Kommandofreie Interaktionen

JOHANN HABAKUK ISRAEL

Zentrum Mensch-Maschine-Systeme, Technische Universität Berlin

Schlüsselwörter: Mensch-Rechner-Interaktion, WIMP-Schnittstellen, Kommandofreie Interaktion, Virtual Reality, Blickgesteuerte Interaktion, Gazetracking

1. Einleitung

Charakteristisch für heutige Computersysteme ist, daß sie einen sehr hohen Umfang an Funktionalität zur Verfügung stellen, es aber relativ schwierig ist, diese Funktionalitäten einzusetzen. Computerarbeit erfordert eine lange Einarbeitungszeit und den Willen und das Können zum Abstrahieren, da sie sich stark von herkömmlichen Arbeitsformen unterscheidet. Ein Grund hierfür sind die vorherrschenden Mensch-Maschine-Modelle, die zur Modellierung von Computerschnittstellen herangezogen werden (Geiser 1990).

Computer sind heute aufgrund ihrer hohen Leistungsfähigkeit potentiell in der Lage, dem Benutzer viel besser zu dienen, als sie es tatsächlich tun. Ziel dieser Arbeit ist es, dieses Defizit aufzuzeigen und das Konzept der kommandofreien Interaktion einzuführen, die computergestützte Arbeit einfacher, intuitiver und natürlicher machen soll. Ein wichtiger, wenn nicht sogar der entscheidende Schritt dafür ist das gedankliche Trennen von den Erfahrungen, die man mit herkömmlichen Benutzerinterfaces gemacht hat. Erst dann kann man sich unvoreingenommen die Frage stellen: „Was will der Benutzer eigentlich tun?“ (Laurel & Mountford 1992, S. XIII).

Zu Beginn wird in Kapitel 2 auf die Entwicklung der Computer-Benutzer-Schnittstellen eingegangen, um darauf aufbauend in Kapitel 3 deren heutige Ausprägungen erklären zu können. Dem wird in Kapitel 4 das Konzept kommandofreier Interaktion entgegengesetzt, und am Beispiel blickbasierter Interaktion konkretisiert.

2. Entwicklung der Mensch-Computer-Schnittstelle

Computer sind nicht in erster Linie mathematische Rechner. Viel wichtiger ist ihre Fähigkeit, Symbole, mit denen Menschen eine Bedeutung assoziieren, darzustellen, zu verknüpfen und zu manipulieren. Computer sind Simulationswerkzeuge für gedankliche Modelle, sie sind ein Medium, das andere Medien darstellen kann, auch solche, die physisch nicht existieren können (Gassée & Rheingold 1992). Die Art

und Weise, wie gedankliche Modelle vom Computer simuliert und dargestellt werden, hat sich über die Zeit geändert, und damit auch die Vorstellung der Computerbenutzer vom dem, was machbar und was nicht machbar ist.

Der iterative Prozeß der Entwicklung der Computertechnik hat bisher vier Generationen von Computern hervorgebracht. Meistens werden diese Generationen über die fundamentalen Veränderungen in der zugrundeliegenden Hardwaretechnologie beschrieben. Alternativ könnten sie auch über ihre Benutzerschnittstelle definiert werden (Tabelle 2-1) (Henning 1997; Nielsen 1996; Skerjanc & Pastoor 1997).

Tabelle 2-1a: Zusammenfassung von Computergenerationen nach Nielsen (1996, S 2).

Generation	**Hardware Technologie**	**Bedienung**	**Programmier-sprache**
0 **0 – 1945** **Prä-historisch**	Mechanisch, Elektromechanisch (Zuse Z3)	Keine tatsächliche Benutzung, außer durch Experten	Umstecken und Bewegen von Kabeln
1 **1945 – 1955** **Pioniere**	Riesige Maschinen, Vakuumröhren, enormer Kühlungsaufwand, hohe Fehlerrate	Nur ein Benutzer kann eine Maschine zur gleichen (begrenzten) Zeit benutzen (besitzen)	Maschinensprache (001100111101)
2 **1955 – 1965** **Historisch**	Transistoren, höhere Zuverlässigkeit, Computer werden erstmals außerhalb von Labors benutzt	Stapelverarbeitung („Computer als Tempel“, dem man Offerten macht, um eine Antwort des Orakels zu bekommen)	Assembler (ADD A,B)
3 **1965 – 1980** **Traditionell**	Integrierte Schaltkreise in hohen Stückzahlen; kosteneffizienter Einsatz in Unternehmen	Zeitteilung	„High-level“ Sprachen wie FORTRAN und PASCAL
4 **1980 – 1997** **Modern**	Preiswerte, leistungsfähige Personal-Computer (PCs) werden privat eingesetzt	Einbenutzer Personal-Computer	Problemorientierte Sprachen
5 **1997 - ?** **Zukunft**	Computer-on-a-chip; Personen können *viele* Computer kaufen	Vernetzte Einbenutzersysteme und eingebettete Systeme	Nichtimperative Sprachen, möglicherweise grafisch

Tabelle 2-1b: Zusammenfassung von Benutzerschnittstellen nach Nielsen (1996, S. 2).

Generation	Terminal Technologie	Benutzer-spektrum	Beworbenes Bild	Benutzer Schnittstellen (Beispiele)
0 **0 – 1945** **Prä-historisch**	Auslesen von blinkenden Lichtern und Lochkarten	Die Erfinder selbst	Keines	Keine (Nur direkter Zugriff auf Hardware)
1 **1945 – 1955** **Pioniere**	Typenraddrukker. Benutzung nur in Computerzentren	Experten, Pioniere	Computer als (Be-) Rechner	Programmierung
2 **1955 – 1965** **Historisch**	Zeilenbasierte Terminals	Technokraten, professionelle Computerspezialisten	Computer als Informations-verarbeiter	Kommando-sprachen
3 **1965 – 1980** **Traditionell**	Bildschirm-Terminals, ausschließlich Alphanumerische Zeichen. Fernzugriff verbreitet	Gruppen ohne Computerwissen (z. B. Bankangestellte)	Mechanisierung von Vorgängen	Streng hierarchische Menüs und Formulare
4 **1980 – 1997** **Modern**	Grafische Displays mit guter Auflösung. Schreibtisch-computer und schwere Portable	Geschäftliche, berufliche Nutzer, interessierte Privatleute	Computer als Werkzeuge. Erhöhung persönlicher Produktivität	WIMP (Windows, Icons, Menus, Pointing device)
5 **1997 - ?** **Zukunft**	Leichte, handliche Portable mit Multimedia I/O und Modem	Jeder	Computer als Unterhalter	Syntaxfreie Schnittstellen

Computer der ersten Generation waren praktisch nur von Experten zu bedienen. Auch die Kommandozeilen-Interfaces der zweiten Generation verlangten vom Benutzer viele Vorkenntnisse. Kommandozeilen-Interfaces (wie MS-DOS) sind relativ einfach zu programmieren, sind aber in anderen Anwendungsfällen unangebracht. Das Problem liegt darin, daß ein unerfahrener Benutzer die anwendbaren Komman-

dos und die zu verwendende Syntax nicht kennt. Deshalb muß er mutmaßen, oft mit fatalen Folgen. Um dieses Problem zu umgehen, wurden Hilfen wie Referenzkarten, Online-Hilfesysteme oder Shells entwickelt, die das Problem mit sich brachten, daß die Suche nach einem Kommando mit wachsendem Vokabular der Kommandosprache immer länger dauerte.

Mit den Menüsystemen der dritten Generation wurden die dem Benutzer zur Verfügung stehenden Optionen explizit aufgelistet. Im Prinzip wurde damit das, was bisher das Online-Hilfesystem war, zur Benutzerschnittstelle. In Menüsystemen sieht der Benutzer Optionen und wählt eine davon aus. Damit ist ein Problem der Kommandozeilen-Interfaces gelöst, denn die verfügbaren Kommandos sind ohne Vorkenntnisse erreichbar. Menüsysteme haben aber ihre eigenen Probleme. Das erste ist der begrenzte Platz auf dem Bildschirm. Wenn jede Option mit all ihren Eingaben im Detail beschrieben würde, wäre der Bildschirm schnell mit einigen wenigen Optionen gefüllt. Werden die Optionen alternativ nur knapp erläutert, haben Menüsysteme nicht mehr den Vorteil, selbsterklärend zu sein. Fortgeschrittene Benutzer fühlen sich durch Menüsysteme oft behindert und gelangweilt, da sie die von ihnen benutzten Funktionen kennen und der Auflistung aller Optionen in einem Menü nicht mehr bedürfen. Für sie wurden Funktionstasten (shortcuts) eingeführt, die durch das Drükken von meist ein oder zwei Tasten die gewünschte Funktion sofort auslösen, ohne daß der Benutzer sie in einem Menü auswählt.

Wenn ein großes Kommandovokabular mit Menüs dargestellt werden soll, stößt man auf neue Probleme. Menüs mit mehr als ca. zehn Einträgen werden unhandlich. Der Benutzer muß jeden Eintrag der Liste überfliegen, bei langen Listen wird dieser Vorgang mühsam. Außerdem sind die meisten Einträge für den Benutzer in seiner augenblicklichen Situation nicht von Interesse, bzw. können nicht angewandt werden, so daß er beim Lesen irrelevanter Optionen Zeit verschwenden muß. Alternativ können verschachtelte Menüsysteme eingesetzt werden, in denen Menüs wieder neue Menüs hervorbringen können. Dadurch sieht der Benutzer nur kurze Menüs mit sachdienlichen Einträgen, es entsteht aber das Problem der Menünavigation. Der Benutzer kann beim Traversieren des Menüsystems leicht den Überblick verlieren.

Die Grafischen Benutzerschnittstellen der 4. Generation brachten einen erneuten Fortschritt in Hinblick auf intuitive Bedienbarkeit von Computern. Diese sogenannten WIMP-Interfaces sind die heute am weitesten verbreitete Schnittstelle zwischen Mensch und Computer. Auf die Arbeit mit grafischen Benutzerschnittstellen wird im nächsten Kapitel eingegangen.

