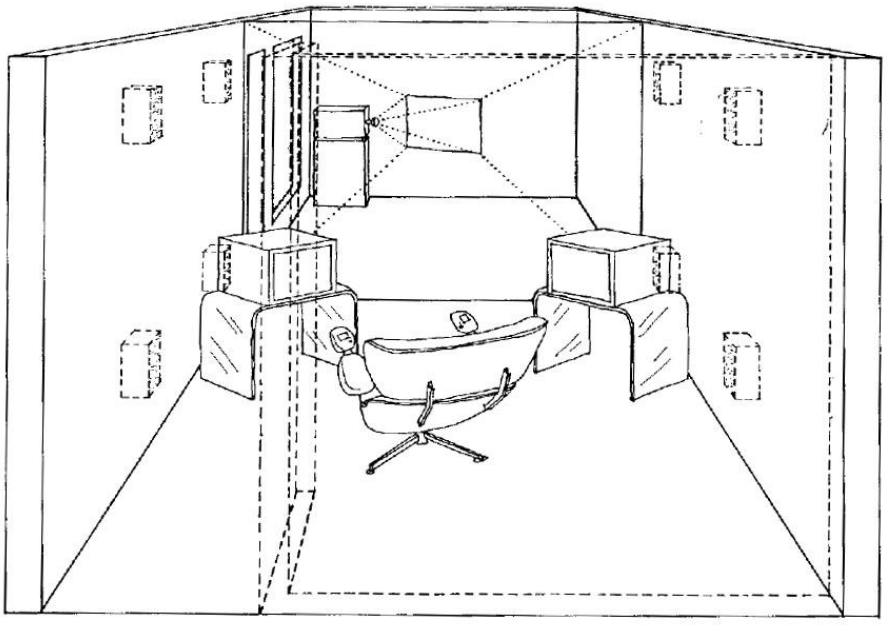

Abbildung 4: Bolt's Media Room (Bolt 1980)

Bolt verwendete das „DP-100" Connected Speech Recognition System (CSRS) von NEC. Ein System, welches im Vergleich zu anderen Systemen nicht nur einzelne Wörter, sondern kontinuierliche Sätze mit einer Länge von bis zu 5 Wörtern erkennen konnte. Das System konnte Sätze aus 120 unterschiedlichen Wörtern erkennen. Bei einzelnen, nicht zusammenhängenden Wörtern sogar an die 1000 Einzelwörter. Das System musste von jedem Benutzer separat

trainiert und kalibriert werden. Dabei mussten Zahlen von Eins bis Zehn je zweimal, alle anderen Wörter je einmal vorgesprochen werden.

Informationen über die räumliche Orientierung lieferte das ROPAMS (Remote Object Position Attitude Measurement System). Dabei wurden Drahtspulen und deren Magnetfelder für die Positionsbestimmung verwendet. Je drei Drahtspulen wurden in einem Kunststoffwürfel in X, Y und Z-Richtung montiert. Dabei diente ein Würfel als Sender und einer als Empfänger. Der Sender, der nicht ohne Kabel auskam war ungefähr 2 cm gross und konnte auf der Hand oder einem Fingerring montiert werden. Die Distanz des Senders zum Empfänger wurde aufgrund der abfallenden Feldstärke berechnet. Da der Benutzer im oben genannten Sessel sass, war der Aktionsradius relativ klein und konnte so genügend genau bestimmt werden.

Das Demonstrationssystem hatte eine einfache Karte als Hintergrundbild, welche zu Beginn leer war. Der Benutzer konnte nun einfache Formen wie Kreise, Quadrate oder Rauten erstellen, welche die Attribute Farbe und Grösse hatten. Eingefügt wurden diese Objekte via Spracherkennung.

„Create a blue square there“

Mit diesem Befehl konnte ein blaues Quadrat erstellt werden. Da keine Grösse angegeben worden ist, wurde die Standardgrösse „medium“ verwendet. Grösse und Farbe wurden im Vokabular vordefiniert. Dabei standen die drei Grössen „large“, „medium“ und „small“ zu Verfügung. Das „there“ bezog sich auf die Position der Hand die von ROPAMS ermittelt wurde.

Objekte konnten via Sprachbefehl verschoben werden.

„Move the blue triangle to the right of the green square“

Dieser Befehl funktioniert jedoch nur wenn genau ein blaues Dreieck und genau ein grünes Quadrat existieren. Die Position von „to the right“ wurde im System definiert.

“Move that to the right of the green square“

Im Vergleich zum vorherigen Befehl konnte der Benutzer anstelle des Befehls „blue triangle“ auch deiktische Gesten verwenden um das Objekt zu verschieben. Dabei bezog sich „that“ auf die Position der Hand. Dieser multimodale Ansatz ist weniger restriktiv und lässt genauere Positionierungen zu.

“Put that there”

Dieser Befehl ist mit Sicherheit der berühmteste von den hier aufgelisteten. Dabei nehmen die Befehle „that“ und „there“ Bezug auf die Position der Hand. In Abbildung 5 sind die zwei Momentaufnahmen von „that“ und „there“ überlagert dargestellt. Es ist zu beachten, dass dieser Befehl nicht fliessend ausgesprochen wird, da sich „that“ auf das auszuwählende Objekt bezieht und „there“ auf die zu verschiebende Position. Zwischen „that“ und „there“ muss der Benutzer also zuerst die Position der Hand verändern. Oviatt kritisiert, dass dieses „speak-and-point Muster nur 14% aller spontanen multimodalen Äusserungen ausmacht“ (Oviatt et al. 1997).

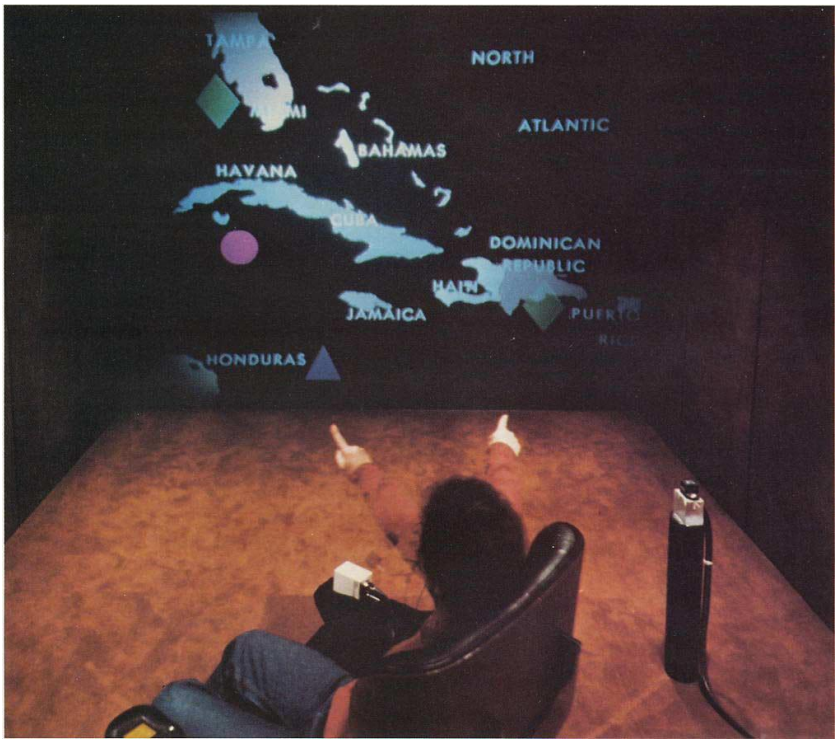

Abbildung 5: Zwei überlagerte Bilder mit der Geste für „That“ und „There“ (Bolt 1980)

Weitere Befehle, wie “Make that smaller” oder “Make that like that” vermochten den Funktionsumfang noch erweitern. Mit diesen und ähnlichen Befehlen war es möglich Objekte nachträglich zu verändern. Hatte das mit „that“ und Geste selektierte Objekt die der Grösse „large“ wurde es durch den Befehl „smaller“ auf die Grösse „medium“ verkleinert.

Die Kombination von deiktischen Gesten und Sprache, die Verwendung von Touchscreens zur Positionierung des Kartenausschnitts auf dem grossen Display sowie Joystick und Touchpad für weitere Funktionen des Systems machten den Media Room zu einem multimodalen System mit vielen Eingabemöglichkeiten. Bolt hat damit viele Wissenschaftler zur Nachahmung und Weiterentwicklung dieser und ähnlicher Ansätze ermutigt.

In seinem System konnte Bolt keine kontinuierlichen Manipulationen an den Objekten vornehmen. Für Grösse und Farbe gab es lediglich ein paar wenige, diskrete Merkmale. Damit werden die Möglichkeiten des Benutzers zur Interaktion stark eingeschränkt. Es wäre wünschenswert, wenn bei solchen Systemen alle Parameter stetig und diskret eingegeben werden könnten.

Das Konzept von Nipper ist in seinen Grundzügen dem Media Room von Bolt sehr ähnlich. Auch Nipper verwendet die Kombination von Sprachbefehlen mit einem Zeigegerät. Daher wird im Folgenden zuerst auf die Spracherkennung danach auf einzelne Zeigegeräte eingegangen.

3.2 Spracherkennung

3.2.1 Entwicklung

Die Spracherkennung ist ein wichtiger Teil der multimodalen Interaktion, insbesondere bei einem „Speak-and-Point"-Konzept. Seit dem Media Room gab es viele Systeme die versucht haben Sprache als Eingabemittel zu verwenden. Dabei muss beachtet werden, dass die Spracherkennung in den letzten Jahren deutliche Fortschritte gemacht hat. Bei der Beurteilung von Konzepten sollte dieser Aspekt stets berücksichtigt werden.

Die Spracherkennung hatte ihre Anfänge in den 1960er Jahren. Bereits damals gab es Anstrengungen Sprachsignale so zu verarbeiten, dass sie von einem Computer verstanden werden konnte. Das US Verteidigungsdepartment (DARPA) rief 1971 das "Speech Understanding Research" Programm (SUR) ins Leben. Erklärtes Ziel war es, einen Computer zu entwickeln der kontinuierliche[5] Sprache erkennen konnte. „Lawrence Roberts wollte ein System, welches ein Vokabular von 10'000 englischen, gesprochenen Wörter von einer beliebigen Person erkennen sollte. Allen Newell reduzierte das Ziel jedoch auf 1'000 Wörter in ruhiger Umgebung, limitierte Anzahl Personen und ein restriktives Vokabular" (Newell et al. 1971).

Beim SUR gab es vier Etappen der Erkennung: Merkmalsextraktion, Segmentierung, Erkennung und Wortvergleich. Bei der Merkmalsextraktion wurde das Sprachsignal 20'000 mal pro Sekunde gesampelt und digitalisiert. Alle 10 Millisekunden extrahierte das System eine gewisse Anzahl Parameter von dem digitalisierten Signal. Mit linearen Vorhersagealgorithmen wurden die Formanten extrahiert. Diese wurden dann segmentiert und zu phonetischen Seg-

[5] Bei der Spracherkennung wird unterschieden zwischen Befehlen die nur ein Wort beinhalten und kontinuierlicher Sprache.

menten zusammengefügt. Diese Segmente wurden in unterschiedliche phonetische Klassen unterteilt und dann aufgrund statistischer Klassifikatoren zu einem Phonem zusammengefügt. Diese Phoneme wurden dann mit den zugelassenen Wörtern in der Grammatik verglichen und das Wort bei dem die Wahrscheinlichkeiten der einzelnen Phoneme am besten zutraf wurde als erkannt gewertet. Mit diesen Verfahren konnte ein Vokabular von 100 bis 1000 Wörter verarbeitet werden (Juang & Rabiner 2004).

Um Schwankungen in der Tonlage des Sprechers zu verhindern und somit die Genauigkeit der Spracherkennung zu verbessern entwickelte Makhoul (1976) „Helium Speech". Um der Taucherkrankheit vorzubeugen, atmen Taucher mit Helium angereicherte Luft. Der resultierende „Donald Duck Effekt" in der Stimme hat zur Folge, dass sich die Tonhöhe nicht mehr ändert. Dieser Effekt hat zur Folge, dass sich alle Stimmen in einem ähnlichen Frequenzbereich befinden. Damit wollte Makhoul die Erkennungsgenauigkeit verbessern. Ähnliche Konzepte versuchten unter Verwendung von Sprachmodifikationssystemen zu erreichen, dass die Stimmen unterschiedlicher Benutzer akustisch möglichst gleich klingen. Solche Versuche brachten aber keine markanten Verbesserungen. Erst 1984 durch die Verwendung eines Hidden-Markov Modells (HMM) machte die Spracherkennung einen grossen Schritt nach vorne. Mit BYBLOS entwickelten Schwartz et al. (1989) ein System, welches bei einem 1000 Wörter Vokabular eine sehr gute Word Error Rate (WER) von 2.9% erreichte.

Mit dem N-Best Algorithmus[6] (Schwartz & Chow 1990) wurde 1993 das erste System vorgestellt, welches 20'000 Wörter in kontinuierlicher Sprache erkennen konnte. Ebenfalls in den 90er Jahren kombinierten Zavaliagkos und Austin (1993) Neuronale Netze (KNNs) mit Hidden-Markov Modellen und erreichten so eine bessere Genauigkeit der Spracherkennung. Da Neuronale Netze sehr rechenintensiv waren, konnten diese zu dem Zeitpunkt noch nicht effizient eingesetzt werden. Dieser Ansatz wurde von Bourlard und Morgan (1997) weiterentwickelt.

Aktuelle Spracherkennungssysteme bauen im wesentlichen auf den oben erwähnten Vorgehen auf. In der englischen Sprache bilden 61 Phoneme über 500'000 Wörter (Garfolo et al. 1992). Phoneme sind „Laute, die in derselben Stellung einen Bedeutungsunterschied hervorrufen" (Grebe 1966). Phoneme werden häufig mittels Fourier Transformationen (Graps 2004) aus dem digitalen Sprachsignal extrahiert. Um die Phoneme mit linguistischen Modellen oder Grammatiken zu vergleichen, werden unterschiedliche Verfahren angewandt. Das oben erwähnte Hidden Markov Modell (Rabiner 1989) aber auch Support Vector Machines (SVM)

[6] Der Algorithmus wurde 1990 entworfen, aber erst 1993 in einem funktionierenden System vorgestellt.

(Kecman 2001) und Künstliche Neuronale Netze (Richard & Lippermann 1991) sind für diesen Zweck geeignet. Die rechenintensiven KNNs haben den grossen Vorteil, dass sich diese durch „Training“ verbessern lassen. Nachteile von KNNs sind die schlechte Erkennung von kontinuierlicher Sprache. Bourlard und Morgan (1997) schlagen daher ein Hybrides System vor, welches die Eigenschaften von KNNs und HMMs kombiniert.

3.2.2 Die Grammatik der Spracherkennung

Eine der wichtigsten Komponenten der Spracherkennung ist die Grammatik[7]. Neben dem Audiosignal ist die Grammatik der zweite Input der Sprachsteuerung und definiert die Wörter und die Wortmuster, welche von den oben genannten Algorithmen zur Erkennung benötigt werden. Die Syntax wird im XML-Format vom World Wide Web Consortium W3C (W3C 2004) als kontextfreie Grammatik definiert. Die Grammatik nach W3C beinhaltet:

- Wörter die gesprochen werden können,
- Muster in denen diese Wörter enthalten sein können,
- Gesprochene Sprache von jedem Wort.

Grammatiken können beliebig komplex definiert werden. Von einzelnen Stichwörtern bis hin zu kompletten, sich wiederholenden Abfolgen von Wörtern und Sätzen ist alles möglich, was die Kontextfreiheit der Grammatik nicht verletzt.

Abbildung 6 zeigt schematisch einen Ausschnitt aus der Grammatik die in Nipper verwendet wird. Ein ganz einfacher Sprachbefehl, der durch die gezeigte Grammatik definiert wird, ist der Befehl „Kreis“. Der Pfad führt vom „Start“ zum Befehl „Kreis“ in der Gruppe „Funktionen“ und dann direkt zur „Ausgabe“. Dieser Befehl lässt sich sinngemäss erweitern zu „Kreis - Hintergrundfarbe – rot“.

Über eine wohldefinierte Grammatik können falsche Eingaben von vornherein blockiert werden. Damit muss bei der Interpretation der Befehle in der Anwendung selbst nicht mehr überprüft werden, ob eine Eingabe korrekt ist oder nicht. Unzulässige Befehle wie „Hintergrundfarbe – 5 Pixel“ werden so von der Erkennung ausgeschlossen.

Ein weiteres Konstrukt, welches die Ausgabe der Spracherkennung effizienter gestaltet, ist die Vergabe von semantischen Schlüsselbefehlen. Über die Semantik können unterschiedliche Sprachbefehle mit gleicher Bedeutung auf eine einzige Semantik reduziert werden. Bei der in Abbildung 6 gezeigten Grammatik generieren die beiden Befehle „Linienbreite 5 Pixel“ und

[7] Die von der Spracherkennung akzeptierten Befehle, werden in einem Regelsystem definiert, welches in Analogie zu den Regelsystemen natürlicher Sprache, Grammatik genannt wird (Wegener, 2005).

„Linienbreite 5“ denselben semantischen Schlüssel, obwohl die Pfade im Schema unterschiedlich sind.

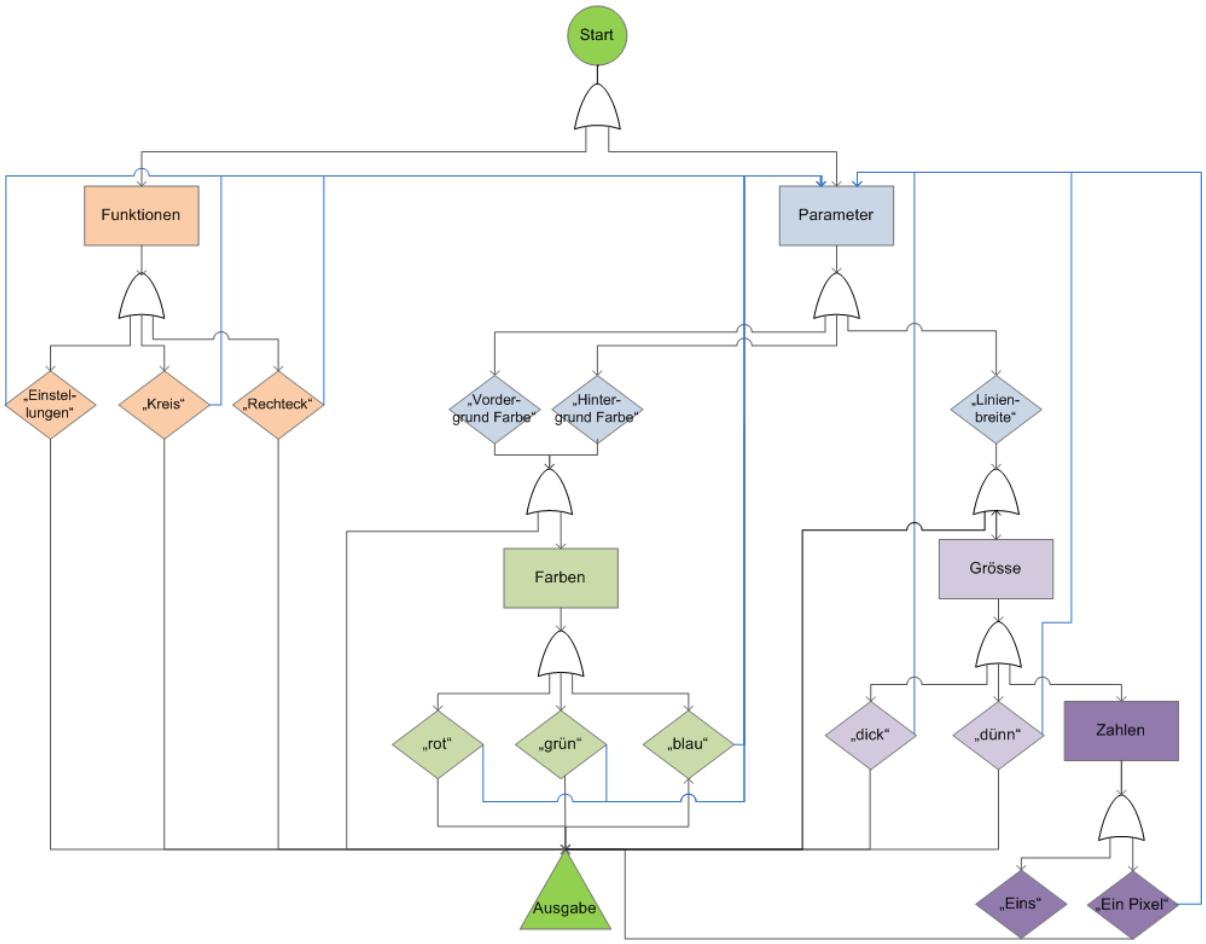

Abbildung 6: Schematische Darstellung aus der Nipper-Grammatik

Durch die Verwendung von semantischen Schlüsseln können sogar bei unterschiedlichen Sprachen identische Ausgaben generiert werden. Auch Zahlen können über die Semantik definiert werden. Die Zahl „Fünfzehn“ setzt sich aus „Fünf“ und „Zehn“ zusammen. Liessen sich semantische Schlüssel nur hintereinander reihen, hätte die Erkennung von „Fünfzehn“ das falsche Ergebnis „510“. Daher lassen sich beliebige Operationen auf diesen Schlüsseln ausführen, so auch Additionen. Je komplexer die Grammatik, desto schwieriger wird es, die semantischen Schlüssel korrekt zu definieren. Der Sprachbefehl „Ich hätte gerne ein rotes Textfeld mit weisser Schrift“ kann mit der entsprechenden Grammatik dieselbe Semantik erzeugen, wie der etwas abstraktere Sprachebefehl „Textfeld - Hintergrundfarbe rot – Schriftfarbe weiss“. Bei Nipper sind die Schlüssel so definiert, dass diese von der Anwendung einfacher Interpretiert werden können. Dabei werden Funktionen mit „::FUNKTION“ und Parameter mit „_PARAMETER_“ gekennzeichnet. So können beliebig viele Befehle kombiniert und von der Anwendung interpretiert werden.