3. Kommando- und WIMP- basierte Benutzerschnittstellen

3.1 Struktur von WIMP-Schnittstellen

Seit 1983 gibt es immer mehr Computer mit WIMP-Benutzerschnittstellen (Fenstersysteme). Windows, Icon, Menu und Pointer, dies sind die Elemente, mit denen Benutzer ihre Arbeit am Computer realisieren müssen, über sie wird der Mensch-Maschine-Dialog organisiert. Computerprogramme (Applikationen), die Werkzeuge des Benutzers, deren unterschiedliche Funktionalitäten und die zu bearbeitenden Daten werden in einem Window dargestellt. Funktionalitäten werden Benutzern über Icons und Menüs angeboten, die sie mit Pointing-Devices wie der Maus auswählen und

auslösen müssen (Reichmann 1986). Diese vier genannten Elemente ordnen sich an unterschiedlichen Stellen in das Konzept von Fenstersystemen ein, würden aber ohne weitere Komponenten nicht arbeiten können (Abbildung 3-1). Seit 1983 gibt es immer mehr Computer mit WIMP-Benutzerschnittstellen (Fenstersysteme). Windows, Icon, Menu und Pointer, dies sind die Elemente, mit denen Benutzer ihre Arbeit am Computer realisieren müssen, über sie wird der Mensch-Maschine-Dialog organisiert.

Computerprogramme (Applikationen), die Werkzeuge des Benutzers, deren unterschiedliche Funktionalitäten und die zu bearbeitenden Daten werden in einem *Window* dargestellt. Funktionalitäten werden Benutzern über *Icons* und *Menüs* angeboten, die sie mit *Pointing-Devices* wie der Maus auswählen und auslösen müssen (Reichmann 1986). Diese vier genannten Elemente ordnen sich an unterschiedlichen Stellen in das Konzept von Fenstersystemen ein, würden aber ohne weitere Komponenten nicht arbeiten können. (Abbildung 3-1).

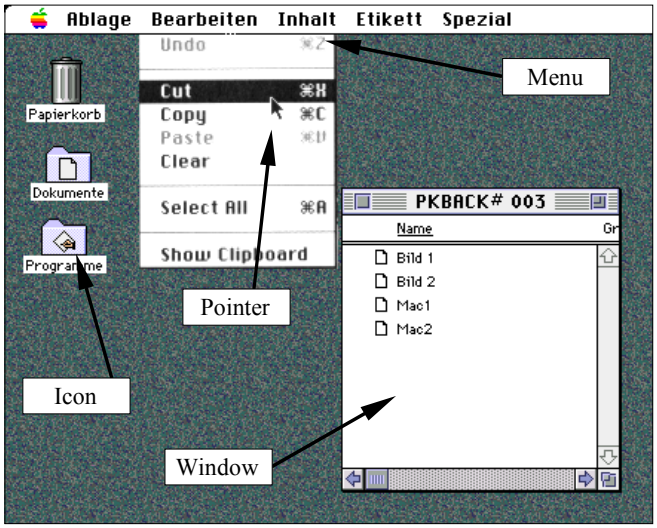

Abbildung 3-1: Elemente von WIMP-Schnittstellen

Schon zwischen 1971 und 1981 wurde am XEROX Forschungszentrum Parc das erste Fenstersystem Star entwickelt. Xerox Star und sein Nachfolger ViewPoint waren zwar wenig verbreitet (wenige tausend Exemplare), haben aber die Macintosh-Welt nachhaltig geprägt, nachdem 1979 Star Entwickler zu Apple wechselten. Dort entstand nach kurzer Entwicklungszeit 1983 ein Rechner namens Lisa, der von Anfang an eine grafische, fensterorientierte Benutzeroberfläche besaß. Lisas niedrige Geschwindigkeit, zahlreiche Kinderkrankheiten und der hohe Preis verhinderten jedoch ihre Ausbreitung. Durch seine Grafikfähigkeiten und die Einfachheit der Benutzung konnte dann der Macintosh als überarbeitete Version über Jahre erfolgreich sein. Seit 1990 ist MS-Windows der de facto Standard in der PC-Welt. Es hat inzwischen einige wesentliche Ideen des Macintosh in etwas anderer grafischer Aufmachung adaptiert. MS-Windows hat durch seine rasante Verbreitung dafür gesorgt, daß Fenstersysteme heute die dominierenden Benutzerschnittstellen sind. Eine ähnliche Rolle fällt dem 1986 entstandenen X in der Welt der Workstations zu. Obwohl X unabhängig vom Betriebssystem ist, gilt es als Unix-Fenstersystem. Es existieren auch Portierungen auf PCs und Großrechner. X ist ein verteiltes Fenstersystem und erlaubt Zusammenarbeit über Bildschirme hinweg. Diese Möglichkeiten bieten auch

Fenstersysteme wie NeWS und Andrews, die aber nie einen vergleichbaren Stellenwert wie X erreicht haben (Klingert 1996).

Als Mindestanforderungen an die Hardware eines heutigen Fenstersystems nennt Klingert (1996) für die Informationsausgabe einen hochauflösenden Rasterbildschirm und zur Informationseingabe durch den Benutzer ein Zeigeinstrument (meist Computermaus) und eine Tastatur. Nach Klingert sind Fenstersysteme Teil eines Betriebssystems. Sie übernehmen Aufgaben, denen sich das Betriebssystem nicht widmet, z. B. die grafische Ausgabe. Dem Window Manager (Fensterverwalter) ist ein Basisfenster (Rootwindow) zugeordnet, das im Hintergrund den gesamten Bildschirm ausfüllt. Die Windows der Anwendungen sind diesem Fenster hierarchisch untergeordnet. WIMP-Schittstellen liegt die Desktop-Metapher zugrunde, nach der der gesamte Bildschirm (das Rootwindow) den Schreibtisch des Benutzers abbilden soll, wobei die geöffneten Fenster zu bearbeitende Dokumente darstellen. Der Computer soll den herkömmlichen Schreibtischarbeitsplatz durch einen Computerarbeitsplatz substituieren.

Das softwaretechnische Grundprinzip von Fenstersystemen ist, „daß alle Dialogobjekte auf ein einheitliches Grundobjekt abgebildet werden. Dieses Grundobjekt ist das Fenster (Window); es bildet die Einheit, die verwaltet wird. Das Fenstersystem verwaltet alle Objekte dieses Typs nach dem gleichen Muster“ (Klingert 1996, S. 35). Ein funktionaler Vorteil von WIMP Interfaces gegenüber älteren Schnittstellen ist, daß die Fähigkeit des Computers zum Multitasking leicht zugänglich ist. Durch das Öffnen unterschiedliche Programmfenster kann der Benutzer zur gleichen Zeit mehrere Aufgaben mit dem Computer bearbeiten und zwischen ihnen umschalten. Problematisch dabei ist das Wechseln des Eingabefokus in einer angemessenen Geschwindigkeit. Unter dem Aspekt, daß Fenster Daten und Prozeduren (Methoden) als ein nach außen geschlossenes Objekt miteinander vereinigen, können Fenstersysteme als (softwaretechnisch) objektorientiert gelten.

Die vom Window Manager verwaltete Anzeige von Fenstern gestaltet sich relativ übersichtlich, solange auf Multitasking verzichtet wird, da das arbeitende Programm den gesamten Bildschirm ausfüllen kann. Sobald mehr als ein Window gleichzeitig dargestellt wird, steht der Benutzer vor der Frage, wie er den knappen Bildschirmplatz unter den Programmen aufteilen soll. Er hat dann folgende drei Möglichkeiten für die Bildschirmaufteilung (Abbildung 3-2):

- *Überlappende Darstellung* der Windows (overlapping).
- *Kacheldarstellung* der Windows (tiled), die Fenster füllen dabei zusammen den Bildschirm komplett aus, die Fensterrahmen schließen aneinander an.
- *Maximierte Darstellung* eines Windows (maximized), das den gesamten Bildschirm ausfüllt. Alle anderen Windows sind dann meist ikonisiert (minimiert).

Die *Interface-Kontrolle* muß bei WIMP-Schnittstellen direkt vom Benutzer vorgenommen werden, dazu wird ihm die Schnittstelle zum Window Manager und Ressourcen-Verwalter u.a. über grafische Elemente zugänglich gemacht (Klingert 1996; Nielsen 1996). Kommandos zur Interface-Kontrolle können durch Eingabe über Push-Buttons (wie „verschiebe Fenster“, „starte eine Anwendung“, „ikonifiziere Fenster“) und die implizite Vergabe des *Eingabefokus* durch Mausklicks in geöffnete Fenster erfolgen. Der Eingabefokus hat bei WIMP-Applikationen in der Regel nur

Auswirkung auf die Umleitung des Eingabestroms der Tastatur. Die Applikation, deren Fenster den Eingabefokus besitzt, erhält die Tastatureingaben.

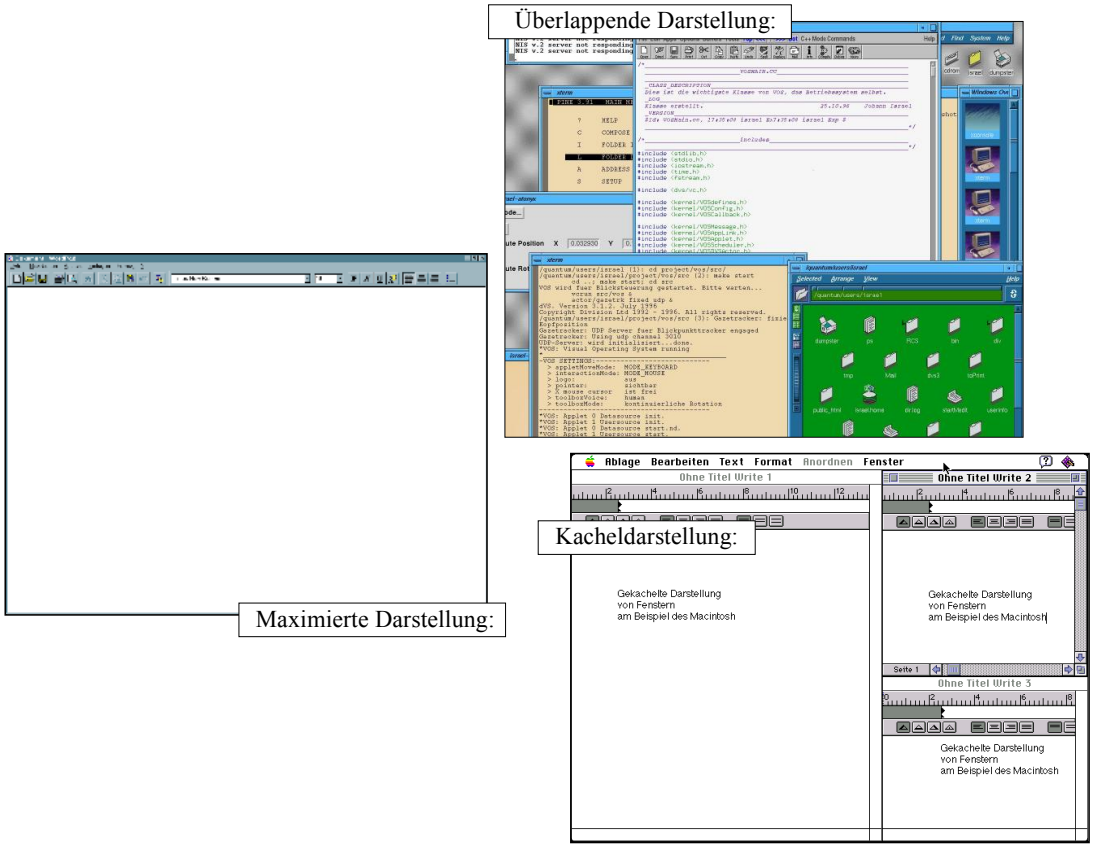

Abbildung 3-2: Mögliche Fensteranordnungen in WIMP-Benutzerschnittstellen

WIMP-Applikationen können auch Fenster sein, in denen Benutzerschnittstellen vergangener Generationen ausgeführt werden, z. B. Kommandozeilen-Schnittstellen und reine Menüsysteme. Typisch ist aber eine Benutzerschnittstelle, die durch Kompositionen und Verfeinerungen grafischer oder textueller Dialogbausteine entstanden ist (Klingert 1996). Einerseits können sie zum *Auswählen von Kommandos* und andererseits zum *Spezifizieren von Optionen* dienen. Abbildung 3-3 zeigt beispielhaft Standard-Dialogbausteine, die ein Entwicklungssystem zur Gestaltung für WIMP-Schnittstellen bereitstellt. Der hier abgebildete Pushbutton ist ebenfalls ein Element zum Auswählen von Kommandos.

Wichtige und namengebende Bestandteile von WIMP-Schnittstellen sind Menüs und Icons. Die bereits in Kapitel 2 beschriebenen Menüs kommen in WIMP Schnittstellen vor allem als feste Menüs mit Pulldown-Untermenüs (Macintosh) und Pop-Up-Menüs mit kaskadierenden Untermenüs (Motif) vor. Icons sind grafische und/oder textuelle Piktogramme mit unterschiedlichen Bedeutungen. Einerseits verkörpern sie ähnlich wie Push-Buttons eine Zusammenfassung von Kommandos (Makro-Grundeinheiten), deren Charakter auf dem Icon durch das Symbol ersichtlich ist. Der Benutzer kann den Sinn der Kommandos erahnen oder sich an deren Wirkung erinnern. Andererseits repräsentieren Icons grafische Austauschobjekte für minimierte Fenster, die durch Deikonisieren in die Form vor ihrer Ikonifizierung zurückversetzt werden können. Die Manipulation von Icons wird häufig als *direkte Manipulation* bezeichnet (Reichmann 1986).

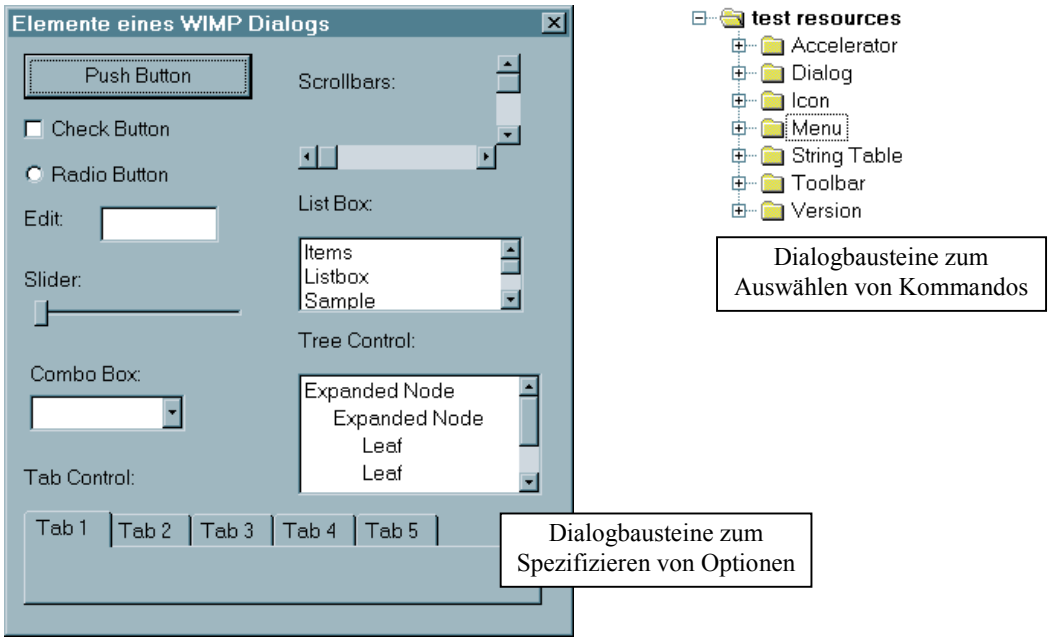

Abbildung 3-3: Dialogbausteine für die Gestaltung von WIMP-Schnittstellen

Eine in WIMP-Schnittstellen besonders häufige Form für die Informationseingabe sind Dialoge, nicht zu verwechseln mit dem Mensch-Maschine-Dialog. Dialoge sind Windows, in denen Dialogelemente zur Spezifizierung von Optionen komponiert werden. Hier unterscheidet man den Frage-Antwort-Dialog und den weiter verbreiteten Formular-Dialog. Der Frage-Antwort-Dialog findet dort Anwendung, wo der Benutzer ungeübt ist, und die Reihenfolge der einzugebenden Daten eine Rolle spielt. Der Formular-Dialog (auch Formular-Maske) besteht aus Eingabefeldern, die an Optionen unterschiedlicher Wertebereiche gekoppelt sind. Über die Manipulation der grafischen oder textuellen Repräsentation der Eingabefelder kann der Benutzer diese Optionen spezifizieren. Bei modalen Dialogen erfolgt nach der Informationseingabe in der Regel eine Bestätigung über die Betätigung eines Push-Buttons. Für die Erledigung gleichartiger Aufgaben wie Löschen, Kopieren, Öffnen und Speichern haben sich Standarddialoge etabliert, die applikationsübergreifend Anwendung finden (Anwenderwerkzeuge).

Herkömmliche WIMP-Applikationen und -Schnittstellen sind kommandobasiert. Die Interaktionen mit WIMP-Systemen erfolgen nach der Syntax einer Kommandosprache, d. h. der Reihenfolge der Selektionen und Spezifikationen von Kommandos, Objekten und Parametern erfolgt nach den Regeln einer Grammatik. Zu der aus Menüsystemen bekannten Strategie der Auflistung von Kommandos und deren Optionen kommen hier weitere Elemente zum Auswählen von Kommandos hinzu. Kommandos werden durch die Betätigung von Push-Buttons, Icons, Funktionstasten oder Menüeinträgen ausgelöst. Dabei ist das Kommando meist durch textuelle oder grafische Abstraktion auf dem entsprechenden WIMP-Element erkennbar (Abbildungen 3-1). Üblicherweise werden zuerst das Kommando und danach ggf. dessen Parameter und Objekte spezifiziert. Beispiel dafür ist eine Standardfunktionalität von Computerprogrammen: das explizite Laden von Daten. Hierzu muß aus einem Menü die Funktion zum Öffnen ausgewählt werden. In einem folgenden Dialog wird die zu öffnende Datei bestimmt, das Argument des Kommandos. Ändert der Benutzer Daten, so werden diese Änderungen erst nach einem expliziten Abspeichern permanent. Auch die Kontrolle des WIMP-Interface über Window Manager und Ressourcen-Verwalter ist

kommandobasiert, da auch sie auf die Dialogbausteine von WIMP-Applikationen zurückgreift (z. B. Push-Buttons und Funktionstasten). Moderne WIMP-Schnittstellen versuchen, objektorientierte Ansätze einfließen zu lassen, indem sie dem Benutzer nach der Spezifikation eines Objekts (z. B. Daten, Dokumente oder Programme) die darauf ausführbaren Operationen in Menüform anbieten (Objekt-Aktion-Paradigma; Klingert 1996).

3.2 Einsatz von WIMP-Schnittstellen

Eine Studie der Universität Strathclyde in Schottland, Großbritannien, gibt Aufschluß über die Benutzung von WIMP-Interfaces (Brooks & Johnston 1990). Tabelle 3-1 gibt die Ergebnisse der Studie wieder. Untersucht wurden der Gebrauch von Windows (überlappend oder gekachelt), die Benutzung von Tastenkombinationen und die Ausrichtung der Maus (links- oder rechtshändig). Es wurden 50 Testpersonen herangezogen, die aus dem Sekretariats- und Verwaltungsbereich (12, 11w, 1m) und wissenschaftlich-/technischen Bereich (38, 5w, 33m) kamen.

Tabelle 3-1 Studie über Benutzung von WIMP Interfaces in der täglichen Arbeit (Brooks & Johnston 1990, S. 7).

Kategorie	Werte
Alter	17 - 51 Jahre; ∅ 26.5 Jahre
Geschlecht	16 weiblich, 34 männlich
Beruf	38 technisch, 12 nicht technisch
WIMP Erfahrung	1 - 72 Monate; ∅ 42 Monate
Benutztes Computersystem	26 Workstation, 16 Macintosh, 8 andere
Anzahl der benutzten Windows	1 - 12 Windows; ∅ 3 Windows
Anordnung der Windows	31 überlappend, 7 gekachelt, 12 k. A.
Empfundene Effizienzverbesserung gegenüber Kommandozeilen-Interfaces	39 ja; 11 nein
Neuanordnung der Windows während der Sitzung	17 ja; 33 nein
Handausrichtung	44 Rechtshänder; 6 Linkshänder
Benutzung von Kommandotasten	31 ja; 19 nein

Von besonderem Interesse ist der Durchschnittswert 3 für die Anzahl gleichzeitig dargestellter Windows, in dem inaktive Fenster wie Bildschirmuhren oder Statusfenster nicht enthalten sind. Alle sieben Testpersonen, die professionell Workstations benutzen, haben gekachelte Windows eingesetzt und die Windows während der Sitzung nicht viel verschoben. Daraus schlußfolgern die Veranstalter der Studie, daß die

Benutzung von gekachelten Windows ein Zeichen für gut organisierte Computerarbeit ist. Zur Anordnung der Windows wurden weiterhin folgende Beobachtungen gemacht:

6,3 % aller Windows-Kommandos während des normalen Betriebs betreffen die Bewegung und Größenänderung von Windows (29,5 % während des Bildschirmaufbaus).