Ist die Grammatik einmal korrekt definiert, können auf modernen Rechnern, die mit Prozessoren mit mehreren Gigahertz arbeiten, Rechenintensive Algorithmen und komplexe Grammatiken in einer Geschwindigkeit verarbeitet werden, welche den Arbeitsfluss nicht wesentlich beeinflusst. Befehle, die in der Grammatik definiert sind, werden mit einer Genauigkeit von über 99% erkannt. Die Genauigkeit beim Diktat wird mit 95-99% angegeben[8]. Wie eine Untersuchung bei Radiologen des RCI in Iowa zeigte, wies ein transkribierter Bericht eine Fehlerrate von 13% auf, während die Berichte die mit Spracherkennung erstellt wurden lediglich 9% aufwiesen (HealthImagingNews 2007). Desweiteren wurden lediglich 4-8% der transkribierten Berichte in weniger als einer Stunde verfasst. Nach einiger Angewöhnungszeit an die Spracherkennung wurden 82% der Berichte in weniger als einer Stunde diktiert.

Für die Erkennung von Sprache bietet der Markt einige Lösungen an, zum einen Softwarebasierte (SRSP) und zum andern Hardwarebasierte (SRHM) Erkennungsmodule. SRHM sind effizienter als SRSP, allerding nur in wenigen Sprachen erhältlich. Softwarebasierte Lösungen sind für Desktop-Computer besser geeignet, vor allem auch, da diese von Entwicklern einfacher verwendet werden können. Nach Kibira (2005) gibt es lediglich zwei Systeme, welche die deutsche Sprache unterstützen (Siehe Tabelle 2). Dragon Naturally Speaking von Nuance unterstützt seit der Version 7 die Erkennung von mehreren Sprachen. Das zweite System ist von IBM und heisst ViaVoice 10. Beides Systeme werden von Nuance vertrieben und sind kostenpflichtig.

Tabelle 2: Spracherkennungsmodule nach Kibira (2005, überarbeitet)

Spracherkennungsmodule für Entwickler	Verkäufer	Sprachen
IBM Via Voice 10	IBM / Nuance (Nuance 2008b)	4 Sprachen (deutsch)
Dragon Naturally Speaking 10 SDK	Nuance (Nuance 2008a)	6 Sprachen (deutsch)
Voxit	Voxit (Voxit 2008)	schwedisch
Java Speech API	Sun Microsystems (Microsystems 2008)	englisch
CMU Sphinx	Carnegie Mellon(Group 2008)	englisch
SpeechStudio Suite	SpeechStudio Inc. (SpeechStudio 2008)	Englisch
Microsoft Speech API (SAPI 5.3)	Microsoft (Microsoft 2008)	8 Sprachen (deutsch)

[8] Die Genauigkeit der Spracherkennung ist abhängig vom verwendeten System. Die oben angegebenen Werte sind nicht eindeutig belegt. Entsprechen aber den Eindrücken die aus den Tests von Nipper gewonnen werden konnten und den Daten vom RCI (HealthImagingNews, 2007). Die Prozentangaben stammen von Oliver Scholz, Vista Speech UX Program Manager (Zeng, 2006).

Das vermutlich neuste System, daher von Kibira noch nicht erwähnt, ist die SAPI 5.3 von Microsoft. Die SAPI 5.3 ist bei Windows Vista standardmässig dabei und ist mit Vista somit frei zugänglich. Kennzahlen dieser Systeme sind entweder nicht verfügbar oder nur schwer zugänglich. Daher ist ein Vergleich der oben genannten Systeme an dieser Stelle nicht möglich. Weitere teilweise frei Verfügbare Systeme wie CMU Sphinx[9] oder JSAPI[10] bieten keine Unterstützung für Deutsch an und werden daher in diesem Zusammenhang nicht genauer betrachtet, da es sinnvoll erscheint für diese natürliche Art der Interaktion die Muttersprache zu verwenden.

3.2.3 Sprachbasierter Mauszeiger

Karimullah und Sears (2002) stellen in ihrer Arbeit ein Konzept einer „sprachgesteuerten Maus“ vor. Um alle Funktionen eines Computers mit einer GUI effektiv kontrollieren zu können, muss der Benutzer Daten eingeben sowie den Mauszeiger manipulieren können. Möchte man diese Funktionen alleine mit der Sprache steuern, so zeigt sich schnell, dass eine effektive Positionierung des Mauszeigers mittels Sprache sehr schwer fällt, ganz zu schweigen von Aktionen wie „Ziehen und Ablegen“. Karimullah und Sears stellen zwei verbessernde Methoden, die Zielbasierte- und Richtungsbasierte Navigation, vor.

Beim der Zielbasierten Navigation wird davon ausgegangen, dass das Ziel benennt werden kann. Dies kann ein Wort in einem Text sein, ein Menüeintrag oder eine Region auf dem Bildschirm. Je mehr Ziele sich auf dem Bildschirm befinden, desto anfälliger auf Fehler wird diese Form der Navigation. Beispielsweise könnte das Wort „Kalender“ mehrmals auf dem Bildschirm vorkommen. Mögliche Sprachbefehle wären:

- „Selektiere Kalender“ damit wird das Wort „Kalender“ in einem Text selektiert
- „Bewege Maus nach links oben“ platziert den Cursor in der Mitte vom ersten Quadranten des Bildschirmes.

Bei der Richtungsbasierten Navigation wird zwischen kontinuierlicher oder diskreter Bewegung unterschieden. Für diskrete Bewegungen gibt der Benutzer Richtung und Distanz via Sprachbefehl ein.

- „Bewege drei Wörter nach links“
- „Bewege 10 Zentimeter nach Oben“

[9] CMU Sphinx, Open Source Speech Recognition Engines wurde vom DARPA gegründet: http://cmusphinx.sourceforge.net/
[10] Java Speech API: http://java.sun.com

Für kontinuierliche Bewegungen gibt der Benutzer lediglich die Richtung an. Mit dem Befehl „Maus nach links“ beginnt sich der Mauszeiger nach Links zu bewegen bis der Benutzer „Stopp“ sagt.

Diese Formen der Navigation stellten sich entweder als ineffizient oder fehlerhaft dar. Karimullah und Sears schlagen daher einen prädikativen Mauszeiger (predictive Cursor) vor. Bei dieser kontinuierlichen, richtungsbasierten Navigation müssen drei Verzögerungen berücksichtigt werden. Zuerst muss der Benutzer eine verbale Reaktion auf ein visuelles Signal erzeugen. Zweitens muss der Benutzer den gewünschten Befehl aussprechen und drittens braucht das System eine gewisse Zeit den Befehl zu interpretieren. Um zu vermeiden, dass der durch Sprachbefehl in Bewegung versetzte Cursor das Ziel verfehlt, sollen diese Verzögerungen berücksichtigt und damit eine schnellere Bewegung des Cursors ermöglicht werden. Da die Reaktionszeit bei jedem Benutzer unterschiedlich ist, wird der prädikative Cursor für jeden Benutzer kalibriert. Je nach Bewegungsrichtung des Cursors wird nun ein zweiter Cursor angezeigt. Dieser soll als Indikator für ein „Stopp“ oder „Click“ Befehl gelten. Die Effizienzsteigerung durch diese Massnahme hielt sich jedoch in Grenzen. Wesentliche Verbesserungen wurden dadurch jedoch nicht erzielt.

Manaris und Harkreader (Manaris & Harkreader 1998) kombinierten Zielbasierte und Richtungsbasierte Navigation. Sie definierten fünf Regionen auf dem Bildschirm, welche Zielbasiert schnell erreicht werden konnten. Die restliche Navigation erfolgte Richtungsbasiert.

Kevin et al. (Kevin et al. 2000) entwickelten eine sprachbasierte Navigation für Webseiten. Dabei musste nicht die Maus auf den Link positioniert, sondern lediglich der Text des Links als Sprachbefehl verwendet werden. Diese zielbasierte Navigation zeigte sich als relativ Robust, weicht aber durch die Verwendung von Links der eigentlichen Problematik der Cursor-Bewegung aus.

Die Sprache, ein sehr mächtiges Instrument mit beinahe unendlich vielen Befehlen und Befehlskombinationen, eignet sich vor allem dort, wo direkt auf Elemente oder Funktionen zugegriffen werden kann. Bei Positionsangaben ist die Sprache, wie oben erwähnt, äusserst ineffizient. Die Ineffizienz dieser sprachlichen Umwege verstärkt sich bei LHRDs noch mehr, da dort Objekte mehrere Tausend Pixel oder mehrere Meter auseinanderliegen können. Daher wird bei Nipper, wie auch schon von Bolt vorgeschlagen, die Spracherkennung mit einem Zeigegerät kombiniert.

3.3 Zeigegeräte für LHRDs

In den vorhergegangenen Kapiteln wurde schon mehrfach Kombination von Spracherkennung mit Zeigegeräten erwähnt. Für LHRDs sind nicht alle Zeigegeräte gleich gut geeignet. Die meisten Point & Click Eingabegeräte gehen entweder davon aus, dass sich der Benutzer nahe am Bildschirm befindet und via Stift oder Berührung interagiert oder dass er sich an einem stationären Arbeitsplatz befindet. Wie oben erwähnt kann bei LHRDs nicht von diesen Annahmen ausgegangen werden. Nach Vogel und Balakrishnan (2005) soll ein Eingabegerät folgende Charakteristiken[11] aufweisen:

Genauigkeit (Accuracy): Das System muss in der Lage sein, kleinste Objekte verlässlich selektieren zu können, egal ob sich der Benutzer nahe am Display oder weit davon entfernt aufhält. Die Benutzerschnittstelle so zu gestalten, dass nur grosse Ziele angezeigt werden, würde die Problematik zwar entschärfen, ist jedoch kein realistisches Design. Auf einem LHRD Ziele grösser darzustellen, nur um diese mit dem Eingabegerät treffen zu können widerspricht der Idee der hohen Auflösung.

Inbetriebnahme (Acquisition Speed): Bei LHRDs kann nicht davon ausgegangen werden, dass ein Benutzer ständig mit dem Display interagiert. Der Benutzer hat möglicherweise andere Aufgaben zu erledigen oder es interagieren mehrere Personen mit dem Display. Zhai (1998) argumentiert, dass „eines der grössten Probleme mit 3D Eingabegeräten welche frei beweglich sind, die Inbetriebnahme des Eingabegerätes ist“. Die Inbetriebnahme des Eingabegerätes soll daher mit minimalem Aufwand möglich sein.

Interaktionsgeschwindigkeit (Pointing and Selection Speed): Bei LHRDs muss der Benutzer Objekte selektieren können, die eventuell mehrere Meter oder mehrere Tausend Pixel auseinander liegen. Das Eingabegerät soll daher schnell und dennoch präzise vom einen Objekt zum anderen bewegt werden können. Die Selektion an und für sich, sprich der Klick, soll ebenfalls schnell und einfach durchgeführt werden können.

Komfort (Comfortable Use): Das Gerät soll einfach zu bedienen und verstehen sein. Bei freien Bewegungen ist vor allem auch darauf zu achten, dass diese für den Benutzer körperlich nicht anstrengend sind.

[11] Diese Charakteristiken richten sich nach den Bestimmungen der ISO-Norm 9241-9 (2000) unter spezieller Berücksichtigung der Anforderungen von LHRDs.

Interaktionsdistanz: Das Zeigegerät soll konsistente Interaktion unabhängig vom Abstand zum Display ermöglichen. Übergänge zwischen Positionen nahe am Display und Positionen weit vom Display entfernt sollen so weich wie möglich gestaltet werden.

Im Folgenden werden zwei Zeigegeräte vorgestellt, die sich gut als aktives Zeigegerät für LHRDs eigenen. Zudem wird mit dem Eye-Tracking noch eine weitere Möglichkeit zur Positionsangabe vorgestellt.

3.3.1 Freihand Gesten

Vogel und Balakrishnan schlagen eine Technik für grosse Displays vor, die es ermöglicht Zeige- und Klickinteraktionen nur mit der Hand zu tätigen. Damit können zusätzliche physische Eingabegeräte umgangen und das „natürliche Zeigegerät" des Menschen verwendet werden, die Hand (Vogel & Balakrishman 2005).

Aktuelle Trackingsysteme benötigen Reflektoren zur Positionsbestimmung (Vgl. Abb. 7.c). Die Erkennung von Position und Bewegung der blossen Hand im dreidimensionalen Raum soll hingegen schon bald robust genug sein, um es für solche Interaktionen einsetzten zu können (Nickel & Stiefelhagen 2003). Es ist daher realistisch anzunehmen, dass in naher Zukunft keine Reflektoren mehr benötigt werden. Damit kommt diese Interaktionstechnik der Vision der völlig freien Interaktion schon sehr nahe.

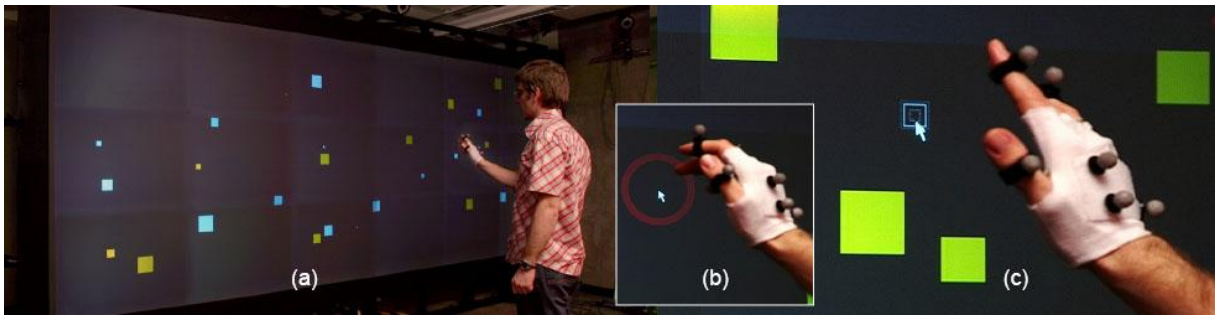

Abbildung 7: Hand-Tracking mit Hilfe eines Handschuhs und Reflektoren (Vogel & Balakrishman 2005)

Eines der klassischen Probleme bei der Interaktion ohne physisches Eingabegerät ist das Klicken ohne Tasten oder Knöpfe. Vogel und Balakrishnan streben die Verwendung bestehender Point & Click GUIs an. Daher werden Alternativen benötigt, um die Tastenfunktionen der Maus zu simulieren. Dazu schlagen sie zwei Klick-Techniken vor, die ohne physisches Eingabegerät auskommen.

Der „AirTap" ist dem Klicken mit der Maus nachempfunden. Der Klick wird mit dem Zeigefinger ausgelöst, sobald sich dieser nach untern bewegt. Problem dabei ist jedoch, dass es keine definierte Stoppposition gibt. Visuelles und akustisches Feedback bestätigt einen Klick

(Vgl. Abb. 8). Um einen Klick zu erkennen, werden Geschwindigkeit und relative Änderung zu den anderen Fingern berücksichtigt. Es wurde festgestellt, dass Benutzer Klicks zwar markant ausführen, diese sich aber von Benutzer zu Benutzer stark unterscheiden. Daher muss das System für jeden Benutzer kalibriert werden.

Mit dem „Thumb Trigger“, dem Daumen-Auslöser wird der Daumen zum Zeigefinger hinbewegt, um einen Klick auszulösen. Theoretisch hat diese Klick-Technik einen klaren Vorteil gegenüber dem AirTap, da der Daumen am Zeigefinger einen Anschlag hat und somit klar definiert ist, wann der Klick ausgeführt werden soll. Durch die Berührung mit dem Zeigefinger kriegt der Benutzer kinästhetisches Feedback. Tests haben jedoch gezeigt, dass die Benutzer diese Art des Klickens als unangenehm empfinden. Abbildung 8 zeigt die einzelnen Gesten die zum Klicken, Ziehen und Loslassen benötigt werden im Vergleich.

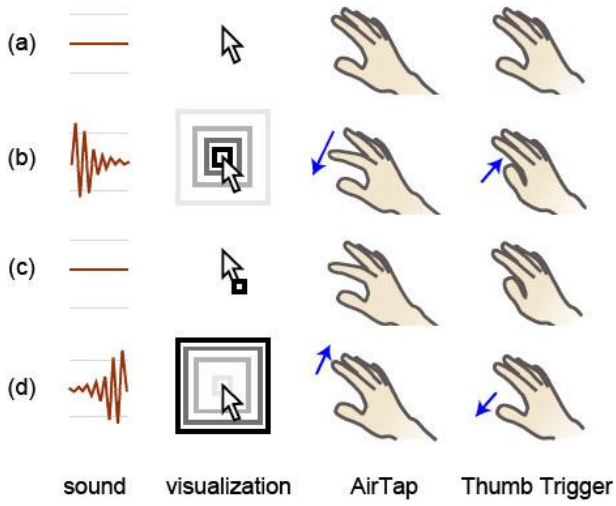

Abbildung 8: AirTap und Thumb Trigger im Vergleich (Vogel & Balakrishman 2005)

Die Powerwall der Universität Konstanz verfügt über ein ähnliches System. Das optische Verfolgungssystem der Firma A.R.T. tastet mit vier Infrarotkameras den Bereich vor dem Display ab. Diese Kameras können spezielle Reflektoren mit einer Genauigkeit von weniger als einem Millimeter erfassen. Bei Föhrenbach et al. (2008) werden solche Reflektoren auf einem dünnen Handschuh befestigt. Dieser Handschuh kann als Zeigegerät verwendet werden, bei dem ein Klick mit Daumen und Zeigefinger ausgeführt (Vgl. Abb. 9) und die Richtung anhand des Handrückens bestimmt wird (Vgl. Abb. 10). So verändert sich die Position des Mauszeigers weniger stark, wenn der Benutzer einen Klick ausführt.

Das menschliche Zittern der Hand und grobmotorische Bewegungen reduzieren die Treffsicherheit bei kleinen Objekten. Dieser Effekt verstärkt sich mit zunehmender Entfernung zum Display. Mit einem Kalman-Filter (Oh & Stuerzlinger 2002) oder andern Filterkonzepten

kann dies reduziert aber nicht ganz eliminiert werden. Diese Ungenauigkeiten sind ein zu berücksichtigender Punk bei der Entwicklung von GUIs für diese Interaktionstechnik.

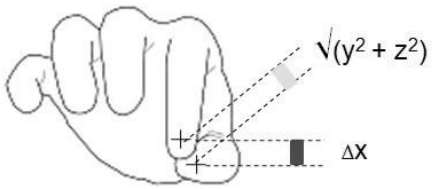

Abbildung 9: Geste zur Selektion (Föhrenbach et al. 2008)

Abbildung 10: Der Handrücken als Richtungsweiser (Föhrenbach et al. 2008)

Abgesehen von der Genauigkeit sind Handgesten gut für LHRDs geeignet. Die Evaluation in Kapitel 5 hat bezüglich Komfort, Interaktionsgeschwindigkeit und Inbetriebnahme keine markanten Defizite ergeben. Im Hinblick auf zukünftige Systeme, die keine Reflektoren mehr benötigen, könnte diese Interaktionstechnik eine grundlegende Komponente für die völlig freie Interaktion sein.

3.3.2 Laserpointer-Interaktion

Ein weiteres Eingabegerät welches sich optimal zur Interaktion mit LHRDs eignet ist der Laserpointer. Die Idee der Laserpointer-Interaktion ist der Verwendung von Laserpointern in Präsentationen nachempfunden. Bei Präsentationen werden Laserpointer verwendet, um den Zuschauern zu deuten, worüber gerade gesprochen wird. Diese „intuitive Verlängerung der Hand“ eignet sich auch für die Interaktion mit LHRDs (König et al. 2007a). Auch die dafür notwendige Mobilität ist gegeben, da nicht die Position des Gerätes sondern der Schnittpunkt des Laserstrahls mit dem Display für die Eingabe verwendet wird. Der Benutzer kann so beliebig nah am Display, aber auch aus grosser Entfernung mit dem System interagieren. König et al. (2007b) stellen seine flexible und skalierbare Interaktionsbibliothek vor, welche „eine sehr direkte und nahezu verzögerungsfreie Eingabe mittels Laserpointer-Tracking auch für grosse, hochauflösende Displays ermöglicht“. Drei Spezialkameras mit Abtastraten von 80fps[12] sind, beim Beispiel der Powerwall Konstanz, hinter dem Display angebracht und erkennen den gebündelten Infrarotlaserstrahl des Laserpointers. Durch die Verwendung des für Menschen unsichtbaren Infrarotlasers sieht der Benutzer lediglich den errechneten Schnittpunkt in Form des Mauszeigers. Sichtbare Laser könnten Abweichungen des optischen und errechneten Schnittpunktes aufweisen, was zu Irritationen seitens des Benutzers führen kann.

[12] Frames per second (fps) bezeichnet die Anzahl Aufnahmen die pro Sekunde gemacht werden,

Mit den drei Eingebauten Tasten kann der Laserpointer die Maus in jeder Point & Click – Anwendungen ersetzten. Die Klicks werden über ein Funkmodul schnurlos an die Powerwall übermittelt (König et al. 2008).

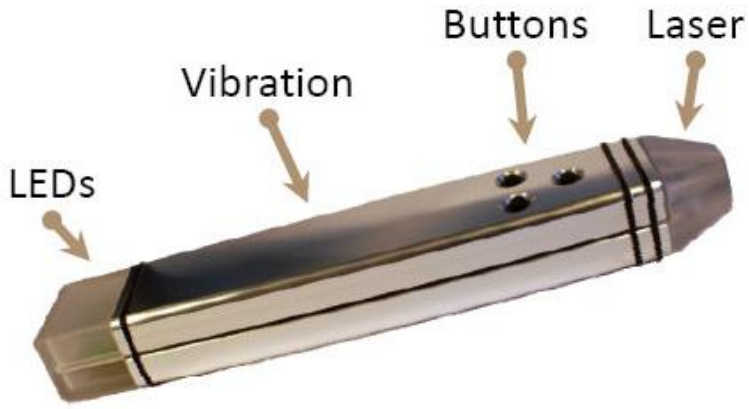

Abbildung 11: Der Laserpointer, entwickelt an der Universität Konstanz (König et al. 2008)

Als absolutes Zeigegerät kann der Laserpointer sehr schnell über die gesamte Fläche eines LHRDs bewegt werden. Bei der Zielgenauigkeit treten ähnliche Probleme auf wie bei den Handgesten. Kleine Objekte sind auch mit dem Laserpointer schwierig zu treffen. Abgesehen von dieser Zielgenauigkeit ist der Laserpointer ein sehr intuitives Eingabegerät und kann sofort von jedem verwendet werden. Durch die automatische Kalibrierung des Gerätes kann dieses ohne weiteres von Benutzer zu Benutzer weitergegeben werden. Dies Begünstigt kollaborative Interaktion, da der Wechsel des Eingabegerätes in wenigen Sekunden von statten gehen kann, was nach Zhai (1998) ein wichtiger Aspekt bei kollaborativer Interaktion ist.