9 von 11 Frauen, die im Sekretariats-/Verwaltungsbereich arbeiten, ordneten die Windows während der Arbeit kaum neu an.

Linkshänder ordnete die Windows während der Arbeit selten neu an. Außerdem benutzten sie die Maus alle in der rechten Hand.

Folgende Aussagen wurden u. a. von Testpersonen gemacht:

Die Windows wurden nur zu Beginn angeordnet. (5 Testpersonen)

Die Neuanordnung von Fenstern kann unnötig Zeit verschwenden. (2)

Neuanordnung ist ein integraler Bestandteil der Benutzung von Windows. (4)

Obwohl die Studie schon sieben Jahre alt ist, gibt sie Aufschluß über das, was erfahrenen und unerfahrenen Computerbenutzern zugemutet werden kann. Wissen über das Benutzerverhalten, das aus Untersuchungen wie der beschriebenen gewonnen wird, kann in die Modelle einfließen, die der Entwickler und das System vom Benutzer haben.

3.3 WIMP-Interaktionen aus Benutzersicht

Zur Zeit ihrer Einführung stellten kommandobasierte WIMP-Computerschnittstellen in vielerlei Hinsicht einen Fortschritt dar. Aus der Erfahrung im Umgang mit solchen Schnittstellen können heute eine Reihe von Verbesserungsvorschlägen und Kritikpunkten benannt werden.

Mit Abstand betrachtet, werfen WIMP-Schnittstellen unzählige Fragen auf. In WIMP-Schnittstellen gelten eigene, abstrakte Regeln und Gesetzmäßigkeiten, die den nicht eingeweihten Benutzern fremd sind, und die er lernen muß. Klingert (1996) erklärt, warum dies so ist:

„Fenstersysteme sind auf Grund abstrakter ergonomischer Überlegungen entstanden. Sie wurden nicht durch systematische Analyse der Bedürfnisse des Menschen entwickelt. Vielmehr hat man es hier mit einem evolutionären Ansatz zu tun. Das Ziel, einen benutzerangemessenen Umgang mit Rechnersystemen zu erfinden, war einigermaßen klar. Der Weg dorthin führte jedoch nur in Einzelfällen über das Studium psychologischer Fakten, physiologischer Erkenntnisse oder über Resultate der Arbeitswissenschaft. Der Weg ist also durch Versuch und Irrtum charakterisierbar“ (Klingert 1996, S. 12f).

Daraus folgt die entscheidende Frage, ob aufgrund neuer technischer Möglichkeiten ein *Neuanfang* in der Konzeption der Mensch-Maschine-Interaktionen notwendig ist, der die menschlichen Fähigkeiten als Entwicklungsgrundlage einbezieht. Klingert verneint das und entwickelt WIMP-Schnittstellen weiter. Gordon Kurtenbach weist dagegen am Beispiel menschlicher Gesten auf die Probleme hin, die sich aus der

Vernachlässigung des menschlichen Verhaltens ergeben: „Gesten sind so sehr Teil des Lebens, daß wir sie selten bemerken (...) Aber wenn wir uns vor den Computer setzen, hören unsere Gesten auf (...) Wie wichtig Gesten sind, wird deutlich, wenn wir uns Kommunikation und andere Aktivitäten ohne sie vorstellen (...) Alle diese Aktivitäten müssen warten bis ein Computersystem entwickelt ist, das die Gesten erkennen kann, die schon ein zweijähriges Kind versteht“ (Kurtenbach & Hulteen 1992, S. 309f.).

Zwischen herkömmlicher Arbeit z. B. mit Schreibwerkzeugen oder Elektrogeräten und der am Computer besteht immer noch ein großer Unterschied. „Computerbenutzer müssen während ihrer Arbeit traditionsgemäß sehr vorsichtig mit ihrem System umgehen. Das führt dazu, daß man die Benutzung eines Computers oft als genau das empfindet: die Benutzung eines Computers, und nicht als das direkte Bearbeiten einer Aufgabe. Von Benutzern wird erwartet, daß sie die richtigen Kommandos herausfinden und Kommandospezifikationen in korrekter Syntax zusammenstellen“ (Nielsen 1996, S. 11). Das heißt, WIMP- und kommandobasierte Computerschnittstellen sind *syntaktisch*[1].

In herkömmlichen Computerinterfaces gibt es keine Gewichtung von Kommandos und Funktionalitäten in Bezug auf deren Anordnung und Zugänglichkeit. Der Aufwand für die Steuerung des Fensterverwalters ist im Verhältnis zur Ausführung gewichtiger Kommandos wie dem Formatieren einer Festplatte relativ hoch. Die Gestaltungsrichtlinien, nach denen WIMP-Schnittstellen entworfen werden, sind meist historisch gewachsen. Um Benutzern das Erlernen neuer Programm-Funktionalitäten zu erleichtern, haben sich *generische Dialoge* etabliert, die in unterschiedlichem Kontext gleiche oder ähnliche Funktionen erfüllen (Geiser 1990). Die Menüleiste eines WIMP-Programms beginnt z. B. häufig mit den Kategorien „Datei“ bzw. „Ablage“, „Bearbeiten“, ..., „Hilfe“, die in keinem logischen Zusammenhang stehen. Durch dem Einsatz generischer Dialoge wird zwar Einarbeitungszeit gespart, andererseits werden die Nachteile von WIMP-Schnittstellen manifestiert. Viele WIMP-Applikationen bilden ihre interne Funktionalität direkt auf die Benutzerschnittstelle ab und schaffen computerfreundliche, statt benutzerfreundliche Schnittstellen.

Die Interaktionen in WIMP-Schnittstellen sind vorrangig *indirekte Manipulationen* an Objekten. Die für die Manipulation zur Verfügung stehenden WIMP-Dialogbausteine haben keine unmittelbar erkennbare Verbindung zu den durch sie beeinflußten Objekten, sondern müssen vom Benutzer aufgrund seiner Erfahrungen im Umgang mit WIMP-Schnittstellen miteinander assoziiert werden. Z. B. wird der Schriftstil eines Textes in Textverarbeitungs-programmen über Menüeinträge oder lose gruppierte Icons ausgewählt, die in keinem räumlichen Bezug zu Text oder Schreibwerkzeug (Cursor, Tastatur) stehen. Natürliche Interaktionen sind dagegen immer direkte Manipulationen am Objekt, z. B. wird über das Wechseln der Schreibfeder das Schreibwerkzeug verändert. Der Benutzer von WIMP-Schnittstellen kann also sein erlerntes Interaktionsverhalten nicht anwenden und wird gezwungen, neue Interaktionssprachen zu erlernen.

Die Organisation von Dokumenten und Daten wird in WIMP-Schnittstellen häufig in einer Ordner/Dokument-Hierarchie vorgenommen, die sich an die Verwahrung von stofflichen Dokumenten in Ordnern im Arbeitsleben anlehnen soll. Sie wird direkt

[1] Syntaktisch gefügte Gruppe von Kommandos, bei der jedes Glied seinen Wert erst durch die Fügung bekommt.

auf die Struktur physischer Speichermedien abgebildet. Eine Trennung von Dokumenten und anderen Dateien wie ausführbaren Programmdateien (Werkzeugen) gibt es nicht. Die Ordner-Metapher verliert deshalb ihre Verständlichkeit für unerfahrene Benutzer.

Durch WIMP-Interaktionen, die meistens dialogbasiert sind, entsteht ein einziger, *sequentieller Ein-/Ausgabestrom*. Der Benutzer kommuniziert im „Ping-Pong-Modus“ alternierend mit dem Computer, d. h. auf eine Eingabe von ihm folgt eine Ausgabe des Computers. Auch die an den Computer angeschlossenen Eingabegeräte ermöglichen nur eine sequentielle Eingabe, selbst wenn sie technisch parallel ausgelesen werden könnten. Benutzer bedienen z. B. nicht parallel Maus und Tastatur, weil sie es erstens nicht können und dies zweitens konzeptionell auch gar nicht vorgesehen ist. Durch die mangelhaften und langsamen Eingriffsmöglichkeiten, die der Benutzer in das System hat, entsteht ein Eingabestrom mit einer sehr *niedrigen Bandbreite Mensch -> Computer*. Durch die hohe Leistungsfähigkeit der Computersysteme wird der Benutzer mit einer Menge an Informationen versorgt, es entsteht

ein Ausgabestrom mit einer *sehr hohen Bandbreite Computer -> Mensch* (Jacob 1996) (Abbildung 3-4).

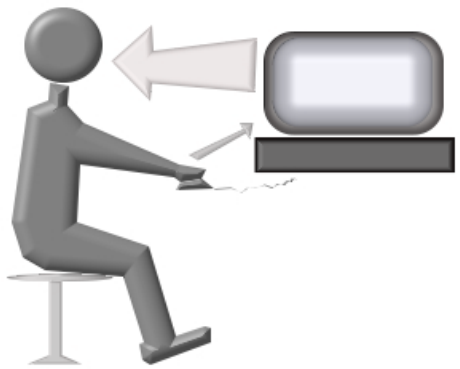

Abbildung 3-4: Herkömmliche Benutzerschnittstelle mit hoher Bandbreite Computer-Mensch und niedriger Bandbreite Mensch-Computer

Um die Arbeitsgeschwindigkeit erträglich zu halten, richten sich Benutzer ihre Oberflächen (bewußt oder unbewußt) so ein, daß alle während des Arbeitens nötigen Fenster und Kommandos sofort zugänglich sind. Deshalb sind WIMP-Schnittstellen nach kurzer Arbeitszeit in der Regel mit Windows, Icons, Buttons etc. überfrachtet. Es kommt zu einer Informationsüberflutung. Mit der wachsenden Komplexität einer zu lösenden Aufgabe wächst auch die Anzahl der nötigen Fenster (siehe Abbildung 3-2, überlappende Windows). Dem Benutzer fehlt der *Platz,* seine Arbeit vor sich ausbreiten zu können.