3.3.3 Eye-Tracking

Eine weitere Methode zur Positionsbestimmung, die neben Laserpointer und Handgesten noch erwähnt werden sollte, ist das Eye-Tracking. Das Eye-Tracking ist daher interessant, da dem System mitgeteilt werden kann, auf welchen Bereich sich der Benutzer konzentriert. Die Bewegung des Auges kann zur Analyse, aber auch zur Kontrolle von Benutzerschnittstellen verwendet werden. Bei der Analyse wird die Bewegung der Augen aufgezeichnet und später ausgewertet. Mittels Eye-Tracking, auch Blicksteuerung (Bieg 2008) oder Augenbewegungsmessung (Schneider & Eggert 2006) genannt, teilt der Benutzer dem Computer bewusst oder unbewusst mit, wo gerade sein visueller Fokus liegt. Um die Blickrichtung des Benutzers zu erkennen, gibt es drei Methoden die mehrheitlich Verwendet werden (Bieg 2008):

- Elektrookulographie: (EOG) Mit seitlich von den Augen auf die Haut geklebte Elektroden kann bei horizontalen Augenbewegungen eine Spannungsänderung zwischen den beiden Elektroden gemessen werden. (Fuhry).
- Search-Coils: Kontaktlinsen die mit mechanischen oder optischen Elementen, wie beispielsweise Magnetspulen versehen sind, welche das Tracking ermöglichen.
- Videookulografie (VOG): Videobasiertes Tracking, welches mit Videokameras die Bewegung des Auges aufzeichnet.

Da die EOG Störungen des Gleichgewichtssystems, Schwindel oder Übelkeit verursachen kann, ist diese Methode für den alltäglichen Gebrauch nicht geeignet (Fuhry). Mit den Search-Coils können sehr genaue Ergebnisse ermittelt werden. Der Einsatz von Search-Coils kann jedoch das Auge austrocknen und es kann zu Deformationen an der Hornhaut kommen, was wiederum die Sehschärfe einschränken kann (Schneider & Eggert 2006). Da zudem Kontaktlinsen nicht von allen Personen getragen werden können, ist auch dies kein geeignetes Mittel um die Blicksteuerung im Alltag zu verwenden. In den meisten Studien, wie auch an der Universität Konstanz, wird das Video basierte Tracking verwendet.

Die Universität Konstanz verwendet das EyeLink II System von SR-Research (SR-Resarch 2008). Die beiden Kameras (Vgl. Abb. 12 und 13) nehmen das Auge 500-mal pro Sekunde auf und ermitteln so die Richtung der Pupille relativ zu den Kameras. Da sich ein Person vor der Powerwall frei bewegen kann, muss zusätzlich die Richtung des Kopfes ermittelt werden. Dies geschieht über das Trackingsystem der Powerwall, das die Position des Eye-Trackers ermitteln kann. Mit der absoluten Position und Ausrichtung des Kopfes und der relativen Position der Pupille kann das System einen Fokuspunkt auf der Powerwall errechnen.

Abbildung 12: EyeLink II von SR-Research (SR-Resarch 2008)

Abbildung 13: EyeLink II, Seitenansicht (SR-Resarch 2008)

„Die visuelle Wahrnehmung erzeugt die Illusion einer stabilen visuellen Welt, die innerhalb des gesamten Fixationsfeldes kontinuierlich zur Verfügung steht.“ (Schneider & Eggert 2006)

Die sakkadischen Augenbewegungen mit denen die visuelle Umgebung abgetastet wird, fallen dem Menschen im Allgemeinen nicht auf. Einfach ausgedrückt ist das Auge ständig in Bewegung und kann nicht Linear bewegt werden wie beispielsweise ein Mauszeiger. Die Verwendung des Eye-Trackers als gewöhnliches Zeigegerät ist daher nicht unproblematisch. Hinzu kommt, dass aktive Kontrolle des Auges ein hohes Mass an Konzentration erfordert und kann sehr schnell zu Ermüdungserscheinungen führen. Desweiteren wird das Eye-Tracking mit zunehmender Entfernung zum Display ungenauer. Tabelle 3 zeigt mögliche Abweichungen des Eye-Trackings vom eigentlichen Ziel am Beispiel der Powerwall.

Tabelle 3: Abweichungen in Pixel bei 22.7ppi Auflösung (Bieg 2008)

Distanz (m)	**Pixel**
0.5	7.8
1	15.6
3	47.8
6	93.6

Das Eye-Tracking kann auch zur „Adaption gewisser Aspekte der Interaktion“ verwendet werden (Bieg 2008). Solche Systeme verwenden das Eye-Tracking als zusätzlichen Input und nicht zur Kontrolle an und für sich. So lassen sich zusätzliche Informationen ermitteln, welche bei der Interaktion nützlich verwendet werden können.

Jedes Eingabegerät hat Vor- und Nachteile. Bis auf den Laserpointer haben jedoch alle vorgestellten Interaktionstechniken das Potential, in Zukunft ohne Sensoren am Körper zu funktionieren. Eine völlig freie Interaktion, die ohne Eingabegerät, nur mit Stimme, Hände und Augen funktioniert. Auch wenn diese Systeme noch nicht ganz soweit sind, können damit jetzt schon optimale Interaktionstechniken entwickelt werden. Im Folgenden soll ein System gezeigt werden, welches die oben vorgeschlagenen Konzepte vereint und neue Konzepte hinzufügt.

4 Multimodale Interaktion an der Powerwall

Ziel des praktischen Teils dieser Arbeit ist die Konzipierung und Umsetzung einer multimodalen Anwendung für die Powerwall der Universität Konstanz. Dabei werden die in den vorhergegangenen Kapiteln diskutierten Konzepte in einem System integriert und für die erkannten Probleme werden Lösungsvorschläge eingebracht. Dieses System mit dem Namen Nipper[13] wird dazu in drei Teile unterteilt. Da es vom Laserpointer, von der Handgestenerkennung und vom Eye-Tracking[14] bereits funktionierende Prototypen für die Powerwall gibt, beschäftigt sich der erste Teil mit der Spracherkennung. Somit sind für den nächsten Teil alle in Kapitel 3 vorgeschlagenen Eingabegeräte auf der Powerwall verfügbar. Im zweiten Teil wird „NipMap“ vorgestellt. NipMap ist eine Anwendung, welche mit den eben genannten Eingabegeräten multimodal bedient werden kann. Damit können, wie in Abbildung 14 gezeigt, einfache geometrische Objekte und Textfelder erstellt und deren Parameter manipuliert werden. Im letzten Teil werden die beiden Konzepte, „WYSIWYS“ und „Input-Everywhere“, zur Verbesserung der multimodalen Interaktion mit Spracherkennung auf LHRDs vorgestellt.

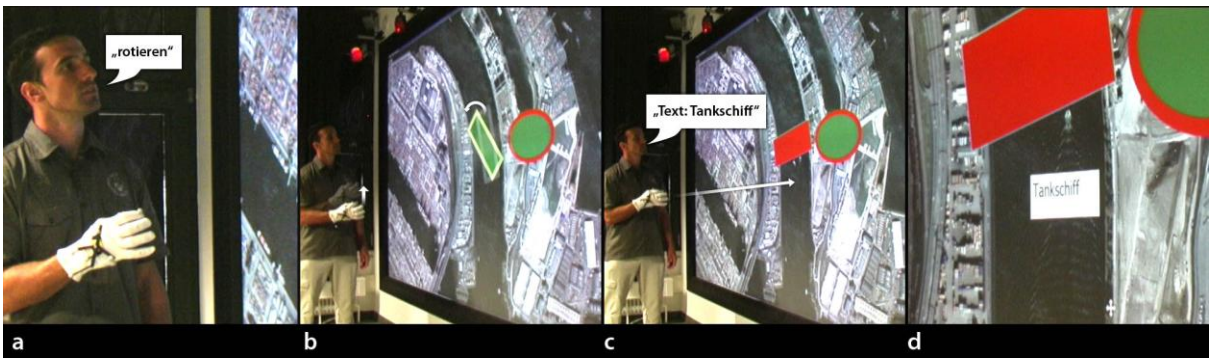

Abbildung 14: Multimodale Interaktion mit NipMap auf der Powerwall[15]

Die Bildsequenz in Abbildung 14 stammt aus dem beiliegenden Demo-Video. In diesem Video werden die einzelnen Funktionen vor der Powerwall demonstriert.

4.1 Das Nipper-System

4.1.1 Spracherkennung bei Nipper

Um das „Speak-and-Point“-Konzept aufgreifen zu können, muss eine gut funktionierende Spracherkennung realisiert werden. Eine erste und wichtige Entscheidung bei der Entwick-

[13] Nipper ist der Name des Hundes von Francis Barraud, dem Maler des bekannten Gemäldes mit diesem weissen Hund, der vor dem Grammophontrichter sitzt und „der Stimme seines Herrn“ lauscht. Das Bild wurde als Logo von „His Master’s Voice“, einem Hersteller von Schallplatten bekannt(Designeboom.com, 2008).

[14] Die Powerwall verfügt ebenfalls über ein Eye-Trackingsystem, welches jedoch aus technischen Gründen nicht zur Verfügung stand.

[15] Die Illustrationen sind im Allgemeinen so zu verstehen, dass in einem Bild ein Sprachbefehl gegeben wird und im nächsten wurde dieser vom System ausgeführt.

lung eines sprachgesteuerten Systems ist die Wahl der Spracherkennung. Für Nipper fiel die Wahl aus den in Kapitel 3.2 aufgeführten Systemen auf die Microsoft SAPI 5.3. Diese Wahl ist damit zu begründen, dass die SAPI 5.3 in deutsch und mit Windows Vista auch frei Verfügbar ist. Desweitern beinhaltet das .NET Framework 3 für die Entwicklung benötigte Bibliotheken. Die Vorgängerversion SAPI 5.0 wurde schon zu einem früheren Zeitpunkt auf der Powerwall installiert, allerdings sind die Ergebnisse dieser Spracherkennung relativ schlecht und als Sprache ist nur Englisch verfügbar. Die Powerwall wird mit Windows Server 2003 betrieben und mit dem .NET Framework steht auch die Sprachbibliothek „System.Speech" zur Verfügung. Allerdings ist die SAPI 5.3 für Windows Vista konzipiert und mit Windows Server 2003 nicht kompatibel. Daher kann SAPI 5.3 nicht direkt auf der Powerwall verwendet werden.

Die beste Möglichkeit die SAPI 5.3 mit der Powerwall zu verwenden erscheint daher eine Client-Server Anwendung über ein Netzwerk zu sein. Die Powerwall ist der Server zu dem sich der Client Verbinden kann. Bei dem Client handelt es sich dabei in diesem Fall um einen Windows Vista Rechner. Nipper ist jedoch so konzipiert, dass jede beliebige Spracherkennung und jedes Betriebssystem als Client verwendet werden kann, sofern die serialisierten Objekte des Clients mit der Schnittstelle des Servers überein stimmen. Auf diese Weise können auch problemlos andere Sprachen realisiert werden. Da die Interpretation der Befehle auf dem Client geschieht, ist der Server nicht abhängig von der Spracherkennung. Dieser leitet lediglich die erkannten Befehle in Form von Zeichenketten vom Client an die Anwendung weiter. Diese Vernetzung hat, abgesehen von der Flexibilität der Sprache, noch weitere Vorteile:

- Sofern es die Netzwerkverbindungen des Servers zulassen, können auf diese Weise mehrere Clients an einen Server angeschlossen werden. Wie bei Tse et al. (2006) können so mehrere Personen mit dem System interagieren.
- Ein Benutzer, der die Spracherkennung auch sonst verwendet, kann sein eigenes Notebook an den Server anschliessen und sein persönliches Sprachprofil verwenden. Die SAPI 5.3 funktioniert am Besten wenn es für jeden Benutzer ein eigenes, gut trainiertes Sprachprofil gibt.

Client, Server und NipMap verfügen zusammen über drei Grammatiken, die jeweils Befehle für eine Komponente beinhalten (Vgl. Kapitel 3.2.2). Beim Verbinden des Clients mit dem Server werden die Grammatiken vom Server an den Client geschickt. Der Sprachbefehl des Benutzers wird von einem Mikrofon aufgezeichnet. Bei der Powerwall der Universität Kon-

stanz handelt es sich dabei um ein Bügelmikrofon, welches über Funk[16] mit dem Client verbunden wird. Das Bügelmikrofon eignet sich sehr gut, da es sehr leicht zu tragen ist und sich das Mikrofon an der optimalen Position befindet. Durch die Funkverbindung hat der Benutzer die nötige Bewegungsfreiheit. Die Sprachsignale werden von der SAPI mit den Grammatiken verglichen und interpretiert. Jede der drei Grammatiken wird für unterschiedliche Funktionen verwendet. In Abbildung 15 wird der Aufbau des Nipper-Systems mit den drei Grammatiken schematisch dargestellt.

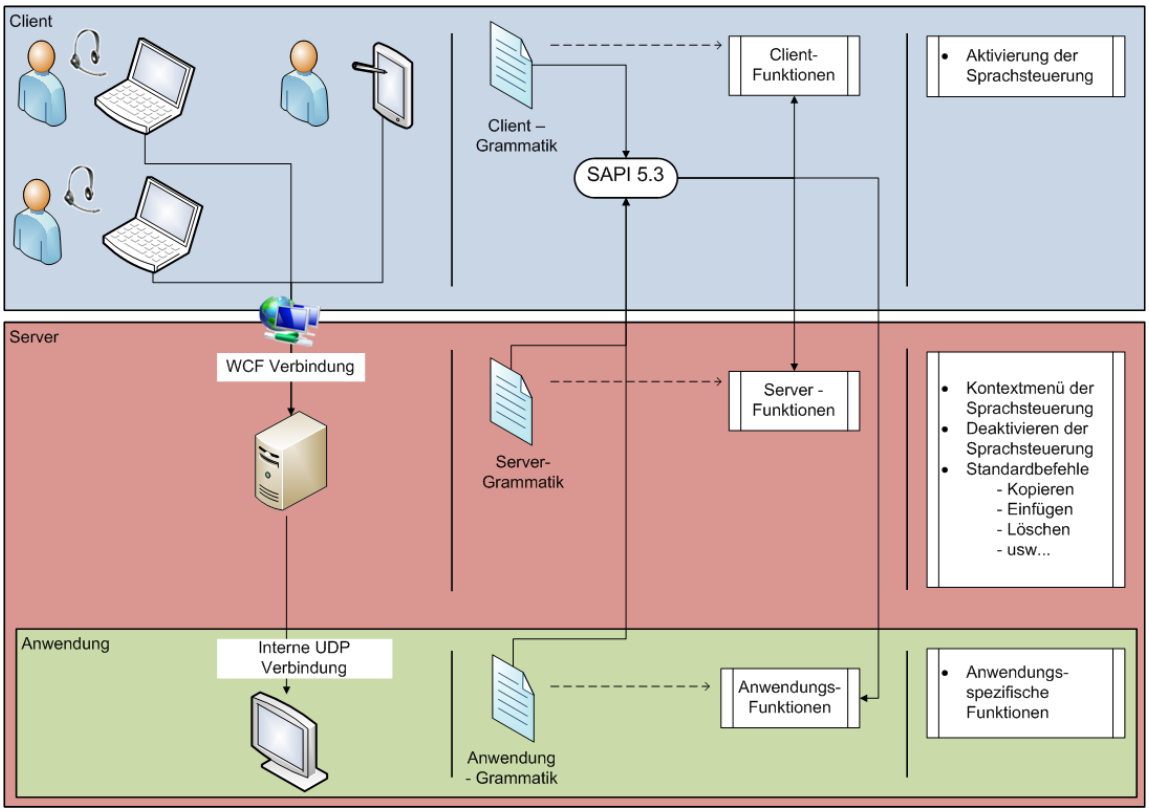

Abbildung 15: Schematische Darstellung des Nippersystems mit den drei Komponenten

Die Client-Grammatik ist lediglich dann aktiv, wenn die Spracherkennung ausgeschaltet ist und einzig für die Aktivierung der Spracherkennung verantwortlich. Wenn die Client- oder Aktivierungs-Grammatik aktiv ist, dann sind alle anderen Grammatiken deaktiviert. Die Server-Grammatik definiert anwendungsunabhängige Standard-Befehle und Befehle, die für den Server benötigt werden. Diese Befehle umfassen Aktionen wie „deaktivieren der Sprachsteuerung“, „Kontextmenü der Sprachsteuerung“ und ähnlichem. Standard-Befehle wie „Kopieren“, „Einfügen“, „Schliessen“, „Löschen“ oder „Beenden“ sind in der aktuellen Version von Nipper nicht definiert, würden jedoch auch in die Server-Grammatik gehören, da diese anwendungs-unabhängig definiert werden können. Die anwendungsspezifische Grammatik ist

[16] Sennheiser EW 352 (http://www.sennheiser.de), bei diesem Mikrofon wurde speziell darauf geachtet, dass es mit einer Richtkapsel ausgestattet ist, um möglichst wenig Hintergrundgeräusche aufzunehmen (Faeh, 2008).

die umfangreichste und komplexeste von den Dreien. Diese beinhaltet nicht mehr nur einfache Befehle wie „Schliessen“ oder „Öffnen“ sondern kann beliebig komplexe Abfolgen von Befehlen definieren. Die Semantik der Grammatik muss von der Anwendung interpretiert werden können, daher ist es sinnvoll für jede Anwendung eine eigene Grammatik zu erstellen. Eine durch die Semantik definierte Abfolge von einzelnen oder mehreren Schlüsselbegriffen wird vom Client an den Server gesendet. Je nach Befehl, wird dieser entweder direkt vom Server ausgeführt oder an die Anwendung weitergeleitet. Bei der SAPI gibt es zwei unterschiedliche Stufen der Erkennung. Diese werden unterteilt in Spracherkennung und Sprachhypothese[17].

Die Sprachhypothese ist wesentlich schneller, aber auch ungenauer als die Spracherkennung. Bei der Sprachhypothese werden die einzelnen erkannten Teilbefehle direkt ausgegeben. Die Ausgabe kann jedoch Fehler enthalten. Tabelle 4 zeigt zwei mögliche Hypothesen des Befehls „Text, Hintergrundfarbe rot, Vordergrundfarbe blau“. Die Sprachhypothese liefert direkt nach jedem erkannten Wort eine Ausgabe. Diese wird laufen durch neue Fragmente ergänzt, wobei die vorangegangenen Fragmente erneut mit ausgegeben werden. So kann es sein, dass sich die Hypothese aufgrund statistischer Modelle ändert. So im Beispiel in der Tabelle 4 zwischen Zeile 2 und 3. Diese Zeilen werden so nacheinander, direkt nach jedem Wort ausgegeben.

Tabelle 4: Möglicher Fehler in der Sprachhypothese

Gute Hypothese	**Fehlerhafte Hypothese**
Text	Text
Text Hintergrundfarbe	*Text hinten umfahren*
Text Hintergrundfarbe rot	Text Hintergrundfarbe rot
Text Hintergrundfarbe rot Vordergrundfarbe	Text Hintergrundfarbe rot Vordergrundfarbe

Diese Ausgabe wird während dem Sprechen erzeugt und ist somit sehr schnell verfügbar und Aktionen können bereits während des Sprechens ausgeführt werden. Die Sprachhypothese ist weniger strikt an die Grammatik gebunden und lässt daher auch Begriffe zu, die nicht definiert wurden.

Die Spracherkennung ist wesentlich genauer und lässt nur die genau definierten Grammatiken zu. Daher ist sie auch wesentlich langsamer und liefert, nach abgeschlossenem Erkennungsvorgang, nur ein einzelnes, komplettes Ergebnis. Bei der Spracherkennung wie auch bei der Sprachhypothese liefert die SAPI je einen Genauigkeitswert mit, der anzeigt wie genau die

[17] Bei Nipper wird lediglich die Spracherkennung verwendet, daher ist hier im Allgemeinen von der Spracherkennung die Rede. Wird die Sprachhypothese verwendet, so wird dies explizit betont.

Erkennung ist. Diese Werte werden zusammen mit den Resultaten der Spracherkennung und Sprachhypothese, dem Namen der Grammatik und dem der Grammatikregel an den Server geschickt. Beim Server kann der Benutzer je ein Limit für die Genauigkeitswerte eingeben. Liegt die Erkennungsgenauigkeit unter diesem Limit, werden die Ergebnisse verworfen. Andernfalls sendet der Client die Ergebnisse, zusammen mit Angaben der Genauigkeit sowie dem Namen der Regel, in welcher der Befehl definiert wurde, an den Server. Abhängig von dem Namen der Regel, kann der Server entscheiden ob die Eingabe für ihn oder für die Anwendung bestimmt ist.

Neben den Erkennungsfunktionen bietet die SAPI auch Funktionen an, die zusätzliche Informationen über die Erkennung liefern. Eine ebenfalls bei Nipper verwendete Funktion ist die „Speech Rejected"-Funktion. Diese wird dann Aufgerufen, wenn die Erkennungsgenauigkeit nicht erreicht wurde und der Sprachbefehl deshalb nicht als Erkannt gewertet werden kann. Desweiteren liefert die SAPI Informationen darüber, ob gerade gesprochen wird, ob ein Problem mit der Erkennung aufgetreten ist und weitere Informationen, die dem Entwickler nützlich sein können.

4.1.2 Feedback

Wie oben erwähnt ist die Spracherkennung nicht so schnell wie die Sprachhypothese. Je komplexer der Befehl, desto länger dauert dessen Interpretation. Daraus ergeben sich Verzögerungen, die dem Benutzer angezeigt werden müssen, da neue Eingaben erst möglich sind, wenn der Vorherige verarbeitet wurde. „Ein Eingabegerät wird als ansprechbar bezeichnet, wenn die seine Betätigung kennzeichnende Rückmeldung kontinuierlich und genau erfolgt. Ein Eingabemittel vermittelt eine angemessene Rückmeldung, wenn dem Benutzer eine unmittelbare wahrnehmbare und verständliche Anzeige gegeben wird, dass das Gerät auf die Betätigung durch den Benutzer reagiert." (ISO 9241-9:2000 2000) Das Feedback der Spracherkennung soll für alle Anwendungen identisch sein, daher wird dieses vom Server gesteuert. Bei LHRDs sollte das Feedback so angezeigt werden, dass dieses vom Benutzer wahrgenommen werden kann. Meldungen in der Statusleiste oder in Popup-Fenster sind aus den in Kapitel 2.1 genannten Gründen nicht geeignet.

Als beste Möglichkeit erscheint daher ein visuelles Feedback an prominenter Stelle positioniert. Dies wäre mittels Eye-Tracking möglich, sodass das visuelle Feedback dort angezeigt wird, wo der Benutzer gerade hinschaut. Technisch einfacher zu realisieren ist ein Feedback beim Mauszeiger. Während dem Arbeitsvorgang ist der Mauszeiger im Fokus des Benutzers, daher sollte ihm das dort angezeigte Feedback während einer Interaktion nicht entgehen. Bei

Nipper wird jedoch nicht der Mauszeiger verändert, da sonst die Mauszeiger für jede Anwendung angepasst werden müssten. Anstelle eines modifizierten Mauszeigers werden Icons hinter dem Mauszeiger eingeblendet. Diese sind unabhängig von der Anwendung und des gerade verwendeten Mauszeigers. Der Sever gibt bei den fünf in Tabelle 5 aufgelisteten Fällen Feedback. Ist die Sprachsteuerung deaktiviert wird dies dem Benutzer permanent und markant angezeigt. Bei deren Aktivierung wird das Icon allerdings nur kurz eingeblendet und liefert so dem Benutzer lediglich bei Statusänderungen ein Feedback. Da die aktive Sprachsteuerung als „normal Zustand“ betrachtet wird, muss dies nicht permanent angezeigt werden. Wie schon oben erwähnt treten, bei der Spracherkennung Verzögerungen auf. Diese Situation entspricht derjenigen, wenn eine Anwendung ausgelastet ist, und die Sanduhr als Cursor angezeigt wird. Da sich die Verzögerung jedoch nur auf die Spracherkennung bezieht, wird bei Nipper keine Sanduhr sondern ein „Pausensymbol“ verwendet. Dieses Symbol wird solange angezeigt, bis der Sprachbefehl verarbeitet wurde und die Spracherkennung wieder bereit ist. Auch wenn keine Verbindung zum Client besteht oder diese aus irgendwelchen Gründen getrennt wurde, wird dies dem Benutzer angezeigt.

Tabelle 5: Icons und Funktion

Icon	Funktion
	Spracherkennung deaktiviert (permanent angezeigt)
	Spracherkennung aktiv (dreimaliges Aufblinken)
	Spracherkennung zu ungenau (dreimaliges Aufblinken)
	Spracherkennung beschäftigt (solange beschäftigt)
	Keine Verbindung zum Client (permanent angezeigt)

4.1.3 Aktivierung und Deaktivierung der Spracherkennung

Ein weiterer wichtiger Punkt ist die Aktivierung und Deaktivierung der Spracherkennung. Ist die Spracherkennung permanent aktiv, kann es vorkommen, dass die SAPI diese Gespräche mit anderen Personen zu interpretieren versucht. Dies kann zur Überlastung der Spracherkennung führen, was im schlimmsten Fall zum Absturz der Software führen kann. Ein weniger schlimmer Fall ist, wenn die Spracherkennung nach gewisser Zeit die ungewollte Eingabe verarbeitet hat und dann wieder Bereitschaft signalisiert. Dennoch kann dies zu längeren Un-

terbrechungen im Arbeitsfluss führen und dadurch den Benutzer verunsichern. Um solche Situationen zu vermeiden deaktiviert sich die Sprachsteuerung von selbst, wenn nach einer gewissen Zeit kein verwertbares Sprachsignal erkannt wird. Wie lange diese Timeout Zeitspanne eingestellt, wird sollte vom Arbeitsvorgang abhängig gemacht werden. Wird hauptsächlich mit dem Computer gesprochen und lediglich zwischendurch mit einer andern Person, so kann diese Zeitspanne relativ lang gewählt werden. Bei Nipper sind standardmässig zehn Sekunden eingestellt. Diese Zeitspanne erlaubt kurze Wortwechsel zwischen den Benutzern, ohne dass die Spracherkennung deaktiviert wird. Wird jedoch hauptsächlich mit anderen Personen gesprochen, dann sollte diese Zeitspanne verkürzt werden. So kann zum einen fehlerhaften Erkennungen vorgebeugt werden und zum anderen könnte es für die anderen Personen angenehm sein, wenn der Benutzer den Computer explizit mit dem Sprachbefehl „Computer“ aktiviert. Dies wird automatisch der Fall, wenn sich die Spracherkennung aufgrund eines kurzen Timeouts schnell deaktiviert. Für die Aktivierung der Spracherkennung schlägt Cohen (1998) eine Zeitspanne von zwei Sekunden nach dem Präfix „Computer“ vor. Erkennt der Computer innerhalb dieser zwei Sekunden kein Input wird die Spracherkennung wieder deaktiviert. Der Ansatz bei Nipper hat einen ähnlichen Hintergrund, die Überlegung das Timeout nach dem Präfix „Computer“ zu starten wird jedoch nicht übernommen, da davon ausgegangen wird, das nach dem Befehl „Computer“ auch die Intention besteht, einen Sprachbefehl zu erteilen. Falls diese Annahme nicht korrekt wäre und der Benutzer keine Eingabe tätigen würde, bleibt die Spracherkennung aktiv. Sofern der Benutzer nichts sagt, hat dies auch keine Folgen für die Spracherkennung und sie bleibt aktiv. Spricht er mit einer anderen Person, wird sich die Spracherkennung wieder ausschalten. Das oben erwähnte Timeout wird erst dann gestartet, wenn die Spracherkennung einen Input erhält, diesen aber nicht verwerten kann, also eine Art „Plapper-Timeout“. Dies hat den Vorteil, dass der Benutzer die Sprachsteuerung nicht für jeden Befehl explizit aktivieren muss. Dies ist nur dann erforderlich, wenn der Benutzer für längere Zeit mit einer anderen Person spricht.

4.1.4 Kontextmenü

Zur Einstellung von Parametern für die Spracherkennung gibt es in Nipper ein Kontextmenü, welches über den Befehl „Kontextmenü“ oder „Spracherkennung“ eingeblendet wird. Damit das Kontextmenü auf dem Display nicht lange gesucht werden muss, wird dieses, wie auch schon die Icons für das Feedback, direkt an der Position der Maus eingeblendet. Im Gegensatz zu den Icons bewegt sich das Menü nicht mit dem Mauszeiger mit, sondern behält die Position bei. Da dieses Menü absichtlich vom Benutzer aufgerufen wird, kann dieses durchaus den aktiven Arbeitsbereich des Benutzers überdecken. Das Kontextmenü ist im Wesentlichen eine

Fernsteuerung für den Nipper-Client. Damit können Einstellungen zur Erkennungsgenauigkeit gemacht und der Nipper-Server neu gestartet oder ganz ausgeschalten werden. Sofern es das Netzwerk zulässt könnte mit diesem Neustart auch der Nipper-Client neugestartet werden und sich automatisch wieder mit dem Nipper-Server Verbinden. Diese Funktion wurde aufgrund technischer Schwierigkeiten bei Nipper deaktiviert. Je nach dem wo sich der Nipper-Client befindet, kann so eine Fernsteuerung durchaus praktisch sein.

Mit dem Nipper-Client und dem Nipper-Server wurde die Basis für die Spracherkennung und somit für das multimodale „Speak-and-Point"-Konzept geschaffen. Das Client-Server-System bietet eine Schnittstelle, um via Sprache mit anderen Anwendungen zu kommunizieren. Der Server stellt die Befehlsketten der Spracherkennung über UDP[18] bereit. Eine oder mehrere Anwendungen können auf diese Befehlsketten zugreifen, sofern sie diese interpretieren können.

4.2 NipMap: Nipper's Map Application

NipMap, die dritte Komponente des Nipper-Systems, auch „Nippers's Map Application" genannt ist die erste Anwendung dieses Systems, welche die Eingaben unterschiedlicher Eingabegeräte interpretieren kann und somit multimodale Interaktion auf der Powerwall ermöglicht. NipMap ermöglicht multimodale Interaktion mit Spracherkennung, kombiniert mit wahlweise Laserpointer oder Handgestenerkennung. Das statische Hintergrundbild von NipMap zeigt eine Luftaufnahme vom Containerhafen Rotterdam. Der Benutzer, beispielsweise der Hafenmeister, kann geometrische Primitive und Textfelder erstellen. Damit können Schiffe markiert, Bereiche gekennzeichnet, Route eines Containers visualisiert, Container beschriftet, Sperrzonen also solche dargestellt werden und vieles mehr. Dafür werden folgende Objekte benötigt:

- Rechteck
- Ellipse / Kreis
- Linie
- Polygonzug
- Freihand-Linie
- Textfeld

[18] Das User Datagram Protocol (UDP) wird verwendet, da damit keine direkten Verbindungen zwischen dem Server und den Anwendungen benötigt werden.

Rechteck, Ellipse und Textfeld sind bereits in NipMap integriert worden. Die restlichen Objekte müssen noch implementiert werden. Die zentralen Aufgeben können so allerdings bereits ausgeführt werden.

Bei NipMap wird versucht, möglichst viele Muster bereits bekannter Interaktionskonzepte alltäglicher Systeme zu übernehmen, sofern diese mit den Interaktionen bei LHRDs vereinbar sind. Die Übergänge von bekannten zu den neuen Konzepten sollen dabei möglichst sanft und intuitiv gestaltet werden. Dazu werden im Folgenden zwei neue Konzepte für NipMap vorgestellt, die dies ermöglichen sollen.

4.2.1 WYSIWYS: What you see is what you say

Wie in Kapitel 2.1 bereits erwähnt, können herkömmliche GUIs weder die Vorteile von LHRDs noch die Funktionalität multimodaler Benutzerschnittstellen optimal ausnützen. Im Zusammenhang mit der Spracherkennung treten bei normalen GUIs noch weitere Probleme auf. Graphische Menüs sind für die Verwendung von Spracherkennung nicht immer geeignet, da der Benutzer mit einem Icon nicht zwingend einen korrekten Sprachbefehl assoziiert. Dennoch sind graphische Menüs aus modernen GUIs nicht mehr wegzudenken. „Wenn Icons benutzt werden, um die Erkennung des Benutzers bezüglich einer Funktion, Objekts oder Namens zu fördern, dann sollten diese Eindeutig sein, den Erwartungen des Benutzers entsprechen und der Aufgabe angemessen sein.“ (ISO 9241-14:2003) Mit rein grafischen Mitteln beim Benutzer den richtigen Sprachbefehl zu assoziieren ist beinahe unmöglich. In der ISO-Norm wird beschrieben, dass „Text-Labels nicht benötigt werden, wenn die Icons selbstbeschreibend sind“ (ISO 9241-14:2003). Umgekehrt bedeutet dies, dass Funktionen, die mit Icons nicht eindeutig identifiziert werden können, ein Text-Label benötigen. Um sicher zu gehen, dass alle Funktionen mit dem korrekten Sprachbefehl assoziiert werden können, verwendet NipMap Icons und Text-Labels stets gemeinsam. Somit dient das Icon, wie in der ISO-Norm erwähnt, zur Wiedererkennung einer Funktion und um sicher zu gehen, dass das Icon vom Benutzer nicht „falsch ausgesprochen wird“, steht der Sprachbefehl in textueller Form gleich daneben. Dieses Konzept wird bei NipMap als „What you see is what you say“ (WYSIWYS) bezeichnet. Auf Deutsch: „Was du siehst, dass kannst du sagen“. Somit kann jedes Wort das im Menü angezeigt wird direkt als Sprachbefehl verwendet werden. Der Sprachbefehl könnte mit einem Mausklick bei einem herkömmlichen Menü verglichen werden, nur mit dem Vorteil, dass der Sprachbefehl auch dann ausgeführt werden kann, wenn kein Menü angezeigt.

Bei LHRDs ist die Verwendung von Text und Icons prinzipiell unproblematisch, da viel Platz zur Verfügung steht. Trotzdem soll dieser Platz nicht verschwendet werden. Daher soll das Menü gerade so gross angezeigt werden, dass der Text noch ohne weitere Schwierigkeiten gelesen werden kann. Ob der Text gelesen werden kann, ist abhängig von der Position des Benutzers, respektive vom Abstand zwischen dem Benutzer und dem Menü. Um das Menü an der richtigen Stelle und in der richtigen Grösse anzeigen zu können, soll ein Eye-Tracker verwendet werden, der zum einen die Blickrichtung zum andern die Position des Benutzers anzeigt. Ist kein Eye-Tracker verfügbar so wird das Menü relativ zum Mauszeiger in einer optimalen Standartgrösse angezeigt. Da für diese Funktion nicht die volle Genauigkeit des Eye-Trackers benötigt wird, würde es auch ausreichen, lediglich die Richtung und die Position des Kopfes zu erfassen. Diese würde genügend genaue Daten bezüglich Blickrichtung und Distanz zum Display liefern und wäre technisch weniger Anspruchsvoll als das Eye-Tracking[19].

Um flüssige Arbeitsabläufe zu ermöglichen, soll der Benutzer nicht durch Popups und Fenster unterbrochen werden (Guimbretière et al. 2001). Aus diesem Grund wird das Menü nicht im Zentrum des fokalen Bereichs angezeigt, sondern so, dass es aktive Objekte nicht überdeckt, aber dennoch im fokalen Bereich des Benutzers liegt. So kann der Benutzer, sollte er das Menü nicht verwenden wollen, weiter arbeiten, ohne dass das Menü störend wirkt. Wird das Menü benötigt, dann befindet sich dieses an einer optimalen Position und bietet dem Benutzer schnellen Zugriff auf die aktiven Funktionen. Das Menü kann vom Benutzer über die Sprachbefehle „Menü“ oder „Einstellungen“ aufgerufen und mit „Ok“ oder „Abbrechen“ wieder ausgeblendet werden. Das Menü wird aber auch dann angezeigt, wenn ein Sprachbefehl keine Parameter enthält. Der Sprachbefehl „Linienbreite“ öffnet das Menü direkt mit aktiviertem Slider für die Linienbreite (Vgl. Abbildung 16.c). Wird ein Sprachbefehl direkt mit Parametern eingegeben, also „Linienbreite 5 Pixel“, dann wird das Menü nicht angezeigt, da die Eingabe schon gemacht wurde. So soll gewährleistet werden, dass das Menü nur dann angezeigt wird, wenn es auch benötigt wird. Das System lässt auch kombiniert Befehle zu. So können mehrere Attribute in einem Sprachbefehl inklusive Parameter genannt werden. Fehlen bei einem Attribut die Parameter, wird dieses Attribut im Menü angezeigt, damit die Einstellungen mit dem Zeigegerät vorgenommen werden können. Die im Befehl bereits benannten Attribute werden im Menü verkleinert dargestellt, da diese Bereits eingestellt wurden. Der Benutzer kann die Sprachbefehle in beliebiger Reihenfolge sagen und das Menü wird in genau

[19] Siehe Kapitel 3.3.3: Beim Eye-Tracking müssen zusätzlich zur Position des Kopfes noch die Position der Pupillen erfasst werden.

dieser Reihenfolge aufgebaut. Der zuerst gesagte Sprachbefehl wird somit im Menü auch an erster Position angezeigt.

Abbildung 16:WYSIWYS-Menü in drei unterschiedlichen Stadien

Das Menü ist so gestaltet, dass nur die Funktionen angezeigt werden, welche beim aktuellen Arbeitsschritt benötigt werden. Ist das Rechteckwerkzeug selektiert, dann stehen Einstellungen zur Schrift nicht zur Verfügung, beim Kreiswerkzeug wird auch die Option für gerundete Ecken ausgeblendet. Um das Menü übersichtlicher zu gestalten, sind die Menüeinträge teilweise gruppiert. Wie alle anderen Menüeinträge können Untergruppen ebenfalls über den Namen der Gruppe geöffnet werden.

Passend zum WYSIWYS-Konzept ist das gesamte Menü in „Leserichtung“ orientiert. Diese horizontale Anordnung von links nach rechts, soll dem Benutzer die Eingabe mit der Sprache erleichtern. Die erste Stelle eines Menüeintrages wird durch das Icon belegt, welches zur schnellen Auffindung des Eintrages dienen soll. An zweiter Stelle steht der Text, welcher als Sprachbefehl verwendet werden kann. Gleich dahinter werden die Parameter in einem Feld angezeigt. Diese Anordnung soll dem Benutzer die direkte Spracheingabe verdeutlichen. Nach dem Lesen des Textes als Sprachbefehl folgt direkt der Parameter. Auch die Möglichkeit zur Verknüpfung von mehreren Sprachbefehlen soll so visualisiert werden, da auf das Text-Feld des Parameters direkt der nächste Menüeintrag folgt. So kann der Benutzer das Menü wie ein Lückentext mit seinen Parameter durchlaufen. Durch die oben erwähnte adaptive Anordnung der Menüoptionen, wird dieser Vorgang noch intuitiver gestaltet. Abbildung 16.a zeigt das Menü, wie es dem Benutzer angezeigt wird, solange noch nichts eingestellt wurde. Mit dem Befehl „Farbe“ öffnet sich in Abbildung 16.b das Untermenü für die Farbe. Der Befehl „Linienbreite“ öffnet in Abbildung 16.c den Slider zur Eingabe der Linienbreite. Der Benutzer hat nun die Möglichkeit die Linienbreite mit dem Slider oder direkt über die Sprache einzugeben. Die Funktionsweise des Menüs wird auch im Video (Anhang F) interaktiv demonstriert.

Zum WYSIWYS-Konzept gehört auch ein Hilfemenü. In diesem Hilfemenü werden alle Befehle angezeigt, die nicht im Hauptmenü vorhanden sind. Vom Prinzip her ist das Hilfemenü, mit Text und Icons, ebenfalls nach dem WYSIWYS-Konzept aufgebaut. Damit stehen dem

Benutzer alle im System verfügbaren Befehle zur Verfügung, wobei das Hilfemenü auch alternative Befehle einer Funktion anzeigt.

Alles in allem wurden bei diesem Konzept einige Faktoren berücksichtigt, die den Benutzer so wenig wie möglich unterbrechen sollen. Das Menü ist so konzipiert, dass es automatisch aufgerufen wird, wenn der Benutzer die Parameter nicht benennen kann. Die optimale Positionierung des Menüs und die Anordnung der Menüoptionen unterstützen den Arbeits- respektive den Sprachfluss des Benutzers. Kaum sind die Einstellungen vorgenommen, wird das Menü wieder ausgeblendet. Während das Menü dem unerfahrenen Benutzer im Umgang mit den Sprachbefehlen weiterhelfen soll, benötigt der erfahrene Benutzer das Menü lediglich dann, wenn er entweder einen Befehl vergessen hat oder die Parameter über einen Farbwähler oder einen Slider, mit Hilfe des Zeigegeräts einstellen möchte.

4.2.2 Input-Everywhere

Das Input-Everywhere-Konzept soll die Interaktion mit den Zeigegeräten effizienter gestalten. Dieses Konzept kann sowohl bei Transformationen als auch bei Parametereingaben im Menü angewendet werden. Die Idee bei diesem Konzept ist, die ganze Fläche des Displays zur Eingabe zu benutzen.

Zum Vergrössern eines Objektes muss nicht, wie bei herkömmlichen Grafikprogrammen, eine Ecke oder Kante genau ausgewählt werden. Dies ist aufgrund der in Kapitel 3.3 erwähnten Ungenauigkeit von Freihand-Eingabegeräten sehr schwierig. Mit NipMap ist lediglich die relative Positionsänderung des Zeigegerätes während des Klicks für die Transformation des Objekts entscheidend. Soll beispielsweise ein Rechteck an der linken, oberen Ecke aufgezogen werden, so muss kein Anfasser angeklickt werden. Es reicht, nach dem Befehl „Vergrössern“ das Zeigegerät nach links oben zu bewegen. Durch diese initiale Richtung wird die entsprechende Ecke oder Kante ausgewählt. Die Grösse und genaue Richtung der Vergrösserung bestimmt das Zeigegerät, welches sich so verhält, als wäre der Mauszeiger zur selektierten Ecke des Objekts gesprungen. Es spielt dabei keine Rolle, ob sich der Mauszeiger innerhalb oder ausserhalb des Objekts befindet. Um der Erwartungskonformität gerecht zu werden, kann der Benutzer auch die Ecke selbst festhalten. Die Objekte haben absichtlich keine Anfasser und nur eine ganz dünne Linie, die sie als selektiert bezeichnet. Dadurch fällt es dem Benutzer schwerer die Ecke genau zu treffen. Damit soll erreicht werden, dass der Benutzer trotz vermeintlichem „nicht Treffen“ der Ecke die gewünschte Aktion ausführen kann. Mit der Zeit könnte so dem Benutzer klar werden, dass diese Aktion unabhängig von der aktuellen Position, sondern lediglich abhängig von der Positionsänderung bestimmt wird. Diese Kom-

bination aus Erwartungskonformität und Selbstbeschreibungsfähigkeit soll dem Benutzer den Umgang und das Verständnis für diese Interaktionstechnik vermitteln.

Das Prinzip der weiteren Transformationen ist ähnlich. Beim Verschieben wird das Objekt relativ zur vorherigen Zeigerposition verschoben. Bei der Rotation löst eine Bewegung des Zeigers nach links eine Rotation nach links aus, wobei die Distanz zur ursprünglichen Zeigerposition den Winkel bestimmt.

Auch bei der Interaktion mit dem Menü wird dieses Konzept verwendet. Die bei NipMap verwendeten Parameter sind entweder Pixel- oder Farbangaben. Für die Farbangaben wurde ein Farbwähler von Microsoft verwendet, welcher dieses Konzept nicht unterstützt. Die restlichen Parameter können mit Hilfe dieses Konzepts verändert werden.