Interessanterweise gibt es ein Softwaregenre, in dem von Anfang an auf höchste Interaktivität und intuitive Handhabung geachtet wurde: Computerspiele (Crawford 1992). Anders als bei Standardsoftware unterliegen Benutzerschnittstellen von Computerspielen einem sehr hohen Anspruch: sie müssen Spaß machen. Spiele mit umständlicher Handhabung haben keine Chance und werden gnadenlos verdrängt. Die hier geltenden Kriterien kann man als *Forderungen an WIMP- und kommandobasierte Applikationen* aufstellen:

Parallele Eingabegeräte und -ströme, wie bei Autorennspielen, die über Joystick/Lenkrad und Fußpedal gesteuert werden können.

Intuitive Handhabung ähnlich der in Adventures, bei denen Gegenstände selbsterklärend sein müssen und Spieler sofort wissen, was sie mit ihnen zu tun haben.

Simulation realer Abläufe und Umgebungen, so fängt z. B. beim Autofahren der Motor an zu heulen, wenn er überdreht wird, und das Bild wackelt, wenn man über holprige Straßen fährt.

Auch in der Anwendung von 3D in Benutzerschnittstellen sind Spiele Vorreiter. Umgebungen, in die sich ein Spieler begibt, werden mehr oder weniger räumlich exakt dargestellt, Licht und Schatten lassen Objekte eindeutig im Raum erscheinen. So gelingt es dem Spieler, sein natürliches räumliches Orientierungsvermögen einzusetzen.

Zusammenfassend kann festgestellt werden, daß bei der Benutzung von WIMP-Interfaces aus der Fülle der motorischen und sensorischen Fähigkeiten des Menschen nur der visuellen Sinn und die Motorik der Hand nennenswert zum Einsatz kommen. Die abstrakte Funktions- und Kommandostruktur der Maschine wird häufig in der WIMP-Schnittstelle sichtbar, die kognitiven Fähigkeiten des Menschen werden selten bewußt eingeplant, effiziente sensomotorische Prozesse sind kaum möglich.

4. Kommandofreie Interaktionen

In den vorigen Abschnitten wurde knapp das heutige Verhältnis Mensch-Computer dargestellt. Es wurde gezeigt, daß trotz graphischer Benutzerschnittstellen Probleme und Unstimmigkeiten bestehen, die den Umgang mit Computern nachhaltig prägen. In diesem Kapitel wird nun das Konzept kommandofreier Interaktion eingeführt, das helfen soll, die Beziehung Mensch-Computer zu entspannen.

Kommandofreie Interaktionen verlangen vom Benutzer nicht, spezifische Funktionen auszulösen, sondern interpretieren seine Eingaben und handeln danach (Green & Jacob 1992; Jacob 1993; Nielsen 1996). Kommandofreie Interaktionen sind objektorientiert, d. h. dem Benutzer werden Objekte zur Verfügung gestellt, mit denen er interagieren kann. Aktionen des Benutzers werden immer im Zusammenhang mit den betroffenen Objekten interpretiert, Reaktionen des Computers beziehen sich immer auch auf das betroffene Objekt. Nach Reichmann kann dies als *direkte Interaktion* verstanden werden (Reichmann 1986). Der Charakter kommandofreier Interaktion wird im folgenden Beispiel von Jacob Nielsen deutlich:

„Gestenbasierte Schnittstellen, wie beispielsweise Computerstifte könnten den Eindruck von papierartigen Schnittstellen vermitteln, und man denkt beim Schreiben, Malen oder Editieren auf Papier-Notizzetteln sicherlich nicht an Syntax. Erlaubt man z. B. dem Benutzer, Texte zu editieren, indem er Korrekturmarkierungen direkt in den Text schreibt, gibt es keine Notwendigkeit für eine Syntax, die zwischen verschiedenen Indikatoren unterscheidet, welche Funktion ausgeführt und welchem Objekt sie zugewiesen werden soll, da beide durch eine einzige Korrekturmarkierung spezifiziert sind, z. B. das Löschen eines Wortes, indem man es durchstreicht. Die Schlüsselaussage ist hier, daß die Spezifizierung von Aktion und Objekt zu einer einzigen Eingabe (input token) vereinigt wird, im Gegensatz zur Komposition eines

Stroms aus Benutzereingaben“ bei herkömmlichen Benutzerschnittstellen (Nielsen 1996, S. 86).

Neben kommadofreien existieren *syntaxfreie* und *non-WIMP-Schnittstellen*[1] als weitere Begriffe für eine neue Klasse von Benutzerschnittstellen. Non-WIMP-Benutzerschnittstellen (Windows, Icon, Menu, Pointer, siehe Kapitel 3) verzichten darauf, daß der Benutzer den Computer durch die Eingabe von Kommandos steuert (bedient). Sie basieren sie nicht mehr auf der Desktop Metapher, nach der der Computerbildschirm Abbild des gesamten Schreibtischs des Benutzers ist.

In kommandobasierten Benutzerschnittstellen wurde bisher durch die Reihenfolge von Benutzereingaben eine Reaktion des Computers ausgelöst. Syntax war notwendig, da frühere Schnittstellen auf einem begrenzten Benutzervokabular basierten, das kombiniert werden mußte, um komplexe Aktionen zu spezifizieren. Im Gegensatz dazu werden zukünftige Benutzerschnittstellen, die auf Gesteneingabe basieren, eine fast unendlich große Zahl verschiedener Eingabeelemente haben, die komplexe Absichten spezifizieren können. Die Zusammengehörigkeit von Objekten und Handlungen wird dabei implizit erkannt. Zukünftige Benutzerschnittstellen werden also *syntaxfrei* sein (Nielsen 1996).

Ein positiver Begriff für eine neue Klasse von Benutzerschnittstellen hat sich noch nicht etabliert, die Bezeichnungen „intuitive“ und „interaktive“ Benutzerschnittstellen würden sie aber treffend beschreiben. Der Unterschied zwischen kommandofreien, non-WIMP- und syntaxfreien Schnittstellen ist so gering, daß diese Begriffe außerhalb dieses Abschnitts synonym gebraucht werden.

4.1 Anforderungen an kommandofreie Interaktionen

Interaktionen können nicht entworfen werden. Kommandofreie Interaktionen werden realisiert, indem man das Interaktionsverhalten der Benutzerschnittstelle während der Mensch-Maschine-Interaktion modelliert; es entstehen *kommandofreie Benutzerschnittstellen*. Um eine Vorstellung von konkreten kommandofreien Benutzerschnittstellen aufzubauen, werden in diesem Abschnitt bereits bekannte Anforderungen an kommandofreie Benutzerschnittstellen zusammengestellt.

Eine positive Definition von neuartigen, nicht kommandobasierten Benutzerschnittstellen erhält man durch die Aufzählung von allgemeinen Anforderungen an zukünftige Benutzerschnittstellen, welche Green & Jacob (1992) vorschlagen:

4.1.1 Anforderungen, Teil 1: Charakteristik zukünftiger Benutzerschnittstellen (nach Green & Jacob 1992):

Hohe Bandbreite für Ein- und Ausgabe:

Non-WIMP benötigt eine hohe Bandbreite für Ein- und Ausgabe.

[1] Statt „non-WIMP“ trifft die Bezeichnung „post-WIMP“ neuartige Benutzerschnittstellen besser, denn sie schließt Kommandozeilen-Interfaces und noch ältere Schnittstellen aus. Trotzdem wird „non-WIMP“ üblicher Weise im Sinne von „post-WIMP“ gebraucht (Green 1992; Jacob 1993; Nielsen 1996).

Beispiele:
Ausgabe: Virtual Reality (VR) Displays mit einer Bildwiederholrate von 15 /s.

Eingabe: Zum Zeichnen einer Handschrift muß die Position eines Stifts mehrmals pro Sekunde ausgelesen werden. Auch Spracherkennung und Eyetracking benötigen regelmäßig viele Daten vom Benutzer, um sinnvoll arbeiten zu können.

Viele Freiheitsgrade:

Non-WIMP-Schnittstellen geben dem Benutzer viele Freiheitsgrade in Applikationen und Interaktionsgeräten.

Beispiel:
Mit Hilfe eines Datenhandschuhs mit 16 Freiheitsgraden kann ein Objekt in einer VR Umgebung manipuliert werden. Es besteht das Problem der Abbildung der Freiheitsgrade des Eingabegerätes auf Reaktionen der Anwendung.

Echtzeit Systemantworten:

Um die hohe Bandbreite von non-WIMP ausnutzen zu können, müssen Systemantworten unmittelbar auf Eingaben erfolgen. Dies erfordert eine hohe Rechenleistung.

Beispiele:
Im Falle eines head-mounted Displays muß spätestens 0,4 Sekunden nach dem Bewegen des Kopfes ein Bild aus der neuen Position generiert werden (Green & Jacob 1992). Beim Schreiben mit einem Computerstift muß die virtuelle Tinte der Bewegung des Stiftes folgen. Wird das Auge des Benutzers zur Eingabe herangezogen, so ist eine unmittelbare Auswertung der Augenposition erforderlich, und dem Benutzer eine Reaktion anzuzeigen.

Kontinuierliche Systemantworten und Rückkopplungen:

In non-WIMP-Schnittstellen gibt es keine expliziten Kommandos, statt dessen gibt es eine kontinuierliche Interaktion zwischen Benutzer und Applikation. Die Eingaben des Benutzers können nicht als diskrete Ereignisse mit einem festen Anfang und Ende interpretiert werden, deshalb muß das System kontinuierlich auf die Eingaben reagieren.

Beispiel:
In einer VR Umgebung mit Gravitationssimulation müssen Position und Verhalten von Objekten kontinuierlich anhand der sie beeinflussenden Faktoren bestimmt werden, z.B. wenn ein Benutzer eines VR Systems eine Teekanne über ihren Schwerpunkt hinaus an den Rand eines Tisches schiebt.

Wahrscheinlichkeitsbasierte Eingabe:

Hinter einer Eingabe des Benutzers können meist verschiedene Absichten stehen. Da non-WIMP Systemen unmittelbar auf Eingaben reagieren müssen, ist es in solchen Situationen erforderlich, zu vermuten, was der Benutzer will, und entsprechend zu reagieren. War die Vermutung falsch, muß es dem Benutzer einfach gemacht werden, die Reaktion des Systems rückgängig zu machen (trivial undo; Jacob 1990).

Beispiel:
Spracherkennung mit 70%iger Erkennungsrate produziert eine Menge möglicher

Wörter, aus denen die wahrscheinlichste ausgewählt und vom Benutzer im Fehlerfall per Hand korrigiert werden kann.