Wie in den gängigsten Zeichenprogrammen können Attribute, wie die Linienbreite, über einen Slider eingestellt werden. Diesen Slider zu bedienen ist mit den bei NipMap verwendeten Zeigegeräten äusserst ineffizient, da der Slider deutlich vergrössert werden müsste, damit dieser schnell und präzise getroffen würde. Durch die Anwendung des Input-Everywhere-Konzepts kann der Slider, ähnlich wie bei Transformationen, durch die relative Positionsänderung des Zeigegeräts bewegt werden. Der Slider wird über den entsprechenden Sprachbefehl aktiviert. Ist ein Slider aktiv, kann das gesamte Display als Slider benutzt werden. Dieser wird jedoch trotzdem angezeigt, da dieser den Benutzer auf die Art der Interaktion hinweisen soll. Abgesehen von der Schwierigkeit den Slider zu treffen, spricht nichts dagegen den Slider auf herkömmliche Weise zu bedienen. Auch in diesem Fall könnte ein unerfahrener Benutzer das Konzept selber entdecken, wenn trotz nicht getroffenem Slider die Aktion ausgeführt wird.

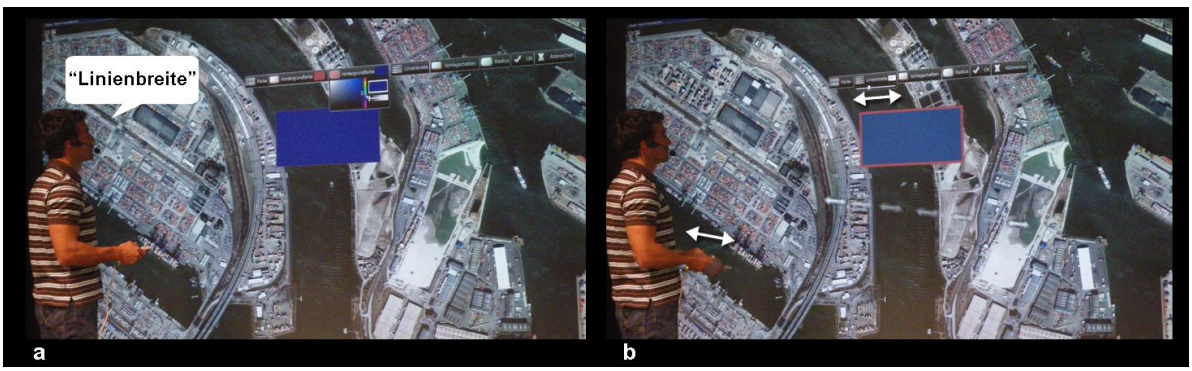

Abbildung 17: Multimodale Eingabe für die Linienbreite

In Abbildung 17 kann der Benutzer dank dem Input-Everywhere Konzept die Linienbreite mit einem einfachen Befehl aktivieren (17.a) und deren Stärke durch die relative Positionsänderung einstellen (17.b). Um die Konsistenz des Systems zu gewährleisten verändert sich nicht

nur die Linienbreite, es reagieren auch Slider und Textfeld im Menü auf die Eingabe des Zeigegeräts.

Mit dieser Interaktionstechnik spielen Ungenauigkeiten bei Zeigegeräten eine untergeordnete Rolle. Und auch wenn die Präzision der Zeigegeräte in Zukunft noch verbessert wird, dürfte dieses Konzept deutlich schneller sein, da der Benutzer die Hand nicht einmal in die Richtung des Interaktionselementes bewegen muss. Alle Eingaben beziehen sich lediglich auf relative Positionsänderungen und sind so völlig unabhängig von der Position des Benutzers oder Zeigegeräts.

Mit den beiden vorgestellten Interaktionstechniken können die Funktionen von NipMap effizient benutzt werden.

4.2.3 Funktionsweise von NipMap

Mit Hilfe der vorgestellten Konzepte lassen sich die Funktionen von NipMap einfach bedienen. Im Folgenden werden diese Funktionen und deren Funktionsweise Schritt für Schritt erläutert.

Zeichnen von Objekten: Einer der ersten Schritte bei NipMap ist das Erstellen respektive Zeichnen von neuen Objekten. Da Positionsangaben via Sprachsteuerung sehr umständlich und ineffizient sind (Vgl. Kapitel 3.2), werden solche Aufgaben mit einem Zeigegerät erledigt. NipMap ist so konfiguriert, dass jedes Eingabegerät, welches die Funktionen einer Maus steuern kann, dafür verwendet werden könnte. Aus bereits benannten Gründen sind Laserpointer und Handgestenerkennung sehr gut dafür geeignet.

Das Erstellen von Objekten wird, wie die meisten Interaktionen, von einem Sprachbefehl eingeleitet. Sofern die Spracherkennung nicht deaktiviert ist, aktivieren Befehle wie „Rechteck“, „Kreis“ oder „Text“ die entsprechenden Werkzeuge. Um der Erwartungskonformität nach der ISO 9241-10 gerecht zu werden, hat jedes Werkzeug, wie dies auch bei handelsüblichen Zeichenprogrammen der Fall ist, einen eigenen visuellen Mauszeiger. Dieser Mauszeiger ändert sich unmittelbar nach Erkennung des Sprachbefehls, als ob der Benutzer in einem herkömmlichen Zeichenprogramm auf das Icon für das gewünschte Werkzeug geklickt hätte. Diese Veränderung des Mauszeigers ist von enormer Wichtigkeit, da dies, das einzige Feedback auf einen korrekt erkannten Sprachbefehl ist. Bei fehlerhafter Erkennung wird das von Nipper dafür vorgesehene Icon angezeigt. Ist das Werkzeug erstmal selektiert, können beliebig viele Objekte des selektierten Typs gezeichnet werden. Da die von Nipper verwendeten Eingabegeräte auch Mausklicks erzeugen können, funktioniert das Zeichnen von Objekten genau so wie

bei normalen Zeichenprogrammen. Von der Positionierung über sprachliche Attribute, wie bei Bolt's Media Room, wird hier abgesehen, da Positionsbestimmungen via Sprache auf einem LHRD noch ineffizienter ist, als bei normalen Displays. Einzig zur exakten Positionierung von Objekten könnten relative Positionsänderungen zu einer, durch das Zeigegerät bestimmten Position mit Sprachbefehlen sinnvoll sein. Bei Zeichenprogrammen ist es häufig der Fall, dass ein Objekt um genau einen Pixel verschoben werden muss. Mit einem Zeigegerät werden häufig mehrere Versuche benötigt, um genau den richtigen Pixel zu treffen. Bei solchen Detailaufgaben könnten Sprachbefehle die exakten Angaben zur Positionierung steuern. Auch falls ein Objekt beispielsweise genau 100 Pixel vom linken Rand entfernt sein soll, ist die Positionierung über die Sprache schneller als über ein Zeigegerät. Im Allgemeinen sollten, wie von Oviatt und Kuhn (1997) vorgeschlagen, Objektpositionen und andere Positionsangaben über ein Zeigegerät getätigt werden.

Selektion: Prinzipiell ist immer das zuletzt gezeichnete oder veränderte Objekt selektiert. Dies wird durch einen dünnen, hellblauen Rahmen angezeigt. Zur Selektion anderer Objekte gibt es zwei Möglichkeiten. Wie auch beim Erstellen der Objekte aktivieren auch hier Sprachbefehle wie „Selektieren" oder „Auswählen" den Selektionsmodus. Analog ändert sich der Mauszeiger bei dieser Aktion in eine „Hand", welche die Objekte greifen kann. Wiederum wird so unmittelbar nach ausgeführtem Sprachbefehl ein visuelles Feedback gegeben. Ist der Selektionsmodus aktiv, wird ein Objekt durch klicken auf dasselbe selektiert. Die zweite Variante ist erst ansatzweise implementiert und wurde erst mit einem simulierten Auge getestet. Bei dieser Variante ist das Auge für die Selektion verantwortlich. Wird ein Eye-Tracker an das System angeschlossen, dann können Objekte mit den Augen selektiert werden. Dieses Selektionsverfahren ist eine Abwandlung von Bolt's deiktischer Selektion. Während dem Betrachten eines Objektes, kann dieses mit dem weniger abstrakten Sprachbefehl „Dieses Objekt auswählen / selektieren" selektiert werden. Der Sprachbefehl kann aber auch schon selektive Elemente enthalten. Liegen zwei unterschiedliche Objekte übereinander, kann ein verdecktes Objekt, durch Benennung von Attributen, direkt ausgewählt. Im illustrierten Beispiel in Abbildung 18 kann der Benutzer das Rechteck, welches sich hinter dem Kreis befindet durch blosses Anschauen[20] und dem Befehl „Rechteck auswählen" selektieren (18.a). In Abbildung 18.b wurde das Rechteck bereits Selektiert. Ohne den Kreis zu verändern und ohne direktes Klicken auf das Rechteck, kann dieses dank des Input-Everywhere-Konzepts verschoben werden (18.c, 18.d). Wie oben bereits erwähnt könnte auch hier die Richtung und Position des

[20] Der Eye-Tracker konnte aus technischen Gründen nicht verwendet werden, daher wurde ein virtuelles Auge auf die Position des Mauszeigers gelegt. Das System ist jedoch bereist so konfiguriert, dass der Eye-Tracker angeschlossen werden kann.

Kopfes zur Selektion eines Objekts ausreichen. Denn durch die Benennung der Attribute kann ein Objekt auch dann selektiert werden, wenn dessen Position nicht exakt mit der errechneten Position des Auges übereinstimmt. Das System würde dann das nächstgelegene Objekt mit den genannten Attributen auswählen. Durch dieses Selektionsverfahren kann ein verdecktes Objekt bearbeitet werden, ohne dass das verdeckende Objekt verschoben werden muss. Diese selektiven Befehle können auch auf die Position des Mauszeigers angewandt werden, machen dann aber nur bei verdeckten Objekten Sinn. Für normale Selektion ist ein Klick schneller.

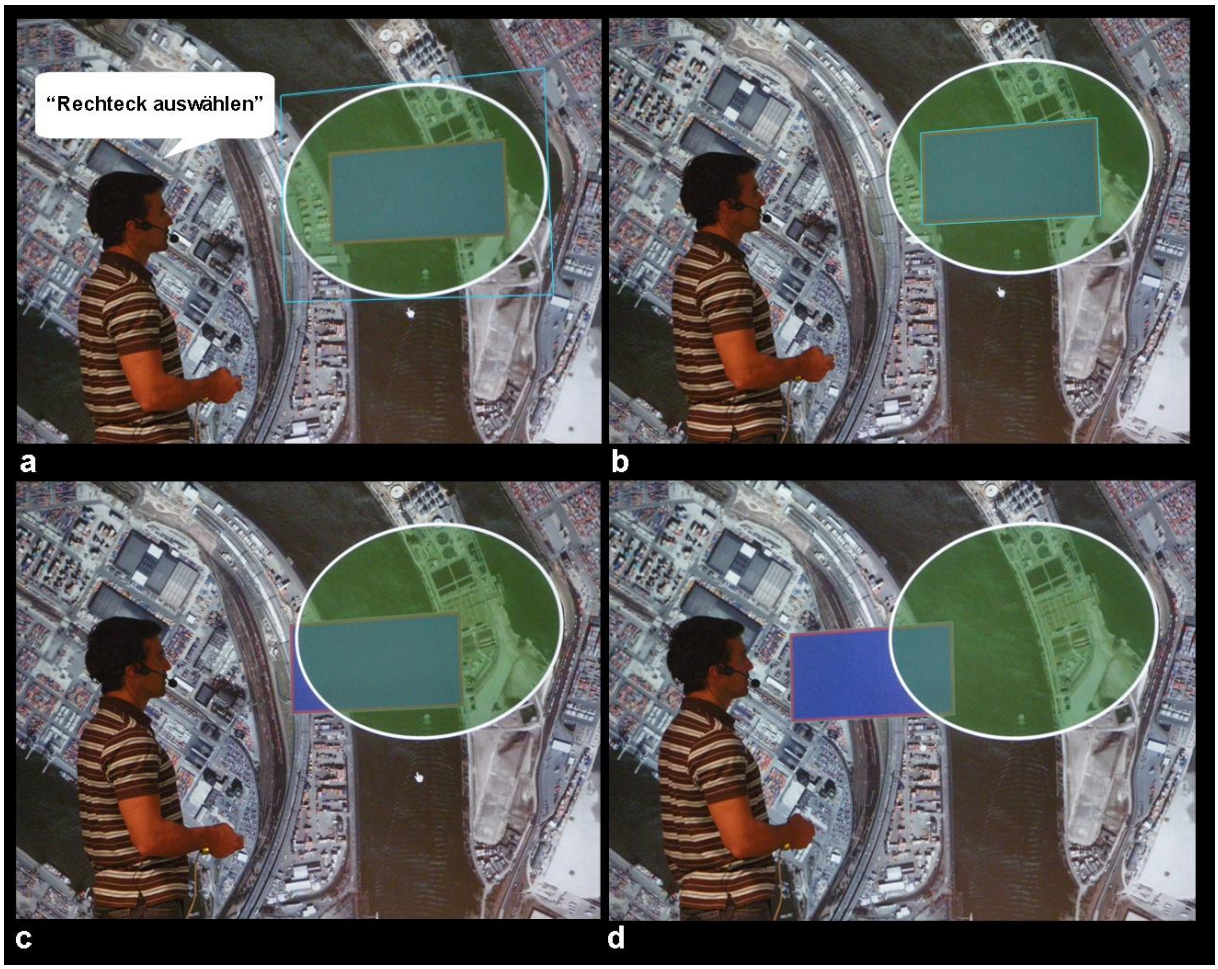

Abbildung 18: Selektion eines verdeckten Objektes

Bolt konnte im Media Room die Objekte nur diskret manipulieren. Bei NipMap hingegen können Pixelzahlen sowie Farben stetig und diskret manipuliert werden. Dank multimodaler Interaktion können folgende Attribute auf beide Arten verändert werden:

- Farbe
  - Vordergrundfarbe
  - Hintergrundfarbe
- Linienbreite

- Runde Ecken (beim Rechteck)
- Schlagschatten-Effekt
    - Tiefe
    - Rauschen
    - Transparenz

Dabei wird dem Benutzer überlassen, ob dieser die Parameter der Attribute unimodal über die Sprache oder multimodal über Sprache und Zeigegerät eingeben möchte.

Parametereingabe via Sprache: Die Parametereingabe via Sprache ist dann sinnvoll, wenn die Parameter des Attributs bekannt sind. Je nach Attribut stehen dem Benutzer vordefinierte, diskrete oder stetige Parameter zur Auswahl. Die Parameter des Attributs „Linienbreite“ lassen sich über den kombinierten Befehl „Linienbreite <ZAHL> Pixel“ ändern. Wobei <ZAHL> eine beliebige Zahl und das Suffix „Pixel“ fakultativ ist.

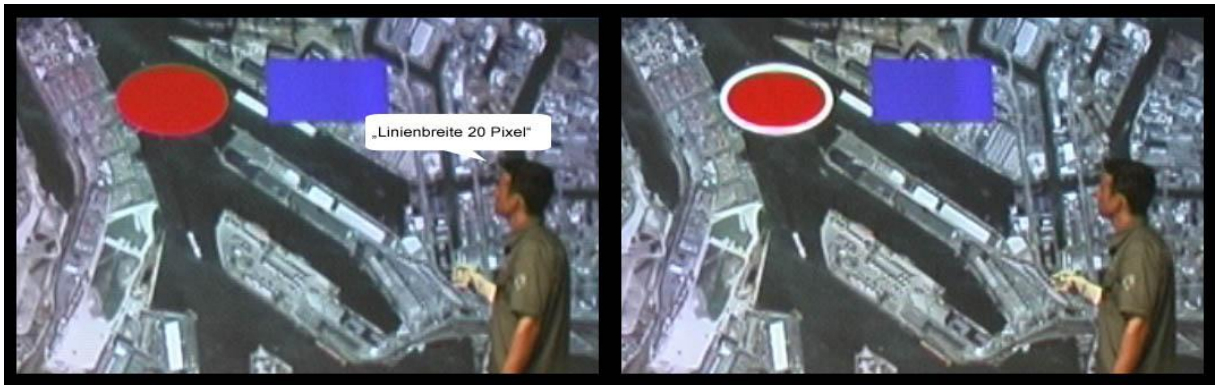

Abbildung 19: Das System reagiert direkt auf die Spracheingabe, mit einer 20 Pixel dicken Linie

Mit oder ohne Suffix „Pixel“ generiert die SAPI den semantischen Schlüssel „::LINEWIDTH_5_“ und sendet diesen via Server an NipMap. Es wäre denkbar, dass Anstelle der Zahl ähnlich wie bei Bolt (1980) ein diskretes Attribut folgt. Diese Attribute müssten dann vom System vordefiniert werden. So könnten Attribute wie „dick“, „dünn“ oder „fein“ als zusätzliche Sprachbefehle definiert werden. Diese Attribute müssten dann zusätzlich im System vom Benutzer definiert werden können. Analog kann der Benutzer auch Farben direkt über die Sprache festlegen, sofern dieser die Farbe benennen kann. Die Farbgebung ist wie bei Zeichenprogrammen üblich, in Vordergrund- und Hintergrundfarbe aufgeteilt. Der Sprachbefehl „Vordergrundfarbe Rot“ führt dazu, dass die Vordergrundfarbe oder auch Linienfarbe des Objekts rot eingefärbt wird. Dabei wird von Nipper wiederum ein semantischer Schlüssel „::FOREGROUNDCOLOR_RED_“ generiert auf den NipMap direkt mit der Ände-

rung der Objektfabre reagiert. Unabhängig davon, welche Parameter eingestellt werden, erhält der Benutzer stets direktes Feedback nach erfolgreicher Eingabe.

Multimodale Parametereingabe: Sprachliche Definitionen von Farbe sind im Gegensatz zu Pixelzahlen relativ schwierig. Es gibt unzählige Farbnamen, die den meisten Menschen nicht geläufig sind. Im Internet finden sich diverse Listen mit gegen 180 Farbnamen[21] wie beispielsweise „Frühlingsgrün“, „Geisterweiss“ „Goldrute“ und viele mehr. Für den Computer ist es kein Problem diese Farben der Sprachgrammatik zuordnen zu können. Dem Benutzer werden die meisten Farbnamen jedoch kaum geläufig sein. Die Benennung aller 16.7 Millionen Farben, die aktuelle Systeme anzeigen können, ist hingegen völlig unmöglich. Daher wird dem Benutzer die Möglichkeit gegeben die Farben, wie bei bekannten Grafikprogrammen, über ein Farbwähler auszuwählen. Dazu wird die Spracheingabe mit dem Zeigegerät kombiniert. Über den Sprachbefehl „Vordergrundfarbe“ wird der Farbwähler aktiviert und der Benutzer kann wie, bei jedem herkömmlichen Grafikprogramm, die gewünschte Farbe mit dem Zeigegerät selektieren.

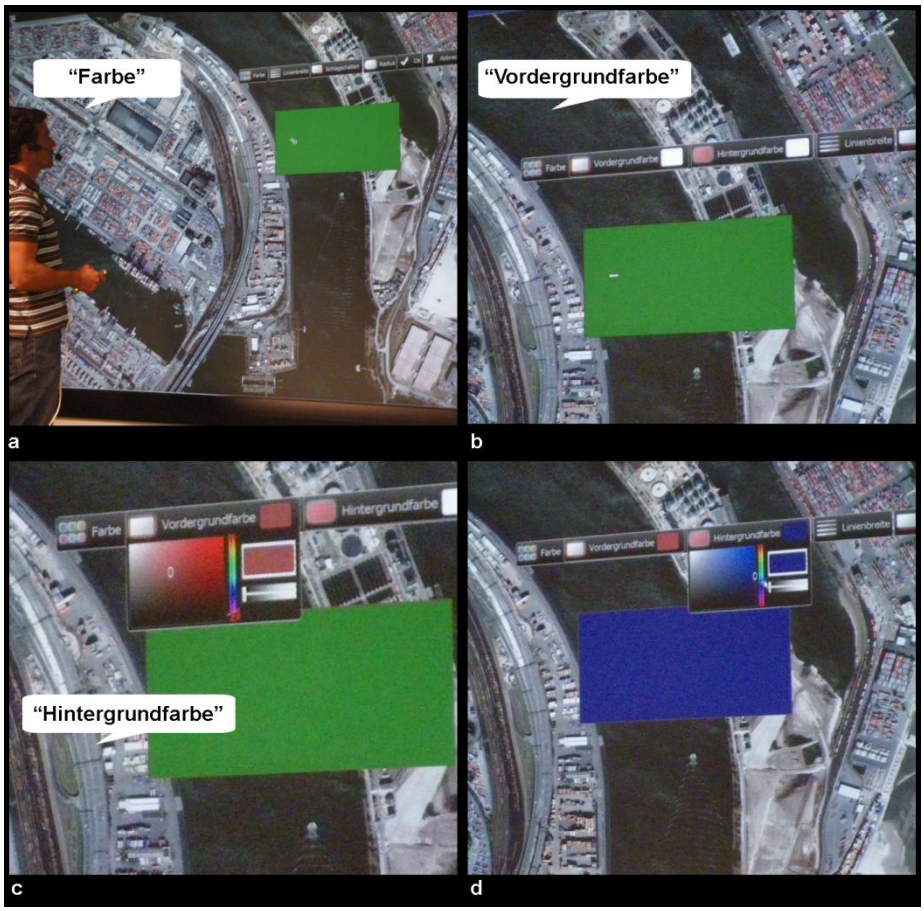

Abbildung 20: Multimodale Farbselektion mit Sprache und Handgesten

[21] Nach j-a-b.net sind dies 176 unterschiedliche Farbnamen (Brückmann 2005).

Abbildung 20 zeigt wie ein Benutzer via Sprachbefehl die Vorder- und Hintergrundfarbe eines Rechtecks einstellt. In 20.a gibt er den Befehl zum Öffnen des Farbmenüs. In 20.b reagiert das System und zeigt das Untermenü der Farbe an. Über den Befehl „Vordergrundfarbe“ öffnet der Benutzer den Farbwähler (20.c) und wählt einen roten Farbton aus. Analog für die Hintergrundfarbe in 20.d.