Jacob Nielsen formuliert Dimensionen, in denen sich zukünftige Benutzerschnittstellen von heutigen unterscheiden werden. Er streift dabei auch Gebiete, die für die Realisierung kommandofreier Interaktionen nicht unmittelbar von Belang sind:

4.1.2 Anforderungen, Teil 2: Dimensionen, in denen sich zukünftige Benutzerschnittstellen von heutigen unterscheiden werden, nach (Nielsen 1996):

Syntax:

Während der Interaktion mit Computern müssen Benutzer bisher eine Syntax beachten, die die Reihenfolge von Kommandos und des Gebrauchs von Dialogbausteinen vorschreibt. Es existiert eine Interaktionssprache, in der Objekte (Substantive) und die darauf auszuführenden Aktionen (Verben) durch syntaktische Mittel explizit zusammengefügt werden müssen. Zukünftige Benutzerschnittstellen werden dagegen syntaxfrei sein, da sie gestenbasiert Handlung und bezogenes Objekt als zusammengehörige Einheiten (interaction units) erkennen.

Aufmerksamkeit (Fokus) des Benutzers:

Benutzer müssen heute der Kontrolle des Computersystems hohe Aufmerksamkeit widmen. Zukünftige Benutzerschnittstellen werden dem Benutzer dagegen erlauben, sich auf die Arbeit zu konzentrieren (focus on).

Rolle des Computers:

Die Rolle des Computers besteht heute darin, aufs Wort (Kommando) zu gehorchen, obwohl es vielen Benutzern wahrscheinlich lieber wäre, der Computer würde das tun, was sie tatsächlich wollen, statt dem, was sie sagen zu wollen. In Zukunft werden Computer den Benutzer beobachten, seine Aktionen interpretieren und dem entsprechend handeln.

Kontrolle der Schnittstelle:

Die Computerschnittstelle wird bisher durch den Benutzer kontrolliert, die Schnittstelle wird dazu meist explizit sichtbar gemacht. Künftige Schnittstellen werden das dagegen dem Computer überlassen. Manchmal werden Computer auch Aktionen ausführen, ohne daß der Benutzer eine entsprechende Eingabe gemacht hat.

Sichtbarkeit von Objekten:

In zukünftigen Benutzerschnittstellen sollen Informationsüberflutungen durch eine zu große Anzahl sichtbarer Objekte vermieden werden. Als Nebenneffekt von Benutzereingaben, oder durch Handlungen von Intelligenten Agenten, werden dazu Objekte unsichtbar gemacht oder manipuliert.

Interaktions-Strom:

Heutige Benutzerschnittstellen können nur einen Eingabestrom gleichzeitig verarbeiten. Zukünftige Benutzerdialoge werden dagegen mehrkanalig sein. Der Benutzer wird verschiedene Eingabegeräte gleichzeitig bedienen können.

Interaktions-Bandbreite:

Die Bandbreite heutiger Benutzerschnittstellen ist sehr gering (Tastatur) bis gering (Maus). In Zukunft wird sie hoch bis sehr hoch sein (VR-Tracking Systeme) .

Tracking-Feedback:

Durch ein frühzeitiges Erkennen der Intentionen des Benutzers kann man ihn auf die Folgen seines Handeln hinweisen und ihm so helfen, Fehler zu vermeiden, bevor die Aktion vollendet ist. Bisher geschah dies z. B. beim Einsatz von Icons, die sich veränderten, wenn sie über andere Icons gezogen wurden (z. B. Papierkorb). Dieses Feedback kann jetzt kontinuierlich und in semantischem Zusammenhang gegeben werden.

Kontinuierliche Benutzerdialoge:

Traditionelle Benutzerschnittstellen sind dialogbasiert in dem Sinne, daß Benutzer und Computer wechselseitig Mitteilungen austauschen. Während der Computer auf Kommandos des Benutzer wartet, bleibt er untätig, ebenso wie die Eingaben des Benutzers während Antwort-Wartezeiten nicht verarbeitet werden (turn-taking). Im Gegensatz dazu werden zukünftige Benutzerschnittstellen keine definierten Wendepunkte für Ein- und Ausgabe haben, sondern kontinuierlich aufeinander reagieren. Der Benutzer wird seine Eingabe nicht mehr als diskrete Kommandos wahrnehmen, sondern sich in einen kontinuierlichen Kommunikationszustand versetzt fühlen.

Räumlicher Ort der Schnittstelle:

Benutzer waren bisher so sehr an den Bildschirm gekettet, daß viele nicht-technischen Benutzer über den Bildschirm reden, als wäre es der Computer. Es gibt aber eine Vielzahl von menschlichen Tätigkeiten, die nicht am Schreibtisch ausgeführt werden. Deshalb werden Computer zukünftig in die Umgebung des Benutzers, seinen Arbeitsraum und -haus, eingebunden sein (siehe dazu auch Buxton 1995).

Programmierung:

Programmiert wird bisher in textuellen imperativen Sprachen und Makrosprachen. Programmierer grafischer Schnittstellen müssen deshalb Objekte, die sie für die Modellwelt konstruieren, in der Konversationswelt entwickeln. Die künftige Programmierung grafischer Benutzerschnittstellen wird dagegen auch grafisch und objektorientiert erfolgen, z. B. Ereignis-basierte Programmierung, die auf Gesten-muster reagiert (siehe auch Jacob 1983, 1996), oder „visuelle Programmierung“, z.B. mit grafisch repräsentierten Produktionsregeln (siehe auch Yamamoto 1996).

Software-Pakete:

Wenn nach der Programmierung auch die Betriebssysteme objektorientiert werden, verschwindet die Notwendigkeit für Applikationen, die ihre gesamte Funktion monolithisch gebündelt haben. Dagegen wird es plug-in-Komponenten geben, die der Benutzer zu Programmen hinzufügen und entfernen kann. So wird es z. B. nicht mehr nötig sein, mehrere Programme mit Rechtschreibprüfung auf einem Rechner installiert zu haben, sondern alle textverarbeitenden Module können auf dasselbe Rechtschreibmodul zugreifen.

Die von Green & Jacob (1992) und Nielsen (1996) vorgebrachten Vorschläge zur Verbesserung heutiger Benutzerschnittstellen, die kommandofreie Interaktionen ermöglichen würden, sind für sich genommen alle sinnvoll; sie beleuchten jeweils un-

terschiedliche Aspekte künftiger Benutzerschnittstellen. Durch sie wird jedoch nicht sichergestellt, daß alle Aspekte von Benutzerschnittstellen betrachtet werden, bzw. wichtige Aspekte nicht unberücksichtigt bleiben. Dazu bedarf es einer Systematik wie sie z. B. Geiser (1990) vorschlägt, indem er am Menschen orientierte Gestaltungsaufgaben in technischen Gestaltungsbereichen definiert.

4.2 Ansätze kommandofreier Interaktionen in der Virtual Reality

In der Virtual Reality (VR) Technologie werden schon heute, wenn auch wahrscheinlich unbeabsichtigt, viele Anforderungen an kommandofreier Interaktionen und non-WIMP-Schnittstellen erfüllt. Es lohnt sich daher eine Analyse dieser Technologie, um Anregungen zu übernehmen und Verbesserungsvorschläge machen zu können. Zukünftige Benutzerschnittstellen werden zum Teil auf Systemen und nach Konzepten realisiert werden, die ihren Ursprung in der Virtual Reality haben. Das gilt besonders für die visuelle und akustische 3D-Ausgabe und das hohe Maß an Interaktivität. Das originäre Einsatzgebiet der VR liegt im Einsatz in immersiven Umgebungen, d. h. daß der „Cybernaut“ (Benutzer) durch die Benutzung der zur Verfügung stehenden Ein- und Ausgabetechnologien in eine virtuelle Welt (VR) oder virtuelle Umgebung (Virtual Environment - VE) eintaucht und mit seiner natürlichen Umgebung nicht mehr interagieren kann.

Burdea & Coiffet (1994) nennen neben der *Immersion* auch die *Interaktion* und *Imagination* als zentrale Bestandteile der VR (die drei I’s der VR).

Immersion (Eintauchen) in die virtuelle Welt bedeutet, daß dem Menschen in erster Linie visuell das Gefühl vermittelt wird, in diese Welt integriert zu sein. Ein hoher Grad an Immersion wird dann erreicht, wenn die menschlichen Sinne entsprechend manipuliert werden können. Bisher werden außer dem visuellen auch der akustische und der Tastsinn einbezogen.

Interaktion mit der virtuellen Umgebung und den darin vorhandenen Objekten integriert den Benutzer als Handelnden in die virtuelle Realität. Der Nutzen eines VR-Systems ist stark abhängig von der Möglichkeit, die der Benutzer hat, in das System einzugreifen und es nach seinen Vorstellungen zu verändern.

Die *Imagination* (Vorstellungskraft) des Benutzers ist es schließlich, die den Eindruck des Erlebens der virtuellen Realität entstehen läßt. Die Imagination ist abhängig von der Qualität der Interaktion und Immersion (bzw. Integration s.u.).

Hervorzuheben ist hier das hohe Maß an Interaktivität, das VR-Systeme ihren Benutzern ermöglichen und das durch die Ein- und Ausgabetechno-logien realisiert wird. Dabei sind in VR-Systemen weiterhin aus WIMP- und Komman-do-basierten Schnitt-stellen bekannte Inter-aktionsformen vorhan-den. So setzen z.B. viele VR-Systeme Toolboxen ein, mit denen der Benutzer dem VR-System Kommandos geben kann (Abbildung 4-1). Aufgrund der an die menschliche Sensorik und Motorik angepaßten Ein- und Ausgabetechnologien sind in VR-Systemen syntaxfreie Interaktionen möglich. Hier sind vor allem die Ausrichtung auf stereoskopische visuelle Anzeigen (Sutherland 1968), die Interaktionsmöglichkeiten über Hand und Körpergesten und die Einbeziehung von akustischer Ausgabe zu nennen. Beispiele für kommandofreie Interaktionen in VR-Systemen sind:

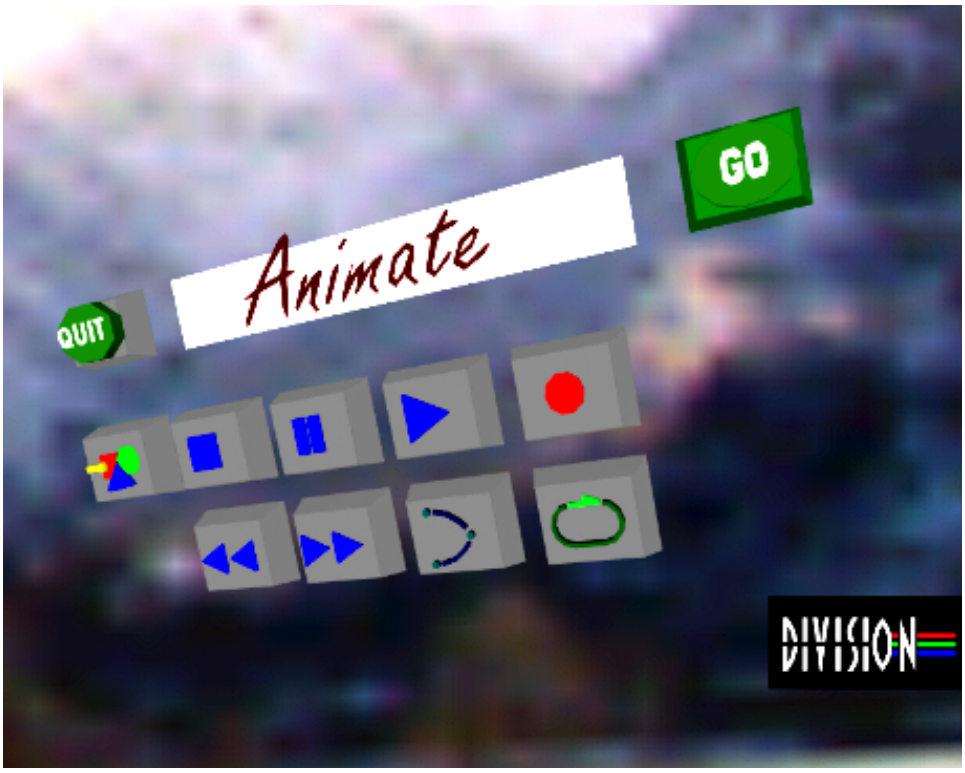

Abbildung 4-1: Kommandobasierte, WIMP-ähnliche Interaktion über eine Toolbox im VR-System dVise (1996).

Direkte Manipulation von Objekten, z. B. das Greifen und Manipulieren von Objekten mit der Hand durch den Einsatz von Datenhandschuhen

Mitführung der Szene bei Kopfbewegungen (dynamische Perspektive)

Hohe Bandbreite für Eingabe (Datenhandschuh, Positionstracker) und Ausgabe (3D-Bildausgabe mit hoher Wiederholfrequenz)

Echtzeitfähigkeit z. B. bei Simulationen und interaktiven Szenarien

Integration von Wahrnehmungs- und Handlungsraum bei immersiven Systemen

Anpassung an die menschliche Sensorik durch stereoskopische 3D-Visualisierung und akustische Stereoausgabe

Anpassung an die menschliche Motorik durch Einbeziehung von Handapparat, Kopf- und Körperposition als Informationskanäle

Anpassung an die menschliche Kognition durch Simulation natürlicher Interaktionsbedingungen wie Gravitation, Beleuchtung, Akustik.

Problematisch ist die Ausrichtung der VR auf Immersion in Bezug auf Akzeptanz und Einsatzmöglichkeiten dieser Technologie. Es ergeben sich sowohl ergonomische Probleme als auch eine Sinnkrise in den Anwendungsmöglichkeiten. Beim Ansatz der (nicht immersiven) Desktop VR steht die *Integration* des VR-Systems in die Arbeitswelt des Benutzers im Vordergrund, was sich in der verwendeten Ein- und Ausgabetechnologie niederschlägt. Desktop-VR-Systeme erreichen gegenüber immersiven VR-Systemen eine hohe und kurzfristige Verfügbarkeit, sie ermöglichen die

Ausführung anderer herkömmlicher Arbeiten und Tätigkeiten parallel zum Betrieb des VR-Systems.

4.3 Blickgesteuerte Interaktionen

4.3.1 Das Potential und Probleme blickgesteuerter Interaktionen

Computer versorgen ihre Benutzer mit einer Vielzahl von Informationen, aber die Informationen, die der Computer bislang vom Benutzer erhält, sind spärlich und langsam (siehe Kapitel 3). Dieses unausgewogene Verhältnis kann durch schnelle und einfache Benutzereingabe ausgeglichen werden. Dazu das menschliche Auge heranzuziehen, ist eine interessante Herausforderung. Durch seine Flexibilität und die hohe Geschwindigkeit seiner Bewegungen kann das Auge viele Informationen transportieren, sein Verhalten macht eine Auswertung jedoch schwierig (Jacob 1990).

Abbildung 4-2: Gazetracking-Kamera: Mit einem Infrarotlicht wird das Auge beschienen. Die bewegliche Kamera richtet sich auf das Auge aus und nimmt es auf.

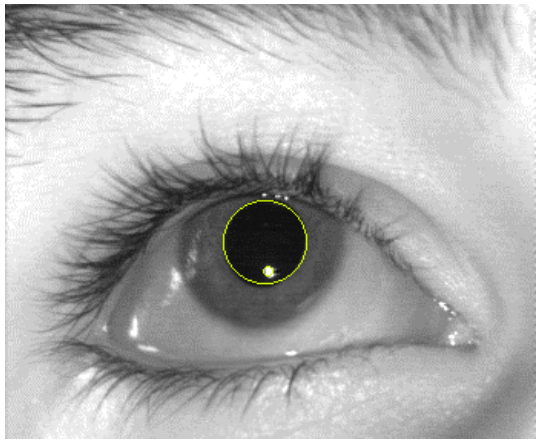

Abbildung 4-3: Kornea-Reflex-Methode: Aus der Position der Pupille und eines Reflexpunktes wird die Blickrichtung ermittelt. Der Reflexpunkt wird durch Infrarotlicht erzeugt und bleibt für den Benutzer unsichtbar. Aus Talmi (1997)

Die Technik des *Gazetracking* wurde bisher vor allem für an den Händen behinderte Menschen eingesetzt, die so mit dem Computer interagieren konnten. Für eine hohe Akzeptanz beim Benutzer ist es besser, davon auszugehen, daß der Benutzer seine Augen natürlich bewegt, anstatt ihn zu trainieren. Deshalb ist es auch fragwürdig, Gazetracking vorrangig als Mausersatz zu sehen. Diese Ansicht rührt aus den Annahmen, daß heutige Computerinteraktion auf der Maus als Eingabegerät beruht und daß dieses Gerät ersetzt werden muß. Die (Computer-) Maus als Eingabegerät ist aber nicht sakrosankt, von neuen Eingabetechnologien kann man sich ähnliche Evolutionssprünge erhoffen, wie ihrerzeit durch die Maus. Tatsächlich eignet sich das Auge aufgrund seiner Charakteristik nicht als Mausersatz, dafür ist es nicht geschaffen, und es würde überanstrengt. Vielmehr kann man anhand dessen, was der Benutzer betrachtet, wertvolle Informationen über seinen „Zustand“ bzw. seine „Wünsche“ und „Intentionen“ erhalten und entsprechend darauf reagieren.

Herkömmliche kommandobasierte Eingabegeräte, die mit der Hand gesteuert werden, haben mindestens zwei Funktionen: 1. ein Kommando anzuwählen (Tastatur:

eintippen; Maus: point) und 2. dieses Kommando zu aktivieren (Tastatur: ENTER-Taste; Maus: click). Mit dem Auge kann man dagegen nichts explizit auswählen, es gibt nur ein point, aber kein click. Es gibt zunächst keine Möglichkeit, zwischen einem bedeutungsvollen und einem bedeutungslosen Blick zu unterscheiden. Somit ist das Auge für kommandobasierte Interaktionen weniger geeignet. Wenn durch das einfache Betrachten des Bildschirms ständig Aktionen ausgelöst werden, entsteht das „Midas Touch Problem“[1], bei dem der Benutzer nirgend wo hin sehen kann, ohne damit Aktionen auszulösen (Jacob 1990). Für Vorgänge, die leicht rückgängig gemacht werden können („trivial undo“), ist eine kontinuierliche Reaktion auf den Blick vertretbar. Aber auch in non-command Schnittstellen gibt es sicherheitskritische Aktionen, die einer Absicherung bedürfen, z. B. das Löschen von Dateien. Die einfachste Methode zum Abfangen ungewollter Computerhandlungen ist die Verwendung von Latenzzeiten („dwell-time“), d. h. das Auslösen zu verzögern und abzubrechen, sobald der Benutzer seinen Blick abwendet. Zu große Latenzzeiten, die bei stark sicherheitskritischen Aktionen notwendig werden, verursachen aber eine Überanstrengung des Benutzers, da er auf Objekte „starren“ muß. Statt dessen sollten Anwendungen unterschiedlich auf den Blick reagieren, und nicht einfach Konzepte aus der WIMP-Welt übertragen. Das Potential blickgesteuerter Interaktion liegt nicht im einheitlichen Reagieren auf Kommandos, sondern in der aufgabenspezifischen Reaktion auf die Benutzerinteressen, die man aus dem Blick (teilweise) herauslesen kann (Jacob 1993).

Um das Auge nur zum Zeigen zu benutzen und Aktionen mit Gedanken auszulösen, müßten Gehirnströme gemessen werden (z. B. EEG Daten). Damit würden viele Probleme gelöst: „point with your eye and click with your mind“. Man kann davon ausgehen, daß die medizinische Forschung auf diesem Gebiet bald für Computerinterfaces eingesetzt werden kann (Hansen & Velichkovsky 1996; Healey & Picard 1997). Die Fragen nach der Akzeptanz und Genauigkeit solcher Interaktionstechniken sind noch ungeklärt.

4.3.2 Verhaltensmuster des Auges, Technik des Gazetrackings

Wenn ein Mensch ein Objekt scharf sehen will, muß er seine Pupille so bewegen, daß das Objekt auf der Fovea, einem kleinen Bereich auf der Retina, erscheint. Deshalb kann anhand der Augenposition ziemlich gut darauf geschlossen werden, welcher Teil einer Szene betrachtet wird. Beim *Gazetracking* wird die Blickrichtung der Augen und die Position des Augapfels im Raum gemessen. Hierfür gibt es unterschiedlich genaue und angenehme Ansätze. So kann über die Messung der natürlichen Spannungsunterschiede zwischen Cornea und Retina die Position ortsunabhängig, d. h. unabhängig von der Position des Betrachters, festgestellt werden. Den Anspruch nach Akzeptanz und Einfachheit kommt man am nächsten, wenn man Videobilder vom menschlichen Auge auswertet (Jacob 1993). Diese Methode verlangt vom Benutzer, daß er sich nicht bewegt, da Bewegungen von Kopf und Pupille nicht unterschieden werden können. Will man Ortsunabhängigkeit erreichen, müssen die Pupille und ein weiterer Referenzpunkt gefunden, und deren relative Bewegung zu-

[1] König Midas, bis 695 v. Chr. Herrscher von Phrygien, war in der Antike vornehmlich als Sagengestalt bekannt. In den „Metamorphosen“ berichtet der römische Dichter Ovid, daß Dionysos, der Gott des Weines, Midas die Gabe verlieh, alles Berührte zu Gold werden zu lassen. Diese Gabe erwies sich vor allem bei der Nahrungsaufnahme als hinderlich.

einander gemessen werden (Talmi 1997) (Abbildung 4-3). Außerdem muß die Kamera der Kopfbewegung nachgeführt werden (Bala 1997) (Abbildung 4-2), oder sie wird fest am Kopf angebracht.