Ähnlich geschieht dies auch mit den anderen Attributen, nur das dann in der Regel ein Slider ausreicht, um die Parameter zu bestimmen. Im oben erwähnten Fall der Linienbreite kann der Benutzer über den Sprachbefehl „Linienbreite“ ohne weitere Parameter einen Slider aktivieren (Vgl. Abb. 17). Dank dem Input-Everywhere-Konzept kann der Benutzer nun wie im Kapitel 4.2.2 erwähnt die Parameter mit dem Zeigegerät verändern. Wie bei handelsüblichen Zeichenprogrammen wird auch hier die Linienbreite direkt am Objekt verändert, damit der Benutzer Feedback zur Interaktion erhält.

Multiple Parametereingabe: Bei NipMap kann der Benutzer mehrere Attribute mit Parameter auf einmal verändern. Mit einem kombinierten Sprachbefehl kann der Benutzer beliebig viele Attribute und in beliebiger Reihenfolge mit den dazugehörigen Parametern diktieren. Es ist somit dem Benutzer überlassen, ob dieser für ein Rotes Rechteck mit weisser Linie, die 10 Pixel stark ist, einzelne Befehle

„Rechteck“ – „Hintergrundfarbe Rot“ – Vordergrundfarbe Weiss“ – „Linienbreite 10 Pixel“

oder in einem Satz

„Rechteck Hintergrundfarbe Rot Vordergrundfarbe Weiss Linienbreite 10 Pixel“

diktieren möchte. Auch das Ändern der Reihenfolge hat keinen Einfluss auf die Interaktion. Der Befehl

„Rechteck Vordergrundfarbe Weiss“ – „Linienbreite 10 Hintergrundfarbe Rot“

liefert das gleiche Ergebnis wie die zwei vorherigen Befehle. Bei NipMap muss lediglich das Werkzeug, sofern dieses noch nicht Ausgewählt wurde, an erster Stelle genannt werden. Mit diesen Massnahmen versucht NipMap der Steuerbarkeit nach ISO 9241-10:1995 gerecht zu werden.

Abbildung 21 zeigt wie ein Benutzer einen kombinierten Befehl gibt (21.a) und dann das Rechteck mit den bereits benannten Einstellungen zeichnet (21.b). Durch das Aufrufen des

Menüs wird in 21.c die Farboption bereits verkleinert dargestellt, da diese im kombinierten Befehl schon vollständig eingestellt wurde.

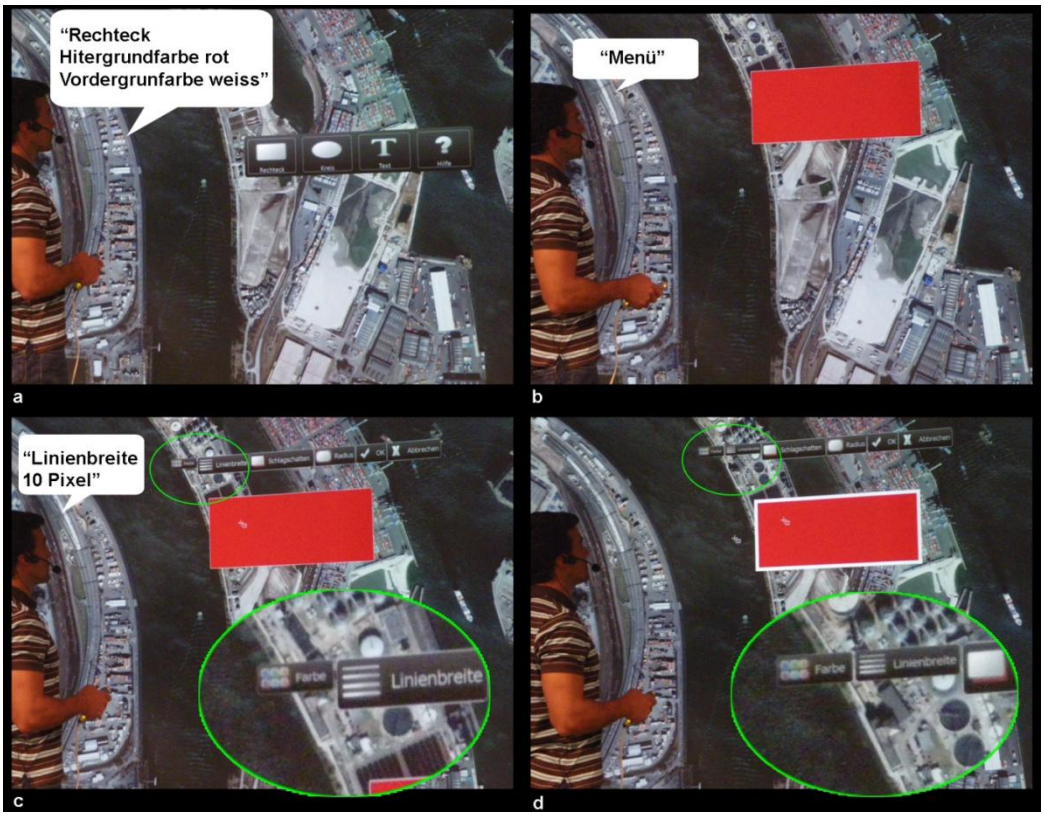

Abbildung 21: Der Benutzer spricht mehrere Befehle in einem Satz. Attribute die bereits eingestellt wurden, werden verkleinert und an erster Stelle im Menü dargestellt.

Transformationen: Neben den oben genannten Attributen kann der Benutzer auch physikalische Veränderungen an den Objekten vornehmen.

- Verschieben
- Vergrössern
- Verkleinern
- Drehen

Auch Transformationen lassen sich durchaus unimodal via Spracherkennung steuern. Möchte der Benutzer ein Objekt um 45 Grad drehen, so könnte dies analog zu den oben genannten Befehlen ausgeführt werden: „Drehen 45 Grad“ (Sharma et al. 2000). Mit einer entsprechend ausgelegten Grammatik könnte der Befehl auch „45 Grad Drehen“ lauten. Auch beim Vergrössern oder Verkleinern von Objekten könnten entweder absolute Angaben „Vergrössern 300 mal 400 Pixel“ oder relative Angaben „Vergrössern um 10%“ zum gewünschten Ergebnis führen. Die aktuelle Version von NipMap unterstützt allerdings nur direkte Manipulation mit

dem Zeigegerät, da diese schneller und intuitiver erscheint. Deren Funktionsweise wurde bereits im Zusammenhang mit dem Input-Everywhere-Konzept erklärt.

Mit NipMap wurde eine Anwendung realisiert, die der Vision der freien, natürlichen Interaktion schon sehr nahe kommt. Mit der Stimme und den Händen können alle Funktionen von NipMap bedient werden. Dank den beiden Konzepten, Input-Everywhere und WYSIWYS, können diese „natürlichen Eingabegeräte“ optimal angewendet werden. Aber auch die kleineren Konzepte, tragen einen wichtigen Teil zu diesem Interaktionskonzept als ganzes bei. Durch die automatische Deaktivierung der Spracherkennung können ohne weiteres Konversationen zwischen Benutzern stattfinden. Das visuelle Feedback informiert den Benutzer optimal über die aktuellen, unsichtbaren Vorgänge der Spracherkennung. Wie gut diese Konzepte vom Benutzer verstanden werden und ob diese die Interaktion wirklich effizienter gestalten, soll in der folgenden Evaluation überprüft werden.

5 Evaluation

Während der Entwicklung von Nipper und NipMap wurden die einzelnen Komponenten jeweils in ruhiger, kontrollierter Umgebung und immer von der gleichen Person getestet. Da das System für die Powerwall konzipiert wurde, ist es unerlässlich zu überprüfen, ob das System auch von anderen Personen bedient werden kann. Dabei sind zwei Aspekte zu beachten:

- Es soll überprüft werden, ob die Spracherkennung andere Personen erkennt und wie viel Training dafür notwendig ist.
- Es soll überprüft werden, ob der Benutzer die Konzepte von NipMap versteht und anwenden kann. Dabei soll vor allem darauf geschaut werden, wie intuitiv und effizient diese Konzepte bedient und eingesetzt werden.

Um dies zu Überprüfen wurde NipMap mit sechs Studenten und Mitarbeiter der Universitäten Konstanz und Zürich getestet. Die Ergebnisse dieser qualitativen Evaluation basieren auf Beobachtungen und Befragungen. Von prozentualen Angaben wird abgesehen, da hier auch Bemerkungen oder Reaktionen einzelner Personen entscheidende Informationen beinhalten können.

5.1 Hypothesen

Bei der Entwicklung von NipMap wurden neue Konzepte entwickelt welche bei der Evaluation überprüft worden sind. Zu diesen Konzepten wurden drei Hypothesen Aufgestellt. Eine weitere Hypothese betrifft die SAPI 5.3 und deren Erkennungsgenauigkeit.

1. *Die SAPI 5.3 wird mit nur wenig Training unterschiedlichste Stimmen erkennen können.* Basierend auf unbestätigten Vermutungen kann die SAPI 5.3 die meisten Benutzer bereits nach einem Durchgang des Sprachtrainingsprogrammes so gut verstehen, dass die Nipper-Befehle erkennt werden können. Jedoch dürften zur genauen Erkennung von diktiertem Text zusätzliche Trainingsdurchläufe nötig sein.
2. *Das WYSIWYS-Konzept erleichtert dem Benutzer die Bedienung von NipMap und liefert diesem die dafür nötigen Informationen und Befehle.* Es soll gezeigt werden, dass alle Informationen die der Benutzer zur Interaktion benötigt, durch das WYSIWYS-Konzept bereitgestellt werden und dass dieses Konzept die Spracheingabe erleichtert.
3. *Das Input-Everywhere erleichtert dem Benutzer die Bedienung mit ungenauen Eingabegeräten, da kein genaues Zielen für die Interaktion erforderlich ist.* Als wesentlicher Punkt dabei ist zu beobachten, ob der Benutzer das Konzept von sich aus versteht und

wie er dieses anwendet. Zusätzlich soll beobachtet werden ob Unterschiede zwischen den beiden verwendeten Eingabegeräten auftreten.

4. *Die Benutzer bevorzugen unimodale Interaktion zur Parametereingabe.* Es wird davon ausgegangen, dass ein Benutzer das Konzept der direkten Parametereingabe über die Sprache favorisieren wird, da dies einfacher und schneller ist.

5.2 Teilnehmer

Für die Evaluation von NipMap wurden 6 Teilnehmer ausgewählt, 3 Männer und 3 Frauen[22]. Die Teilnehmer waren entweder Studenten oder Mitarbeiter der Universitäten Konstanz und Zürich aus den Fachbereichen Psychologie, Agrarwissenschaften und Informatik. Die Hälfte der Versuchspersonen war nicht im Bereich von Informatik oder Informationstechnologie tätig. Damit sollte verglichen werden, ob computeraffine Versuchspersonen ähnliche Probleme mit dem System haben, wie die anderen Versuchspersonen. Die Gruppe der „Nicht-Informatiker" hatten zuvor noch nie mit einem LHRD, einem Laserpointer oder mit Gestenerkennung gearbeitet. Lediglich eine Person hatte einmal bei einer Telefonhotline mit Spracherkennung Kontakt gehabt. Die Gruppe der Informatiker hingegen, war mit den verwendeten Gerätschaften im Allgemeinen, abgesehen von der Spracherkennung, vertraut oder wussten zumindest wie diese theoretisch funktionieren. Zeichnungsprogramme wie MS Paint oder Adobe Photoshop waren allen Testpersonen bekannt. Die Teilnehmer waren im Alter zwischen 21 und 29. Eine Person stammte aus der Schweiz, die übrigen Personen aus Deutschland.

5.3 Materialien

Für die Spracherkennung wurde ein eigener PC eingerichtet, der lediglich zu diesem Zweck verwendet wurde. Dieser PC wurde mit Windows Vista Business betrieben, da diese Version standartmässig die Spracherkennung SAPI 5.3 beinhaltet. Jeder Benutzer erhielt ein eigenes Benutzerkonto, welches teilweise nach dem Test wieder gelöscht worden ist. Diese Benutzerkonten sollten garantieren, dass jeder Benutzer ein eigenes Sprachprofil hatte. Zur Sprachaufzeichnung wurde das oben erwähnte Sennheiser EW 352 Bügelmikrofon verwendet. Als Zeigegeräte wurden ein Laserpointer (König et al. 2007a) und ein Handschuh mit Reflektoren (Föhrenbach et al. 2008) für die Gestenerkennung verwendet. Das Menü wurde mit dem Zeigegerät positioniert, da kein Eye-Tracker zur Verfügung stand.

[22] Der Ausdruck „Teilnehmer" lässt im Folgenden kein Rückschluss auf das Geschlecht zu, sondern wird für alle Versuchspersonen gleich verwendet.

5.4 Aufgaben

Die Teilnehmer mussten im Verlauf des Experiments vier Teilaufgaben bearbeiten. Die erste Teilaufgabe befasste sich alleine mit der Spracherkennung, die restlichen Teilaufgaben mit NipMap.

- *Spracherkennung*: Der Teilnehmer musste, nachdem ihm das Mikrofon angepasst worden ist, fünf Sätze diktieren. Danach wurde das System mit dem Programm zur Verbesserung der Spracherkennung von Windows Vista trainiert. Nach dem der Teilnehmer das Training abgeschlossen hatte, wurden erneut die gleichen Sätze diktiert. Die beiden Diktate wurden je in einem Textfile zu Analysezwecken abgespeichert.
- *NipMap Exploration:* Nach einer kurzen Einführung in die wesentlichen Eigenheiten von Nipper wurde dem Teilnehmer, ohne weitere Erklärungen zu NipMap, fünf einfache Übungsaufgaben gestellt (Siehe Anhang B). Dabei wurde beobachtet, wo der Teilnehmer Hilfe in Anspruch nehmen musste.
- *NipMap Set A:* Nach einer kurzen Einführung in die Konzepte von NipMap musste der Teilnehmer ein Aufgabenset von sieben etwas komplizierteren Aufgaben bearbeiten (Siehe Anhang C). Der Aufbau einer Aufgabe entsprach den einzelnen Schritten der Übungsaufgaben. Der Benutzer musste somit die einzelnen Schritte der Übungsaufgaben für eine einzelne Aufgabe kombinieren. Bei den ersten vier Aufgaben mussten neue Objekte erstellt werden. Zweimal wurde das Objekt mit konkreten Farben und Pixelzahlen definiert und zweimal wurde das Objekt nur vage umschrieben. Bei den letzten drei Aufgaben mussten die bereits erstellten Objekte abgeändert werden. Für die Aufgabensets wurde alternierend der Laserpointer oder der Handschuh als Zeigegerät verwendet.
- *NipMap Set B:* Das Aufgabenset B beinhaltete ähnliche Aufgaben wie Set A, wobei nun das Zeigegerät gewechselt wurde (Siehe Anhang D). Alternierend haben die Teilnehmer mit Set A oder Set B begonnen.

5.5 Ablauf

Alle Teilnehmer wurden einzeln von einem bis maximal zwei Betreuer getestet. Wobei der zweite Betreuer lediglich für die Bedienung der Videokamera zuständig war und mit dem Test an und für sich nichts zu tun hatte. Nach Unterzeichnung der Einverständniserklärung wurde von jedem Teilnehmer ein Pre-Test Fragebogen (Siehe Anhang A) ausgefüllt. Danach gab es eine kurze Einführung zur Spracherkennung. Das Diktat dauerte im Schnitt zwei Minuten. Danach wurde für 4-5 Minuten die Spracherkennung trainiert. Darauf folgte das zweite Dik-

tat. Dem Benutzer wurde dann die Bedeutung des Feedbacks erklärt und wie die Spracherkennung ein- und ausgeschaltet werden konnte. Nach ein paar Test-Befehlen, die benötigt wurden um die Erkennungsgenauigkeit beim Nipper-Server anzupassen, wurden dem Teilnehmer die Übungsaufgaben zur Bearbeitung gegeben. Während den Übungsaufgaben durfte bei Schwierigkeiten Fragen gestellt werden. Danach folgten eine kurze Einführung in die Bedienelemente von NipMap sowie ein kurzer Überblick über die Funktionsweise der Konzepte. Aufgabenset A und Aufgabenset B wurden alternierend mit Laserpointer oder Handgesten bearbeitet. Die Aufgabensets mussten alleine durchgeführt werden. Bei Fehlern im System oder nicht implementierten Funktionen wurde dem Teilnehmer weitergeholfen. Wollte ein Teilnehmer beispielsweise ein „hellgrünes“ Rechteck zeichnen, wurde er darauf hingewiesen, dass die Farbe Hellgrün dem System nicht bekannt ist. Um die Benutzung des WYSIWYS-Konzeptes zu forcieren sind in den Menüs keine Interaktionen mit Mausklicks möglich. Nach der Abarbeitung der Aufgabensets welche im Schnitt etwa 20 Minuten dauerte, wurde der Teilnehmer basierend auf einem Fragekatalog (Siehe Anhang E) zu den Punkten Spracherkennung, Interaktion und Eingabegeräte befragt. Die Teilnehmer erhielten 5 Euro für die Teilnahme, welche im Schnitt 60 Minuten dauerte.

5.6 Ergebnisse

Die Ergebnisse werden analog zu den vier Teilaufgaben separat betrachtet. Dabei ist zu beachten, dass hier teilweise auch auf spezielle Beobachtungen eingegangen wird, die nur bei einer einzelnen Versuchsperson aufgetreten sind.

5.6.1 Spracherkennung

Bei dem Test zur Spracherkennung sollte getestet werden, wie gut die Diktierfunktion ohne teilnehmerspezifisches Training ist und wie gross der Unterschied zum zweiten Diktat nach dem Training war. Die Ergebnisse waren sehr unterschiedlich. Beim ersten Diktat traten 0 – 8 Fehler auf. Beim zweiten Diktat war die Fehlerrate mit 0 – 3 Fehlern deutlich niedriger. Wobei anzumerken ist, dass derjenige Benutzer, der beim ersten Diktat null Fehler gemacht hatte, beim zweiten Diktat zwei Fehler machte. Dies könnte darauf zurückzuführen sein, dass sich der Benutzer nach dem Sprachtraining sicherer fühlte und dementsprechend das zweite Diktat weniger genau diktierte. Die erste Hypothese „*Die SAPI 5.3 wird mit nur wenig Training unterschiedlichste Stimmen erkennen können“* wurde somit im Wesentlichen bestätigt. Erstaunlicherweise waren Fehler beim Diktat deutlich seltener als erwartet. Im Gegensatz dazu, war die Erkennungsgenauigkeit bei den Nipper-Befehlen deutlich niedriger als angenommen. Dies führte dazu, dass es Befehle gab welche bei einzelnen Versuchspersonen nicht verwen-

det werden konnten. Der Befehl „Farbe“ bereitete bei drei Versuchspersonen Probleme, bei einer Person wurde dieser Befehl überhaupt nicht erkannt. Auch der Befehl „Kreis“ wurde bei einer anderen Versuchsperson nicht erkannt. Bei zwei Benutzern kam es sehr häufig vor, dass diese die Befehle wiederholen mussten. Abbildung 22 zeigt eine Versuchsperson bei der Eingabe von „Farbe“. Da sie das „r“ nicht ausspricht, interpretiert das System das gesagte nicht als Befehl, sondern als diktiertes Wort und fügt den erkannten Begriff „Fabel“ in das zu bearbeitende Textfeld ein.

Abbildung 22: Versuchsperson mit einer fehlerhaften Erkennung von "Farbe"

Im Allgemeinen schnitt die Spracherkennung bei Nipper deutlich schlechter ab als erwartet. Konnte während der Entwicklung die Erkennungsgenauigkeit beim Nipper-Server auf über 0.5 eingestellt werden, musste diese während der Evaluation auf Werte unter 0.2 eingestellt werden. Um Frustrationen zu vermeiden, wurden bei einer Versuchsperson sogar Werte unter 0.1 eingestellt. Je komplexer ein Befehl war, desto höher war die Erkennungsgenauigkeit. Einzelne Befehle wie eben „Farbe“ wurden deutlich häufiger Falsch erkannt, als komplexere Befehle wie „Linienbreite 10 Pixel“. Es sei hier jedoch anzumerken, dass trotz dieser geringen Werte nie eine falsche Aktion auf einen Sprachbefehl folgte, die Befehle wurden lediglich als „nicht-erkannt“ gewertet. Wenn eine Aktion vom System ausgeführt wurde, dann entsprach diese in allen Fällen der vom Teilnehmer erwarteten Aktion. Da es bei allen Teilnehmern vor kam, dass sie laut geäusserte Gedanken und Sprachbefehle zu kurz hintereinander ausgesprochen haben, war es in gewissen Fällen nicht möglich zu entscheiden, was der wirkliche Grund er Ungenauigkeit war.

5.6.2 NipMap Exploration

Bei der Exploration gingen die Teilnehmer mit unterschiedlichen Vorkenntnissen an die Aufgaben. Bei den ersten zwei Teilnehmern folgten häufig Sprachbefehle auf andere Äusserungen. Die darauf folgenden Benutzer wurden dann bei der Einführung jeweils darauf hingewie-

sen, zwischen normaler Konversation oder Selbstgespräch und Sprachbefehlen eine kurze Pause einzulegen. Dennoch kamen solche zusammenhängende Äusserungen bei allen Benutzern vor. Die Übungsaufgaben konnten in den meisten Fällen gut gelöst werden, da diese Schritt für Schritt auszuführen waren. Folgende Beobachtungen konnten gemacht werden:

- Keiner der Benutzer hat von sich aus die Befehle für Objektattribute gefunden. Dies lag daran, dass die Befehle, die im Menü aufgeführt sind, nicht in der Hilfe angezeigt werden und der Befehl um das Menü zu öffnen, wegen schlechter Bezeichnung, nicht gefunden worden ist.
- Bei fünf Benutzern ist aufgefallen, dass diese das Menü in kleinen Einzelschritten bedient haben. So wurden über Befehle wie „Linienbreite“ das Menü mit dem Slider für die Linienbreite geöffnet. Dann wurde nach einer kurzen Pause die Pixelzahl genannt, die auf dem Aufgabenblatt stand. Dieser Zwischenschritt wird vom System jedoch nicht unterstützt und führte daher zu keinem Ergebnis.

5.6.3 NipMap Aufgabensets

Nach den Übungsaufgaben und einigen Tipps zur Benutzung des Programms konnten die Benutzer die Aufgabensets erfolgreich bearbeiten. Der Zeitaufwand pro Aufgabenset variierte zwischen 15 und 25 Minuten. Auffallend war, dass jeder Teilnehmer mindestens bei einer Aufgabe, die Spracheingabe auch für die nicht genau spezifizierten Aufgaben verwendeten. So wurde für eine „helle, dünne Linie“ die Linienbreite häufig über einen Sprachbefehl auf 1-2 Pixel eingestellt und die helle Farbe auf weiss oder gelb eingestellt. Bei konkreten Aufgaben verwendeten vier Teilnehmer ausschliesslich die Spracheingabe. Die Hypothese „*Die Benutzer bevorzugen unimodale Interaktion zur Parametereingabe“* kann damit ebenfalls bestätigt werden. Diese Aussagen wurden auch im Interview von 4 Teilnehmern bestätigt. Wohingegen lediglich ein Teilnehmer versucht hat Transformationen unimodal, mit der Spracherkennung auszuführen.

Das Input-Everywhere Konzept, welches das genaue Zielen auf das Interaktionselement erübrigt, geriet häufig in Vergessenheit. Erst nach 2-3 erfolgslosen Zielversuchen erinnerten sich die Teilnehmer daran, dass sie die Eingabe überall auf dem Display tätigen können. Bei der Hälfte der Teilnehmer trat dieses Problem beim zweiten Aufgabenset nicht mehr auf. Die Hypothese „*Das Input-Everywhere erleichtert dem Benutzer die Bedienung mit ungenaueren Eingabegeräten, da genaues Zielen für die Interaktion nicht erforderlich ist“* kann somit bestätigt werden. Diese Bestätigung fand sich in den Beobachtungen wie auch in dem Interview

nach dem Test, in dem alle Teilnehmer[23] eine Erleichterung bei der Eingabe bestätigten. Bei den Beobachtungen wurde dies vor allem bei den Aufgaben deutlich, bei denen die Transparenz der Farbe verändert werden musste. Um die Transparenz der Farben einzustellen, musste ebenfalls ein Slider innerhalb des Farbwähler-Dialogs bedient werden. Da der Farbwähler eine Standardkomponente von Microsoft ist, konnte das Input-Everywhere Konzept auf diesen Slider nicht anwenden. Auffallend war, dass alle Teilnehmer Probleme hatten, diesen Slider zu bedienen und sich nach wenigen Versuchen an das Input-Everywhere-Konzept erinnerten und versuchten dieses anzuwenden.

Um die Hypothese „*Das WYSIWYS-Konzept erleichtert dem Benutzer die Bedienung von NipMap und liefert diesem die dafür nötigen Informationen und Befehle"* zu bestätigen reichte die Teilnehmerzahl nicht aus. Da bei jedem Teilnehmer unterschiedliche Reaktionen festgestellt wurden, wären weitere Tests notwendig um festzustellen ob noch weitere Reaktionen festzustellen wären. Die untenstehenden Teilnehmer zeigten deutlich unterschiedliche Verhaltensmuster.

- Ein Teilnehmer hat fast jeder Aufgabe beim Hilfemenü begonnen, dann das Menü eingeblendet und dann die Menüpunkte zur Erinnerung an die Sprachbefehle benutzt. Das Menü zur Interaktion mit dem Zeigegerät hat dieser Benutzer kaum verwendet.
- Ein Teilnehmer, ohne jegliche Vorkenntnisse des Systems, hat nach wenigen Aufgaben das Menü kaum mehr gebraucht und Intuitiv die richtigen Befehle verwendet, obwohl die teilweise bei vorherigen Aufgaben noch gar nie benutzt wurden.
- Ein Teilnehmer hat Aufgaben mit konkreten Pixelzahlen mit der Spracherkennung gelöst. Auffallend war, dass bei Aufgaben mit vagen Beschreibungen, feine Ausprägungen mit Sprachbefehlen und grobe Ausprägungen mit den Slidern gelöst wurden. Eine „dünne Linie" wurde beispielsweise mit „Linienbreite 1 Pixel" eingestellt und eine „dicke Linie" wurde mit dem Slider eingestellt.

Im Allgemeinen konnte dennoch festgestellt werden, dass alle Benutzer, wenn auch auf unterschiedliche Art und Weise, mit dem Menü zurechtgekommen sind.

Im Folgenden werden einige Funktionen von NipMap separat betrachtet.

Feedback: Das Feedback (Vgl. Kapitel 4.1.2) wurde im Interview von allen Teilnehmern als gut bezeichnet. Bei vier Benutzern konnte jedoch beobachtet werden, dass diese nicht erkannt

[23] Ein Teilnehmer, der die Slider beim ersten Aufgabenset nie benutzt hat, wurde nachträglich gebeten das Input-Everywhere-Konzept zu verwenden, um zu sehen wie dieser damit klar kam.

haben, dass das Werkzeug gewechselt wurde und daher den Befehl unnötigerweise wiederholten. Das kann daran liegen, dass beim Wechseln des Werkzeuges lediglich der Cursor ändert und dies auf dem grossen Display womöglich nicht ausreichend visualisiert wird. Auch minimalen Veränderungen am Objekt, beispielsweise die Änderung der Linienbreite von 3 auf 4 Pixel, wurden von den Teilnehmern teilweise nicht wahrgenommen.

Parametereingabe: Bei der Eingabe der Parameter bietet NipMap zwei voneinander getrennte Möglichkeiten an. Zum einen die direkte Eingabe via Sprache und zum andern die Eingabe via Sprache und Menü. Wie bereits in Kapitel 5.6.2 beschrieben haben fünf Teilnehmer intuitiv versucht, trotz bereits geöffnetem Menü, die Parameter über die Sprache einzugeben. Das System ist für diesen Zwischenschritt nicht ausgelegt. Da dies jedoch der häufigste Fehler war, sollte diese Möglichkeit der Eingabe auf jeden Fall mit eingeplant werden.

Bei der Aufgabe zur Erstellung eines Schlagschattens kamen die Teilnehmer mit den angezeigten Optionen nicht zurecht. Es schien keinem Teilnehmer bekannt zu sein, dass ein Schlagschatten nur dann sichtbar ist, wenn eine Tiefe angegeben wird.

Rotation: Bei der Rotation von Objekte ist die lineare Einstellung des Winkels bei den wenigsten Benutzern intuitiv. Bei der Rotation wird der Winkel durch die relative Positionsänderung des Zeigegerätes auf der Horizontalen bestimmt. Alle haben bei der Rotation erwartungsgemäss versucht das Objekt mit einer Rotationsbewegung zu drehen. Da der Winkel lediglich durch die horizontale Komponente bestimmt wird, entsprach das Ergebnis nicht den Erwartungen der Benutzer.

5.7 Empfehlungen zur Verbesserung von Nipper

Die oben genannten Ergebnisse zeigten einige positive Aspekte von Nipper auf. Drei Hypothesen konnten dabei bestätig werden. Für die erkannten Probleme werden im Folgenden Lösungsansätze vorgeschlagen.

5.7.1 Spracherkennung

Für Benutzer ohne Kenntnisse eines sprachgesteuerten Computers, ist dessen Verwendung zwar nicht unangenehm, dennoch müssen diese zwei wichtige Punkte beachten.

- Die Spracherkennung benötigt kurze Pausen zwischen den Audiosignalen, um die Sprachbefehle von normaler Konversation unterscheiden zu können.

- Emotionale Betonungen der Sprachbefehle kann dazu führen, dass der Sprachbefehl nicht erkennt wird.

Die stetig sinkende Erkennungsgenauigkeit bei Nipper und die teilweise sehr guten Ergebnisse beim ersten Diktat könnten ein Hinweis darauf sein, dass die Sprachprofile unterschiedlicher Benutzerkonten nicht vollständig unabhängig sind. Es wäre möglich, dass bereits in anderen Benutzerkonten definierte Sprachprofile entweder miteinbezogen oder sogar durch die neuen Trainingsdaten ergänzt werden. Dies würde dazu führen, dass zwar neue Benutzer schneller erkannt werden, genaue Erkennungen jedoch deutlich mehr Training benötigt. Eine andere Erklärung für die grossen Genauigkeitsunterschiede zwischen dem Entwickler und den Testpersonen könnte die Adaption des Entwicklers an das System sein. Um diesbezüglich genauerer Aussagen treffen zu können, müsste entweder bekannt sein, wie die Sprachprofile von Windows Vista intern abgelegt und verwaltet werden oder es müssten zusätzliche Tests gemacht werden, welche sich auf diesen Aspekt spezialisieren. Interessante Fragen dabei wären:

- Wie häufig und wie lang muss eine SAPI, die bereits in anderen Benutzerkonten trainiert wurde, von einem neuen Benutzer trainiert werden, damit die Erkennungsgenauigkeit gut ist?
- Können mehrere Benutzer auf dem gleichen Benutzerkonto die SAPI so trainieren, dass jeder Benutzer gut Erkannt wird?

Sollten diese Tests ergeben, dass die Spracherkennung nur dann genau ist, wenn genau ein Benutzer den Computer mit der Spracherkennung verwendet, könnten Konzepte zur effizienten Fehlerkorrektur weiterhelfen.

Das von Tan et al. (2003a) vorgeschlagene Konzept wurde für die sprachliche Texteingabe in Formularen entwickelt. Tan et al. kombinieren Spracherkennung mit einem Eye-Tracker. Bei zweimaliger, fehlerhafter Erkennung wird eine virtuelle Tastatur eingeblendet. Diese virtuelle Tastatur hat, ähnlich wie die Tastatur eines Telefons, jeweils drei Buchstaben in einem Feld (Vgl. Abb. 23). Kombiniert mit Sprache und Blicksteuerung wird ein Buchstabe ausgewählt.

Die Eingabe des Buchstabierdialogfeldes wird mit einer Wortdatenbank verglichen. Werden weniger als neun Treffer gefunden, die zur Eingabe passen, so werden diese in einem weiteren Dialogfeld angezeigt. Sollte dieses Konzept auch zur Eingabe von Sprachbefehlen verwendet werden, dann könnten zusätzlich zur Datenbank auch die Grammatiken der Anwen-

dung durchsucht werden. Zur Selektion des Wortes respektive des Sprachbefehls spricht der Benutzer den Sprachbefehl erneut und schaut dabei Feld in dem das Wort angezeigt wird.

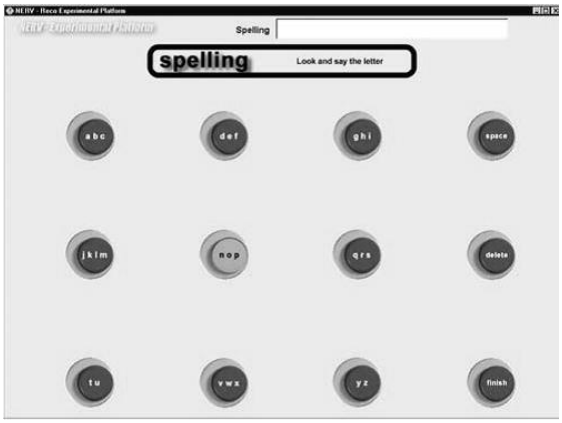

Abbildung 23: Virtuelle Tastatur (Tan et al. 2003a)

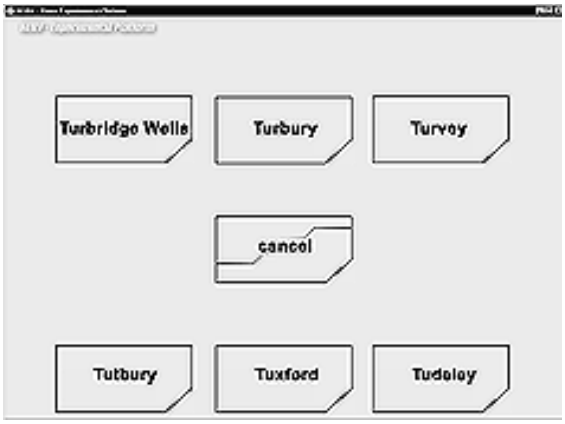

Abbildung 24: Wortliste mit ähnlichen Wörtern und einem "Cancel" Button (Tan et al. 2003a)

In einem Experiment vergleichen Tan et al. (2003b) ihr System mit einem System welches ein fehlerhaftes Wort durch erneutes Sprechen des Wortes korrigiert (RESER). Alle Testpersonen bezeichneten das System von Tan et al. als exakt, schnell und natürlich. Dieses System könnte bei der aktuellen Erkennungsgenauigkeit von Nipper die Effizienz enorm steigern. Wie oben erwähnt waren es vor allem kurze Befehle die nicht erkannt werden konnten. Das Problem der Teilnehmerin in Abbildung 22 mit dem Wort „Farbe“ könnte so in wenigen Augenblicken gelöst werden. Das Dialogfeld wird nach Tan et al. automatisch geöffnet, daraufhin würde bei der aktuellen Grammatik bereits der Buchstaben „F“ ausreichen, damit der Befehl „Farbe“ in der Auswahl angezeigt würde.

Eine andere Möglichkeit zur effizienten Fehlerkorrektur verwendet das „Speech-Rejected“-Ereignis der SAPI. Wie in Kapitel 4.1.1 erwähnt wird dieses Ereignis ausgegeben, wenn ein Befehl erkannt wird, aber nicht die gewünschte Erkennungsgenauigkeit erreicht. In Nipper ist eine Funktion integriert, welche dem Benutzer diesen verworfenen Befehl anzeigen und fragen kann, ob dies der gewünschte Befehl sei. Der Benutzer braucht dann lediglich mit „Ja“ oder „Nein“ zu Antworten. Bei Nipper wurde diese Funktion jedoch deaktiviert, da das „Speech-Rejected“-Ereignis keine semantischen Schlüssel mitliefert und somit von NipMap nicht verarbeitet werden kann. Ein Algorithmus der in der Grammatik nach dem Befehl sucht und die semantischen Schlüssel generiert, könnte dieses Problem lösen. Etwas komplizierter umzusetzen, jedoch zuverlässiger als das „Speech-Rejected“-Ereignis, wäre die Verwendung

der Sprachhypothese. Für die Spracherkennung und die Sprachhypothese können unterschiedliche Genauigkeiten eingestellt werden. So könnte mit Hilfe der Sprachhypothese, die ebenfalls keine semantischen Schlüssel ausgibt, und einem ähnlichen Algorithmus, diejenigen Befehle in der Grammatik gesucht werden, die der Sprachhypothese am ähnlichsten sind. Ähnlich wie bei Tan et al. (2003a) könnte dem Benutzer eine Trefferliste zur Auswahl angezeigt werden.

Die Spracherkennung zeigte sich bei Nipper als sehr zuverlässig solange das System ausreichend trainiert ist. Die fünf Minuten Training der Testpersonen haben jedoch deutlich schlechtere Ergebnisse gebracht als erwartet. Mit den vorgeschlagenen Verbesserungen dürfte die Spracherkennung in Zukunft auch im Mehrbenutzer-Betrieb zuverlässige Ergebnisse liefern.

5.7.2 NipMap

Im Folgenden werden, aufgrund der oben genannten Ergebnissen, Lösungsvorschläge für die einzelnen Funktionen von NipMap gemacht.

Feedback: Das Feedback sollte so gestaltet werden, dass jede Interaktion eindeutig angezeigt wird, auch bei geringfügigen Veränderungen am Objekt. Solcher Veränderungen, beispielsweise die Veränderung der Linienbreite von 2 auf 3 Pixel, konnte von den Teilnehmern teilweise nicht wahrgenommen werden. Dieser Aspekt gilt vor allem bei unerfahrenen Benutzern. Dem erfahrenen Benutzer fallen auch kleinste Veränderungen auf, da er weiss, worauf er achten muss. Folgende zwei Möglichkeiten könnten das Feedback verbessern:

- Das Feedback kann vom Nipper-Server gesteuert werden, was bedeutet, dass dieses sich darauf beschränken würde, ob der Befehl erkannt wurde oder nicht. Ähnlich wie bei den Icons in Tabelle 5, könnte ein zusätzliches Icon eingefügt werden, welches immer dann angezeigt wird, wenn ein Sprachbefehl als richtig erkannt wurde. Da bei den Tests keine falschen Erkennungen aufgetreten sind, wäre diese Variante denkbar. Sollten aber dennoch Befehle falsch erkannt werden, dann wäre das Feedback nicht erwartungsgemäss.
- Die zweite Möglichkeit der Visualisierung wäre direkt in NipMap integriert. Dieses Feedback wäre dann nicht für alle Befehle gleich, sondern abhängig von der ausgeführten Aktion. Es könnte entweder ein Icon, das für den Befehl repräsentativ, ist kurz eingeblendet werden oder ein optischer Effekt an der betreffenden Stelle könnte ange-

zeigt werden. Wird die Linienbreite verändert, so könnte dies mit einer kurz eingeblendeten Umrandung des Objekts dargestellt werden.

Parametereingabe: Die Parametereingabe muss zu jedem Zeitpunkt der Interaktion konsistent bleiben. Wurde das Menü zur Parametereingabe via Spracheingabe geöffnet, soll es dem Benutzer auch möglich sein mit der Sprache eine Eingabe zu tätigen. Dabei sollen auch einzelne Zahlen oder Farben als Eingabe akzeptiert werden. Da das Menü bereits aktiviert wurde, können diese einzelnen Zahlen oder Farben eindeutig zugeordnet werden und benötigen kein erneutes Benennen des Attributes mehr. Die Grammatik ist daher so zu verändern, dass Zahlen zu jedem Zeitpunkt als einzelne Eingabe möglich sind.

Rotation: Das Input-Everywhere-Konzept muss so erweitert werden, dass bei der Rotation eine Art relative Rotation gemessen werden kann. Es soll weiterhin möglich sein die Rotation im Stil des Input-Everywhere-Konzepts aufzuführen, aber die Interaktion soll nicht nur von der horizontalen Komponente abhängen.

Transformationen: Die aktuelle Version von NipMap verwendet bei Freihand-Gesten lediglich das Zeigen und Klicken. Für Transformationen könnten zusätzliche Gesten verwendet werden. Angenommen es würden beide Hände von den Kameras erfasst, dann könnten ähnliche Gesten zum vergrössern oder drehen von Objekten verwendet werden, wie beispielsweise bei Han‘s Perceptive Pixel (Han 2006). Das Drehen von Objekten könnte so mit dem Drehen der Hand verknüpft werden und die Objekte liessen sich mit beiden Händen auf- oder zusammenziehen. Für solche Funktionen können Gesten durchaus intuitiver sein als Sprachbefehle. Eine Geste, um die Farbe oder Linienbreite einzustellen, ist hingegen eher weniger intuitiv. Die Erweiterung des Systems um weitere Eingabe- und Kontrollmöglichkeiten beruht auf der Idee, dass jede Modalität nur diejenigen Interaktionen unterstützen soll, für die es am besten geeignet ist, respektive der Benutzer die Modalität verwenden kann, die für ihn am intuitivsten ist.

Selektion: Die Selektion von Objekten war bei allen Benutzern unproblematisch. Wenn jedoch gerade ein Slider aktiv ist, können keine andern Objekte ausgewählt werden, da die gesamte Fläche für die Interaktion mit dem Slider verwendet wird. Bei handelsüblichen Zeichenprogrammen können zwar auch keine Objekte selektiert werden, wenn nicht das entsprechende Werkzeug verwendet wird. Nichts desto trotz sollte die Selektion mit dem Eye-Tracker, wie in Kapitel 4.2.3 beschrieben wird, ergänzt werden, um die Selektion intuitiver und effizienter zu gestalten.

Hilfe: Das Hilfemenü bei NipMap beinhaltet zwar die meisten Befehle. Dieses ist jedoch statisch aufgebaut und die Befehle für Parametereingabe werden lediglich im Menü angezeigt, nicht aber in der Hilfe. Wie bei den Ergebnissen schon erwähnt, haben Benutzer auch die Sprachbefehle zur Parametereingabe in der Hilfe gesucht. Die Hilfe sollte daher ähnliche Adaptiv aufgebaut sein, wie das Menü. Es sollen alle Befehle angezeigt werden, die in der aktuellen Situation verwendet werden können. Bei Befehlen die nicht verwendet werden können, sollte Begründet werden warum diese deaktiviert sind und welche Schritte notwendig sind um diese zu aktivieren. Wird gerade ein Rechteck bearbeitet, dann sollte angezeigt werden, weshalb Schriftfunktionen deaktiviert sind und das diese nur bei Textfeldern aktiv sind.

Alternative Sprachbefehle: Wie in der aktuellen Hilfe von NipMap schon angedeutet und auch in der Grammatik teilweise schon integriert, sollten alternative Sprachbefehle verfügbar sein. Zudem sollte es Möglich sein, intuitive Befehle während der Laufzeit zu definieren. Die SAPI 5.3 unterstützt die Veränderung der Grammatik zur Laufzeit. So ist es möglich, laufend neue Sprachbefehle hinzuzufügen.

Multimodalität erweitern: Abgesehen von der Texteingabe, wäre es möglich alle Funktionen nur über das Zeigegerät zu steuern. Das WYSIWYS-Menü soll so erweitert werden, dass Klicks möglich sind. So könnte der Benutzer das Menü auch ohne Spracheingabe unimodal mit dem Zeigegerät bedienen. Diese Option überlässt es jeder Zeit dem Benutzer ob er via Sprache, Zeigegerät oder multimodal interagieren möchte.

Texteingabe: Ein Textfeld ist nach dem es gezeichnet wurde hellhörig auf jegliche diktierte Spracheingabe. Im Normalfall ist dies auch unproblematisch. Im kollaborativen Umfeld könnte es vorkommen, dass der Benutzer, nachdem er das Textfeld erstellt hat, sich zuerst mit einer anderen Person unterhält und erst danach die Texteingabe macht. Dies hätte zur Folge, dass die Spracherkennung versucht die Konversation als Diktat zu interpretieren. Ähnlich wie bei Tan et al. (2003b)(Tan et al. 2003b)(Tan et al. 2003b) könnte der Eye-Tracker verwendet werden, um das Textfeld zu selektieren. So wäre das Textfeld nur dann aktiv, wenn der Benutzer seinen Fokus darauf richtet. Auf diese Wiese wäre es auch einfacher möglich, schon bearbeitete Textfelder mit Text zu erweitern.

Bei der gesamten Evaluation wurden einige, eher kleinere, Probleme erkannt. Für die meisten wurden schon Lösungsvorschläge gemacht, die diese Probleme mit grosser Wahrscheinlichkeit beheben können.