Das für die Messungen relevante Verhalten von Augen wird von Robert Jacob folgendermaßen beschrieben: „Die häufigste Art der Augenbewegung ist eine plötzliche, ballistische und fast augenblickliche Sakkade. Sie wird typischerweise von einer Fixierung gefolgt, einer 200-600 ms langen Periode der relativen Stabilität, während der das Objekt betrachtet werden kann. Während der Fixierung führt das Auge trotzdem noch kleine, zitternde Bewegungen aus, die nicht größer als ein Grad sind.“ Nach Jacob sind gleichmäßige Augenbewegungen viel seltener als Sakkaden und treten nur in Zusammenhang mit Objekten auf, die sich durch das Betrachtungsfeld bewegen. „Das Gesamtbild der Augenbewegungen eines Benutzers, der vor einem Computer sitzt, ist eine Ansammlung starrer (aber leicht zitternder) Fixierungen, die durch plötzliche, schnelle Sakkaden verbunden sind.“ Jacob weist auch auf den grundsätzlichen Unterschied zwischen Blicksteuerung und Mauseingabe hin. „Verglichen mit der langsamen und überlegten Art und Weise, mit der Menschen die Maus oder andere manuelle Eingabegeräte benutzen, verlaufen Augenbewegungen völlig unkontrolliert über den Bildschirm. Während einer Fixierung denkt ein Benutzer normalerweise, daß er starr auf ein einzelnes Objekt sieht, und ist sich der kleinen, zitternden Bewegungen nicht bewußt. Daraus folgt, daß Mensch-Computer-Dialoge auf diese Bewegungen auch nicht reagieren sollten. Statt dessen sollten sie darauf reagieren, was der Benutzer denkt zu tun, und nicht darauf, was seine Augenmuskeln tatsächlich tun“ (Jacob 1990, S. 12).

Diese Beschreibung zeigt, daß man Augenbewegungen nicht in der gleichen Art wie herkömmliche Eingabegeräte verwenden kann. Vielmehr bedarf es des Herausfilterns der Sakkaden, um das Auge zu einem nützlichen Eingabegerät werden zu lassen (Hansen & Velichkovsky 1996). Entscheidend ist dabei, ab wann der „Augenblick“ Auswirkungen auf das System hat, („dwell-time“, „Midas Touch Problem“). Dafür gibt es keine allgemeine Lösung, dieses Problem ist nur in Zusammenhang mit der jeweiligen Anwendung zu klären.

Die Genauigkeit des Gazetracking beträgt mit den bisher bekannten Technologien allgemein zwei Grad, unter besonders günstigen Umständen (guter Kalibrierung etc.) bis zu einem Grad des Blickwinkels (Jacob 1993). Ein im Heinrich-Hertz-Institut in Berlin entwickeltes Verfahren, erreicht eine Genauigkeit von bis zu 0,3 Grad (Abbildung 4-3) (Talmi 1997).

5. Zusammenfassung

„Und wie kann man klicken?“ ist die häufigste Frage, die man hört, sobald von blickgesteuerter Interaktion erzählt wird. Das Konzept der kommandofreien, blickbasierten Interaktion ist gerade denen schwer begreiflich, die froh sind, ihren eigenen Computer mittlerweile so zu beherrschen, daß er das tut, was sie wollen oder meinen zu wollen.

Bisher wurde vom Menschen verlangt, daß er seine Probleme, so er sie mit dem Computer lösen wollte, in computerverständliche Teilprobleme und Aussagen zerlegen mußte. Das Bedienen und Kommandieren des Computers bedeutete für den Menschen nicht nur eine große Last, sondern führte auch zu einer Verschiebung der

Wahrnehmung und Verkennung von Problemen. Feinheiten, Differenzierungen und außerordentliche Spezifika gingen verloren, indem natürliche Sachverhalte in computerverständliche Ja/Nein-Aussagen zerteilt wurden. Es galt, daß ein Problem erörtert ist, sobald es der Computer verarbeiten kann.

Demgegenüber wird der monolithische „Computer“ bald Geschichte sein. Statt dessen wird es viele, unterschiedliche, intelligente Objekte und Dinge geben, die man anfassen, greifen und mit denen man reden und kommunizieren kann. Mit wachsenden Erwartungen an das natürliche Interaktionsverhalten von Objekten steigt jedoch auch die Gefahr der Enttäuschung, wenn Interaktionen nicht befriedigend gelingen. Schließlich ist die ethische Frage zu klären, welche Rollen „Computer“ in unserem Leben einnehmen und wie „intelligent“ und „menschlich“ sie sein dürfen, damit ihre Auswirkungen abgeschätzt werden können.

Voraussetzung für eine erfolgreiche Fortentwicklung der Mensch-Maschine-Beziehung ist ein Umdenken bei denen, die Benutzerschnittstellen und Maschinen entwerfen. Ernüchternd ist die Betrachtung des bisherigen Einflusses von am Mensch orientierten Wissenschaften wie Kommunikationswissenschaften, Psychologie, Motologie, Sinnesphysiologie, Erziehungswissenschaften, Medizin, Kunst- und Kulturwissenschaften auf die *inhaltliche* Organisation der Mensch-Maschine-Kommunikation. Wer Dinge entwirft, mit denen Menschen arbeiten sollen, muß aber wissen, *wie* Menschen arbeiten. Grundlage dafür ist eine interdisziplinäre Ausbildung und Forschung und ein Bewußtsein für den Vorgang Interfacedesign.

6. Literaturverzeichnis

Brooks, A. & Johnston, F. (1990). An inductive analysis of a snapshot of everyday WIMP usage. *University of Strathclyde, Research Report HCI-2-90*.

Burdea, G. & Coiffet, P. (1994). *Virtual Reality Technology*. John Wiley & Sons, Inc.

Buxton, W. (1995). Ubiquitous Media and the Active Office. *Nikkei Electronics 3.27 (no. 632)*, S. 187-195.

Crawford, C. (1992). Lessons from Computer Game Design. In B. Laurel (ed.). *The Art of Human-Computer Interface Design* (S. 103–111). Reading, Mass.: Addison Wesley.

dVise Users Guide (1996). Bristol, Division Limited, UK.

Gassée, J.L. & Rheingold, H. (1992). The Evolution of Thinking Tools. In B. Laurel (ed.). *The Art of Human-Computer Interface Design* (S. 225-227). Reading, Mass.: Addison Wesley.

Geiser, G. (1990). *Mensch-Maschine-Kommunikation*. München, Wien: Oldenbourg Verlag.

Green, M. & Jacob, R. (1992). Software Architectures and Metaphors for Non-WIMP User Interfaces. *ACM SIGGRAPH’92*.

Hansen, J.P. & Velichkovsky, B.M. (1996). New Technological Window into Mind: There is More in Eyes and Brains for Human-Computer Interaction. *CHI Proceedings*.

Healey, J. & Picard, R.W. (1997). Affective Wearables. *M.I.T. Media Laboratory Perceptual Computing Section Technical Report No. 432*. IEEE ISWC proceedings Vol.1, No.1, October 1997.

Henning, A. (1997). *Die andere Wirklichkeit: Virtual Reality - Konzepte, Standards, Lösungen*. Bonn: Addison Wesley.

Jacob, R.J.K. (1983). Using Formal Specifications in the Design of a Human-Computer Interface. *Communications of the ACM 26(4)*, S. 259-264.

Jacob, R.J.K. (1990). What You Look At Is What You Get: Eye Movement-Based Interaction Techniques. *ACM CHI'90 Proceedings* (S. 11-18). Reading, Mass.: Addison-Wesley/ACM Press.

Jacob, R.J.K. (1993). Eye-Movement-Based Human Computer Interaction Techniques: Toward Non-Command Interfaces. In H.R. Hartson & D. Hix (eds.). *Advances in Human-Computer Interaction, Vol.4*. Norwood: Ablex Publishing Co.

Jacob, R.J.K. (1996). A Visual Language for Non-WIMP User Interfaces. *Proc. IEEE Symposium in Visual Languages* (S. 231-238). IEE Computer Society Press.

Klingert, A. (1996). *Einführung in Grafische Fenstersysteme – Konzepte und reale Systeme.* Berlin, Heidelberg, New York: Springer-Verlag.

Kurtenbach, G. & Hulteen E. A. (1992). Gestures in Human-Computer Communication. In B. Laurel (ed.). *The Art of Human-Computer Interface Design* (S. 309–317). Reading, Mass.: Addison Wesley.

Laurel, B. & Mountford, S.J. (1992). Introduction. In B. Laurel (ed.). *The Art of Human-Computer Interface Design* (S. XI – XVI). Reading, Mass.: Addison Wesley.

Nielsen, J. (1996). Noncommand User Interfaces. *Communications of the ACM 36*, 4. überarbeitete Ausgabe, S. 83-99.

Reichmann, R. (1986). Communication Paradigms for a Window System. In D. Norman & S. Drapper (ed.). *User Centered System Design* (S. 285–313). London: Lawrence Erlbaum Associates, Publishers.

Skerjanc, R. & Pastoor, S. (1997). New Generation of 3-D desktop computer interfaces. *Proceedings of „Stereoscopic Displays and Virtual Reality Systems IV"*, San José, S. 439-447.

Talmi, K. (1997). *Verfahren zur Bestimmung der Blickrichtung und Kopfposition des Betrachters.* Diplomarbeit Technische Universität Berlin, Fachbereich Informatik.

Yamamoto, K. (1996). 3D-Visulan: A 3D Programming Language for 3D Applications. *Pacific Workshop on Distributed Multimedia Systems (DMS96)*, S. 199 – 206.