6 Fazit und Ausblick

Im Verlaufe dieser Arbeit wurde Nipper, das multimodale System für die Powerwall der Universität Konstanz vorgestellt. Dieses System ermöglicht die Verwendung der Spracherkennung auf der Powerwall mit einer Client-Server-Lösung. Die dritte Komponente des Nipper-Systems, genannt NipMap, verwendet die Spracherkennung zusammen mit dem Laserpointer oder der Handgestenerkennung. Bei der Verwendung des Laserpointers und der Handgestenerkennung wird auf zwei Systeme zurückgegriffen, die ebenfalls für die Powerwall entwickelt wurden. Mit dieser Kombination ermöglicht NipMap multimodale Interaktion mit unterschiedlichen Zeigegeräten. NipMap unterstützt damit nicht nur die in Kapitel 2.3 motivierte multimodale Interaktion sondern lässt zusätzlich die Möglichkeit offen, das bevorzugte Eingabegerät zu wählen. Durch die zwei in NipMap integrierten Konzepte ergeben sich keine wesentlichen Unterschiede zwischen den beiden Eingabegeräten. Diese Konzepte, das WYSIWYS-Konzept und das Input-Everywhere-Konzept, vereinfachen die Interaktion mit dem System und bieten die nötigen Informationen für die Spracherkennung. Das WYSIWYS-Konzept vereinfacht die Verwendung der Spracherkennung und ermöglicht zugleich die multimodale Interaktion. Das Konzept beinhaltet ein adaptives Menü, welches die Eingabe von Sprachbefehlen begünstigt und zugleich auch Eingaben mit einem Zeigegerät emöglicht. Die Menüoptionen passen sich dem aktuellen Arbeitsvorgang an und versuchen dem Benutzer zu jeder Zeit die Informationen anzuzeigen, die dieser für die nächsten Schritte benötigt. Das zweite Konzept, das Input-Everywhere-Konzept vereinfacht die Interaktion mit den Zeigegeräten. Interaktionselemente müssen dabei nicht mehr exakt mit dem Mauszeiger getroffen werden, es kann die gesamte Fläche des Displays kann zur Eingabe verwendet werden. Weitere Funktionen, wie die automatische Deaktivierung der Spracherkennung und das visuelle Feedback für die sonst „unsichtbare“ Sprache, sorgen für eine möglichst fehlerfreie Interaktion mit der Spracherkennung.

In einer Abschliessenden Evaluation wurde gezeigt, dass NipMap und die integrierten Konzepte von den Versuchspersonen durchwegs Positiv beurteilt wurden. Zusammen mit den durch die Evaluation motivierten Lösungsvorschlägen, kann das System als robust und zuverlässig angesehen werden.

Das Nipper-System kommt mit der Handgestenerkennung und dem Funkmikrofon der Vision der völlig freien und natürlichen Interaktion schon sehr nahe. Die Handschuhe zur Gestenerkennung sollten schon bald verschwinden oder sind in anderen Systemen teilweise bereits

verschwunden. Festinstallierte Mikrofone und Kameras machen, mit Hilfe der audio-visuellen Spracherkennung, das Funkmikrofon schon bald überflüssig. Personenerkennung via Stimme oder Kamera sorgt dafür, dass stets die richtigen Trainingsdaten und Profile aktiv sind, auch wenn mehrere Benutzer mit dem System interagieren. Weitere sinnvolle Kombinationen mit anderen Eingabekonzepten können das System so erweitern, dass in jeder Situation und für jede Aufgabe die dafür am besten geeigneten Konzepte verwendet werden können. Die Erweiterung der Gestenerkennung von einfachen Klicks zur Interaktion mit beiden Händen wurde bereits angesprochen. Mit einem berührungsempfindlichen Display könnten zusätzliche Interaktionsschemata aus dem Bereich Multi-Touch-Technologie[24] integriert werden. Es gibt unzählige Konzepte die in so ein System eingebracht werden könnten. Denn auch wenn das Nipper-System 28 Jahre jünger ist, als das erste vergleichbare System von Bolt, steckt dieses System noch in den Kinderschuhen und lässt viele kreative Ideen für die Zukunft zu.

[24] Multi-Touch: Berührungsempfindliche Displays, die mehrere Finger unabhängig voneinander erfassen können(Tse et al. 2006).

Anhang A: Pre-Test Fragebogen

Zur Person

Alter: _____

Geschlecht:

☐ männlich

☐ weiblich

Momentante Tätigkeit: _____
(bei Studium auch den Studiengang bitte nennen)

Wie viele Stunden verbringen Sie pro Tag an einem Computer?

0 – 1 Stunde ☐

1 – 2 Stunden ☐

2 – 3 Stunden ☐

Mehr als 3 Stunden ☐

Haben Sie schon einmal einen Handschuh für Virtual-Reality Anwendungen ausprobiert?

Ja ☐ Nein ☐

Haben Sie schon einmal einen Laserpointer ausprobiert?

Ja ☐ Nein ☐

Haben Sie schon einmal eine Anwendung mit Spracherkennung bedient, oder mit Spracherkennung gearbeitet?

Ja ☐ Nein ☐

Haben Sie schon einmal mit einem Zeichenprogramm gearbeitet?

Ja ☐ Nein ☐

Haben Sie bereits einmal mit einem Beamer (Projektor) oder der Powerwall gearbeitet?

Ja ☐ Nein ☐

Anhang B: Übungsaufgaben

Bedienung des Programms Nipper

Das Programm „Nipper“ gibt Ihnen die Möglichkeit, Objekte zu zeichnen und zu verändern. Dies können Sie über die Spracherkennung in Kombination mit einem Laserpointer oder auch Handgesten.
Die Spracherkennung wird mit dem Befehl „**Computer**“ aktiviert. Ist die Spracherkennung aktiv (d.h. kein rotes Icon angezeigt) können die Befehle direkt ausgesprochen werden. Es muss nicht mit „Computer“ begonnen werden.
Ist das Gesprochene nicht an den Computer gerichtet, schaltet sich die Spracherkennung nach 10 Sekunden automatisch aus und es wird das rote Icon angezeigt. Der Befehl „Computer“ aktiviert die Spracherkennung wieder.
Das Pausensymbol zeigt an, dass die Spracherkennung die Eingabe verarbeitet.

Icon	Funktion
	Spracherkennung deaktiviert (permanent angezeigt)
	Spracherkennung aktiv (blinkt auf und wird ausgeblendet)
	Spracherkennung beschäftigt
	Keine Verbindung zum Client
	Erkennung nicht Ausreichend (blinkt auf und wird ausgeblendet)

Mit dem Befehl „**Hilfe**“ wird das Hilfemenü aufgerufen. Darin befinden sich die verfügbaren Werkzeuge zur Ansicht, diese können durch Aussprechen ihrer Bezeichnung – z.B. „Rechteck“ ausgewählt werden. Das derzeit aktive Werkzeug ist mit dem Icon ersichtlich z.B. ein T für das Textwerkzeug. Werkzeuge können auch ohne das Hilfemenü direkt über die Sprache ausgewählt werden.
Eigenschaften von Objekten können entweder ausschließlich über die Sprache angegeben werden, z.B. „Hintergrundfarbe rot“ oder aus Menüs ausgewählt werden. So öffnet sich zum Beispiel bei dem Befehl „Hintergrundfarbe“ die Farbauswahl.
Soll etwas aus einem Menü ausgewählt werden ist das ausschließlich mit Sprachbefehlen möglich, Klicks können nicht zur Auswahl verwendet werden.

Übungsaufgaben

Bitte lesen Sie jede Aufgabe laut vor, bevor Sie mit Ihr beginnen.
Aktivieren Sie anschließend die Sprachsteuerung mit dem Befehl „**Computer**“.
Nachdem Sie eine Aufgabe bearbeitet haben, deaktivieren Sie bitte die Sprachsteuerung mit dem Befehl „**nicht mehr zuhören**“

1. Rufen Sie das Hilfemenü auf und suchen Sie den Befehl für „Kreis“. Zeigen Sie auf das entsprechende Icon.

2. Zeichnen Sie einen Kreis

3. Ändern Sie die Hintergrundfarbe des Kreises

4. Ändern Sie die Linie in eine dicke blaue Linie

5. Zeichnen Sie ein rotes Rechteck mit einer grünen 5 Pixel Linie und einem leichten Schatten

Anhang C: Aufgabenset A

Bitte lesen Sie jede Aufgabe laut vor, bevor Sie mit Ihr beginnen.
Aktivieren Sie anschließend die Sprachsteuerung mit dem Befehl „**Computer**“.
Nachdem Sie eine Aufgabe bearbeitet haben, deaktivieren Sie bitte die Sprachsteuerung mit dem Befehl „**nicht mehr zuhören**“

1. Zeichnen Sie einen **kleinen, gelben Kreis,** mit einer **blauen 2 Pixel Linie** auf der **linken Hälfte** der Powerwall.

2. Zeichnen Sie ein **kleines, weisses Rechteck** mit **schwarzer 18 Pixel Linie** und **gerundeten Ecken** mit einem **Radius von 14 Pixel** in der **Mitte** der Powerwall.

3. Zeichnen Sie ein **dunkles Rechteck** mit einer **hellen, dünnen Linie** und einem **leichten Schatten** auf der **linken Hälfte** der Powerwall.

4. Erstellen Sie ein **Textfeld** mit **hellem, halb transparentem Hintergrund** und **dunkler Schrift** mit dem Text **„Hafenbecken“**.

5. **Verschieben** Sie das **dunkle Rechteck** auf die **rechte Seite** der Powerwall und **runden Sie die Ecken stark ab**.

6. **Vergrössern** Sie das **weisse Rechteck** an der **oberen rechten Ecke** und ändern Sie die **Linienbreite auf 15 Pixel**.

7. **Drehen** Sie das **Textfeld nach rechts** und **verändern Sie die Schriftfarbe**.

Anhang D: Aufgabenset B

Bitte lesen Sie jede Aufgabe laut vor, bevor Sie mit Ihr beginnen.
Aktivieren Sie anschließend die Sprachsteuerung mit dem Befehl „**Computer**“.
Nachdem Sie eine Aufgabe bearbeitet haben, deaktivieren Sie bitte die Sprachsteuerung mit dem Befehl „**nicht mehr zuhören**“

1. Zeichnen Sie einen **kleinen, roten Kreis**, mit einer **grünen 4 Pixel Linie** auf der **rechten Hälfte** der Powerwall.

2. Zeichnen Sie ein **kleines, blaues Rechteck** mit **weisser 22 Pixel Linie** und **gerundeten Ecken** mit einem **Radius von 10 Pixel** in der **Mitte** der Powerwall.

3. Zeichnen Sie ein **helles Rechteck** mit einer **dunklen, dünnen Linie** und einem **leichten Schatten** auf der **rechten Hälfte** der Powerwall.

4. Erstellen Sie ein **Textfeld** mit **dunklem, halb transparentem Hintergrund** und **heller Schrift** mit dem Text **„Tankschiff“**.

5. **Verschieben** Sie das **helle Rechteck** auf die **linke Seite** der Powerwall und **runden Sie die Ecken stark ab**.

6. **Vergrössern** Sie das **blaue Rechteck** an der **oberen rechten Ecke** und ändern Sie die **Linienbreite auf 15 Pixel**.

7. **Drehen** Sie das **Textfeld nach links** und **verändern Sie die Schriftfarbe**.

Anhang E: Interview

Spracherkennung

Ist ihnen die Bedienung leicht gefallen?

ja □ nein □

Glauben sie, dass die meisten Personen die Bedienung schnell erlernen könnten?

ja □ nein □

Würden sie lieber nur über die Sprachsteuerung gehen, oder lieber nur über das Menü?

Sprachsteuerung □ Menü □ Hängt von der Aufgabe ab □

Wie beurteilen sie das visuelle Feedback

gut □ ausreichend □ zu wenig □ unverständlich □

Wie sehr beeinflusste sie die Verzögerung bei der Spracherkennung?

stark □ kaum □ gar nicht □

Würden sie die Kombination von Sprache und Gesten als intuitiv beurteilen?

stark □ kaum □ gar nicht □

Wie sehr erleichtert das Input-Everywhere Konzept die Aufgabe?

stark □ kaum □ gar nicht □ ist mir nicht aufgefallen □

Präferenz Laserpointer ↔ Handgesten

Welches Gerät …

… war genauer

Laserpointer □ Handgesten □ Kein Unterschied □

… war schneller

Laserpointer □ Handgesten □ Kein Unterschied □

… war anstrengender zu benutzen (Ermüdung der Finger, Handgelenk, Arm, Schulter)

Laserpointer □ Handgesten □ Kein Unterschied □

… hat ihnen mehr Spaß gemacht

Laserpointer □ Handgesten □ Weiß nicht □

… würden sie lieber für eine solche Anwendung verwenden

Laserpointer □ Handgesten □ Beide □ Keines □

Anhang F: Demo-Video

Die beigelegte CD enthält ein ca. 8-minütiges Demo-Video, in dem die Funktionen von Nip-Map demonstriert werden.

Das Video enthält folgende Punkte:

- Feedback & Kontextmenü
- Hilfemenü
- WYSIWYS-Menü
- Input-Everywhere
- Texteingabe
- Bewegungsfreiheit
- Multimodale Eingabe

Bei dem Video kommen zwei unterschiedliche Aufnahmemethoden zum Einsatz. Ein grosser Teil der Aufnahmen wurde mit einer Videokamera vor der Powerwall aufgenommen. Um Details bei der Interaktion mit den Menüs besser zu zeigen, wurden einige Ausschnitte zusätzlich mit einem Screen-Recorder aufgezeichnet. Diese Ausschnitte sind etwas neuer und können daher kleine Veränderungen des Menüs aufweisen. Die Qualität der Audioaufzeichnungen ist sehr unterschiedlich, aber durchaus interessant. Bei den Aufnahmen mit der Videokamera wird deutlich, wie schlecht die Akustik in dem Powerwall-Raum ist. Bei den Aufnahmen mit dem Screen-Recorder, wurde dasselbe Audiosignal wie für die Spracherkennung verwendet. Dieses Signal ist sehr stumpf und wurde um 400% verstärkt, damit es für das Video überhaupt verwendet werden kann.

Literaturverzeichnis

BIEG H-J (2008) Laserpointer and Eye Gaze Interaction Design and Evaluation. In: Information Engineering) University of Konstanz, Konstanz.

BOLT RA (1980) Put-That-There: Voice and Gesture at the Graphics Interface. In: Proceedings of the 7th annual conference on Computer graphics and interactive techniques) ACM, Seattle, Washington, United States.

BOUCHET J & NIGAY L (2004) ICARE: a component-based approeach for the design and development of multimodal interfaces. In: Proceedings of CHI'04), 1325-1328.

COHEN PR, JOHNSTON M, MCGEE D, et al. (1997) QuickSet: Multimodal Interaction for Distributed Applications. In: Proceedings of the fifth ACM international conference on Multimedia) ACM, Seattle, Washington, United States.

COHEN PR, MCGEE D & CLOW J (2000) The efficiency of multimodal interaction for a map-based task. In: Proceedings of the sixth conference on Applied natural language processing) Morgan Kaufmann Publishers Inc., Seattle, Washington.

FÖHRENBACH S, KÖNIG WA, GERKEN J & REITERER H (2008) Natural Interaction with Hand Gestures and Tactile Feedback for large, hight-res Displays. MITH'08: Workshop on Multimodal Interaction Through Haptic Feedback.

FUHRY L Elektrookulographie (EOG) = Elektronystagmographie (ENG).) Klinikum Ingolsadt.

GARFOLO J, LAMEL L, FISHER W, FISCUS J & PALLET D (1992) TIMIT Acoustic-Phonetic Continuous Speech Corpus. Linguistic Data Consortium, Philadelphia, USA.

GERAMISOS P, CHALAPATHY N, GRAVIER G, ASHUTOSH G & ANDREW WS (2003) Recent Advances in the Automatic Recognition of Audiovisual Speech. In: Proceedings of the IEEE).

GRAPS A (2004) Fourier Analysis.).

GREBE P (1966) Duden - Fremdwörterbuch. Dudenverlag, Mannheim.

GROUP S (2008) CMUSphinx: teh Carnegie Mellon Sphinx Porject.) Carnegie Mellon.

GUIMBRETIÈRE F, STONE M & WINOGRAD T (2001) Fluid interaction with high-resolution wall-size displays. In: Proceedings of the 14th annual ACM symposium on User interface software and technology) ACM, Orlando, Florida.

HAN J (2006) Perceptive Pixel Site.).

HEALTHIMAGINGNEWS (2007) Speech recognition demonstrates fewer errors than transcribed reports.).

ISO 9241-9:2000 (2000) Ergonomic requirements for office work with visual display terminals (VDTs) - Part 9: Requirements for non-keyboard input devices.).

JUANG BH & RABINER L (2004) Automatic Speech Recognition – A Brief History of the Technology Development Abstract.) Rutgers University and the University of California, Santa Barbara.

KECMAN V (2001) Learning and Soft Computing - Support Vector Machines, Neural Networks and Fuzzy Logic Models.) The MIT Press, Cambridge, MA.

KEVIN C, BILL K, BEN S & ADEL Y (2000) A comparison of voice controlled and mouse controlled web browsing. In: Proceedings of the fourth international ACM conference on Assistive technologies) ACM, Arlington, Virginia, United States.

KÖNIG WA, BIEG H-J & REITERER H (2007a) Laserpointer-Interaktion für grosse, hochauflösende Displays. In: Mensch & Computer 2007: Interaktion im Plural, 7. Konferenz für interaktive und koorperative Medien) Oldenbourg Verlag, 69-78.

KÖNIG WA, BIEG H-J, SCHMIDT T & REITERER H (2007b) Position-Indepentent Interaction for large high-resolution Displays. IHCI'07: Proceedings of IADIS International Conference on Interfaces and Human Computer Interaction 2007.

KÖNIG WA, BÖTTGER J, VÖLZOW N & REITERER H (2008) Laserpointer-Interaction between Art ans Science. IUI'08: Proceedings of the 13th international conference on Intelligent User Interfaces, 423-424.

MANARIS B & HARKREADER A (1998) SUITEKeys: A speech understanting interface for the motor-control challenged. In: Proceedings of the 3rd International ACM SIGCAPH).

MARTELL D (2007) Intel's Moore muses on end of Technology Maxim.) Reuterns 2007.

MCNEILL D (1992) Hand and Mind: What Gestures Reveal about Thought.) University of Chicago Press, Chicago.

MICROSOFT (2008) Microsoft Speech API (SAPI 5.3).).

MICROSYSTEMS S (2008) Java Speech API.).

MOORE GE (1965) Cramming More Components Onto Integrated Circuits. In: Proceedings of the IEEE), 82 - 85.

NEWELL A, BARNETT J, FORGIE J, GREEN C & KLATT D (1971) Speech-Understanding System: Final Reports of a Study Group.) CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE.

NICKEL K & STIEFELHAGEN R (2003) Pointing gesture recognition based on 3D-tracking of face, hands and head orientation. In: Proceedings of the 5th international conference on Multimodal interfaces) ACM, Vancouver, British Columbia, Canada, 140-146.

NUANCE (2008a) Dragon NaturallySpeaking 10.

NUANCE (2008b) IBM ViaVoice Releas 10.).

OH J-Y & STUERZLINGER W (2002) Laserpointer as Collaborative Pointing Deivce. Graphics Interface, 141-149.

OVIATT S (1997) Multimodal interactive maps: Designing for human performance. Human-Computer Iteraction 12.

OVIATT S (1999a) Mutual disambiguation of recognition errors in a multimodal architecture. Proceedings of the Conference on Human Fatrocs in Computer Systems, CHI'99.

OVIATT S (1999b) Ten Myths of Multimodal Interaction. Communications of the ACM 42.

OVIATT S (2002) Multimodal Interfaces. Handbook of Human-Computer Interaction.

OVIATT S, DEANGELI A & KUHN K (1997) Integration and synchronization of input modes during mulitmodal human-computer interaction. Proceedings of the Internaitonal Conference on Spoken Language Processing.

OVIATT S & KUHN K (1997) Referential features and linguistic indirection in multimodal language. . Proceedings of the Internaitonal Conference on Spoken Language Processing.

RABINER L (1989) A tutorial on Hidden Markov Models and selected Applications in Speech Recognition. Proceedings of the IEEE 77, 257 - 286.

RICHARD M & LIPPERMANN R (1991) Neuronal network classifiers estimate bayesian a posteriori probabilities. Neural Computation 3, 461- 483.

SCHAFER RW (1995) Scientific bases of human-machine communication by voice.) Departement of Electrical and Computer Engineering, erogia Institut of Technology, Atlanta, Georgia, USA.

SCHNEIDER E & EGGERT T (2006) Methoden der Augebwegegunsmessung.) Neurologische Poliklinik, Klinikum der Universität München - grosshadern, München.

SCHWARTZ R & CHOW YL (1990) The N-Best Algorithm: En Efficient and Exact Procedure for finding the N Most likely Sentence Hypotheses. Proceedings of the Internaitonal Conference on Acousitg, Speech and Signal Processing.

SHARMA R, ZELLER M, PAVLOVIC VI, HUANG TS & SCHULTER K (2000) Speech/ Gesture Interface to a Visual-Computing Environment. IEEE Computer Graphics and Applications.

SHNEIDERMAN B & PLAISANT C (2005) Designing the User Interface. Pearson Education Inc.

SPEECHSTUDIO (2008) Speech Studio Inc.).

SR-RESARCH (2008) SR Research EyeLink II - Head Mounted System Overview.).

TAN YK, NASSER S & ALLEN T (2003a) Error Recovery in a Blended Style Eye Gaze and Speech Interface. ICMI'03.

TAN YK, SHERKAT N & ALLEN T (2003b) Eye Gaze and Speech for Data Entry: A Comparison of Different Input Methods. Accepted for publication in IEEE Internacional Conference on Multimedia and Expo (ICME).

VOGEL D & BALAKRISHMAN R (2005) Distant Freehand Pointing and Clicking on Very Large, High Resolution Displays.) Departement of Computer Science, University of Toronto, Toronto, Canada.

VOXIT (2008) Voxit.).

W3C (2004) Speech Recognition Grammar Specification Version 1.0.).

WEISER M (1993) Some Computer Science Issues in Ubiquitous Computing. Communications of the ACM 36.

YOST B & NORTH C (2006) The Perceptual Scalabiilty of Visualization. In: IEEE Transactions on Visualization and Computer Graphics).

Abbildungsverzeichnis